GENERIC INFERENCE

GENERIC INFERENCE

A Unifying Theory for Automated Reasoning

Marc Pouly
Cork Constraint Computation Centre
University College Cork, Ireland

Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg

Jürg Kohlas
Department of Informatics
University of Fribourg, Switzerland

WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at www.wiley.com.

Library of Congress Cataloging-in-Publication Data:

Pouly, Marc, 1980–, author.
Generic Inference : A Unifying Theory for Automated Reasoning / Marc Pouly, Jürg Kohlas.
p. cm
Includes bibliographical references and index.
ISBN 978-0-470-52701-6 (hardback)
1. Valuation theory. 2. Algorithms. 3. Algebra, Abstract. I. Kohlas, Jürg, 1939–,
author. II. Title.
QA162.P65 2011
519.5'4—dc22 2010042336

Printed in Singapore.

oBook ISBN: 9781118010877
ePDF ISBN: 9781118010846
ePub ISBN: 9781118010860

10 9 8 7 6 5 4 3 2 1

To Marita and Maria

CONTENTS

PART II GENERIC CONSTRUCTIONS

PART III APPLICATIONS

List of Instances and Applications

List of Figures

Acknowledgments

The authors would like to thank our collaborators from the Department of Informatics at the University of Fribourg (Switzerland), the Cork Constraint Computation Centre at the University College Cork (Ireland) and the Interdisciplinary Centre for Security, Reliability and Trust at the University of Luxembourg for the many interesting discussions that helped to improve the content and presentation of this book. In particular, we are grateful for the indispensable support of Christian Eichenberger during our journey through the Gaussian world and to Radu Marinescu and Nic Wilson for providing expert knowledge regarding inference, search methods and semiring systems. Jutta Langel and Jacek Jonczy both corrected parts of this book and provided valuable feedback. Special thank also goes to Cassie Craig from John Wiley & Sons, Inc. for the editing support and to the Swiss National Science Foundation, which financed the research stay of Marc Pouly at the Cork Constraint Computation Centre.

Marc Pouly & Jürg Kohlas

Introduction

Generic Algorithms

Abstract mathematical structures usually have a large number of models. Algorithms based on operations and laws of such structures therefore have an identical form: they are *generic*. This means that for each particular instance of the structure, only the basic operations have to be adapted, whereas the overall algorithm remains the same. The simplest and typical problem is the one of sorting. It applies to *totally ordered* structures with an *order relation* $\leq$. Any sorting algorithm can be formulated in terms of the order relation $\leq$ and it can be adapted to any particular model of an order relation by specifying this relation; for example for integers, real numbers or for a lexicographical order relation of alphanumeric strings. In addition, many programming languages allow for *generic programming*, i.e. in their syntax they provide the means to formulate generic algorithms and to specialize them to particular instances. In this book, an abstract algebraic structure called *valuation algebra* is proposed. It provides the foundation to formulate an abstract inference problem, to construct a graphical data structure and a generic inference algorithm to solve the inference problem. Usually, one arrives at such general structures from concrete problems and algorithms by lifting them to their most abstract form. This also holds for the present case. In 1988, Lauritzen and Spiegelhalter (Lauritzen & Spiegelhalter, 1988) proposed an algorithm for solving the inference problem for Bayesian networks based on a paradigm called *local computation*. Soon after, Shenoy and Shafer (Shafer & Shenoy, 1988; Shenoy & Shafer, 1990) noted that the same algorithm could also be applied to solve inference problems with belief functions. They proposed a small but sufficient system of axioms for an algebraic framework that makes possible the application of the generic inference algorithm. The algebraic theory of valuation algebras was developed in (Kohlas, 2003), and it was shown that they permit, under certain varying additional conditions, four different generic inference architectures. In (Pouly, 2008) a computer implementation of these generic architectures together with a number of concrete instances is described, see also (Pouly, 2010).

In the meantime, many different models of valuation algebras from very distant fields of Mathematics and Computer Science were identified. They range from probabilistic and statistical systems, different uncertainty formalisms to constraint systems, passing by various systems of logic, relational databases and systems of linear equations over fields and semirings. Many of these instances are presented in this book. Their computational interest can always be expressed in terms of the general inference problem, which can be solved by the same generic algorithm or inference architectures. This is first and foremost of great practical interest. Instead of developing and programming inference algorithms for each instance separately, it is possible to use a single generic system. Of course, in the past, individual solution algorithms and computer programs have been proposed and written mostly independent from each other, using the same basic concepts but often naming them quite differently. This adds to a certain Babylonian confusion, especially for students who should get an overall view of Computer Science. Further, it also represents a dissipation of valuable human resources in research and problem solving.

Second, the process of abstraction is of great theoretical importance. It provides a unifying view of seemingly different fields of Computer Science and allows to elaborate both similarities and differences between problem classes. In fact, valuation algebras allow for a very appealing general view of information processing: information comes in pieces, usually from different sources. Each piece of information concerns a particular domain and answers, possibly only partially, specific questions. Pieces of information can be aggregated or combined, which is one of the basic operations of a valuation algebra. So, the general inference problem essentially consists of combining all available information. But usually one is only interested in some particular facet or aspect of the total information. Therefore, the parts of the total information, which are relevant to precise questions, must be extracted. Extraction or projection of information is the second basic operation of a valuation algebra. In relational databases for example, combination corresponds to the *join operation*, whereas information extraction corresponds to the usual operation of *projection* in the relational algebra. So, valuation algebras show that this scheme of relational algebra is much more general. The form and nature of information elements may be very different from model to model.

Complexity Considerations

Combining pieces of information to solve the inference problem leads to what can be called a *race of dimensionality*. It was said above that each piece of information is associated with a certain domain. Combining information results in information on larger domains. So, when many pieces of information are combined, the domains may become very large. This often leads to serious problems of computational complexity, as we are next going to illustrate with two simple examples (Aji & McEliece, 2000):

Example I.1 *Let $f(X_1, X_2)$, $g(X_2, X_3)$ and $h(X_2, X_4, X_5)$ be three real-valued functions, where X_1 to X_5 are variables taking values from a finite set of n elements. Such discrete functions can always be represented in tabular form. This leads to tables with n^2 entries for f and g, and a table with n^3 entries for the function h. If we identify combination with component-wise multiplication, and projection with the summation of unrequested variables, we may define the following inference problem:*

$$\sum_{X_3, X_4, X_5} f(X_1, X_2) \cdot g(X_2, X_3) \cdot h(X_2, X_4, X_5). \tag{I.1}$$

Note that we combine all available information and project the result afterwards to the domain of interest, represented by the variable set $\{X_1, X_2\}$. This set is also called the query *of the inference problem, whereas the set of factors f, g and h is called* knowledgebase. *Although each knowledgebase factor can be represented by a small table of at most n^3 entries, the total product already needs a table of n^5 entries. In our small example, this may still be tractable, but the reader will recognize that with growing numbers of factors and variables, computing this product will fast become intractable. In fact, if s denotes the total number of variables, the complete table over all variables has n^s entries. We can see that this exhibits an* exponential growth *in the total number of variables. Thus, although the size of each factor in the knowledgebase may be small, computing their combination may be intractable. The next example points out a more realistic situation.*

Example I.2 *For $1 \leq i \leq 100$ we consider 100 real-valued functions, where each function f_i is defined over exactly three binary variables X_i to X_{i+2}. Again, each of these functions is representable by a table of $2^3 = 8$ entries. Similarly to the foregoing example, we then choose the variable set $\{X_{101}, X_{102}\}$ as query and obtain the following inference problem:*

$$\sum_{X_1, \ldots, X_{100}} f_1(X_1, X_2, X_3) \cdot \ \ldots \ \cdot f_{100}(X_{100}, X_{101}, X_{102}). \tag{I.2}$$

Here, each factor is represented by a small table of 2^3 entries. The total product on the other hand needs a table of 2^{102} entries that requires 10^{18} terabytes of memory. There is no computer with such a large memory capacity. If we further assume that a computer performs one multiplication per nanosecond, then these computations require at least 10^{12} years, which is 100 times longer than the estimated age of *the universe. We therefore conclude that the above sum cannot be computed by first computing the total product and then summing out the hundred variables. This example is quite representative for the inference problems considered in this book.*

So, what is then the escape from this race of dimensionality? We sketch the basic idea using the above examples. The secret lies in a clever use of the *distributive law* of arithmetics, which can often be used to arrange sequences of summations and multiplications such that all intermediate results stay manageable. This allows us to write equation (I.1) as

$$f(X_1, X_2) \cdot \left(\sum_{X_3} g(X_2, X_3) \right) \cdot \left(\sum_{X_4, X_5} h(X_2, X_4, X_5) \right).$$

We immediately observe that the largest intermediate result produced during the computation of this formula contains 2 variables and thus requires a tables of only n^2 entries. This is a considerable increase of efficiency which becomes even more apparent by applying the same idea to equation (I.2). We obtain

$$\sum_{X_{100}} \cdots \left(\sum_{X_2} \left(\sum_{X_1} f_1(X_1, X_2, X_3) \right) \cdot f_2(X_2, X_3, X_4) \right) \cdots f_{100}(X_{100}, X_{101}, X_{102}) \tag{I.3}$$

Here, the largest intermediate result contains only 3 variables that requires a table of 8 entries. Thus, instead of this huge amount of memory and computational time, the required resources remain proportional to the input size. Note also that besides the distributive law, we used the associative laws of addition and multiplication.

We may change the operation that is associated with projection in the above examples. Instead of addition, we now take the maximum value in Example I.1 that then leads to the following computational task:

$$\max_{X_3, X_4, X_5} f(X_1, X_2) \cdot g(X_2, X_3) \cdot h(X_2, X_4, X_5). \tag{I.4}$$

Both expressions (I.1) and (I.4) model inference problems over different valuation algebras, whereas the second problem now takes the shape of an optimization task. Obviously, the same complexity considerations with respect to the size of intermediate tables apply in both cases, and because the distributive law still holds between multiplication and maximization, we may also perform the same improvement:

$$f(X_1, X_2) \cdot \left(\max_{X_3} g(X_2, X_3) \right) \cdot \left(\max_{X_4, X_5} h(X_2, X_4, X_5) \right).$$

We thus have two different problems with different semantics and perhaps different application fields, but from the algebraic perspective they are both specializations of the same generic problem and can be solved by the same generic algorithm. This is exactly the idea of generic inference, and the secret of efficiency consists of a clever arrangement of the computations (combinations and projections) based on a generalization of the distributive law. Essentially the same technique was applied in the past for the processing of every specific inference formalism, which attests that the developers of these algorithms did not only solve the same generic problem, but they were also confronted with the same complexity concerns and finally hit upon the same solution. This is once more a convincing argument for the necessity of a generic inference theory, as it will be developed in the first part of this book.

The essence of valuation algebras consists in capturing these associative and distributive laws with respect to abstract operations of combination and projection. This will then be sufficient to rearrange the computations in the inference problem such that they remain feasible at each stage. In fact, this technique tends to keep storage proportional to the domain size of the input factors of the inference problem. One speaks of *local computation*, i.e. local on the domains of individual factors. Local computation is closely related to a well-known technique called *tree-decomposition* of graphs. Suppose a graphical representation of an inference problem where each node represents a variable. In addition, two nodes are linked by an edge, if their variables occur in the domain of a common factor or in the query. Figure I.1 shows the graph associated with the inference problem of Exercise I.1. If we again assume that there are s variables, each with a domain of n values, then the total information represented by the graph needs a table of n^s entries.

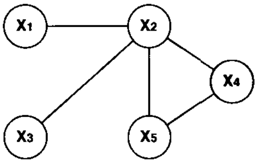


Figure I.1 The graph associated with the inference problem of Exercise I.1.

The key parameter, called *treewidth* ω of the graph, represents an important structural property. For the time being, we can imagine the treewidth as some measure related to the sparsity of the graph. Then, instead of the intractable complexity of n^s associated with the total graph, the operations of the inference problem can be rearranged in such a way that the complexity is reduced to $g(s) \cdot n^w$, where g is a *linear function*. Observe that the complexity is still *exponential*, but now in the treewidth ω instead of the total number of variables s. Since in many cases ω is much smaller than s, it represents a big gain and often turns a intractable problem into a tractable problem. This is a case of what has recently been called *parametric* or *parameterized complexity*; see for example (Downey & Fellows, 1999). However, we should also emphasize that the complexity of inference problems is not always exponential. There are important cases where it may be *polynomial*, for instance s^3 with s still being the number of variables. But we then still have the change in complexity from s^3 to $g(s) \cdot \omega^3$. It may thus be said that the polynomial or exponential complexity of the inference problem is a dimension somehow *orthogonal* to the framework of valuation algebras; but it always allows to reduce the complexity to a product of a linear function of problem size s and a polynomial or exponential function of treewidth ω according to the nature of the problem. That is essentially what valuation algebras achieve in terms of complexity.

Generic Constructions

From a practical point of view, the relevance of a generic theory can best be measured by the number of instances and applications covered by the theory. Here, a formalism is called an instance of a valuation algebra, if it satisfies the properties or axioms of the algebra. Besides the many concrete, individual instances presented in this book, we also describe generic approaches to construct whole *classes* of valuation algebras. We call this *generic constructions*. Essentially, these methods derive valuation algebras in a constructive manner from other algebraic structures such as fields, vector spaces or semirings. For example, we will see that matrices over particular semirings, systems of linear equation over arbitrary fields or mappings from configurations to semiring values always form a valuation algebra. Another generic construction identifies algebras derived from structures of *languages* with *models*, as in logic and many other fields. This adds to picture elements of valuation algebras as pieces of information, since information is often expressed in terms of a language; i.e. a formal language in our case. Generic constructions further allow to verify axioms and properties for whole classes of formalisms on a more abstract level and thereby simplify the theory. Finally, they also identify new formalisms that can be processed efficiently by the generic inference algorithms but which are yet unknown to the community.

The Content of this Book

This book is divided into three parts. The first part introduces the algebraic system of a valuation algebra and defines generic inference algorithms for their processing. The second part is entirely dedicated to generic constructions and the identification of new valuation algebra instances. Finally, the third part studies some selected applications of local computation. Some of these applications even go beyond the pure computation of inference, but they are nevertheless based on the same computational techniques. Typical examples are the construction of solutions for constraint and equation systems and sparse matrix techniques. Let us consider the organisation of the three parts in more detail:

Part I: Local Computation

Chapter 1 introduces the valuation algebra framework upon which all later chapters are based. For simplicity, we do not stress the most general axiomatic system that is based on arbitrary lattices with partial projection. Instead, we restrict ourselves to valuation algebras with full projection over variable systems that cover most practical applications. The two generalizations will be discussed in the appendix of this chapter. Also, we give first examples of well-known formalisms that satisfy the structure of a valuation algebra, including crisp constraints, relations, probability mass functions, belief functions or density functions.

Chapter 2 is dedicated to the definition of the generic inference problem that reflects the fundamental computational interest in valuation algebras. It will be distinguished between single-query and multi-query inference problems. We also introduce several possibilities to represent the structure of knowledgebases and give important applications that require the solution of inference problems with knowledgebases from different valuation algebras. This includes reasoning in Bayesian networks and Dempster-Shafer theory, satisfiability in logic and constraint systems, or discrete Fourier and Hadamard transforms.

Chapter 3 provides generic algorithms for the solution of single-query inference problems. In particular, it introduces the fusion algorithm, the bucket-elimination scheme and two variations of the collect algorithm. We also provide a detailed complexity analysis of these first local computation methods.

Chapter 4 extends the collect algorithm to take multiple queries into consideration, leading to the Shenoy-Shafer architecture. We will see that its complexity can be improved if the valuation algebra provides some concept of division. Based on this additional operator, three further local computation architectures for the solution of multi-query inference problems called Lauritzen-Spiegelhalter, Hugin and idempotent architecture are derived. The study of the algebraic requirements of a valuation algebra to provide a division operator is a more ambitious topic. Its comprehensive discussion is therefore postponed to the appendix of this chapter.

Part II: Generic Constructions

Chapter 5 presents the first generic construction that identifies valuation algebras by means of a simple mapping from configurations to values of a commutative semiring. This family of valuation algebras for example includes probability potentials, crisp constraints, weighted constraints, probabilistic constraints, possibilistic constraints and assumption-based constraints. A second family of valuation algebras is obtained from mapping sets of configurations to semiring values and includes in particular the formalism of belief functions from Dempster-Shafer theory.

Chapter 6 introduces path problems and shows that their computation amounts to the solution of a semiring fixpoint equation system. Based on this observation, two generic constructions related to matrices with semiring values are identified. Typical members and application fields of these families of valuation algebras are shortest path problems, maximum capacity problems, connectivity problems, path reliability, path counting, Markov chains or partial differentiation. Moreover, this chapter provides the preliminary work for the discussion of sparse matrix techniques in Chapter 9.

Chapter 7 deals with the duality between information and its representation in terms of a language. Typical examples of this generic construction are different kinds of logic and systems of equations and inequalities.

Part III: Applications

Chapter 8 points out that many valuation algebra have an associated notion of a solution. This is typically the case for systems of equations and inequalities, but it also holds in logic and constraint systems. The identification of solutions in arbitrary valuation algebras is the main topic in this chapter. It will be shown that based on a previous execution of a local computation architecture, solutions can be found without increasing the complexity of the local computation scheme. This results in two further generic algorithms to compute either a single solution or all solutions of a valuation given as factorization. In the second part of Chapter 8, this theory is applied to semiring constraint systems for the computation of solutions in optimization or constraints problems. Further specializations for equation systems will be discussed in Chapter 9.

Chapter 9 deals with sparse matrix techniques. The first part of the chapter focuses on sparse, linear systems over fields. It introduces Gaussian elimination and different decomposition approaches and establishes the connection to the valuation algebra of affine spaces introduced in Chapter 7. The famous least squares method provides an interesting case study. In the second part, we consider path problems that induce sparse fixpoint equation systems over semirings. This is based on the valuation algebras from Chapter 6. We will further show how local computation is used to solve the single-source, multiple-pairs or all-pairs version of different path problems.

Chapter 10 looks at linear systems with stochastic disturbances. In many important applications, these disturbances may be assumed to have a Gaussian distribution. Together with observations, such systems provide Gaussian information. It will be shown that several valuation algebras including Gaussian potentials and Gaussian hints hide behind Gaussian information. Also, this chapter investigates inference in Gaussian systems with local computation and provides an in-depth study of statistical, assumption-based reasoning.

All chapters in this book are completed by a collection of exercises and open problems. We distinguish three degrees of difficulty. One-star exercises are either simple finger exercises to actively digest the theory of the chapter, straightforward applications of the theory to concrete problems or omitted proofs with a reference to another textbook or journal article that contains the complete proof. Two-star exercises are small research projects that extend the theory of the chapter and establish links to related research topics. Finally, three-star exercises label comprehensive and mainly open research questions that could be included in a research program on valuation algebras and local computation.

Beyond the Content of this Book

This book is about generic local computation or tree-decomposition methods that are derived from the general valuation algebra framework. Additionally, we discuss several specializations and extensions of these algorithms for particular families of valuation algebras in the third part. However, it should also be mentioned that other techniques for the processing of inference problems exist. Depending on the concrete application, these methods may be less or more efficient than the class of algorithms presented in this book. But we also point out that it is in general not possible to apply these techniques to arbitrary valuation algebras, because they usually require more information about the concrete buildup of formalisms. A popular class of such algorithms are *search methods*, see for example (Nilsson, 1982; Pearl, 1984; Kanal & Kumar, 1988; Dechter, 2003). Search methods work through a search space that is obtained from a knowledgebase by assigning different values to variables. It is therefore clear that search methods are suitable only for formalisms that consist of mappings from finite variable assignments to some values. In particular, this covers the large family of semiring valuation algebras introduced in Chapter 5, but we will also meet numerous formalisms in this book that do not provide this structure. Search methods and local computation have also been combined to *hybrid methods* (Larrosa & Dechter, 2003; Dechter, 2006). A closely related aspect concerns the representation of valuations. This is again specific to each family of valuation algebras and goes along with tailored inference methods. Important techniques are *AND-OR search* (Dechter & Mateescu, 2007; Marinescu & Dechter, 2009a; Marinescu & Dechter, 2009b), *OR search with caching* (Bacchus *et al.*, 2003), methods based on *automata* (Vempaty, 1992; Fargier & Vilarem, 2004), *decision diagrams* (Wilson, 2005; Nicholson *et al.*, 2006) and various other techniques related to *knowledge compilation* (Darwiche, 2001; Darwiche & Marquis, 2001; Darwiche & Marquis, 2002; Wachter & Haenni, 2006; Fargier & Marquis, 2007; Wachter *et al.*, 2007; Wachter, 2008). We further restrict ourselves to *exact inference* in this book. There are several methods to perform *approximated inference* with local computation as for example the *mini-bucket scheme* (Dechter & Rish, 2003). We further refer to (Haenni, 2004) for approximate inference with *ordered valuation algebras* and to (Kohlas & Wilson, 2008) for semiring valuation algebras. It should also be mentioned that other algebraic frameworks for generic inference exists, e.g. (Pralet *et al.*, 2007), but these systems generally include the valuation algebra framework. Finally, it has already been mentioned in the first publications about local computation that these algorithms qualify for an implementation on distributed and parallel computing environments. In particular, local computation is successfully applied for inference in *sensor networks* (Paskin *et al.*, 2005). This raises many interesting questions that go beyond the scope of this book. Examples are the management of computing resources, the minimization of communication costs (Pouly, 2008) or the increase of parallelism. These questions are discussed for probabilistic reasoning in (Kozlov & Singh, 1994; Kozlov & Singh, 1996; Namasivayam & Prasanna, 2006; Yinglong & Prasanna, 2008), and we also refer to the comprehensive literature on parallel and distributed databases.

PART I

LOCAL COMPUTATION

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

CHAPTER 1

VALUATION ALGEBRAS

The valuation algebra framework provides the algebraic foundation for the application of all generic inference mechanisms introduced in this book and therefore marks the beginning of our studies. Comparable to the total order that is required for the application of sorting procedures, all formalisms must satisfy the structure of a valuation algebra in order to be processed by these generic inference tools. Further, this framework mirrors all the essential properties we naturally associate with the rather imprecise notions of knowledge and information. Let us engross ourselves in this thought and put our daily perception of information into words: information exists in pieces and comes from different sources. A piece of information refers to some specific questions that we later call the domain of an information piece. Also, there are two principal operations to manipulate information: we may combine or aggregate pieces of information to a new information piece and we may project a piece of information to some specific question which corresponds to information extraction. Depending on the operation, we either get a broader or more focused information. In the following section, we give a formal definition of the valuation algebra framework consisting of its operations and axioms. Along the way, we further bear on our idea of valuations as pieces of knowledge or information to clarify the rather abstract and formal structures.

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

1.1 OPERATIONS AND AXIOMS

The basic elements of a valuation algebra are so-called *valuations* that we subsequently denote by lower-case Greek letters $\phi, \psi, \ldots$ Intuitively, a valuation can be regarded as a representation of information about the possible values of a finite set of variables. We use Roman capitals $X, Y, \ldots$ to refer to variables and lower-case letters $s, t, \ldots$ for sets of variables. Thus, we assume that each valuation ϕ refers to a finite set of variables $d(\phi)$, called its *domain*. For an arbitrary, finite set of variables s, Φ_s denotes the set of all valuations ϕ with $d(\phi) = s$. With this notation, the set of all possible valuations for a countable set of variables r can be defined as

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

Let $D = \mathcal{P}(r)$ be the powerset (the set of all subsets) of r and Φ the set of valuations with their domains in D. We assume the following operations defined in $\langle \Phi, D \rangle$:

1. *Labeling:* $\Phi \rightarrow D$; $\phi \mapsto d(\phi)$;

2. *Combination:* $\Phi \times \Phi \rightarrow \Phi$; $(\phi, \psi) \mapsto \phi \otimes \psi$;

3. *Projection:* $\Phi \times D \rightarrow \Phi$; $(\phi, x) \mapsto \phi^{\downarrow x}$ for $x \subseteq d(\phi)$.

These are the three basic operations of a valuation algebra. If we readopt our idea of valuations as pieces of information, the labeling operation tells us to which questions (variables) such a piece refers. Combination can be understood as aggregation of information and projection as the extraction of the part we are interested in. Sometimes this operation is also called*focusing* or *marginalization*. We now impose the following set of axioms on Φ and D:

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

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

$$d(\phi \otimes \psi) = d(\phi) \cup d(\psi). \tag{1.1}$$

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

$$d(\phi^{\downarrow x}) = x. \tag{1.2}$$

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

$$(\phi^{\downarrow y})^{\downarrow x} = \phi^{\downarrow x}. \tag{1.3}$$

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

$$(\phi \otimes \psi)^{\downarrow z} = \phi \otimes \psi^{\downarrow z \cap y}. \tag{1.4}$$

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

$$\phi^{\downarrow x} = \phi. \tag{1.5}$$

These axioms require natural properties regarding knowledge or information. The first axiom indicates that Φ is a commutative semigroup under combination. If information comes in pieces, the sequence of their aggregation does not influence the overall result. The labeling axiom tells us that the combination of valuations yields knowledge about the union of the involved domains. Variables do not vanish, nor do new ones appear. The projection axiom expresses the natural functioning of focusing. Transitivity says that projection can be performed in several steps. For the combination axiom, let us assume that we have some information about a domain in order to answer a certain question. Then, the combination axiom states how the answer is affected if a new information piece arrives. We can either combine the new piece to the given information and project afterwards to the specified domain, or first remove the uninteresting parts of the new information and combine it afterwards. Both approaches lead to the same result. In fact, we are going to see in Section 5.3 that this axiom correlates with some generalized distributive law. Finally, the domain axiom ensures that information is not influenced by projection to its own domain, which expresses some kind of stability with respect to trivial projection.

Definition 1.1 *A system $\langle \Phi, D \rangle$ together with the operations of labeling, projection and combination satisfying these axioms is called a* valuation algebra.

In the first appearance of the valuation algebra axioms (Shafer & Shenoy, 1988; Shenoy & Shafer, 1990) only Axioms (A1), (A4) and a simpler version of (A5) were listed. Axiom (A2) was simply assumed in the definition of combination. (Shafer, 1991) mentioned Property (A3) for the first time and also remarked that Axiom (A6) cannot be derived from the others. (Kohlas, 2003) then assembled these early results to a complete and sufficient axiomatic system for generic inference and local computation. But in addition to the above system, all former approaches contained so-called neutral valuations that express vacuous information with respect to a certain domain. Here, we will introduce valuation algebras with neutral elements in Section 3.3 as a special case of the definition given here.

Before we turn towards the first concrete examples of valuation algebras, we list a few elementary properties that are derived directly from the above set of axioms.

Lemma 1.1

1. *If* $\phi, \psi \in \Phi$ *with* $d(\phi) = x$ *and* $d(\psi) = y$*, then*
$$(\phi \otimes \psi)^{\downarrow x \cap y} \quad = \quad \phi^{\downarrow x \cap y} \otimes \psi^{\downarrow x \cap y}. \tag{1.6}$$

2. *If* $\phi, \psi \in \Phi$ *with* $d(\phi) = x$*,* $d(\psi) = y$ *and* $z \subseteq x$*, then*
$$(\phi \otimes \psi)^{\downarrow z} \quad = \quad (\phi \otimes \psi^{\downarrow x \cap y})^{\downarrow z}. \tag{1.7}$$

Proof:

1. By the transitivity and combination axiom:
$$(\phi \otimes \psi)^{\downarrow x \cap y} = ((\phi \otimes \psi)^{\downarrow x})^{\downarrow x \cap y} = (\phi \otimes \psi^{\downarrow x \cap y})^{\downarrow x \cap y} = \phi^{\downarrow x \cap y} \otimes \psi^{\downarrow x \cap y}.$$

2. By the transitivity and combination axiom:
$$(\phi \otimes \psi)^{\downarrow z} = ((\phi \otimes \psi)^{\downarrow x})^{\downarrow z} = (\phi \otimes \psi^{\downarrow x \cap y})^{\downarrow z}.$$

■

1.2 FIRST EXAMPLES

Our first encounter with the valuation algebra framework took place on a very abstract level. To reward the reader for this formal effort, we now consider a first catalogue of concrete formalisms that satisfy the valuation algebra structure. Such formalisms are subsequently called *valuation algebra instances*. For the moment, we content ourselves with the observation of how these formalisms fit in the valuation algebra framework. Later sections then inform about their exact purpose and semantics. However, these instances are all based on the important concepts of *configurations*, *tuples* or *vectors* that shall first be introduced separately.

Frames, Tuples, Configurations and Vectors:

Consider a countable set r of variables where each variable $X_i \in r$ is referenced by its index $i \in \mathbb{N}$. Conversely, every index $i \in \mathbb{N}$ refers to a unique variable $X_i \in r$. This one-to-one correspondence between variables and indexes allows us subsequently to identify the two concepts. Further, we assume for every variable $X \in r$ a set Ω_X of possible values, called its *frame*. Such variable frames are sometimes assumed to be finite or countable, but this is not a general requirement. If a frame contains exactly two elements, the corresponding variable is said to be *binary*. Moreover, if the two elements represent the states *true* and *false*, the variable is called *propositional* or *Boolean*. A *tuple* or *configuration* with finite *domain* $s \subseteq r$ is a function

$$\mathbf{x} : s \rightarrow \Omega_s \;=\; \prod_{X \in s} \Omega_X$$

that associates a value $\mathbf{x}(X) \in \Omega_X$ with each variable $X \in s$. By convention, the single tuple with empty domain is identified by the diamond symbol $\diamond$, and we use bold-face, lower-case letters $\mathbf{x}, \mathbf{y}, \ldots$ to refer to tuples. Subsequently, we write $\mathbf{x} \in \Omega_s$ to mark $\mathbf{x}$ as a tuple with domain s, for which we also use the shorthand term *s-tuple* or *s-configuration*.

It is often convenient to decompose tuples according to some variable partition. This operation is also called *projection*, although it is not directly related to valuation algebras. Given an s-tuple $\mathbf{x}$ and $t \subseteq s$, the projection of $\mathbf{x}$ to t is defined by a t-tuple $\mathbf{y}$ such that $\mathbf{y}(X) = \mathbf{x}(X)$ for all $X \in t$. We subsequently write $\mathbf{x}^{\downarrow t}$ for the projection of $\mathbf{x}$ to t. Note that $\diamond$ denotes the projection of any tuple to the empty set. Further, this notation allows us to write $\mathbf{x} = (\mathbf{x}^{\downarrow t}, \mathbf{x}^{\downarrow s-t})$ for the decomposition of $\mathbf{x}$ with respect to t and $s - t$. Observe also that $(\mathbf{x}, \diamond) = (\diamond, \mathbf{x}) = \mathbf{x}$. Similar to frames of single variables, the set Ω_s represents all possible s-tuples and is therefore called the *frame* of the variable or index set s. In particular, $\Omega_\emptyset = \{\diamond\}$. Then, a *tuple set* with domain s is simply a subset $S \subseteq \Omega_s$. If a tuple set consists of only one element, then it is also called a *singleton*.

Example 1.1 *To describe the attributes color, speed and prize of a car, we assume the set of variables $r = \{C, S, P\}$ with variable frames: $\Omega_C = \{\text{red}, \text{blue}, \text{black}\}$, $\Omega_S = \{\text{slow}, \text{fast}\}$ and $\Omega_P = \mathbb{N}$. Two possible r-tuples are $\mathbf{x} = (\text{blue}, \text{fast}, 30\text{'}000)$ and $\mathbf{y} = (\text{black}, \text{slow}, 10\text{'}000)$. The tuple $\mathbf{x}$ can for example be decomposed into $\mathbf{x}^{\downarrow\{C,P\}} = (\text{blue}, 30\text{'}000)$ and $\mathbf{x}^{\downarrow\{S\}} = (\text{fast})$. Further, the variable set $\{C, S\}$ possesses the frame $\Omega_{\{C,S\}} = \{(\text{red}, \text{slow}), (\text{blue}, \text{slow}), \ldots, (\text{black}, \text{fast})\}$. Observe also that the frame Ω_r contains infinitely many tuples.*

If $X \in r$ is a variable that takes real numbers, we have $\Omega_X = \mathbb{R}$, and for a set $s \subseteq r$ of real variables the *linear space* $\mathbb{R}^s$ corresponds to the frame of all s-tuples. In this case, s-tuples are also called *s-vectors*. It is possible to introduce an *extension operator* that lifts an s-vector $\mathbf{x}$ to some larger domain $t \subseteq s$ by assigning zeros to all variables in $t - s$. We thus have $\mathbf{x}^{\uparrow t}(X) = \mathbf{x}(X)$ if $X \in s$ and $\mathbf{x}^{\uparrow t}(X) = 0$ otherwise. Clearly, the introduction of this operation is not only possible for real number but for all algebraic structures that contain a *zero element*. These definitions of frames, tuples and tuple sets allow for a uniform notation in the following tour through a first collection of valuation algebra instances.

1.1 Indicator Functions - Boolean Functions - Crisp Constraints

An *indicator function* with domain $s \subseteq r$ identifies a subset $S \subseteq \Omega_s$ by specifying for each tuple $\mathbf{x} \in \Omega_s$ whether $\mathbf{x}$ belongs to S or not. If we adopt the usual interpretation of 0 for $\mathbf{x}$ not being an element of S and 1 for $\mathbf{x}$ being in S, an indicator function i is defined by

$$i(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \notin S, \\ 1 & \text{if } \mathbf{x} \in S. \end{cases}$$

Thus, an indicator i with domain $d(i) = s$ is a function that maps every tuple $\mathbf{x} \in \Omega_s$ onto a value $i(\mathbf{x}) \in \{0,1\}$, i.e. $i : \Omega_s \to \{0,1\}$. These are the valuations in the valuation algebra of indicator functions. As introduced in Section 1.1, we usually write Φ_s for the set of all indicator functions with domain s and Φ for the set of all possible indicator functions over subsets of r. Combination of indicator functions is defined by multiplication. If i_1 and i_2 are indicator functions with domain s and t respectively, we define for $\mathbf{x} \in \Omega_{s \cup t}$

$$i_1 \otimes i_2(\mathbf{x}) \quad = \quad i_1(\mathbf{x}^{\downarrow s}) \cdot i_2(\mathbf{x}^{\downarrow t}).$$

Alternatively, we may also define combination in terms of minimization:

$$i_1 \otimes i_2(\mathbf{x}) \quad = \quad \min\{i_1(\mathbf{x}^{\downarrow s}), i_2(\mathbf{x}^{\downarrow t})\}.$$

Projection, on the other hand, corresponds to maximization. For an indicator i with domain s and $t \subseteq s$ we define for all $\mathbf{x} \in \Omega_t$

$$i^{\downarrow t}(\mathbf{x}) \quad = \quad \max_{\mathbf{y} \in \Omega_{s-t}} i(\mathbf{x}, \mathbf{y}).$$

If we consider only variables with finite frames, an indicator function with domain s can be represented by an $|s|$-dimensional table with $|\Omega_s|$ zero-one entries. Below, we simply use a *relational* or *tabular representation*, but emphasize that indicator functions are not limited to this inefficient representation. In order to prove that indicator functions form a valuation algebra, the reader may verify that all axioms are satisfied. However, in Section 5.3 this proof will be made for a whole family of formalisms that also includes indicator functions. Otherwise, a direct proof can be found in (Kohlas, 2003). Depending on its application field, the formalism of indicator functions has different names. In the context of constraint reasoning, they are usually called *crisp constraints*. Otherwise, indicator functions for propositional variables are also called *Boolean functions*.

For a concrete example of computing with indicator functions, consider a set of variables $r = \{A, B, C\}$ with frames $\Omega_A = \{a, \bar{a}\}$, $\Omega_B = \{b, \bar{b}\}$ and $\Omega_C = \{c, \bar{c}\}$. Then, assume two indicator functions i_1 and i_2 with domains $d(i_1) = \{A, B\}$ and $d(i_2) = \{B, C\}$:

$i_1 =$

A	B	
a	b	0
a	$\bar{b}$	1
$\bar{a}$	b	0
$\bar{a}$	$\bar{b}$	0

$i_2 =$

B	C	
b	c	1
b	$\bar{c}$	0
$\bar{b}$	c	1
$\bar{b}$	$\bar{c}$	1

We combine i_1 and i_2 and project the result to $\{A\}$:

$i_3 \;=\; i_1 \otimes i_2 \;=$

A	B	C	
a	b	c	0
a	b	$\bar{c}$	0
a	$\bar{b}$	c	1
a	$\bar{b}$	$\bar{c}$	1
$\bar{a}$	b	c	0
$\bar{a}$	b	$\bar{c}$	0
$\bar{a}$	$\bar{b}$	c	0
$\bar{a}$	$\bar{b}$	$\bar{c}$	0

$i_3^{\downarrow\{A\}} \;=$

A	
a	1
$\bar{a}$	0

1.2 Relational Algebra

The second instance we are going to study is closely related to the first one. It is a subset of the *relational algebra* (Maier, 1983; Ullman, 1988), which traditionally belongs to the most fundamental formalisms for representing knowledge and information. In its usual extent, at least six operations are provided to manipulate knowledge represented as sets of tuples. Respecting the language of relational database theory, variables are called *attributes* and sets of tuples are called *relations*. A relation over $s \subseteq r$ is therefore simply a tuple set $R \subseteq \Omega_s$. It is important to note that variable frames do not need to be finite and consequently, relations can also be infinite sets. Combination is defined by *natural join*: If R_1 and R_2 are relations with domain s and t respectively, we define

$$R_1 \bowtie R_2 \quad = \quad \left\{\mathbf{x} \in \Omega_{s \cup t} : \mathbf{x}^{\downarrow s} \in R_1,\ \mathbf{x}^{\downarrow t} \in R_2\right\}. \tag{1.8}$$

Projection is defined for a relation R with domain s and $t \subseteq s$ as

$$R^{\downarrow t} \quad = \quad \{\mathbf{x}^{\downarrow t} : \mathbf{x} \in R\}$$

Alternatively, the notation $\pi_t(R) = R^{\downarrow t}$ is also used in the context of relational algebras. If we consider only finite variables in r, then the relational algebra over r is isomorphic to the valuation algebra of indicator functions over r. In fact, relations are just another representation of indicators. We obtain the relation R_ϕ associated to the indicator function ϕ with domain $d(\phi) = s$ by

$$R_\phi \quad = \quad \left\{\mathbf{x} \in \Omega_s : \phi(\mathbf{x}) = 1\right\}. \tag{1.9}$$

Conversely, we derive the indicator function ϕ associated to a relation R with domain $s \subseteq r$ by $\phi(\mathbf{x}) = 1$ if $\mathbf{x} \in R$, and $\phi(\mathbf{x}) = 0$ otherwise, for all $\mathbf{x} \in \Omega_s$. Hence, we directly conclude that relations form a valuation algebra in the case of finite variables. This statement also holds for the general case as is will be shown in Chapter 7.3. Let us write the combination axiom for $z = x$ in the notation of relational algebra: For $R_1, R_2 \in \Phi$ with $d(R_1) = x$ and $d(R_2) = y$,

$$\pi_x(R_1 \bowtie R_2) \quad = \quad R_1 \bowtie \pi_{x \cap y}(R_2).$$

The right-hand side is called *semi-join* in database theory (Maier, 1983).

To give an example of computing in the relational algebra, we consider two relations R_1 and R_2 with domain $d(R_1) = \{continent, country\}$ and $d(R_2) = \{country, city\}$:

$R_1 =$

continent	country
Africa	Kenya
Asia	China
Asia	Japan
Europe	Germany
Europe	France

$R_2 =$

country	city
France	Paris
France	Lyon
Germany	Berlin
Italy	Rome
Kenya	Nairobi

Combining R_1 and R_2 gives

$R_3 = R_1 \bowtie R_2 =$

continent	country	city
Africa	Kenya	Nairobi
Europe	Germany	Berlin
Europe	France	Paris
Europe	France	Lyon

and we obtain for the projection of R_3 to $\{continent\}$:

$\pi_{\{continent\}}(R_3) = R_3^{\downarrow\{continent\}} =$

continent
Africa
Europe

1.3 Arithmetic Potentials - Probability Potentials

Probability potentials are perhaps the most cited example of a valuation algebra, and their common algebraic structure with belief functions was originally the guiding theme for the abstraction process that lead to the valuation algebra framework (Shafer & Shenoy, 1988). However, at this point we yet ignore their usual interpretation as discrete probability mass functions and refer to this formalisms as *arithmetic potentials*. These are simple mappings that associate a non-negative real value with each tuple, i.e. $p : \Omega_s \rightarrow \mathbb{R}_{\geq 0}$. Also, we consider from the outset only variables with finite frames and identify $d(p) = s \subseteq r$ to be the domain of the potential p. The tabular representation can again be used for arithmetic potentials. Combination of two potentials p_1 and p_2 with domain s and t is defined by

$$p_1 \otimes p_2(\mathbf{x}) \quad = \quad p_1(\mathbf{x}^{\downarrow s}) \cdot p_2(\mathbf{x}^{\downarrow t}) \tag{1.10}$$

for x $\in \Omega_{s \cup t}$. Projection consists in summing up all variables to be eliminated. If arithmetic potentials are used to express discrete probability mass functions, then this corresponds to the operation of *marginalization* in probability theory. For a potential p with domain s, $t \subseteq s$ and $\mathbf{x} \in \Omega_t$, we define

$$p^{\downarrow t}(\mathbf{x}) \quad = \quad \sum_{\mathbf{y} \in \Omega_{s-t}} p(\mathbf{x}, \mathbf{y}). \tag{1.11}$$

Arithmetic potentials belong to the same family of instances as indicator functions. Section 5.3 provides a generic proof that all formalisms of this family satisfy the valuation algebra axioms. Moreover, we will also conclude from this result that other sets (such as integers, rational or complex numbers) may be used instead of real numbers to define arithmetic potentials.

We pointed out in the introduction of this book that the distributive law from arithmetics is used to make the computational process of inference more efficient. Also, we said that this property is contained in the valuation algebra axioms. This becomes evident by writing the combination axiom in the notation of arithmetic potentials: For $p_1, p_2 \in \Phi$ with $d(p_1) = s$, $d(p_2) = t$ and $\mathbf{x} \in \Omega_s$

$$\sum_{\mathbf{y} \in \Omega_{t-s}} p_1(\mathbf{x}) \cdot p_2(\mathbf{x}^{\downarrow s \cap t}, \mathbf{y}) \quad = \quad p_1(\mathbf{x}) \cdot \sum_{\mathbf{y} \in \Omega_{t-s}} p_2(\mathbf{x}^{\downarrow s \cap t}, \mathbf{y}).$$

For an example of how to compute with arithmetic potentials, consider a set $r = \{A, B, C\}$ of three variables with finite frames $\Omega_A = \{a, \overline{a}\}$, $\Omega_B = \{b, \overline{b}\}$ and $\Omega_C = \{c, \overline{c}\}$. We define two arithmetic potentials p_1 and p_2 with domain $d(p_1) = \{A, B\}$ and $d(p_2) = \{B, C\}$:

$p_1 \quad =$

A	**B**	
a	b	0.6
a	$\overline{b}$	0.4
$\overline{a}$	b	0.3
$\overline{a}$	$\overline{b}$	0.7

$p_2 \quad =$

B	**C**	
b	c	0.2
b	$\overline{c}$	0.8
$\overline{b}$	c	0.9
$\overline{b}$	$\overline{c}$	0.1

We combine p_1 and p_2 and project the result to $\{A, C\}$:

$p_3 \quad = \quad p_1 \otimes p_2 \quad =$

A	**B**	**C**	
a	b	c	0.12
a	b	$\overline{c}$	0.48
a	$\overline{b}$	c	0.36
a	$\overline{b}$	$\overline{c}$	0.04
$\overline{a}$	b	c	0.06
$\overline{a}$	b	$\overline{c}$	0.24
$\overline{a}$	$\overline{b}$	c	0.63
$\overline{a}$	$\overline{b}$	$\overline{c}$	0.07

$p_3^{\downarrow\{A,C\}} \quad =$

A	**C**	
a	c	0.48
a	$\overline{c}$	0.52
$\overline{a}$	c	0.69
$\overline{a}$	$\overline{c}$	0.31

1.4 Set Potentials - Belief Functions

As we already mentioned, *belief functions* (Shafer, 1976; Kohlas & Monney, 1995) rank among the few instances that originally initiated the abstraction process culminating in the valuation algebra framework (Shafer & Shenoy, 1988). Belief functions are special cases of *set potentials*. Therefore, we first focus on this slightly more general formalism and refer to Instance D.9 and D.6 for a discussion of belief functions and its alternative representations. The theory is again restricted to variables with finite frames. A set potential $m : \mathcal{P}(\Omega_s) \to \mathbb{R}_{\geq 0}$ with domain $d(m) = s \subseteq r$ is a function that assigns non-negative real numbers to all subsets of the variable set frame Ω_s. Combination of two set potentials m_1 and m_2 with domains s and t is defined for all tuple sets $A \subseteq \Omega_{s \cup t}$ by

$$m_1 \otimes m_2(A) \quad = \sum_{B \bowtie C = A, B \subseteq \Omega_s, C \subseteq \Omega_t} m_1(B) \cdot m_2(C). \tag{1.12}$$

This is a simplified version of Dempster's rule of combination (Dempster, 1968; Shafer, 1991). Projection of a set potential m with domain s to $t \subseteq s$ is given for $A \subseteq \Omega_t$ by

$$m^{\downarrow t}(A) \quad = \sum_{\pi_t(B) = A, B \subseteq \Omega_s} m(B). \tag{1.13}$$

Observe that these definitions use the language from the relational algebra to deal with tuple sets. This particular notation will later be helpful. Since relations satisfy the valuation algebra axioms, we may call in its properties to derive important results about set potentials. Note also that the formalism of set potentials essentially differs from the other instances introduced beforehand that map single tuples to some value. Indeed, it belongs to a second family of valuation algebras that will be introduced in Section 5.7 where we also provide a generic proof that all formalisms of this family satisfy the valuation algebra axioms. Before presenting a concrete example, it is important to remark that although all variable frames are finite, a simple enumeration of all assignments as proposed for indicator functions or arithmetic potentials is beyond question since set potentials assign values to all elements of the powerset of Ω_s. Instead, only those entries are listed whose values differ from zero. These tuple sets are usually called *focal sets*.

Consider a set of two variables $r = \{A, B\}$ with finite frames $\Omega_A = \{a, \bar{a}\}$ and $\Omega_B = \{b, \bar{b}\}$. We define two set potentials m_1 and m_2 with domains $d(m_1) = \{A, B\}$ and $d(m_2) = \{A\}$:

$$m_1 = \begin{array}{|c||c|} \hline \{(a,b)\} & 0.7 \\ \{(\bar{a},b),(a,\bar{b})\} & 0.1 \\ \{(a,b),(\bar{a},\bar{b})\} & 0.2 \\ \hline \end{array} \qquad m_2 = \begin{array}{|c||c|} \hline \{(\bar{a})\} & 0.6 \\ \{(a)\} & 0.4 \\ \hline \end{array}$$

The task of computing the combination of m_1 and m_2 is simplified by constructing the following table as an intermediate step. The first column contains m_1 and the top row m_2, both extended to the union domain $d(m_1) \cup d(m_2) = \{A, B\}$. Then, every internal cell contains the intersection between the two corresponding tuple sets and the product of the corresponding values. This corresponds to the natural join in equation (1.12).

	$\{(\bar{a},\bar{b}),(\bar{a},b)\}, 0.6$	$\{(a,\bar{b}),(a,b)\}, 0.4$
$\{(a,b)\}, 0.7$	$\emptyset, 0.42$	$\{(a,b)\}, 0.28$
$\{(\bar{a},b),(a,\bar{b})\}, 0.1$	$\{(\bar{a},b)\}, 0.06$	$\{(a,\bar{b})\}, 0.04$
$\{(a,b),(\bar{a},\bar{b})\}, 0.2$	$\{(\bar{a},\bar{b})\}, 0.12$	$\{(a,b)\}, 0.08$

To complete the combination, it is then sufficient to add the values of all internal cells with equal tuple set. The result of the combination is projected afterwards to $\{A\}$ using equation (1.13)

$$m_3 \quad = \quad m_1 \otimes m_2 \quad = \quad \begin{array}{|c||c|} \hline \emptyset & 0.42 \\ \{(a,b)\} & 0.36 \\ \{(a,\bar{b})\} & 0.04 \\ \{(\bar{a},b)\} & 0.06 \\ \{(\bar{a},\bar{b})\} & 0.12 \\ \hline \end{array} \qquad m_3^{\downarrow\{A\}} \quad = \quad \begin{array}{|c||c|} \hline \emptyset & 0.42 \\ \{(\bar{a})\} & 0.18 \\ \{(a)\} & 0.40 \\ \hline \end{array}$$

1.5 Density Functions

All instances discussed so far are based on variables with finite (or at least countable) frames. Although such a setting is typical for a large number of formalisms used in computer science, it is in no way mandatory. Thus, we next introduce the valuation algebra of *density functions* which are defined over variables taking values from the set of real numbers. For a set of variables $s \subseteq r$, a density function $f : \mathbb{R}^s \rightarrow \mathbb{R}_{\geq 0}$ with domain s is a continuous, non-negative valued function with finite Riemann integral,

$$\int_{-\infty}^{\infty} f(\mathbf{x})d\mathbf{x} \quad < \quad \infty. \tag{1.14}$$

Note that $\mathbf{x}$ denotes a vector in the linear space $\Omega_s = \mathbb{R}^s$. The combination $f \otimes g$ of two density functions f and g with domain s and t is defined for $\mathbf{x} \in \mathbb{R}^{s \cup t}$ by simple multiplication

$$f \otimes g(\mathbf{x}) \quad = \quad f(\mathbf{x}^{\downarrow s}) \cdot g(\mathbf{x}^{\downarrow t}). \tag{1.15}$$

The projection $f^{\downarrow t}$ of a density f with $d(f) = s$ to $t \subseteq s$ is given by the integral

$$f^{\downarrow t}(\mathbf{x}) \quad = \quad \int_{-\infty}^{\infty} f(\mathbf{x}, \mathbf{y})d\mathbf{y} \tag{1.16}$$

where $\mathbf{x} \in \mathbb{R}^t$ and $\mathbf{y} \in \mathbb{R}^{s-t}$. Density functions together with the above definitions of combination and projection form a valuation algebra. The proof of this statement even for the more general case of Lebesque measurable density functions can be found in (Kohlas, 2003). The perhaps most interesting example of a density function in the context of valuation algebras is the *Gaussian density*. In fact, Instance 1.6 below shows that Gaussian densities are closed under combination and projection and therefore establish a subalgebra of the valuation algebra of density functions.

Labeled Matrices

These first examples were all based on tuples and tuple sets. Next, we are going to introduce two further valuation algebra instances based on *labeled matrices*. For finite sets $s, t \subseteq r$ a labeled, real-valued matrix is a mapping $\mathbf{M} : s \times t \rightarrow \mathbb{R}$ such that for $X \in s$ and $Y \in t$ we have $\mathbf{M}(X,Y) \in \mathbb{R}$. We write $\mathcal{M}(\mathbb{R}, s \times t)$ for the set of matrices of this kind. The sum of $\mathbf{M}_1, \mathbf{M}_2 \in \mathcal{M}(\mathbb{R}, s \times t)$ is defined by

$$(\mathbf{M}_1 + \mathbf{M}_2)(X,Y) \quad = \quad \mathbf{M}_1(X,Y) + \mathbf{M}_2(X,Y) \tag{1.17}$$

for $X \in s$ and $Y \in t$. If $\mathbf{M}_1 \in \mathcal{M}(\mathbb{R}, s \times t)$ and $\mathbf{M}_2 \in \mathcal{M}(\mathbb{R}, t \times u)$ for $s, t, u \subseteq r$, $X \in s$ and $Y \in u$ we define the product $\mathbf{M}_1 \cdot \mathbf{M}_2$ by

$$(\mathbf{M}_1 \cdot \mathbf{M}_2)(X,Y) \quad = \quad \sum_{Z \in t} \mathbf{M}_1(X,Z) \cdot \mathbf{M}_2(Z,Y). \tag{1.18}$$

A special element of the set $\mathcal{M}(\mathbb{R}, s \times t)$ is the *zero matrix* defined for $X \in s$ and $Y \in t$ by $\mathbf{O}(X,Y) = 0$. The *projection* of a labeled matrix $\mathbf{M} \in \mathcal{M}(\mathbb{R}, s \times t)$ to $u \subseteq s$ and $v \subseteq t$ is defined by $\mathbf{M}^{\downarrow u,v}(X,Y) = \mathbf{M}(X,Y)$ if $X \in u$ and $Y \in v$, which simply corresponds to dropping the unrequested rows and columns. Similarly to the extension of vectors, we *extend* a labeled matrices $\mathbf{M} \in \mathcal{M}(\mathbb{R}, s \times t)$ to $u \supseteq s$ and $v \supseteq t$ with $X \in u$ and $Y \in v$ by

$$\mathbf{M}^{\uparrow u,v}(X,Y) \quad = \quad \begin{cases} \mathbf{M}(X,Y) & \text{if } X \in s \text{ and } Y \in t, \\ 0, & \text{otherwise.} \end{cases} \tag{1.19}$$

Of special interest are often square matrices $\mathbf{M} \in \mathcal{M}(\mathbb{R}, s \times s)$. We then say that the *domain* of $\mathbf{M}$ is $d(\mathbf{M}) = s$ and use the abbreviation $\mathcal{M}(\mathbb{R}, s)$ for the set of all matrices with domain s. Also, the projection of a matrix $\mathbf{M}$ with domain s to $t \subseteq s$ is abbreviated by $\mathbf{M}^{\downarrow s}$ and accordingly, we write $\mathbf{M}^{\uparrow u}$ for its extension to the larger domain $u \supseteq s$. Finally, we define the *identity matrix* for the domain s by $\mathbf{I}(X,Y) = 1$ if $X = Y$ and $\mathbf{I}(X,Y) = 0$ otherwise. In later chapters, we will also consider labeled matrices with values from other algebraic structures than real numbers. But for the introduction of the following instances, the limitation to real numbers is sufficient.

■ 1.6 Gaussian Densities - Gaussian Potentials

An important family of density functions is provided by *Gaussian distributions*. For a set of real variables $s \subseteq r$ and $\mathbf{x} \in \mathbb{R}^s$ a *Gaussian density* is defined by

$$f(\mathbf{x}) \quad = \quad \sqrt{\frac{|\det(\mathbf{K})|}{(2\pi)^{|s|}}}\, e^{-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{K}(\mathbf{x}-\mu)}, \tag{1.20}$$

where $\mu \in \mathbb{R}^s$ denotes the *mean vector* and $\mathbf{K} : s \times s \to \mathbb{R}$ the symmetric and positive definite *concentration matrix*. The matrix $\mathbf{K}^{-1}$ is the *variance-covariance matrix* of the Gaussian density. Looking at this definition, we remark that a Gaussian density is completely determined by its mean vector and concentration matrix. Such pairs $(\mu, \mathbf{K})$ with $\mu \in \mathbb{R}^s$ and $\mathbf{K} \in \mathcal{M}(\mathbb{R}, s)$ being symmetric and positive definite are called *Gaussian potentials*. The square root in front of the exponential function in equation (1.20) is merely a *normalization factor* guaranteeing that the integral over the density function equals one.

Let f and g be two Gaussian densities with domain $d(f) = s$ and $d(g) = t$, represented by the Gaussian potentials $(\mu_1, \mathbf{K}_1)$ and $(\mu_2, \mathbf{K}_2)$. When combining these two densities by equation (1.15) we can neglect the normalization constants and simply multiply the exponential functions. This in turn means adding their exponents. Doing so, we may also neglect additive terms without variables, since these terms simply go into the normalization factor. In the following, equality must always be taken up to such terms. We then get for the exponent of the combined density function and $\mathbf{x} \in \mathbb{R}^{s\cup t}$

$$\begin{aligned}
(\mathbf{x}^{\downarrow s} - \mu_1)^T\mathbf{K}_1(\mathbf{x}^{\downarrow s} - \mu_1) + (\mathbf{x}^{\downarrow t} - \mu_2)^T\mathbf{K}_2(\mathbf{x}^{\downarrow t} - \mu_2) &= \\
(\mathbf{x} - \mu_1^{\uparrow s\cup t})^T\mathbf{K}_1^{\uparrow s\cup t}(\mathbf{x} - \mu_1^{\uparrow s\cup t}) + (\mathbf{x} - \mu_2^{\uparrow s\cup t})^T\mathbf{K}_2^{\uparrow s\cup t}(\mathbf{x} - \mu_2^{\uparrow s\cup t}) &= \\
\mathbf{x}^T\mathbf{K}_1^{\uparrow s\cup t}\mathbf{x} - \mathbf{x}^T\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} - \mu_1^{T\uparrow s\cup t}\mathbf{K}_1^{\uparrow s\cup t}\mathbf{x} &+ \\
\mathbf{x}^T\mathbf{K}_2^{\uparrow s\cup t}\mathbf{x} - \mathbf{x}^T\mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t} - \mu_2^{T\uparrow s\cup t}\mathbf{K}_2^{\uparrow s\cup t}\mathbf{x} &= \\
\mathbf{x}^T(\mathbf{K}_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t})\mathbf{x} - \mathbf{x}^T(\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t}) &- \\
(\mu_1^{T\uparrow s\cup t}\mathbf{K}_1^{\uparrow s\cup t} + \mu_2^{T\uparrow s\cup t}\mathbf{K}_2^{\uparrow s\cup t})\mathbf{x}. &
\end{aligned}$$

Define now

$$\mathbf{K} \quad = \quad \mathbf{K}_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}.$$

Note that $\mathbf{K}$ is still symmetric and positive definite. Then, we rewrite the last expression using $\mathbf{K}$, add a constant, additive term to the last expression completing it to a quadratic form:

$$\begin{aligned}\mathbf{x}^T\mathbf{K}\mathbf{x} - \mathbf{x}^T(\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t}) &\quad - \\ (\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t})^T\mathbf{x} &\quad + \\ (\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t})^T\mathbf{K}^{-1}(\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t}) &\quad = \\ (\mathbf{x} - \mathbf{K}^{-1}(\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t}))^T\mathbf{K} & \\ (\mathbf{x} - \mathbf{K}^{-1}(\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t})) &\end{aligned}$$

This finally means that the multiplication of two Gaussian densities results again in a Gaussian density with concentration matrix

$$\mathbf{K} \quad = \quad \mathbf{K}_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}. \tag{1.21}$$

and mean vector

$$\mu \quad = \quad \mathbf{K}^{-1}\Big(\mathbf{K}_1^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t}\Big). \tag{1.22}$$

Motivated by this result, we may define the combination of two Gaussian potentials $(\mu_1, \mathbf{K}_1)$ and $(\mu_2, \mathbf{K}_2)$ with $d(\mu_1, \mathbf{K}_1) = s$ and $d(\mu_2, \mathbf{K}_2) = t$ by

$$(\mu_1, \mathbf{K}_1) \otimes (\mu_2, \mathbf{K}_2) \quad = \quad (\mu, \mathbf{K}),$$

where μ is defined by (1.22) and $\mathbf{K}$ by (1.21). We found this combination rule by a purely formal derivation. Its reason and meaning will be given in Section 10.1. The projection of a Gaussian density f with domain $d(f) = s$, mean vector μ and concentration matrix $\mathbf{K}$ to a subset $t \subseteq s$ is again a Gaussian density, as it will be shown in Appendix J.2. Its mean vector is simply $\mu^{\downarrow t}$ and $((\mathbf{K}^{-1})^{\downarrow t})^{-1}$ is its concentration matrix. Consequently, Gaussian densities are closed under combination and projection and form an important subalgebra of the valuation algebra of density functions. The operations of combination and projection can both be expressed in terms of the Gaussian potentials associated with the densities. Gaussian densities and Gaussian potentials will be studied in detail in Chapter 10, which will also clarify the interest in this valuation algebra.

Many further valuation algebra instances will be introduced throughout the subsequent chapters. At this point we interrupt the study of formalisms for the time being and focus on the computational interest in valuation algebras. The following chapter phrases an abstract computational task called inference problem using the language of valuation algebras and also gives a first impression of the different semantics of this problem under different valuation algebra instances. But first, we propose a more far-reaching discussion of the valuation algebra axiomatics in the appendix of this chapter. The valuation algebra framework presented in Section 1.1 is more general than most of its antecessors. Nevertheless, there is still potential for further generalizations. Two such systems will be presented in the following appendix: the first introduces valuation algebras that are no more necessarily based on variable systems. Instead, valuations take their domains from a more general lattice structure. The second framework focuses on a generalization of the projection operator. Here, it is no longer the case that projections on arbitrary subsets are possible, but there is a fourth operator that tells us to which domain a valuation is allowed to be projected. For the understanding of the subsequent chapters, these systems are of minor importance. But it is nevertheless interesting to see that further instances can be covered by a more general and abstract definition of the valuation algebra framework, which still provides enough structure for the application of local computation.

1.3 CONCLUSION

This first chapter introduced the valuation algebra framework upon which all following chapters are based. Valuations can be imagined as pieces of information that refer to a question called the domain of the valuation. This domain is returned by the labeling operation. Two further operations called combination and projection are used to manipulate valuations. Combination corresponds to aggregation, and projection to focussing or extraction of knowledge. In addition, the valuation algebra framework consists of a set of six axioms that determine the behaviour of the three operations. Formalisms that satisfy the structure of a valuation algebra are called instances and occur numerously in very different fields of mathematics and computer science. This chapter gave a first selection of instances including crisp çonstraints, arithmetic and probability potentials, Dempster-Shafer belief functions, density functions and the important family of Gaussian distributions.

Appendix: Generalizations of the Valuation Algebra Framework

We first give a formal definition of lattices with their main variations and refer to (Davey & Priestley, 1990) for an extensive discussion of these an related concepts.

A.1 ORDERED SETS AND LATTICES

Definition A.2 *A* preorder *is a binary relation $\leq$ over a set P which is reflexive and transitive. We have for $a, b, c \in P$:*

- *$a \leq a$ (reflexivity);*
- *$a \leq b$ and $b \leq c$ implies that $a \leq c$ (transitivity).*

A preorder is called partial order *if it is antisymmetric, i.e. if*

- *$a \leq b$ and $b \leq a$ implies that $a = b$ (antisymmetry).*

A set with a partial order is also called a *partially ordered set* or simply an *ordered set.* Chapter 5 starts with an introduction to semiring theory that naturally offers a large number of examples for preorders and partial orders. We therefore refer to the numerous examples of semirings in Chapter 5 for concrete sets providing preorders and partial orders.

Definition A.3 *Let P be an ordered set.*

- *P has a* bottom element *if there exists an element $\perp \in P$ such that $\perp \leq x$ for all $x \in P$.*
- *P has a* top element *if there exists an element $\top \in P$ such that $x \leq \top$ for all $x \in P$.*

Definition A.4 *Let P be a ordered set and $S \subseteq P$. An element $u \in P$ is called* supremum, least upper bound *or* join *of S if*

- *$x \leq u$ for all $x \in S$;*
- *for any $v \in P$ such that $x \leq v$ for all $x \in S$ it holds that $u \leq v$.*

Likewise, an element $u \in P$ is called infimum, greatest lower bound *or* meet *of S if*

- *$u \leq x$ for all $x \in S$;*
- *for any $v \in P$ such that $v \leq x$ for all $x \in S$ it holds that $v \leq u$.*

If join or meet of a subset $S \subseteq P$ exist, then they are always unique, and we write $\bigvee S$ for join and $\bigwedge S$ for meet. Moreover, if $S = \{x, y\}$ consists of two elements, we generally write $x \vee y$ or $\sup\{x, y\}$ for join and $x \wedge y$ or $\inf\{x, y\}$ for meet.

Definition A.5 *Let P be a non-empty, ordered set.*

- *If $x \vee y$ and $x \wedge y$ exist for all $x, y \in P$, then P is called a* lattice.
- *If $\bigvee S$ and $\bigwedge S$ exist for all subsets $S \subseteq P$, then P is called a* complete lattice.

Lattices may satisfy further identities:

Definition A.6 *Let L be a lattice.*

- *L is said to be* bounded, *if it has a bottom and a top element.*
- *L is said to be* distributive, *if for all $a, b, c \in L$*

$$\begin{aligned} a \wedge (b \vee c) &= (a \wedge b) \vee (a \wedge c); \\ a \vee (b \wedge c) &= (a \vee b) \wedge (a \vee c). \end{aligned}$$

This definition lists both statements of distributivity, although they are in fact equivalent. The proof is given in (Davey & Priestley, 1990). Note also that every complete lattice is bounded.

Example A.2 (Powerset or Subset Lattice) *For any set A we consider the set of all subsets called its* powerset *or* subset lattice $\mathcal{P}(A)$. *This set is partially ordered via set inclusion and forms a lattice with set intersection as meet $a \wedge b = a \cap b$ and set union as join $a \vee b = a \cup b$. The powerset lattice is distributive, complete and bounded by A itself and the empty set $\emptyset$.*

Example A.3 (Division Lattice) *The set of* $\mathbb{N} \cup \{0\}$ *also forms a distributive lattice with the least common multiple as join* $\text{lcm}(a, b) = a \vee b$ *and the greatest common divisor as meet* $\text{gcd}(a, b) = a \wedge b$. *The order is defined by divisibility, i.e. $a \leq b$ if a divides b. This lattice is also bounded with* 1 *as bottom and* 0 *as top element.*

An important source of non-distributive lattices are partitions:

A.1.1 Partitions and Partition Lattices

Partition lattices take a universal position among all lattices. (Grätzer, 1978) shows that every lattice is isomorph to some partition lattice.

Definition A.7 *A* partition $\pi = \{B_i : 1 \leq i \leq n\}$ *of a set U called* universe *consists of a collection of subsets $B_i \subseteq U$ called* blocks *such that*

- $B_i \neq \emptyset$;
- $B_i \cap B_j = \emptyset$ *for* $i \neq j$ *and* $1 \leq i.j \leq n$;
- $\bigcup_{i=1}^{n} B_i = U$.

Let $Part(U)$ denote the set of all possible partitions of a universe U. It is then possible to introduce a partial order between its elements. For $\pi_1, \pi_2 \in Part(U)$ we write $\pi_1 \leq \pi_2$ if every block of π_1 is contained in some block of π_2. This is the case if, and only if, every block of π_2 is a union of blocks from π_1. We then also say that the partition π_1 is *finer* than π_2, or conversely that π_2 is *coarser* than π_1. The blocks of the infimum or meet of an arbitrary collection of partitions $P \subseteq Part(U)$ corresponds to the non-empty intersections of all blocks contained in the partitions of P. The definition of the supremum or join especially for the case of universes with an infinite number of elements is more involved. We refer to (Grätzer, 1978) for a discussion of this aspect. Here, we directly conclude that the set $Part(U)$ of all partitions of the universe U forms a complete lattice. The partition $\{\{u\} : u \in U\}$ consisting of all one-element subsets of U is the lower bound, and the partition $\{U\}$ is the upper bound. A lattice whose elements are partitions of a universe U is called *partition lattice*. They are sublattices of $Part(U)$ and generally not distributive. The above refinement relation is the natural way of introducing an order relation between partitions. However, in the context of algebraic information theory (Kohlas, 2003) it is often useful to consider the inverse of the natural order, i.e. $\pi_2 \leq_c \pi_1$ if, and only if, $\pi_1 \leq \pi_2$ or equivalently, if π_2 is coarser than π_1. The motivation is that a finer partition expresses more information than a coarser partition.

Example A.4 (Interval Partitions) *Let $U = [a, b)$ be a semi-closed interval of real numbers. A sequence $x_0 < x_1 < \ldots < x_n$ with $x_0 = a$ and $x_n = b$ establishes a partition $\pi = \{[x_i, x_{i+1}) : 0 \leq i \leq n-1\}$. If an interval partition π_1 contains all elements of the sequence defining the partition π_2, then every bloc of π_1 is contained in some bloc of π_2. In other words, π_1 is a refinement of π_2. The meet of two interval partitions is obtained from the union sequence, and the join from the intersection. If for example $U = [0, 1)$ the partition for* $0 < 0.2 < 0.4 < 0.6 < 0.8 < 1$ *is a refinement of the partition for* $0 < 0.4 < 0.8 < 1$. *If furthermore π_1 is given by the sequence* $0 < 0.5 < 0.8 < 1$ *and π_2 by* $0 < 0.2 < 0.8 < 1$, *their meet is obtained from the sequence* $0 < 0.2 < 0.5 < 0.8 < 1$ *and their join from* $0 < 0.8 < 1$.

A.2 VALUATION ALGEBRAS ON GENERAL LATTICES

The definition of a valuation algebra given at the beginning of this chapter is based on a particular lattice, namely on the powerset lattice of variables (see Example A.2). This lattice is distributive. However, it turns out that the valuation algebra framework can even be generalized to arbitrary lattices (Shafer, 1991; Kohlas, 2003). Thus, let D be a lattice with a partial order $\leq$ and the two operations meet $\wedge$ and join $\vee$. We denote by Φ the set of valuations with domains in D and suppose the following three operations defined on Φ and D.

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

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

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

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

$$d(\phi \otimes \psi) \quad = \quad d(\phi) \vee d(\psi). \tag{A.1}$$

(A3′) *Projection:* For $\phi \in \Phi$, $x \in D$ and $x \leq d(\phi)$,

$$d(\phi^{\downarrow x}) \quad = \quad x. \tag{A.2}$$

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

$$(\phi^{\downarrow y})^{\downarrow x} \quad = \quad \phi^{\downarrow x}. \tag{A.3}$$

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

$$(\phi \otimes \psi)^{\downarrow z} \quad = \quad \phi \otimes \psi^{\downarrow z \wedge y}. \tag{A.4}$$

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

$$\phi^{\downarrow x} \quad = \quad \phi. \tag{A.5}$$

Clearly, this definition of a valuation algebra is more general than the framework introduced beforehand. It is indeed surprising that no further properties of the lattice are required to enable the application of local computation (Kohlas & Monney, 1995). We must, however, accept that not all properties of the more restricted concept of a valuation algebra are maintained. Although the subsequent chapters of this book are entirely based on variable systems, we nevertheless present an example from diagnostics that is based on a valuation algebra with domains from a more general lattice, i.e. from a lattice of partitions (Kohlas & Monney, 1995; Shafer *et al.*, 1987).

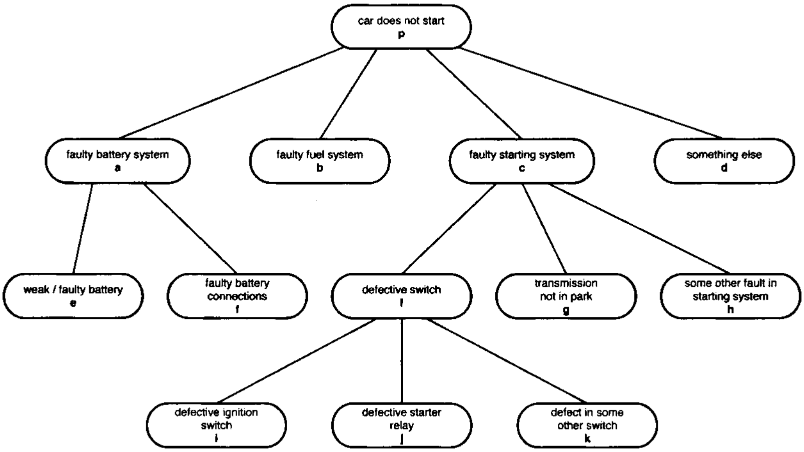


Figure A.1 A tree of diagnoses for stubborn cars.

Example A.5 *Suppose that your car does not start anymore and you want to determine the cause of its failure. The* tree of diagnoses *shown in Figure A.1 provides a structural approach for the identification of the cause of failure. At the beginning, the list of possible diagnoses is fairly coarse and only contains the failure possibilities* $\Omega_1 = \{a, b, c, d\}$. *Note, however, that these options are mutually exclusive and collectively exhaustive. More precise statements are obtained by partitioning* *coarse diagnoses. Doing so, we replace diagnosis a by $\{e, f\}$ and diagnosis c by $\{l, g, h\}$ which leads to the new set of failure possibilities $\Omega_2 = \{e, f, b, l, g, h, d\}$. After a third specialization step we finally obtain $\Omega_3 = \{e, f, b, i, j, k, g, h, d\}$. These step-wise refined sets of diagnoses are called* frames of discernment. *When a frame Ω_i is replaced by a finer frame Ω_{i+1} the substitution of a diagnosis by a set of more precise diagnoses is described by a mapping*

$$\tau \;:\; \Omega_i \;\;\rightarrow\;\; \mathcal{P}(\Omega_{i+1}) \tag{A.6}$$

called refinement mapping *or simply* refinement *(Shafer, 1976). Refinement mappings must satisfy the following requirements:*

1. $\tau(\theta) \neq \emptyset$ *for all* $\theta \in \Omega_i$*;*
2. $\tau(\theta_1) \cap \tau(\theta_2) = \emptyset$ *whenever* $\theta_1 \neq \theta_2$*;*
3. $\cup\{\tau(\theta) : \theta \in \Omega_i\} = \Omega_{i+1}$.

For example, the passage from Ω_1 to Ω_2 is expressed by the mapping

$$\tau(x) \;=\; \begin{cases} \{x\} & \text{if } x \in \{b, d\}, \\ \{e, f\} & \text{if } x = a, \\ \{l, g, h\} & \text{if } x = c. \end{cases}$$

From an alternative but equivalent point of view, this family of related frames can be seen as a collection of partitions of the universe Ω_3. These partitions are:

- $\pi_0 = \{\{e, f, b, i, j, k, g, h, d\}\}$;
- $\pi_1 = \{\{e, f\}, \{b\}, \{i, j, k, g, h\}, \{d\}\}$;
- $\pi_2 = \{\{e\}, \{f\}, \{b\}, \{i, j, k\}, \{g\}, \{h\}, \{d\}\}$;
- $\pi_3 = \{\{e\}, \{f\}, \{b\}, \{i\}, \{j\}, \{k\}, \{g\}, \{h\}, \{d\}\}$.

Note that $\pi_{i+1} \leq \pi_i$, i.e. every block of π_{i+1} is contained in some block of π_i. Because each frame Ω_i corresponds to a partition π_i, we may replace the frames in equation (A.6) by their corresponding partitions. The mapping

$$\delta \;:\; \pi_i \;\;\rightarrow\;\; \mathcal{P}(\pi_{i+1})$$

assigns to each block $B \in \pi_i$ the set of blocks in π_{i+1} whose union is B. The passage from π_1 to π_2 is thus expressed by the mapping

$$\delta(x) \;=\; \begin{cases} \{\{x\}\} & \text{if } x = \{b, d\}, \\ \{\{e\}, \{f\}\} & \text{if } x = \{e, f\}, \\ \{\{i, j, k\}, \{g\}, \{h\}\} & \text{if } x = \{i, j, k, g, h\}. \end{cases}$$

This mapping is referred to as decomposition mapping.

Although many valuation algebras with domains from variable lattices could be adapted to partition and other lattices, the modified valuation algebra of set potentials from Dempster-Shafer theory is probably the most important example for practical applications. In fact, the derivation of partitions from a tree of diagnoses takes center stage in the applications of medical diagnostics in (Gordon & Shortliffe, 1985). Other authors studied Dempster-Shafer theory on unspecified lattices (Grabisch, 2009; Shafer, 1991). Here, we follow the algebraic approach from (Kohlas & Monney, 1995) and first extract the necessary requirements from the above example that later enables the definition of set potentials with domains from partition lattices.

Definition A.8 *A collection of frames $\mathcal{F}$ together with a collection of refinements $\mathcal{R}$ forms a* family of compatible frames *if the following conditions hold:*

1. *For each pair $\Omega_1, \Omega_2 \in \mathcal{F}$ of frames, there is at most one refinement $\tau : \Omega_1 \rightarrow \mathcal{P}(\Omega_2)$ in $\mathcal{R}$.*

2. *There exists a set U and a mapping $\varphi : \mathcal{F} \rightarrow Part(U)$ such that for all frames $\Omega_1, \Omega_2 \in \mathcal{F}$ we have $\varphi(\Omega_1) \neq \varphi(\Omega_2)$ whenever $\Omega_1 \neq \Omega_2$, and such that $\varphi(\mathcal{F}) = \{\varphi(\Omega) : \Omega \in \mathcal{F}\}$ forms a bounded sublattice of $Part(U)$.*

3. *For each frame $\Omega \in \mathcal{F}$ there is a bijective mapping $b : \Omega \rightarrow \varphi(\Omega)$. This mapping identifies for each frame element the corresponding block in the partition that contains this element.*

4. *There is a refinement $\tau : \Omega_1 \rightarrow \mathcal{P}(\Omega_2)$ in $\mathcal{R}$ exactly if $\varphi(\Omega_2) \leq \varphi(\Omega_1)$.*

Condition 2 means that every frame corresponds to a partition and that the corresponding mapping is an embedding of $\mathcal{F}$ into $Part(U)$. Since the least partition of U is in $\varphi(\mathcal{F})$, there is exactly one frame Ω whose partition $\varphi(\Omega)$ corresponds to the least partition. By the mapping of Condition 3, the elements of the frame Ω are in one-to-one correspondence with the blocks of the least partition that are the singletons of the universe U. From a mathematical point of view, the two sets U and Ω are thus isomorphic. Finally, as a consequence of Condition 4, we can introduce a decomposition mapping for each refinement. Indeed, if $\tau : \Omega_1 \rightarrow \mathcal{P}(\Omega_2)$, we have $\varphi(\Omega_2) \leq \varphi(\Omega_1)$ which means that every block in $\varphi(\Omega_2)$ is contained in some block of $\varphi(\Omega_1)$. We may therefore assume a decomposition mapping

$$\delta : \varphi(\Omega_1) \rightarrow \mathcal{P}(\varphi(\Omega_2)) \tag{A.7}$$

that assigns to each block $B \in \varphi(\Omega_1)$ the set of blocks in $\varphi(\Omega_2)$ whose union is B. This mapping can further be extended to sets of blocks $\delta : \mathcal{P}(\pi_i) \rightarrow \mathcal{P}(\pi_{i+1})$ by

$$\delta(A) = \bigcup\{\delta(B) : B \in A\}. \tag{A.8}$$

for $A \subseteq \pi_i$. Such families of compatible frames provide sufficient structure for the definition of set potentials over partition lattices (Kohlas & Monney, 1995).

■ A.7 Set Potentials on Partition Lattices

Let $\langle \mathcal{F}, \mathcal{R} \rangle$ be a family of compatible frames with universe U and partition lattice $Part(U)$. A set potential $m : \mathcal{P}(\pi) \rightarrow \mathbb{R}_{\geq 0}$ with domain $d(m) = \pi \in \varphi(\mathcal{F}) \subseteq Part(U)$ is defined by a mapping that assigns non-negative real numbers to all subsets of the partition π. Let us again consider the inverse natural order between partitions. According to the labeling axiom, the domain of the combination of m_1 and m_2 with domains π_1 and π_2 must be $\pi_1 \vee \pi_2$, which corresponds to the coarsest partition that is finer than π_1 and π_2. We then conclude from Condition 2 that frames Ω_1, Ω_2 and Ω exist such that $\pi_1 = \varphi(\Omega_1)$, $\pi_2 = \varphi(\Omega_2)$ and $\pi_1 \vee \pi_2 = \varphi(\Omega)$. Since $\pi_1, \pi_2 \leq \pi_1 \vee \pi_2$ in the natural order, it follows from Condition 4 that the two refinement mappings $\tau_1 : \Omega \rightarrow \mathcal{P}(\Omega_1)$ and $\tau_2 : \Omega \rightarrow \mathcal{P}(\Omega_2)$ exist in $\mathcal{R}$. We thus obtain the corresponding decomposition functions $\delta_1 : \mathcal{P}(\pi_1 \vee \pi_2) \rightarrow \mathcal{P}(\pi_1)$ and $\delta_2 : \mathcal{P}(\pi_1 \vee \pi_2) \rightarrow \mathcal{P}(\pi_2)$. Altogether, this allows us to define the combination rule: For $A \subseteq \pi_1 \vee \pi_2$ we define

$$m_1 \otimes m_2(A) \quad = \quad \sum_{A = B \cap C} m_1(\delta_1(B)) \cdot m_2(\delta_2(C)) \tag{A.9}$$

For the projection operator, we assume a set potential m with domain π_1 and $\pi_2 \leq \pi_1$. By the same justification we find the decomposition mapping $\delta : \mathcal{P}(\pi_1) \rightarrow \mathcal{P}(\pi_2)$ and define for $A \subseteq \pi_2$ the projection rule as follows:

$$m^{\downarrow \pi_2}(A) \quad = \quad \sum_{B \subseteq \pi_1 : A \cap \delta(B) \neq \emptyset} m(B). \tag{A.10}$$

The proof that set potentials over partition lattices satisfy the axioms of a valuation algebra on general lattices can be found in (Kohlas & Monney, 1995).

A.3 VALUATION ALGEBRAS WITH PARTIAL PROJECTION

The valuation algebra definition given at the beginning of this chapter allows every valuation $\phi \in \Phi$ to be projected to any subset of $d(\phi)$. Hence, we may say that ϕ can be projected to all domains in the *marginal set* $\mathcal{M}(\phi) = \mathcal{P}(d(\phi))$. Valuation algebras with partial projection are more general in the sense that not all projections are necessarily defined. In this view, $\mathcal{M}(\phi)$ may be a strict subset of $\mathcal{P}(d(\phi))$. It is therefore sensible that a fourth operation is needed which produces $\mathcal{M}(d(\phi))$ for all $\phi \in \Phi$. Additionally, all axioms that bear somehow on projection must be generalized to take the corresponding marginal sets into account. Thus, let Φ be a set of valuations over domains $s \subseteq r$ and $D = \mathcal{P}(r)$. We assume the following operations in $\langle \Phi, D \rangle$:

1. *Labeling:* $\Phi \rightarrow D$; $\phi \mapsto d(\phi)$,
2. *Combination:* $\Phi \times \Phi \rightarrow \Phi$; $(\phi, \psi) \mapsto \phi \otimes \psi$,
3. *Domain:* $\Phi \rightarrow \mathcal{P}(D)$; $\phi \mapsto \mathcal{M}(\phi)$,

4. *Partial Projection:* $\Phi \times D \rightarrow \Phi$; $(\phi, x) \mapsto \phi^{\downarrow x}$ defined for $x \in \mathcal{M}(\phi)$.

The set $\mathcal{M}(\phi)$ contains therefore all domains $x \in D$ such that the marginal of ϕ relative to x is defined. We impose now the following set of axioms on Φ and D, pointing out that the two Axioms (A1″) and (A2″) remain identical to the traditional definition of a valuation algebra.

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

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

$$d(\phi \otimes \psi) = d(\phi) \cup d(\psi). \tag{A.11}$$

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

$$d(\phi^{\downarrow x}) = x. \tag{A.12}$$

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

$$x \in \mathcal{M}(\phi) \Rightarrow x \in \mathcal{M}(\phi^{\downarrow y}), y \in \mathcal{M}(\phi) \text{ and } (\phi^{\downarrow y})^{\downarrow x} = \phi^{\downarrow x}. \tag{A.13}$$

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

$$z \cap y \in \mathcal{M}(\psi) \Rightarrow z \in \mathcal{M}(\phi \otimes \psi) \text{ and } (\phi \otimes \psi)^{\downarrow z} = \phi \otimes \psi^{\downarrow z \cap y}. \tag{A.14}$$

(A6″) *Domain:* For $\phi \in \Phi$ with $d(\phi) = x$, we have $x \in \mathcal{M}(\phi)$ and

$$\phi^{\downarrow x} = \phi. \tag{A.15}$$

Definition A.9 *A system $\langle \Phi, D \rangle$ together with the operations of labeling, combination, partial projection and domain satisfying these axioms is called a valuation algebra with partial projection.*

It is easy to see that this system is indeed a generalization of the traditional valuation algebra, because if $\mathcal{M}(\phi) = \mathcal{P}(d(\phi))$ holds for all $\phi \in \Phi$, the axioms reduce to the system given at the beginning of this chapter. Therefore, the latter is also called *valuation algebra with full projection.*

■ A.8 Quotients of Density Functions

Let us reconsider the valuation algebra of density functions introduced in Instance 1.5 and assume that f is a positive density function over variables in $s \subseteq r$, i.e. for $\mathbf{x} \in \mathbb{R}^s$ we always have $f(\mathbf{x}) > 0$. We consider the quotient

$$g(\mathbf{x}) \quad = \quad \frac{f(\mathbf{x})}{f^{\downarrow t}(\mathbf{x}^{\downarrow t})}$$

with $t \subseteq s$. For any $t \subseteq s$ these quotients represent the family of *conditional distributions* of $\mathbf{x}^{\downarrow s-t}$ given $\mathbf{x}^{\downarrow t}$ associated with the density f. However, projection is only partially defined for such quotients since

$$g^{\downarrow t}(\mathbf{x}^{\downarrow t}) = \left(\frac{f(\mathbf{x})}{f^{\downarrow t}(\mathbf{x}^{\downarrow t})}\right)^{\downarrow t} = \frac{f^{\downarrow t}(\mathbf{x}^{\downarrow t})}{f^{\downarrow t}(\mathbf{x}^{\downarrow t})} = 1.$$

This is a constant function with an infinite integral and therefore not a density anymore. Nevertheless, conditional density functions are part of a valuation algebra with partial projection. A formal verification of the corresponding axiomatic system can be found in (Kohlas, 2003).

PROBLEM SETS AND EXERCISES

A.1 ★ Verify the valuation algebra axioms for the relational algebra of Instance 1.2 without restriction to variables with finite frames.

A.2 ★ Reconsider the valuation algebra of arithmetic potentials from Instance 1.3. This time, however, we restrict ourselves to the unit interval and replace the operation of addition in the definition of projection in equation (1.11) by maximization. This leads to the formalism of *possibility potentials* or *probabilistic constraints* that will later be discussed in Instance 5.2. Prove that the valuation algebra axioms still hold in this new formalism.

A.3 ★ Again, reconsider the valuation algebra of arithmetic potentials from Instance 1.3. This time, we replace the operation of multiplication in the definition of combination in equation (1.10) by addition and the operation of addition in the definition of projection in equation (1.11) by minimization. This leads to the formalism of *Spohn potentials* or *weighted constraints* which will later be discussed in Instance 5.1. Prove that the valuation algebra axioms still hold in this new formalism. Alternatively, we could also take maximization for the projection rule.

A.4 ★ Cancellativity is an important property in semigroup theory and will be used frequently in later parts of this book. A valuation algebra $\langle \Phi, D \rangle$ is called *cancellative*, if its semigroup under combination is cancellative, i.e. if for all $\phi \in \Phi$

$$\phi \otimes \psi \quad = \quad \phi \otimes \psi'$$

implies that $\psi = \psi'$. Prove that cancellativity holds in the valuation algebras of arithmetic potentials from Instance 1.3, Gaussian potentials from Instance 1.6 and Spohn potentials from Exercise A.3.

A.5 ★ The domain axiom (A6) expresses that valuations are not affected by trivial projection. Prove that this axiom is not a consequence of the remaining axioms (A1) to (A5), i.e. construct a formalism that satisfies the axioms (A1) to (A5) but not the domain axiom (A6). The basic idea is given in (Shafer, 1991): take any valuation algebra and double the number of elements by distinguishing two versions of each element, one marked and one unmarked. We further define that the result of a projection is always marked and that a combination produces a marked element if, and only if, one of its factors is marked. Prove that all valuation algebra axioms except the domain axiom (A6) still hold in this algebra.

A.6 ★★ Study the approximation of probability density functions by discrete probability distributions and provide suitable definitions of combination and projection.

A.7 ★★★ It was shown in Instance 1.6 that Gaussian potentials form a subalgebra of the valuation algebra of density functions. Look for other parametric classes of densities that establish subalgebras, for example *uniform distributions*, or more general classes like densities with *finite* or *infinite support*. The support of a density corresponds to the part of its range where it adopts a non-zero value.

CHAPTER 2

INFERENCE PROBLEMS

With the valuation algebra framework introduced in the first chapter, we dispose of a system to express the structure of information independently of any concrete formalism. This system will now be used to describe the main computational problem of knowledge representation systems: the task of computing inference. Let us go back to our initial idea of valuations as pieces of knowledge or information. Given a collection of information pieces called *knowledgebase* and some query of interest, inference consists in aggregating all elements of the knowledgebase, followed by projecting the result to a specified domain representing the question of interest. This computational problem called *inference* or *projection problem* will take center stage in many of the following chapters. Here, we start with a short presentation of some important graphical structures used to represent knowledgebases and to uncover their hidden structure. Then, we give a formal definition of inference problems in Section 2.3, followed by a selection of examples that arise from the valuation algebra instances of Chapter 1. Most interesting are the different meanings that the inference problem adopts under different valuation algebras. We therefore not only say that formalisms *instantiate* valuation algebras, but the computational problems based on these formalisms also *instantiate* the generic inference problem. Algorithms for the efficient solution of inference problems will be presented in Chapter 3 and 4.

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

2.1 GRAPHS, TREES AND HYPERGRAPHS

An *undirected graph* $G = (V, E)$ is specified by a set of *vertices* or *nodes* V and a set of *edges* E. Edges are sets of either one or two nodes, i.e. $E \subseteq V_1 \cup V_2$ where V_1 denotes the set of all one-element subsets and V_2 the set of all two-element subsets of V. Without loss of generality, we subsequently identify nodes by natural numbers $V \subseteq \mathbb{N}$. The *neighbors* of node $i \in V$ are all other nodes with an edge to node i, $ne(i) = \{j \in V : \{i,j\} \in E\}$, and the *degree* of a graph G is the maximum number of neighbors over all its nodes:

$$deg(G) \quad = \quad \max_{i \in V} |ne(i)|. \tag{2.1}$$

A sequence of nodes $(i_0, i_1, \ldots, i_n)$ with $\{i_{k-1}, i_k\} \in E$ for $k = 1, \ldots, n$ is called a *path* of *length* n from the *source node* i_0 to the *terminal node* i_n. A graph is said to be *connected* if there is a path between any pair of nodes. A *cycle* is a path with $i_0 = i_n$. A *tree* is a connected graph without cycles and $V_1 = \emptyset$. Trees satisfy the property that removing one edge always leads to a non-connected graph. A subset of nodes $W \subseteq V$ is called a *clique*, if all elements are pairwise connected, i.e. if $\{i,j\} \in E$ for all $i, j \in W$ with $i \neq j$. A clique is *maximal* if the set obtained by adding any node from $V - W$ is no more a clique.

Example 2.1 *Figure 2.1 displays two graphs: The left-hand graph $G_1 = (V_1, E_1)$ is specified by $V_1 = \{1,2,3,4\}$ with $E_1 = \{\{1,2\},\{2,3\},\{3,1\},\{4\}\}$. This graph is not connected, the neighbors of node 2 are $ne(2) = \{1,3\}$, and the node set $\{1,2,3\}$ forms a maximal clique. On the right, $G_2 = (V_2, E_2)$ is given by $V_2 = \{1,2,3,4,5\}$ and $E_2 = \{\{5,1\},\{1,2\},\{1,3\}, \{3,4\}\}$. This graph is connected and since there are no cycles, it is a tree. The neighbors of node 1 are $ne(1) = \{2,3,5\}$, and the degree of G_2 is $deg(G_2) = 3$.*

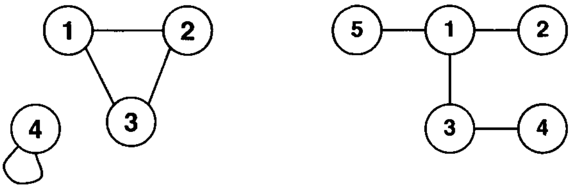


Figure 2.1 Undirected graphs.

In a *directed graph* $G = (V, E)$ edges are ordered pairs of nodes $E \subseteq V \times V$. We thus refer to $(i,j) \in E$ as the directed edge from node i to node j. A sequence of nodes $(i_0, i_1, \ldots, i_n)$ with $(i_{k-1}, i_k) \in E$ for $k = 1, \ldots, n$ is called a *path* of *length* n from the *source node* i_0 to the *terminal node* i_n. If $i_0 = i_n$ the path is again called a *cycle*. A directed graph is connected if for every pair of nodes i and j there exists either a path from i to j or from j to i. For every directed graph we find an associated undirected graph by ignoring all directions of the edges. We then say that a directed graph is a *tree* if its associated undirected graph is a tree. Moreover, if a directed graph is a tree and all edges are directed towards a selected node, then the graph is called a *directed tree* with the selected node as *root node*.

Example 2.2 *Consider the two graphs in Figure 2.2. The left-hand graph* $G_1 = (V_1, E_1)$ *is specified by* $V_1 = \{1,2,3,4\}$ *with* $E_1 = \{(1,3),(2,1),(2,3),(4,4)\}$. *This graph is not connected. On the right,* $G_2 = (V_2, E_2)$ *is given by* $V_2 = \{1,2,3,4,5\}$ *and* $E_2 = \{(5,1),(2,1),(3,1),\ (4,3)\}$. *This graph is a directed tree with root node* 1.

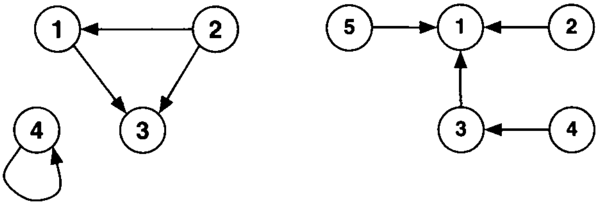


Figure 2.2 Directed graphs.

Next, we consider directed or undirected graphs where each node possesses a label out of the powerset $\mathcal{P}(r)$ of a set of variables r. Thus, a *labeled, undirected graph* $(V, E, \lambda, \mathcal{P}(r))$ is an undirected graph with a *labeling function* $\lambda : V \rightarrow \mathcal{P}(r)$ that assigns a domain to each node. Such a graph is shown in Example 2.3.

Example 2.3 *Figure 2.3 shows a labeled graph* $(V, E, \lambda, \mathcal{P}(r))$ *where* $V = \{1,2,3,4,5\}$, $E = \{\{5,1\},\{1,2\},\{1,3\},\{2,4\},\{4,3\}\}$ *and* $r = \{A,B,C,D\}$. *The labeling function* λ *is given by:* $\lambda(1) = \lambda(4) = \{A,C,D\}$, $\lambda(2) = \{A,D\}$, $\lambda(3) = \{D\}$ *and* $\lambda(5) = \{A,B\}$. *Observe that node labels are not necessarily unique and therefore cannot replace the node numbering.*

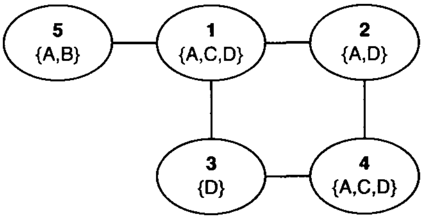


Figure 2.3 A labeled graph.

A *hypergraph* $H = (V, S)$ is specified by a set of vertices or nodes V and a set of *hyperedges* $S = \{S_1, \ldots, S_n\}$ which are subsets of V, i.e. $S_i \subseteq V$ for $i = 1, \ldots, n$.

The *primal graph* of a hypergraph is an undirected graph (V, E_p) such that there is an edge $\{u, v\} \in E_p$ for any two vertices $u, v \in V$ that appear in the same hyperedge,

$$E_p = \big\{\{u,v\} : u, v \in S_i \text{ for some } 1 \leq i \leq n\big\}.$$

The *dual graph* of a hypergraph is an undirected graph (S, E_d) that has a vertex for each hyperedge, and there is an edge $\{S_i, S_j\} \in E_d$ if the corresponding hyperedges share a common vertex, i.e. if $S_i \cap S_j \neq \emptyset$.

Example 2.4 *Consider a hypergraph* (V, S) *with* $V = \{1, 2, 3, 4, 5, 6\}$ *and* $S = \{\{1, 2, 3\}, \{3, 4, 5\}, \{1, 5, 6\}\}$. *Figure 2.4 shows a graphical representation of this hypergraph together with its associated primal and dual graphs.*

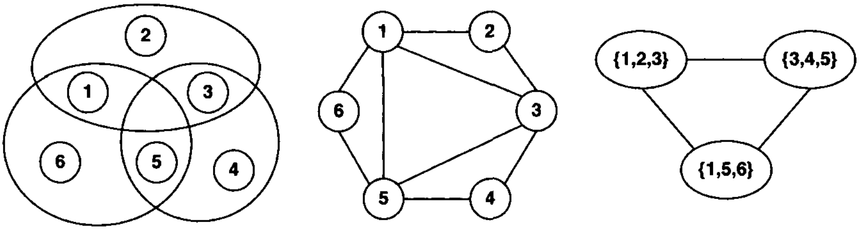


Figure 2.4 The hypergraph and its associated primal and dual graphs of Example 2.4.

2.2 KNOWLEDGEBASES AND THEIR REPRESENTATION

Consider a valuation algebra $\langle \Phi, \mathcal{P}(r) \rangle$ for a set of variables r and a set of valuations $\{\phi_1, \ldots, \phi_n\} \subseteq \Phi$. This valuation set stands for the available knowledge about some topic which constitutes the starting point of any deduction process. Accordingly, we refer to such a set as *knowledgebase*. It is often very helpful to represent knowledgebases graphically, since this uncovers a lot of hidden structure. There are numerous possibilities for this task, but most of them are somehow related to a particular instance. A general representation of knowledgebases are *valuation networks*: A valuation network (Shenoy, 1992b) is a graphical structure where variables are represented by circular nodes and valuations by rectangular nodes. Further, each valuation is connected with all variables of its domain. Valuation networks highlight the structure of a factorization and are therefore also called *factor graphs* (Kschischang *et al.*, 2001) in the literature. Another general representation is given by the hypergraph $H = (r, S)$ with $S = \{d(\phi_1), \ldots, d(\phi_n)\}$ and its associated primal and dual graphs. These possibilities are also illustrated in Example 2.5.

Example 2.5 *Figure 2.5 shows a valuation network, a hypergraph and its associated primal graph for the knowledgebase* $\{\phi_1, \phi_2, \phi_3\}$ *with* $d(\phi_1) = \{A, C\}$, $d(\phi_2) = \{B, C, D\}$ *and* $d(\phi_3) = \{B, D\}$.

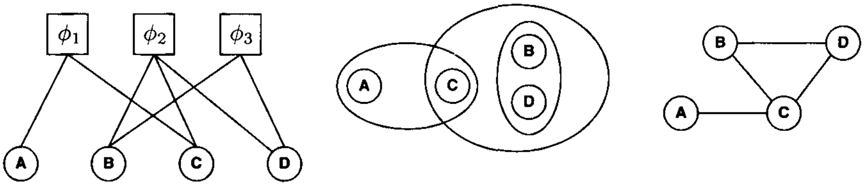


Figure 2.5 Representations for the knowledgebase in Example 2.5.

2.3 THE INFERENCE PROBLEM

The knowledgebase is the basic component of an inference problem. It represents all the available information that we may want to combine and project on the domains of interest. This is the statement of the following definition.

Definition 2.1 *The task of computing*

$$\phi^{\downarrow x_i} \quad = \quad (\phi_1 \otimes \cdots \otimes \phi_n)^{\downarrow x_i} \tag{2.2}$$

for a given knowledgebase $\{\phi_1, \ldots, \phi_n\} \subseteq \Phi$ *and domains* $x = \{x_1, \ldots, x_s\}$ *where* $x_i \subseteq d(\phi_1 \otimes \ldots \otimes \phi_n)$ *is called an* inference problem *or a* projection problem. *The domains* x_i *are called* queries.

The valuation $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ resulting from the combination of all knowledgebase factors is often called *joint valuation* or *objective function* (Shenoy, 1996). Further, we refer to inference problems with $|x| = 1$ as *single-query inference problems* and we speak about *multi-query inference problems* if $|x| > 1$. For a better understanding of how different instances give rise to inference problems, we first touch upon a selection of typical and well-known fields of application that are based on the valuation algebra instances introduced in Chapter 1. Efficient algorithms for the solution of inference problems are discussed in Chapter 3 and 4. Here, we only focus on the identification of inference problems behind different applications.

■ 2.1 Bayesian Networks

A *Bayesian network*, as for instance in (Pearl, 1988), is a graphical representation of a *joint probability distribution* over a set of variables $r = \{X_1, \ldots, X_n\}$. The network itself is a directed, acyclic graph that reflects the conditional independencies among variables, which are associated with a node of the network. Each node contains a *conditional probability table* that quantifies the influence between variables. These tables can be considered as probability potentials, introduced as Instance 1.3. The joint probability distribution p of a Bayesian network is given by

$$p(X_1, \ldots, X_n) \quad = \quad \prod_{i=1}^{n} p(X_i | pa(X_i)) \tag{2.3}$$

where $pa(X_i)$ denotes the parents of X_i in the network. If the set $pa(X_i)$ is empty, the conditional probability becomes an ordinary probability distribution. Consider next a set $q \subseteq r$ called *query variables* and a set $e \subseteq r$ called *evidence variables*. The evaluation of a Bayesian network then consists in computing the posterior probability distribution for the query variables, given some observed values of the evidence variables. Such an observation corresponds to a tuple $\mathbf{e} \in \Omega_e$. Thus, if $\mathbf{q} \in \Omega_q$ denotes a *query event*, we want to compute the *posterior probability* $p(\mathbf{q}|\mathbf{e})$ defined as

$$p(\mathbf{q}|\mathbf{e}) \quad = \quad \frac{p(\mathbf{q}, \mathbf{e})}{p(\mathbf{e})}. \tag{2.4}$$

To clarify how Bayesian networks give rise to inference problems, we consider an example from medical diagnostics (Lauritzen & Spiegelhalter, 1988).

> Shortness-of-breath (dyspnoea) may be due to tuberculosis, lung cancer or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X-ray do not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea. A patient presents himself at a chest clinic with dyspnoea and has recently visited Asia. Smoking history and chest X-ray are not yet available. The doctor wants to know the chance that this patient suffers from bronchitis.

Here, the query and evidence variables are easily identified: the query set $q = \{B\}$ contains the variable standing for bronchitis, and the evidence variables $e = \{A, D\}$ are dyspnoea and the recent visit to Asia. Figure 2.6 shows the Bayesian network of this example that reflects the dependencies among the involved variables $r = \{A, B, D, E, L, S, T, X\}$. All variables are assumed to be binary, e.g. the frame values $\Omega_S = \{s, \overline{s}\}$ refer to a smoking or non-smoking patient respectively. Also, the Bayesian network shows the conditional probability tables that are associated with each node. These are the valuations of our example and can be modelled using the language of probability potentials. For example, we have

$p(T|A) =$

T	A	
t	a	0.05
t	$\overline{a}$	0.01
$\overline{t}$	a	0.95
$\overline{t}$	$\overline{a}$	0.99

Thus, we get a knowledgebase that consists of eight valuations (one probability potential for each node of the Bayesian network) and the joint valuation (joint probability mass function) is given by

$$p(A,B,D,E,L,S,T,X) \quad = \quad p(A) \otimes p(T|A) \otimes \cdots \otimes p(D|E,B). \qquad (2.5)$$

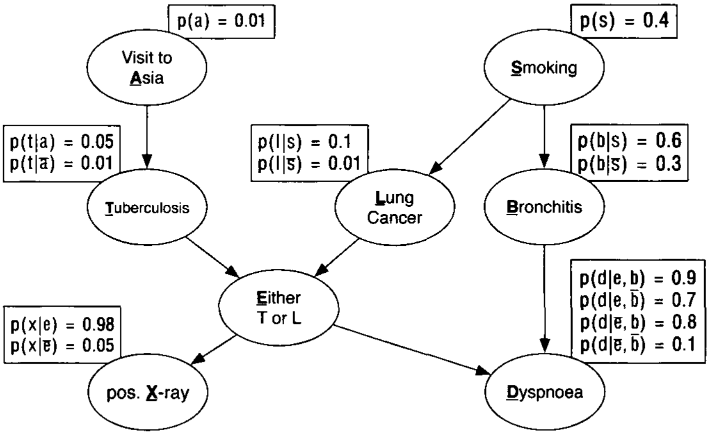


Figure 2.6 The Bayesian network of the medical example. The bold-faced, underlined letters refer to the variable associated with the corresponding node.

To answer the query $p(B|A, D)$ we therefore need to compute

$$p(B|A,D) \;=\; \frac{p(A,B,D)}{p(A,D)} \;=\; \frac{p(A,B,D,E,L,S,T,X)^{\downarrow\{A,B,D\}}}{p(A,B,D,E,L,S,T,X)^{\downarrow\{A,D\}}}.$$

and obtain our final answer by a last table lookup in $p(B|A, D)$ to extract the entry that corresponds to the query and evidence event. To sum it up, evaluating the Bayesian network of the medical example requires to compute an inference problem with knowledgebase $\{p(A), p(T|A), \ldots, p(D|E, B)\}$ and query set $\{\{A, B, D\}, \{A, D\}\}$. In fact, it is sufficient to compute the query $\{A, B, D\}$ because $p(A, D)$ can easily be obtained from this result by one last projection

$$p(A,D) \quad = \quad p(A,B,D)^{\downarrow\{A,D\}}. \qquad (2.6)$$

■ 2.2 Query Answering in Relational Databases

The procedure of query answering in a relational database is another example of an inference problem. The knowledgebase is a set of database tables that are elements of the valuation algebra of relations introduced as Instance 1.2. The inference problem then computes the natural join of all relations in the knowledgebase and projects the result on the attributes of interest. However, it is important to note that the *selection operator*, which is integral part of every relational database system, does not directly have a counterpart in the valuation algebra framework. Nevertheless, there are techniques that allow embedding selections. A particular simple but limited possibility is to transform selections into relations themselves that are then added to the knowledgebase. The advantage of this method is that we stay on a generic level and do not need to care about selections during the computations. On the other hand, transforming selections may create relations of infinite cardinality if special queries over attributes with infinite frames are involved. In such cases, more complex techniques are required that extend the definition of inference problems and treat selections explicitly. We refer to (Schneuwly, 2007) for a detailed discussion of *selection enabled local computation.* In the following (self-explanatory) example we focus on the first method and convert selections into relations.

SID	Student
1	Ann
2	John
3	Laura
⋮	⋮

SID	Grade	LID
1	C	901
2	A	901
2	B	903
⋮	⋮	⋮

LID	Lecture	Semester
901	Chemistry	S1898
902	Mathematics	S1801
903	Physics	A1905

PID	LID
1	903
2	902
3	901

PID	Professor
1	Einstein
2	Gauss
3	Curie

Thus, we get a knowledgebase of five relations $\{r_1, \ldots, r_5\}$. Let us further assume the following database query:

Find all students with grade A or B in a lecture of professor Einstein.

This gives us the single query for the inference problem $\{Student\}$. In addition, it contains two selections which correspond to the following new relations:

$s_1 =$

Grade
A
B

$s_2 =$

Professor
Einstein

Thus, we finally derive the inference problem

$$\left(r_1 \otimes r_2 \otimes r_3 \otimes r_4 \otimes r_5 \otimes s_1 \otimes s_2\right)^{\downarrow\{Students\}}.$$

2.3 Reasoning with Mass Functions

Mass functions correspond to the formalism of set potentials introduced as Instance 1.4 with an additional condition of normalization. They are used to represent knowledge in the *Dempster-Shafer theory of evidence* (Shafer, 1976) and provide similar reasoning facilities as Bayesian networks, but with different semantics as pointed out in (Smets & Kennes, 1994; Kohlas & Monney, 1995). It will be shown in Section 4.7 that normalization in the context of valuation algebras can be treated on a generic level. We therefore neglect normalization and directly use set potentials for reasoning with mass functions in the subsequent example. Normalized set potentials will later be introduced in Instance D.9. The following imaginary example has been proposed by (Shafer, 1982):

> Imagine a disorder called *ploxoma* which comprises two distinct *diseases*: θ_1 called *virulent ploxoma*, which is invariably fatal, and θ_2 called *ordinary ploxoma*, which varies in severity and can be treated. Virulent ploxoma can be identified unequivocally at the time of a victim's death, but the only way to distinguish between the two diseases in their early stages is a blood test with three possible outcomes X_1, X_2 and X_3. The following evidence is available:
>
> 1. Blood tests of a large number of patients dying of virulent ploxoma showed the outcomes X_1, X_2 and X_3 occurring 20, 20 and 60 per cent of the time, respectively.
> 2. A study of patients whose ploxoma had continued so long as to be almost certainly ordinary ploxoma showed outcome X_1 to occur 85 per cent of the time and outcomes X_2 and X_3 to occur 15 per cent of the time. (The study was made before methods for distinguishing between X_2 and X_3 were perfected.) There is some question whether the patients in the study represent a fair sample of the population of ordinary ploxoma victims, but experts feel fairly confident (say 75 per cent) that the criteria by which patients were selected for the study should not affect the distribution of test outcomes.
> 3. It seems that most people who seek medical help for ploxoma are suffering from ordinary ploxoma. There have been no careful statistical studies, but physicians are convinced that only 5-15 per cent of ploxoma patients suffer from virulent ploxoma.

This story introduces the variables D for the disease with the possible values $\{\theta_1, \theta_2\}$ and T for the test result taking values from $\{X_1, X_2, X_3\}$. (Shafer, 1982) comes up with a systematic disquisition on how to translate this evidence into mass functions in a meaningful manner. Here, we do not want to enter into this discussion and directly give the corresponding knowledgebase factors. Instead, we focus on the fact that reasoning with mass functions again amounts to the solution of an inference problem.

The mass function m_1 derived from the first statement is:

$$m_1 \quad = \quad \begin{array}{|c||c|} \hline \{(\theta_1, X_1), (\theta_2, X_1), (\theta_2, X_2), (\theta_2, X_3)\} & 0.2 \\ \{(\theta_1, X_2), (\theta_2, X_1), (\theta_2, X_2), (\theta_2, X_3)\} & 0.2 \\ \{(\theta_1, X_3), (\theta_2, X_1), (\theta_2, X_2), (\theta_2, X_3)\} & 0.6 \\ \hline \end{array}$$

For example, our belief that the blood test returns X_3 for a patient with virulent ploxoma θ_1 is 0.6. This mass function does not give any evidence about the test outcome with disease θ_2. The second factor m_2 is:

$$m_2 = \begin{array}{|c||c|} \hline \{(\theta_2, X_1), (\theta_1, X_1), (\theta_1, X_2), (\theta_1, X_3)\} & 0.85 \cdot 0.75 \\ \{(\theta_2, X_2), (\theta_2, X_3), (\theta_1, X_1), (\theta_1, X_2), (\theta_1, X_3)\} & 0.15 \cdot 0.75 \\ \{(\theta_2, X_1), (\theta_2, X_2), (\theta_2, X_3), (\theta_1, X_1), (\theta_1, X_2), (\theta_1, X_3)\} & 0.25 \\ \hline \end{array}$$

For example, in case of ordinary ploxoma θ_2 the test outcome is X_1 in 85 per cent of all cases. This value is scaled with the expert's confidence of 0.75. There are 25 per cent of cases where the test tells us nothing whatsoever. Finally, the third mass function m_3 is:

$$m_3 = \begin{array}{|c||c|} \hline \{(\theta_1)\} & 0.05 \\ \{(\theta_2)\} & 0.85 \\ \{(\theta_1), (\theta_2)\} & 0.10 \\ \hline \end{array}$$

For example, the belief that a patient, who seeks medical help for ploxoma, suffers from θ_2 is 0.85. Here, the value 0.1 signifies that it is not always possible to distinguish properly between the two diseases.

Let us now assume that we are interested in the reliability of the test result. In other words, we want to compute the belief held in the presence of either θ_1 or θ_2 given that the test result is for example X_1. Similar to Bayesian networks, this is achieved by adding the observation m_o to the knowledgebase which expresses that the test result is X_1:

$$m_o = \begin{array}{|c||c|} \hline \{(X_1)\} & 1.0 \\ \hline \end{array}$$

Thus, we obtain the belief for the presence of either virulent or ordinary ploxoma by solving the following computational task:

$$\left(m_1 \otimes m_2 \otimes m_3 \otimes m_o\right)^{\downarrow T}.$$

This is clearly an inference problem with knowledgebase $\{m_1, m_2, m_3, m_o\}$ and query $\{T\}$. Although this example with only 4 knowledgebase factors, one query and two variables seems easily manageable, it is already a respectable effort to solve it by hand. We therefore refer to (Blau *et al.*, 1994) where these computations are executed and commented.

2.4 Satisfiability

Let us consider a digital circuit build from AND, OR and XOR gates which are connected as shown in Figure 2.7. Each gate generates an output value from two input values according to the following tables:

and	0	1
0	0	0
1	0	1

or	0	1
0	0	1
1	1	1

xor	0	1
0	0	1
1	1	0

This particular circuit is called *full adder* since it computes the binary addition of the three input values as shown in Figure 2.8. The value Out_1 corresponds to the *carry-over bit.*

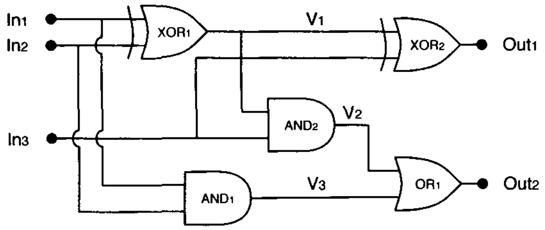


Figure 2.7 A digital circuit of a binary adder.

In_1	In_2	In_3	Out_1	Out_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 2.8 The result table of a full adder circuit.

From the adder circuit in Figure 2.7, we extract the following five Boolean functions that constitute our knowledgebase $\{\phi_1, \ldots, \phi_5\}$:

$$V_1 = xor(In_1, In_2) \qquad Out_1 = xor(V_1, In_3) \qquad Out_2 = or(V_2, V_3)$$

$$V_2 = and(V_1, In_3) \qquad V_3 = and(In_1, In_2)$$

Now, imagine the following situation: we insert the values $In_1 = 0, In_2 = 0$ and $In_3 = 0$ into the above adder circuit and observe $Out_1 = 1$ and $Out_2 = 1$. A quick glance on Figure 2.8 confirms that this result can only be produced by a faulty adder component. Fortunately, we are supplied with a voltage meter to query the actual values of each line V_1 to V_3, which can be used to locate the fault. However, this requires that the correct line values for the particular input are known which cannot be deduced from the above tables. We therefore add the three Boolean function $\phi_6 : In_1 = 0, \phi_7 : In_2 = 0$ and $\phi_8 : In_3 = 0$ to our knowledgebase and compute the inference problem with query $\{V_1, V_2, V_3\}$:

$$\left(\phi_1 \otimes \phi_2 \otimes \cdots \otimes \phi_8\right)^{\downarrow\{V_1,V_2,V_3\}}.$$

Here, we compute in the valuation algebra of Boolean functions introduced in Instance 1.1. Computing this query gives $V_1 = 1$, $V_2 = 0$ and $V_3 = 0$ as the correct line values which now makes the physical monitoring process possible. Alternatively, we may also deduce the correct input or output values from partial observations: If we for example know that $\phi_9 : In_3 = 0$ and $\phi_{10} : Out_2 = 0$, solving the inference problem

$$\left(\phi_1 \otimes \phi_2 \otimes \phi_3 \otimes \phi_4 \otimes \phi_5 \otimes \phi_9 \otimes \phi_{10}\right)^{\downarrow\{In_1,In_2,Out_1\}}$$

gives $In_1 = 1$, $In_2 = Out_1 = 0$. Clearly, it would have been possible to extract this result from Figure 2.8 directly, but building this table becomes quickly intractable even for an only slightly larger circuit. How such larger adder circuits are constructed from the one-bid-adder is shown in Figure 2.9.

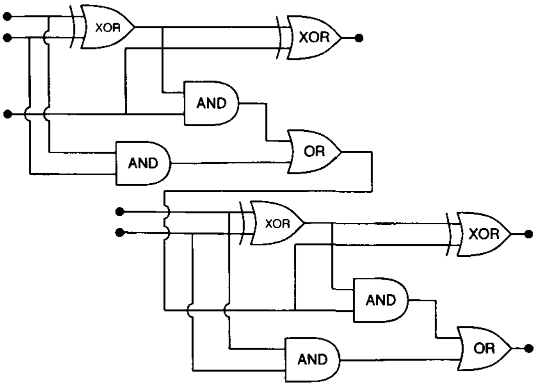


Figure 2.9 Connecting two 1-bit-adders produces a 2-bit-adder.

This example of an inference problem is a very classical application of constraint programming, and many similar applications can be found in the corresponding literature. For example, we refer to (Apt, 2003) for a collection of constraint satisfaction problems. However, let us point out that the various constraint formalisms indeed represent an important class of valuation algebras, but there are many further instances beyond classical constraints. We will discuss different constraint formalism in Chapter 5 and also show that they all instantiate the valuation algebra framework.

2.5 Hadamard Transforms

The *Hadamard transform* (Beauchamp, 1984; Gonzalez & Woods, 2006) is an essential part of many applications that incorporate error correcting codes, random number generation or image and video compression. The unnormalized variant of a Hadamard transform uses a $2^m \times 2^m$ Hadamard matrix H_m that is defined recursively by the so-called *Sylvester construction*:

$$H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_m = \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix}.$$

For instance, we obtain for $m = 3$,

$$H_3 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

Hadamard matrices have many interesting properties: they are symmetric, orthogonal, self–inverting up to a constant and any two rows differ in exactly 2^{m-1} positions. The columns of a Hadamard matrix are called *Walsh functions*. Thus, if we multiply Hadamard matrix with a real-valued vector v, we obtain a transformation of this vector into a superposition of Walsh functions which constitutes the Hadamard transform. For illustration, assume three binary variables X_1 to X_3 and a function $v : \{X_1, X_2, X_3\} \to \mathbb{R}$. This function is completely determined by its values v_i for $i = 0, \ldots, 7$. We then obtain for its Hadamard transform,

$$H_3 \cdot \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{pmatrix} = \begin{pmatrix} v_0 + v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7 \\ v_0 - v_1 + v_2 - v_3 + v_4 - v_5 + v_6 - v_7 \\ v_0 + v_1 - v_2 - v_3 + v_4 + v_5 - v_6 - v_7 \\ v_0 - v_1 - v_2 + v_3 + v_4 - v_5 - v_6 + v_7 \\ v_0 + v_1 + v_2 + v_3 - v_4 - v_5 - v_6 - v_7 \\ v_0 - v_1 + v_2 - v_3 - v_4 + v_5 - v_6 + v_7 \\ v_0 + v_1 - v_2 - v_3 - v_4 - v_5 + v_6 + v_7 \\ v_0 - v_1 - v_2 + v_3 - v_4 + v_5 + v_6 - v_7 \end{pmatrix}.$$

An alternative way to compute this Hadamard transform bears on a duplication of variables (Aji & McEliece, 2000). We introduce the binary variables Y_1 to Y_3 which give a binary representation of the line number in the Hadamard matrix. Then, the same computational task is expressed by the formula

$$\sum_{X_1,X_2,X_3} f(X_1, X_2, X_3) \cdot (-1)^{X_1 Y_1} \cdot (-1)^{X_2 Y_2} \cdot (-1)^{X_3 Y_3}.$$

The reader may easily convince himself of the equivalence between the two computational tasks. In this alternative representation, we discover a sum over a multiplication of four discrete functions taking real values or, in other words, an inference problem over the valuation algebra of arithmetic potentials, where the knowledgebase consists of the four discrete functions, and the single query is $\{Y_1, Y_2, Y_3\}$. Thus, we could alternatively write for the Hadamard transform:

$$\left(f \otimes (-1)^{X_1Y_1} \otimes (-1)^{X_2Y_2} \otimes (-1)^{X_3Y_3}\right)^{\downarrow\{Y_1,Y_2,Y_3\}} \tag{2.7}$$

which is reduced to the solution of a singe-query inference problem.

2.6 Discrete Fourier and Cosine Transforms

Our last example of an inference problem comes from signal processing. For a positive integer $N \in \mathbb{N}$ we consider a function $f : \{0, \ldots, N-1\} \to \mathbb{C}$ that represents a sampled signal, i.e. the value $f(x) \in \mathbb{C}$ corresponds to the signal value at time $0 \leq x \leq N-1$. Thus, the input signal is usually said to be in the *time domain* although any kind of sampled data may serve as input. The *complex, discrete Fourier transform* then changes the N point input signal into two N point output signals which contain the *amplitudes* of the component sine and cosine waves. It is given by

$$F(y) \quad = \quad \sum_{x=0}^{N-1} f(x)e^{-\frac{2i\pi xy}{N}}. \tag{2.8}$$

for $0 \leq y \leq N-1$. Accordingly, F is usually said to be in *frequency domain*. The discrete Fourier transform has many important applications as for example in audio and image processing or data compression. We refer to (Smith, 1999) for a survey of different applications and for a more detailed mathematical background. Figure 2.10 gives an illustration of the discrete Fourier transform.

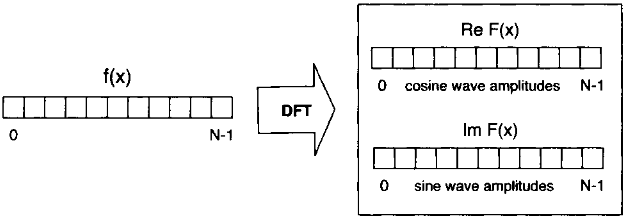


Figure 2.10 Input and output of a complex, discrete Fourier transform.

The following procedure of transforming a discrete Fourier transform into an inference problem was proposed by (Aji, 1999) and (Aji & McEliece, 2000).

Let us take $N = 2^m$ for some $m \in \mathbb{N}$ and write x and y in binary representation:

$$x = \sum_{j=0}^{m-1} X_j 2^j \quad \text{and} \quad y = \sum_{l=0}^{m-1} Y_l 2^l$$

with $X_j, Y_l \in \{0, 1\}$. This corresponds to an encoding of x and y into binary configurations $x = (X_0, \ldots, X_{m-1})$ and $y = (Y_0, \ldots, Y_{m-1})$. Observe that we may consider the components X_i and Y_i as new variables of this system (therefore the notation as capital letters). The product xy then gives

$$xy \quad = \quad \sum_{0 \leq j,l \leq m-1} X_j Y_l 2^{j+l}.$$

We insert this encoding into equation (2.8) and obtain

$$\begin{aligned}
&\sum_{N_0,\ldots,N_{m-1}} f(N_0, \ldots, N_{m-1}) \cdot e^{-\frac{2\pi i \sum_{0 \leq j,l \leq m-1} N_j K_l 2^{j+l}}{2^m}} \\
= \quad &\sum_{N_0,\ldots,N_{m-1}} f(N_0, \ldots, N_{m-1}) \prod_{0 \leq j,l \leq m-1} e^{-\frac{2\pi i N_j K_l 2^{j+l}}{2^m}} \\
= \quad &\sum_{N_0,\ldots,N_{m-1}} f(N_0, \ldots, N_{m-1}) \prod_{0 \leq j,l \leq m-1} e^{-\frac{2\pi i N_j K_l}{2^{m-j-l}}} \\
= \quad &\sum_{N_0,\ldots,N_{m-1}} f(N_0, \ldots, N_{m-1}) \prod_{0 \leq j+l \leq m-1} e^{-\frac{2\pi i N_j K_l}{2^{m-j-l}}}.
\end{aligned}$$

The last equality follows since the exponential factors become unity when $j + l \geq m$. Introducing the short-hand notation

$$e_{j,l} \quad = \quad e^{-\frac{2\pi i N_j K_l}{2^{m-j-l}}}$$

we obtain for the equation (2.8):

$$F(K_0, \ldots, K_{m-1}) \;=\; \sum_{N_0,\ldots,N_{m-1}} f(N_0, \ldots, N_{m-1}) \prod_{0 \leq j+l \leq m-1} e_{j,l}.$$

This is an inference problem over the valuation algebra of (complex) arithmetic potentials. All variables $r = \{N_0, \ldots, N_{m-1}, K_0, \ldots, K_{m-1}\}$ take values from $\{0, 1\}$ and the discrete functions f and $e_{j,l}$ map binary configurations to complex values. The knowledgebase is therefore given by

$$\{f\} \cup \{e_{j,l} : 0 \leq j + l \leq m - 1\}$$

and the query of the inference problem is $\{K_0, \ldots, K_{m-1}\}$. Thus, we could alternatively express the computational task of a discrete Fourier transform as

$$\left(f \otimes e_{0,0} \otimes \cdots \otimes e_{m-1,0} \otimes e_{1,0} \otimes \cdots \otimes e_{1,m-2} \otimes \cdots\right)^{\downarrow\{K_0,\ldots,K_{m-1}\}}. \tag{2.9}$$

Observe that the inference problem behind the discrete Fourier transform has been uncovered by the artificial introduction of variables that manifests itself in the $e_{j,l}$ knowledgebase factors. This knowledgebase has a very particular structure as shown by the board in Figure 2.11. For $m = 8$, each cell of this board corresponds to one factor $e_{j,l}$ with j identifying the row and l the column. Observe, however, that only the shaded cells refer to functions contained in the knowledgebase. As it will be shown in Section 3.11, this *sparse* structure of the knowledgebase enables the efficient solution of this inference problems by the local computation procedures introduced in the next chapter.

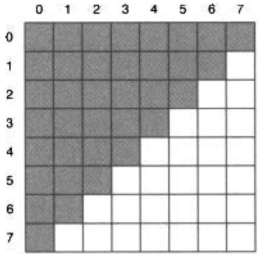


Figure 2.11 Illustration of the $e_{j,l}$ functions in the inference problem of the discrete Fourier transform with $m = 8$. Index j identifies the row, and index l the column. The shaded cells refer to the functions contained in the knowledgebase.

The same technique of introducing binary variables can be applied to other discrete, linear transforms such as the *discrete cosine transform* which for example plays an important role in the JPEG image compression scheme. See Exercise B.1.

2.4 CONCLUSION

Sets of valuations are called knowledgebases and represent the available knowledge at the beginning of an inference process. There are several possibilities for visualizing knowledgebases and their internal structure. Most important are valuation networks, hypergraphs and their associated primal and dual graphs. The second ingredient of an inference task is the query set. According to the number of queries, we distinguish between single-query inference problems and multi-query inference problems. Inference problems state the main computational interest in valuation algebras. They require combining all available information from the knowledgebase and projecting the result afterwards on the elements of the query set. Depending on the underlying valuation algebra inference problems adopt very different meanings. Examples of applications that attribute to the solution of inference problems are reasoning tasks in Bayesian and belief networks, query answering in relational databases, constraint satisfaction and discrete Fourier or Hadamard transforms.

PROBLEM SETS AND EXERCISES

B.1 ★ The *discrete cosine transform* of a signal f is defined by

$$\mathcal{F}(y) \quad = \quad 2\sum_{x=0}^{N-1} f(x) \cdot \cos\left[\frac{\pi \cdot y}{N} \cdot \left(x + \frac{1}{2}\right)\right]. \tag{2.10}$$

Exploit the binary representation of x and y and the identity

$$\cos(\varphi) \quad = \quad \frac{1}{2}\left(e^{i\varphi} + e^{-i\varphi}\right)$$

to transform this formula into two inference problems over the valuation algebra of arithmetic potentials. A similar technique has been applied for the discrete Fourier transform in Instance 2.6. Compare the knowledgebase factors, their domains and the query sets of the two inference problems and also with the inference problem of the discrete Fourier transform.

B.2 ★ Assume that we want to determine the exact weight of an object. It is clear that the result of a weighting process always includes an unknown error term. Using $n \in \mathbb{N}$ different balances, we thus obtain n different observations $x_i = m + \omega_i$ for $i = 1, \ldots, n$, where m refers to the object weight and ω_i to the error term. We further assume that the errors of the different devices are mutually independent and normally distributed with mean value 0 and variance σ_i^2. Rewriting the above equation as $m = x_i - \omega_i$, we obtain a Gaussian potential $\phi_i = (x_i, \frac{1}{\sigma_i^2})$ for each measurement. For simplicity, assume $m = 2$ and determine the combined potential $\phi = \phi_1 \otimes \phi_2$ by the combination rule in Instance 1.6. Then, by choosing $\sigma_1 = \sigma_2$ (balances of the same type), show that the object weight m corresponds to the arithmetic mean of the observed values. These calculations mirror the solution of a (trivial) projection problem where the domain of the objective function ϕ consists of only one variable. A more comprehensive example will be given in Section 10.1.

B.3 ★ We proved in Exercise A.2 that possibility potentials form a valuation algebra. In fact, this corresponds to the valuation algebra of probability potentials with addition replaced by maximization. Apply this algebra to the Bayesian network of Instance 2.1 and show that the projection problem with empty domain identifies the value of the *most probable configuration* in the objective function.

B.4 ★ Figure 2.12 shows a representation of a stochastic process called *hidden Markov chain*. The random variables X_k refer to the state of the process at time $k = 1, \ldots, n$ and are subject to a known transition probability $p(X_{k+1}|X_k)$. Further, we assume that a prior probability $p(X_0)$ for the first variable X_0 is given. However, the states of the hidden Markov chain cannot be observed from the outside. Instead, a second random process with variables Y_k generates an observable state at each point in time that only depends on the real but hidden state X_k with a known probability $p(Y_k|X_k)$. For a fixed $k \geq 0$, a hidden Markov chain is a particular Bayesian network. There are three computational problems related to hidden Markov chains:

- *Filtering:* Given an observed sequence of states $y_1, \ldots y_k$, determine the probability of the hidden state X_k by computing $p(X_k|Y_1 = y_1, \ldots, Y_k = y_k)$.

- *Prediction:* Given an observed sequence of states $y_1, \ldots y_k$ and $n > 1$, determine the probability of the hidden state at some point $k + n$ in the future by computing $p(X_{k+n}|Y_1 = y_1, \ldots, Y_k = y_k)$. This corresponds to repeated filtering without new observations.

- *Smoothing:* Given an observed sequence of states $y_1, \ldots y_k$ and $1 \leq n \leq k$, determine the probability of the hidden state at some point $k - n$ in the past by computing $p(X_{k-n}|Y_1 = y_1, \ldots, Y_k = y_k)$.

Show that the computational tasks of filtering, prediction and smoothing in hidden Markov chains can be expressed by projection problems with different query sets in the valuation algebra of probability potentials. How does the meanings of these projection problems change when the valuation algebra of possibility potentials from Exercise A.2 is used? Hidden Markov chains and other stochastic processes will be reconsidered in Chapter 10.

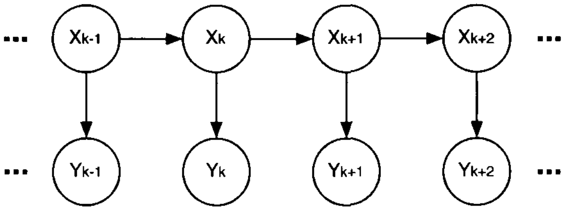


Figure 2.12 A hidden Markov chain.

CHAPTER 3

COMPUTING SINGLE QUERIES

The foregoing chapter introduced the major computational task when dealing with valuation algebras, the inference or projection problem. Also, we gained a first insight into the different semantics that inference problems can adopt under specific valuation algebra instances. The logical next step consists now in the development of algorithms to solve general inference problems. In order to be generic, these algorithms must only be based on the valuation algebra operations without any further assumptions about the concrete, underlying formalism. But before this step, we should also raise the question of whether such algorithms are necessary at all, or, in other words, why a straightforward computation of inference problems is inadequate in most cases. To this end, remember that inference problems are defined by a knowledgebase whose factors combine to the objective function which has to be projected on the actual queries of interest. Naturally, this description can directly be understood as a trivial procedure for the computation of inference problems. However, the reader may have guessed that the complexity of this simple procedure puts us a spoke in our wheel. If the knowledgebase consists of $n \in \mathbb{N}$ valuations, then, as a consequence of the labeling axiom, computing the objective function $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ results in a valuation of domain $s = d(\phi_1) \cup \cdots \cup d(\phi_n)$. In many valuation algebra instances the size of valuations grows at least exponentially with the size of their domain. And even

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

if the domains grow only polynomially, this growth may be prohibitive. Let us be more concrete and analyse the amount of memory that is used to store a valuation. In Definition 3.3 this measure will be called *weight function*. Since indicator functions, arithmetic potentials and all their connatural formalisms (see Chapter 5) can be represented in tabular form, their weight is bounded by the number of table entries which corresponds to the number of configurations of the domain. Clearly, this is exponential in the number of variables. Even worse is the situation for set potentials and all instances of Section 5.7 whose weights behave super-exponentially in the worst case. If we further assume that the time complexity of the valuation algebra operations is related to the weight of their factors, we may come to the following conclusion: it is in most cases intractable to compute the objective function ϕ explicitly. Second, the domain acts as the crucial point for complexity when dealing with valuation algebras. Consequently, the solution of inference problems is intractable unless algorithms are used which confine the domain size of all intermediate results. This essentially is the promise of local computation that organizes the computations in such a way that domains do not grow significantly. Before we start the discussion of local computation methods, we would like to point out that not all valuation algebras are subject to the above complexity concerns. There are indeed instances that have a pure polynomial behaviour, and this also seems to be the reason why these formalisms have rarely been considered in the context of local computation. However, we will introduce a family of such formalisms in Chapter 6 and also show that there are good reasons to apply local computation to polynomial formalisms.

This first chapter about local computation techniques focuses on the solution of single-query inference problems. We start with the simplest and most basic local computation scheme called *fusion* or *bucket elimination* algorithm. Then, a graphical representation of the fusion process is introduced that brings the fundamental ingredient of local computation to light: the important concept of a join tree. On the one hand, join trees allow giving more detailed information about the complexity gain of local computation compared to the trivial approach discussed above. On the other hand, they represent the required structure for the introduction of a second and more general local computation scheme for the computation of single queries called *collect algorithms*. Finally, we are going to reconsider the inference problem of the discrete Fourier transform from Section 2.3 and show in a case study that applying these schemes reveals the complexity of the fast Fourier transform. This shows that applying local computation may also be worthwhile for polynomial problems.

The fusion or bucket elimination algorithm is a simple local computation scheme that is based on variable elimination instead of projection. Therefore, we first investigate how the operation of projection in a valuation algebra can (almost) equivalently be replaced by the operation of variable elimination.

3.1 VALUATION ALGEBRAS WITH VARIABLE ELIMINATION

The axiomatic system of a valuation algebra given in Chapter 1 is based on a universe of variables r whose finite subsets form the set of domains D. This allows us to replace the operation of projection in the definition of a valuation algebra by another primitive operation called *variable elimination* which is sometimes more convenient as for the introduction of the fusion algorithm. Thus, let $\langle \Phi, D\rangle$ be a valuation algebra with $D = \mathcal{P}(r)$ and Φ being the set of all possible valuations with domains in D. For a valuation $\phi \in \Phi$ and a variable $X \in d(\phi)$, we define

$$\phi^{-X} = \phi^{\downarrow d(\phi)-\{X\}}. \tag{3.1}$$

Some important properties of variable elimination, that follow from this definition and the valuation algebra axioms, are pooled in the following lemma:

Lemma 3.1

1. *For $\phi \in \Phi$ and $X \in d(\phi)$ we have*

$$d(\phi^{-X}) = d(\phi) - \{X\}. \tag{3.2}$$

2. *For $\phi \in \Phi$ and $X, Y \in d(\phi)$ we have*

$$(\phi^{-X})^{-Y} = (\phi^{-Y})^{-X}. \tag{3.3}$$

3. *For $\phi, \psi \in \Phi$, $x = d(\phi)$, $y = d(\psi)$, $Y \notin x$ and $Y \in y$ we have*

$$(\phi \otimes \psi)^{-Y} = \phi \otimes \psi^{-Y}. \tag{3.4}$$

Proof:

1. By the labeling axiom

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

2. By Property 1 and the transitivity axiom

$$\begin{aligned}(\phi^{-X})^{-Y} &= (\phi^{\downarrow d(\phi)-\{X\}})^{\downarrow (d(\phi)-\{X\})-\{Y\}}\\ &= \phi^{\downarrow d(\phi)-\{X,Y\}}\\ &= (\phi^{\downarrow d(\phi)-\{Y\}})^{\downarrow (d(\phi)-\{Y\})-\{X\}}\\ &= (\phi^{-Y})^{-X}.\end{aligned}$$

3. Since $Y \notin x$ and $Y \in y$ we have $x \subseteq (x \cup y) - \{Y\} \subseteq x \cup y$. Hence, we obtain by application of the combination axiom

$$\begin{aligned}(\phi \otimes \psi)^{-Y} &= (\phi \otimes \psi)^{\downarrow (x\cup y)-\{Y\}}\\ &= \phi \otimes \psi^{\downarrow ((x\cup y)-\{Y\})\cap y}\\ &= \phi \otimes \psi^{\downarrow y-\{Y\}}\\ &= \phi \otimes \psi^{-Y}.\end{aligned}$$

■

According to equation (3.3) variables can be eliminated in any order with the same result. We may therefore define the consecutive elimination of a non-empty set of variables $s = \{X_1, \ldots, X_n\} \subseteq d(\phi)$ unambiguously by

$$\phi^{-s} \quad = \quad (((\phi^{-X_1})^{-X_2})\ldots)^{-X_n}. \tag{3.5}$$

This, on the other hand, permits expressing any projection by the elimination of all unrequested variables. For $x \subset d(\phi)$, we have

$$\phi^{\downarrow x} \quad = \quad \phi^{-(d(\phi)-x)}. \tag{3.6}$$

To sum it up, equation (3.1) and (3.6) allow switching between projection and variable elimination ad libitum. Moreover, we may, as an alternative to definitions in Chapter 1, define the valuation algebra framework based on variable elimination instead of projection (Kohlas, 2003). Then, the properties in the above lemma replace their counterparts based on projection in the new axiomatic system, and projection can later be derived using equation (3.6) and the additional domain axiom (Shafer, 1991).

3.2 FUSION AND BUCKET ELIMINATION

The *fusion* (Shenoy, 1992b) and *bucket elimination* (Dechter, 1999) algorithms are the two simplest local computation schemes for the solution of single-query inference problems. They are both based on successive variable elimination and correspond essentially to traditional *dynamic programming* (Bertele & Brioschi, 1972). Applied to the valuation algebra of crisp constraints, bucket elimination is also known as *adaptive consistency* (Dechter & Pearl, 1987). We will see in this section that fusion and bucket elimination perform exactly the same computations and are therefore equivalent. On the other hand, their description is based on two different but strongly connected graphical structures that bring the two important perspectives of local computation to the front: the structural buildup of inference problems that makes local computation possible and the concept that determines its complexity. This section introduces both algorithms separately and proves their equivalence such that later considerations about complexity apply to both schemes at once. In doing so, we provide a slightly more general introduction which is not limited to queries consisting of single variables, as it is often the case for variable elimination schemes.

3.2.1 The Fusion Algorithm

Let $\langle\Phi, D\rangle$ be a valuation algebra with the operation of projection replaced by variable elimination. We further assume an inference problem given by its knowledgebase $\{\phi_1, \ldots, \phi_m\} \subseteq \Phi$ and a single query $x \subseteq d(\phi) = d(\phi_1) \cup \ldots \cup d(\phi_m)$ where ϕ denotes the joint valuation $\phi = \phi_1 \otimes \ldots \otimes \phi_m$. The domain $d(\phi)$ must correspond to a set of variables $\{X_1, \ldots, X_n\}$ with $n = |d(\phi)|$. The fusion algorithm (Shenoy, 1992b; Kohlas, 2003) is based on the important property that eliminating a variable $X \in d(\phi)$ only affects the knowledgebase factors whose domains contain X. This is the statement of the following theorem:

Theorem 3.1 *Consider a valuation algebra $\langle \Phi, D \rangle$ and a factorization $\phi = \phi_1 \otimes \ldots \otimes \phi_m$ with $\phi_i \in \Phi$ and $1 \leq i \leq m$. For $X \in d(\phi)$ we have*

$$\phi^{-X} = \left(\bigotimes_{i:X \in d(\phi_i)} \phi_i \right)^{-X} \otimes \left(\bigotimes_{i:X \notin d(\phi_i)} \phi_i \right). \tag{3.7}$$

Proof: This theorem is a simple consequence of Lemma 3.1, Property 3. Since combination is associative and commutative, we may always write

$$\begin{aligned} \phi^{-X} &= \left[\left(\bigotimes_{i:X \in d(\phi_i)} \phi_i \right) \otimes \left(\bigotimes_{i:X \notin d(\phi_i)} \phi_i \right) \right]^{-X} \\ &= \left(\bigotimes_{i:X \in d(\phi_i)} \phi_i \right)^{-X} \otimes \left(\bigotimes_{i:X \notin d(\phi_i)} \phi_i \right). \end{aligned}$$

■

We therefore conclude that eliminating variable X only requires combining the factors whose domains contain X. According to the labeling axiom, this creates a new factor of domain

$$s_X = \bigcup_{i:X \in d(\phi_i)} d(\phi_i) \tag{3.8}$$

that generally is much smaller than the total domain $d(\phi)$ of the objective function. The same considerations now apply successively to all variables that do not belong to the query $x \subseteq d(\phi)$. This procedure is called fusion algorithm (Shenoy, 1992b) and can formally be defined as follows: let $\text{Fus}_Y(\{\phi_1, \ldots, \phi_m\})$ denote the set of valuations after fusing $\{\phi_1, \ldots, \phi_m\}$ with respect to variable $Y \in d(\phi_1) \cup \ldots \cup d(\phi_m)$,

$$\text{Fus}_Y(\{\phi_1, \ldots, \phi_m\}) = \{\psi^{-Y}\} \cup \{\phi_i : Y \notin d(\phi_i)\} \tag{3.9}$$

where

$$\psi = \left(\bigotimes_{i:X \in d(\phi_i)} \phi_i \right).$$

Using this notation, the result of Theorem 3.1 can equivalently be expressed as:

$$\phi^{-X} = \bigotimes \text{Fus}_X(\{\phi_1, \ldots, \phi_m\}). \tag{3.10}$$

By repeated application and the commutativity of variable elimination we finally obtain the fusion algorithm for the computation of a query $x \subseteq d(\phi)$. If $\{Y_1, \ldots, Y_k\} = d(\phi) - x$ are the variables to be eliminated, we have

$$\phi^{\downarrow x} = \phi^{-\{Y_1, \ldots, Y_k\}} = \bigotimes \text{Fus}_{Y_k}(\ldots(\text{Fus}_{Y_1}(\{\phi_1, \ldots, \phi_m\}))\ldots). \tag{3.11}$$

This is a generic algorithm for solving the single-query inference problem over an arbitrary valuation algebra as introduced in Chapter 1. Algorithm 3.1 provides a summary of the complete fusion process which stands out by its simplicity and compactness. Also, Example 3.1 expands the complete computation of fusion for the inference problem of Instance 2.1.

Algorithm 3.1 The Fusion Algorithm

```
input:  $\{\phi_1, \ldots, \phi_m\}$,  $x \subseteq d(\phi)$   output:  $(\phi_1 \otimes \ldots \otimes \phi_m)^{\downarrow x}$

begin
  $\Psi := \{\phi_1, \ldots, \phi_m\}$;
  for each $Y \in d(\phi) - x$ do
    $\Gamma := \{\phi_i \in \Psi : Y \in d(\phi_i)\}$;
    $\psi := \bigotimes \Gamma$;
    $\Psi := (\Psi - \Gamma) \cup \{\psi^{-Y}\}$;
  end;
  return $\bigotimes \Psi$;
end
```

Example 3.1 *To exemplify the fusion algorithm, we reconsider the inference problem of the Bayesian network from Instance 2.1. The knowledgebase consists of eight valuations with domains: $d(\phi_1) = \{A\}, d(\phi_2) = \{A, T\}, d(\phi_3) = \{L, S\}, d(\phi_4) = \{B, S\}, d(\phi_5) = \{E, L, T\}, d(\phi_6) = \{E, X\}, d(\phi_7) = \{B, D, E\}$ and $d(\phi_8) = \{S\}$. We choose $\{A, B, D\}$ as query and eliminate all variables from $\{E, L, S, T, X\}$ in the elimination sequence (X, S, L, T, E):*

Elimination of variable X:

$$\phi^{-\{X\}} = \phi_6^{-X} \otimes \bigotimes\{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_7, \phi_8\}$$

Elimination of Variable S:

$$\phi^{-\{X,S\}} = \phi_6^{-X} \otimes (\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S} \otimes \bigotimes\{\phi_1, \phi_2, \phi_5, \phi_7\}$$

Elimination of Variable L:

$$\phi^{-\{X,S,L\}} = \phi_6^{-X} \otimes ((\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S} \otimes \phi_5)^{-L} \otimes \bigotimes\{\phi_1, \phi_2, \phi_7\}$$

Elimination of Variable T:

$$\phi^{-\{X,S,L,T\}} = \phi_6^{-X} \otimes (((\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S} \otimes \phi_5)^{-L} \otimes \phi_2)^{-T} \otimes \bigotimes\{\phi_1, \phi_7\}$$

Elimination of Variable E:

$$\phi^{-\{X,S,L,T,E\}} = (\phi_6^{-X} \otimes (((\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S} \otimes \phi_5)^{-L} \otimes \phi_2)^{-T} \otimes \phi_7)^{-E} \otimes \phi_1$$

We next present a graphical representation of the fusion process proposed by (Shafer, 1996). For this purpose, we maintain a set of domains that is initialized with the domains of all knowledgebase factors $l = \{d(\phi_1), \ldots, d(\phi_m)\}$. In parallel, we build up a labeled graph (V, E, λ, D) which is assumed to be empty at the beginning, i.e. $V = \emptyset$ and $E = \emptyset$. When variable X_i is eliminated during the fusion process, we remove all domains from the set l that contain variable X_i and compute their union

$$s_i \quad = \quad \bigcup_{s \in l : X_i \in s} s. \tag{3.12}$$

We then add the new domain $s_i - \{X_i\}$ to the set l which altogether updates to

$$(l - \{s \in l : X_i \in s\}) \cup \{s_i - \{X_i\}\}.$$

This corresponds to the i-th step of the fusion algorithm where variable $X_i \in d(\phi) - x$ is eliminated. Then, a new node i is constructed with label $\lambda(i) = s_i$. This node is tagged with a color and added to the graph. We then go through all other colored graph nodes: if the label of such a node $v \in V$ contains the variable X_i, then its color is removed and a new edge $\{i, v\}$ is added to the graph. This whole process is repeated for all variables $d(\phi) - x$ to be eliminated. Finally, one last finalization step has to be performed, which corresponds to the combination in equation (3.11): at the end of the variable elimination process, the set l will not be empty. It consists either of the query variables that are never eliminated, or, if the query is empty, it contains an empty set. We therefore add a last, colored node to the graph whose domain corresponds to the union of all remaining elements in the domain set. Then, we link all nodes that are still tagged with a color to this new node and remove all node colors. Algorithm 3.2 gives a summery of the whole construction process and returns a labeled graph (V, E, λ, D). Since all node labels created during the graph construction process consist of the variables from $d(\phi)$, we may always set $D = \mathcal{P}(d(\phi))$. The yet mysterious name of this algorithm will be explained subsequently.

Algorithm 3.2 Join Tree Construction

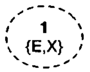
```
input:  $\{d(\phi_1), \ldots, d(\phi_m)\}$,  $x \subseteq d(\phi)$   output:  $(V, E, \lambda)$

begin
  $V$ := $\emptyset$;  $E$ := $\emptyset$;
  $l$ := $\{d(\phi_1), \ldots, d(\phi_m)\}$;

  for each $X_i \in d(\phi) - x$ do      // Start graph construction process.
    $s_i$ := $\bigcup\{s \in l : X_i \in s\}$;
    $l$ := $(l - \{s \in l : X_i \in s\}) \cup \{s_i - \{X_i\}\}$;
    $i$ := new vertex; $\lambda(i)$ := $s_i$; $color(i)$ := $true$;
    for each $j \in V$ do
      if $X_i \in \lambda(j)$ and $color(j) = true$ do
        $E$ := $E \cup \{\{i, j\}\}$;
        $color(j)$ := $false$;
      end;
    end;
    $V$ := $V \cup \{i\}$;
  end;
```

```
    $i$ := new vertex; $\lambda(i)$ := $\bigcup l$;          // Finalization step.
    for each $j \in V$ do
      if $color(j) = true$ do
        $E$ := $E \cup \{\{i, j\}\}$;
        $color(j) := false$;
      end;
    end;
    $V$ := $V \cup \{i\}$;
    return $(V, E, \lambda)$;
end
```

Example 3.2 *We give an example of the graphical representation of the fusion algorithm based on the factor domains of the Bayesian network example from Instance 2.1. At the beginning, we have $l = \{\{A\}, \{A, T\}, \{L, S\}, \{B, S\}, \{E, L, T\}, \{E, X\}, \{B, D, E\}, \{S\}\}$. The query of this inference problem is $\{A, B, D\}$ and the variables $\{E, L, S, T, X\}$ have to be eliminated. We choose the elimination sequence (X, S, L, T, E) as in Example 3.1 and proceed according to the above description. Colored nodes are represented by a dashed border.*

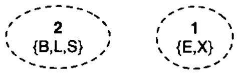
- *Elimination of variable X:*

$$l : \{A\}, \{A, T\}, \{L, S\}, \{B, S\}, \{E, L, T\}, \{E, X\}, \{B, D, E\}, \{S\}$$



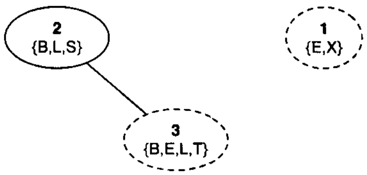
- *Elimination of variable S:*

$$l : \{A\}, \{A, T\}, \{L, S\}, \{B, S\}, \{E, L, T\}, \{E\}, \{B, D, E\}, \{S\}$$



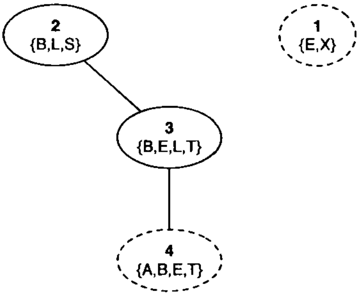
- *Elimination of variable L:*

$$l : \{A\}, \{A, T\}, \{B, L\}, \{E, L, T\}, \{E\}, \{B, D, E\}$$



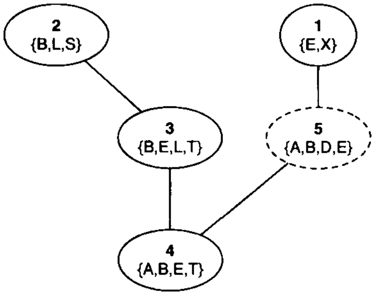
- *Elimination of variable T:*

$$l : \{A\}, \{A, T\}, \{B, T, E\}, \{E\}, \{B, D, E\}$$



- *Elimination of variable E:*

$$l : \{A\}, \{A, B, E\}, \{E\}, \{B, D, E\}$$



- *End of variable elimination: after the elimination of variable E, only the last added node $\{A, B, E, D\}$ is colored. This, however, is due to the particular structure of our knowledgebase, and it is quite possible that multiple colored nodes still exist after the elimination of the last variable. We next enter the finalization step and add a last node whose label corresponds to the union of the remaining elements in the domain list, i.e. $\{A\}$ and $\{A, B, D\}$. This node is then connected to the colored node and all colors are removed. The result of the construction process is shown in Figure 3.1.*

3.2.2 Join Trees

Let us investigate the graphical structure that results from the fusion process in more detail. We first remark that each node can be given the number of the eliminated variable. For example, in Figure 3.1 the node labeled with $\{B, E, L, T\}$ has number 3 because it was introduced during the elimination of the third variable that corresponds to L in the elimination sequence. This holds for all nodes except the one added in the finalization step. We therefore say that variable X_i has been eliminated in node i for $1 \leq i \leq |d(\phi) - x|$. If we further assign number $|d(\phi) - x| + 1$ to the finalization node, we see that if $i < j$ implies that node j has been introduced after node i.

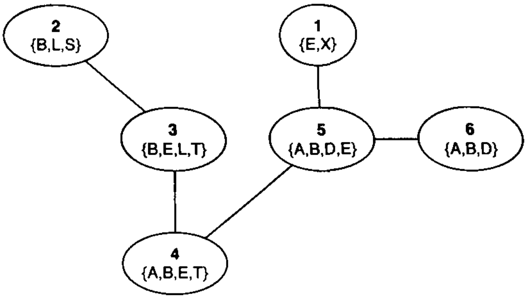


Figure 3.1 Finalization of the graphical fusion process.

Lemma 3.2 *At the end of the fusion algorithm the graph G is a tree.*

Proof: During the graph construction process, only colored nodes are connected, which always causes one of them to lose its color. It is therefore impossible to create cycles. But it may well be that different (unconnected) colored nodes exist after the variable elimination process. Each of them is part of an independent tree, and each tree contains exactly one colored node. In the finalization step, we add a further node and connect it with all the remaining colored nodes. The total graph G must therefore be a tree to which all nodes are connected. ∎

This tree further satisfies the *running intersection property* which is sometimes also called *Markov property* or simply *join tree property*. Accordingly, such trees are named *join trees, junction trees* or *Markov trees.*

Definition 3.1 *A labeled tree (V, E, λ, D) satisfies the* running intersection property *if for two nodes $i, j \in V$ and $X \in \lambda(i) \cap \lambda(j)$, $X \in \lambda(k)$ for all nodes k on the path between i and j.*

Example 3.3 *Figure 3.2 reconsiders the tree G_2 from Example 2.1 equipped with a labeling function λ for $r = \{A, B, C, D\}$. The labels are: $\lambda(1) = \{A, C, D\}$, $\lambda(2) = \{A, D\}$, $\lambda(3) = \{D\}$, $\lambda(4) = \{C\}$ and $\lambda(5) = \{A, B\}$. This labeled tree is not a join tree since $C \in \lambda(1) \cap \lambda(4)$ but $C \notin \lambda(3)$. However, adding variable C to node label $\lambda(3)$ will result in a join tree.*

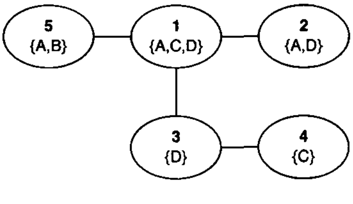


Figure 3.2 This labeled tree is not a join tree since variable C is missing in node 3.

Lemma 3.3 *At the end of the fusion algorithm the graph G is a join tree.*

Proof: We already know that G is a tree. Select two nodes s' and s'' and $X \in s' \cap s''$. We distinguish two cases: If X does not belong to the query, then X is eliminated in some node. We go down from s' along to later nodes, until we arrive at the node where X is eliminated. Let k' be the number of this node. Similarly, we go down from s'' to later nodes until we arrive at the elimination node of X. Assume the number of this node to be k''. But X is eliminated in exactly one node. So $k' = k''$ and the path from s' to s'' goes from s' to $k' = k''$ and from there to s''. X belongs to all nodes of the paths from s' to $k' = k''$ and s'' to $k' = k''$, hence to all nodes of the path from s' to s''. The second possibility is that X belongs to the query. Then, X has never been eliminated but belongs to the label of the node added in the finalization step. Thus, the two paths meet in this node and the same argument applies. ■

Example 3.4 *In Figure 3.1, variable E is eliminated in node* 5 *but appears already in node* 3*, therefore it must also belong to the label of node* 4*. On the other hand, variable B belongs to the query and is therefore part of the node label introduced in the finalization step. This is node number* 6*. But variable B also appears in node* 2 *and therefore does so in the nodes* 3 *to* 5*.*

3.2.3 The Bucket Elimination Algorithm

We reconsider the same setting for the introduction of bucket elimination: let $\langle \Phi, D \rangle$ be a valuation algebra with the operation of projection replaced by variable elimination. The knowledgebase of the inference problem is $\{\phi_1, \ldots, \phi_m\} \subseteq \Phi$ and $x \subseteq d(\phi) = \{X_1, \ldots, X_n\}$ denotes the single query. Similarly to the fusion algorithm, the bucket elimination scheme (Dechter, 1999) is based on the important observation of Theorem 3.1 that eliminating a variable only affects the valuations whose domain contain this variable. We again choose an elimination sequence $(Y_1, \ldots, Y_k)$ for the variables in $d(\phi) - x$ and imagine a *bucket* for each of these variables. These buckets are ordered with respect to the variable elimination sequence. We then partition the valuations in the knowledgebase as follows: all valuations whose domain contains the first variable Y_1 are placed in the first bucket, which we subsequently denote by $bucket_1$. This process is repeated for all other buckets, such that each valuation is placed in the first bucket that mentions one of its domain variables. It is not possible that a valuation is placed in more than one bucket, but there may be valuations whose domain is a subset of the query and therefore do not fit in any bucket. Bucket elimination then proceeds as follows: at step $i = 1, \ldots, k-1$, $bucket_i$ is processed by combining all valuations in this bucket, eliminating variable Y_i from the result and placing the obtained valuation in the first of the remaining buckets that mentions one of its domain variables. At then end, we combine all valuations contained in $bucket_k$ with the remaining valuations of the knowledgebase for which no bucket was found in the initialization process.

In the first step of the bucket elimination process, $bucket_1$ computes

$$\left(\bigotimes_{i:Y_1 \in d(\phi_i)} \phi_i \right)^{-Y_1}$$

and moves the result into the next bucket that contains one of its domain variables. This corresponds exactly to equation (3.9) whereas

$$\left\{\phi_i : Y_1 \notin d(\phi_i)\right\} \quad = \quad \bigcup_{i=2}^{k} bucket_i \cup \left\{\phi_i : d(\phi_i) \subseteq x\right\}.$$

In other words, the first step of the bucket elimination process corresponds to the first step of the fusion algorithm, and the same holds also naturally for the remaining steps. Bucket elimination therefore computes equation (3.11) that implies its correctness and the equivalence to the fusion algorithm.

Theorem 3.2 *Fusion and bucket elimination with identical elimination sequences perform exactly the same computations.*

Example 3.5 *We repeat the computations of Example 3.1 and 3.2 from the perspective of bucket elimination. The knowledgebase consists of eight valuations with domains:* $d(\phi_1) = \{A\}, d(\phi_2) = \{A,T\}, d(\phi_3) = \{L,S\}, d(\phi_4) = \{B,S\}, d(\phi_5) = \{E,L,T\}, d(\phi_6) = \{E,X\}, d(\phi_7) = \{B,D,E\}$ *and* $d(\phi_8) = \{S\}$. *The query is* $\{A,B,D\}$. *We choose the same elimination sequence* (X,S,L,T,E), *construct a bucket for each variable in the elimination sequence and partition the valuations accordingly. Since* $d(\phi_1) \subseteq x$, ϕ_1 *is not contained in any bucket.*

Initialization:

$bucket_X : \phi_6$

$bucket_S : \phi_3,\ \phi_4,\ \phi_8$

$bucket_L : \phi_5$

$bucket_T : \phi_2$

$bucket_E : \phi_7$

Elimination of bucket X:

$bucket_S : \phi_3,\ \phi_4,\ \phi_8$

$bucket_L : \phi_5$

$bucket_T : \phi_2$

$bucket_E : \phi_7,\ \phi_6^{-X}$

Elimination of bucket S:

$$bucket_L : \phi_5,\ (\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S}$$

$$bucket_T : \phi_2$$

$$bucket_E : \phi_7,\ \phi_6^{-X}$$

Elimination of bucket L:

$$bucket_T : \phi_2,\ (\phi_5 \otimes (\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S})^{-L}$$

$$bucket_E : \phi_7,\ \phi_6^{-X}$$

Elimination of bucket T:

$$bucket_E : \phi_7,\ \phi_6^{-X},\ (\phi_2 \otimes (\phi_5 \otimes (\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S})^{-L})^{-T}$$

Elimination of bucket E:

$$(\phi_7 \otimes \phi_6^{-X} \otimes (\phi_2 \otimes (\phi_5 \otimes (\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S})^{-L})^{-T})^{-E}$$

Combining this with valuation ϕ_1 confirms the result of Example 3.2.

Similar to the fusion algorithm, we give a graphical representation of the bucket elimination process. We first create a graph node for each bucket and label it with the corresponding variable. If during the elimination of $bucket_X$ a valuation is added to $bucket_Y$, then an edge $\{X, Y\}$ is added to the graph. Since the elimination of each bucket generates exactly one new valuation that is added to the bucket of a variable which follows later in the elimination sequence, the resulting graph will always be a tree called *bucket-tree*. The bucket-tree of Example 3.5 is shown in Figure 3.3.

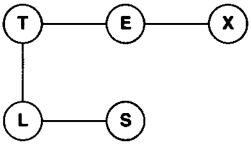


Figure 3.3 The bucket-tree of Example 3.5.

Based on the bucket-tree, we can derive the join tree that underlies the bucket elimination process. We revisit each bucket at the time of its elimination and compute the union domain of all valuations contained in this bucket. This domain is assigned as label to the corresponding node in the bucket tree. Finally, we add a new node with the query of the inference problem as label to the modified bucket-tree and connect this node with the node that belongs to the last variable in the elimination sequence. It follows from Theorem 3.2 and Lemma 3.3 that the resulting graph is a join tree, i.e. we obtain the same join tree as for the fusion algorithm.

Example 3.6 *At the time of elimination, the buckets of Example 3.5 contain valuations over the following variables:*

$$bucket_X : \{E, X\}$$

$$bucket_S : \{B, L, S\}$$

$$bucket_L : \{B, E, L, T\}$$

$$bucket_T : \{A, B, E, T\}$$

$$bucket_E : \{A, B, D, E\}$$

We next assign these domains as labels to the corresponding nodes in the bucket-tree. Finally, we create a new node with the query domain $\{A, B, D\}$ and connect it to the node that refers to the last variable in the elimination sequence. The resulting graph is shown in Figure 3.4. If we further replace the variables from the bucket-tree by their position in the elimination sequence and assign number $k + 1$ to the node with the query domain, we obtain the join tree of Figure 3.1.

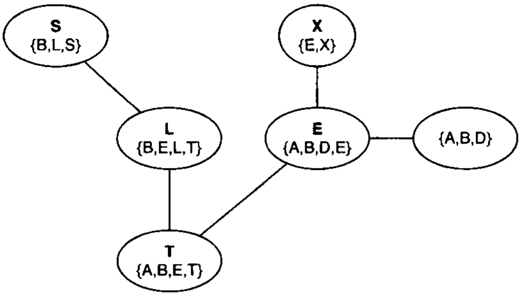


Figure 3.4 The join tree obtained from the bucket-tree of Figure 3.3.

Our presentation of the bucket elimination algorithm differs in two points from the usual way that it is introduced. First, we consider arbitrary query sets with possibly more than one variable. Consequently, there may be valuations that cannot be assigned to any bucket as shown in Example 3.5, and this makes it necessary to add the additional query node when deriving the join tree from the bucket-tree. The second difference concerns the elimination sequence. In the literature, buckets are normally processed in the inverse order to the elimination sequence. This however is only a question of terminology. Before we turn towards a complexity analysis of the fusion or bucket elimination algorithm, we reconsider the computational task of Example I.2 used to illustrate the potential benefit of local computation in the introduction of this book.

Example 3.7 *The formalism used in Example I.2 in the introduction consists of mappings from configurations to real numbers. Combination refers to point-wise* *multiplication and projection to summing up the values of the eliminated variables. In other words, this formalism corresponds to the valuation algebra of arithmetic potentials. The knowledgebase is given by the set of function* $\{f_1, \ldots, f_{100}\}$ *and the query is* $\{X_{101}, X_{102}\}$. *We recall that an explicit computation of the objective function is impossible. Instead, we apply the fusion or bucket elimination algorithm to this inference problem with the elimination sequence* $(X_1, \ldots, X_{100})$. *This leads to the join tree shown in Figure 3.5, and the executed computations correspond to equation (I.3). Thus, the domains of all intermediate factors that occur during the computations contain at most three variables.*

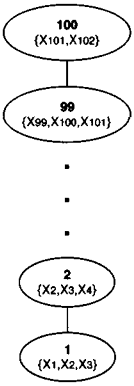


Figure 3.5 The join tree for Example I.2.

3.2.4 First Complexity Considerations

Making a statement about the complexity of local computation for arbitrary valuation algebras is impossible since we do not have information about the form of valuations. Remember, the valuation algebra framework describes valuations only by their operational behaviour. We therefore restrict ourselves first to a particular class of valuations and show later how the corresponding complexity considerations can be generalized. More concretely, we focus on valuations that are representable in tabular form such as indicator functions from Instance 1.1 or arithmetic potentials from Instance 1.3. In fact, we shall study in Chapter 5 the family of semiring valuation algebras which all share this common structure of mapping configurations to specific values. Thus, given the universe of variables with finite frames, we denote by $d \in \mathbb{N}$ the size of the largest variable frame. Then, the space (subsequently called *weight*) of a valuation ϕ with domain $d(\phi) = s$ is bounded by $\mathcal{O}(d^{|s|})$ and similar statements can be made about the time complexity of combination and projection.

When during the fusion process variable $X_i \in d(\phi) - x$ is eliminated, all valuations are combined whose domain contains this variable according to equation (3.9). Then, variable X_i is eliminated. Thus, if m_i valuations contain variable X_i before iteration i of the fusion algorithm, its elimination requires $m_i - 1$ combinations and one variable elimination (projection). The domain s_i of the valuation resulting from the combination sequence is given in equation (3.12). We may therefore say that during the elimination of variable X_i, the time complexity of each operation is bounded by $\mathcal{O}(d^{|s_i|})$ which gives us the following total time complexity for the fusion or bucket elimination process:

$$\mathcal{O}\left(\sum_{i \in d(\phi)-x} m_i \cdot d^{|s_i|} \right).$$

In the graphical representation of the fusion algorithm, the elimination of variable X_i introduces a new join tree node with label $\lambda(i) = s_i$. The size of these labels are bounded by the largest node label $\omega = \max_{i \in V} |\lambda(i)|$. A related measure called *treewidth* will be introduced in Definition 3.2 below. Further, we observe that the fusion process reinserts a valuation into its current knowledgebase after the elimination of each variable. This valuation was called ψ in equation (3.9). We may therefore say that $\sum m_i \leq m + |d(\phi) - x| \leq m + |d(\phi)|$ where m denotes the size of the original knowledgebase of the inference problem and x the query. Putting things together, we obtain the following simplified expression for the time complexity:

$$\mathcal{O}\left(\left(m + |d(\phi)|\right) \cdot d^{\omega} \right). \tag{3.13}$$

Slightly better is the space complexity. During the elimination of variable X_i a sequence of combinations is computed that creates an intermediate result with domain s_i, from which the variable X_i is eliminated. In case of valuations with a tabular representation, the intermediate result does not need to be computed explicitly. This will be explained in more detail in Section 5.6. Here, we content ourselves with the rather intuitive idea that each value (or table entry) of this intermediate valuation can be computed separately and directly projected. This omits the computation of the complete intermediate valuation and directly gives the result after eliminating variable X_i from the combination. The domain size of this valuation is $s_i - \{X_i\} \leq \omega - 1$, and its size is therefore bounded by $\mathcal{O}(d^{\omega-1})$. Further, we have just seen that the number of eliminations is at most $|d(\phi)|$ which gives us a total space complexity of

$$\mathcal{O}\left(|d(\phi)| \cdot d^{\omega-1} \right). \tag{3.14}$$

It is important to remark that both complexities depend on the shape of the join tree, respectively on its largest node label called treewidth. This measure varies under different elimination sequences as illustrated in the following example.

Example 3.8 *Figure 3.6 shows two different join trees for the domain list of Example 3.2 obtained by varying the elimination sequence. In order to get comparable results, we take the empty set as query and eliminate all variables using Algorithm 3.2. The left-hand join tree is obtained from the elimination sequence* (T, S, E, L, B, A, X, D)

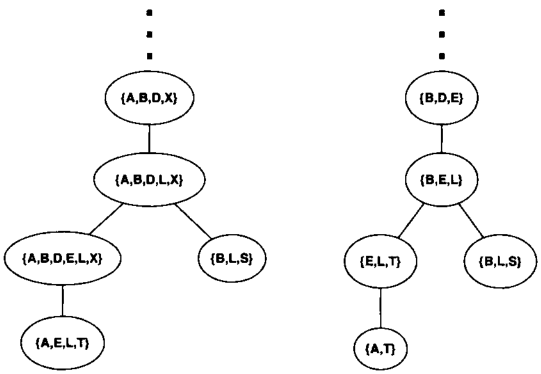


Figure 3.6 Different elimination sequences produce different join trees and therefore influence the size of the largest node label.

and has $\omega = 6$. The right-hand join tree has $\omega = 3$ due to the clever choice of the elimination sequence (A, T, S, L, B, E, X, D). After a certain number of variable elimination steps only a single valuation remains in both examples from which one variable after the other is eliminated. This uninteresting part is omitted in the join trees of Figure 3.6. A third join tree with $\omega = 4$ can be obtained by eliminating the remaining variables in Example 3.2. This corresponds to the elimination sequences that start with (X, S, L, T, E)

The complexity of solving an inference problem with the fusion algorithm therefore depends on the variable elimination sequence that produces a join tree whose largest node label should be as small as possible. This follows from equation (3.13) and (3.14). In order to understand how elimination sequences and the treewidth measure are related, we consider the primal graph of the inference problem introduced in Section 2.1. Since a primal graph contains a node for every variable, an elimination sequence can be represented by a particular node numbering. Here, we number the nodes with respect to the inverse elimination sequence as shown on the left-hand side of Figure 3.7. Such graphs are called *ordered graphs*, and we define the *width of a node in an ordered graph* as the number of its earlier neighbors. Likewise, we define the *width of an ordered graph* as the maximum node width. Next, we construct the *induced graph* of the ordered primal graph by the following procedure: process in the reversed order of the node numbering (i.e. according to the elimination sequence) and add edges to ensure for each node that its earlier neighbors are directly connected. We define the *induced width of an elimination sequence* (or ordered graph) as the width of its induced graph and refer to the *induced width of a graph* ω^* as the minimum width over all possible elimination sequences (orderings) (Dechter & Pearl, 1987). Since a primal graph is associated with each inference problem, we may also speak about the *induced width of an inference problem.*

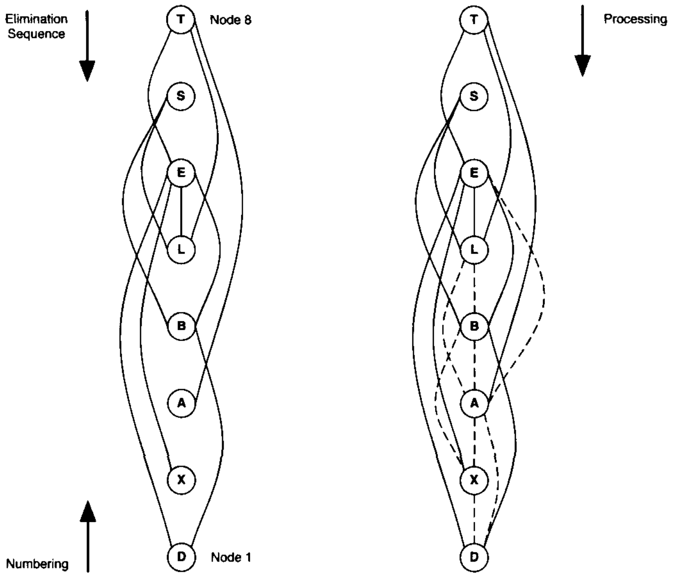


Figure 3.7 A primal graph and its induced graph.

Example 3.9 *The left-hand part of Figure 3.7 shows the primal graph of Example 3.2 ordered with respect to the elimination sequence (T, S, E, L, B, A, X, D). Node E has width 4 since four neighbors posses a smaller node number. Then, the right-hand part of Figure 3.7 shows its induced graph. Here, node E has 5 neighbors $\{A, B, D, L, X\}$ with a smaller number which also determines the width of this elimination sequence. Observe that this corresponds to the largest node label of the join tree on the left-hand side of Figure 3.6, if we add the variable E that has been eliminated in this node.*

The following theorem has been proved by (Arnborg, 1985):

Theorem 3.3 *The largest node label in a join tree equals the induced width of the elimination sequence plus one.*

This relationship allows us subsequently to focus only on join trees when dealing with complexity issues. It is then common to use a second but equivalent terminology called *treewidth*, which has been introduced by (Robertson & Seymour, 1983; Robertson & Seymour, 1986):

Definition 3.2 *The* treewidth of a join tree *(V, E, λ, D) is given by*

$$\max_{i \in V} |\lambda(i)| - 1, \tag{3.15}$$

and we refer to the minimum treewidth over all join trees created from all possible variable elimination sequences as the treewidth of the inference problem.

Since treewidth and induced width are equal according to the above theorem, we reuse the notation ω^* for the treewidth of an inference problem. It is very important to distinguish properly between the two notions treewidth of a join tree and treewidth of an inference problem. The first refers to a concrete join tree whereas the second captures the best join tree over all possible elimination sequences of a given inference problem. The decrement in equation (3.15) ensures that inference problems whose primal graphs are trees have treewidth one. Besides induced width and treewidth, there are other equivalent characterization in the literature. Perhaps most common among them is the notion of a *partial k-tree* (van Leeuwen, 1990).

Example 3.10 *The value ω from Example 3.8 refers to the incremented treewidth of the join tree. We therefore have in this example three join trees of treewidth* 2, 3 *and* 5 *and conclude that the treewidth of the medical inference problem with empty query is at most* 2. *In fact, it is equal to* 2 *because one of the knowledgebase factors has a domain of three variables, which makes it impossible to get a lower treewidth.*

Returning to the task of identifying the complexity of the fusion and bucket elimination algorithm, we obtain a complexity bound for the solution of a single-query inference problem by including the treewidth ω^* of the latter into equation (3.13) and (3.14):

$$\mathcal{O}\left((m + |d(\phi)|) \cdot d^{\omega^*+1}\right) \quad \text{and} \quad \mathcal{O}\left(|d(\phi)| \cdot d^{\omega^*}\right). \tag{3.16}$$

This is called a *parameterized complexity* (Downey & Fellows, 1999) with the treewidth of the inference problem as parameter. Naturally, the question arises how the minimum treewidth over all possible variable elimination sequences can be found, the more so as it determines the complexity of applying the fusion and bucket elimination algorithm. Concerning this important question we only have bad news. It has been shown by (Arnborg *et al.*, 1987) that finding the smallest treewidth of a graph is NP-complete. In fact, the task of deciding if the primal graph of the inference problem has a treewidth below a certain constant k can be performed in $\mathcal{O}(|d(\phi)| \cdot g(k))$ where $g(k)$ is a very bad exponential function in k (Bodlaender, 1998; Bodlaender, 1993). Fortunately, there are good heuristics for this task that deliver join trees with reasonable treewidths as we will see in Section 3.7. It is thus important to distinguish the two ways of specifying the complexity of local computation schemes. Given a concrete join tree or an elimination sequence with treewidth ω, we insert this value into equation (3.16) and obtain the achieved complexity of a concrete run of the fusion algorithm. On the other hand, using the treewidth ω^* of the inference problem in these formulae denotes the best complexity over all possible join trees or variable elimination sequences that could be achieved for the solution of this inference problem by the fusion or bucket elimination algorithm. But since we are generally not able to exactly determine this value, this often remains an unsatisfied wish. However, it is nevertheless common to specify the complexity of local computation schemes by the treewidth of the inference problem and we will also go along with this.

3.2.5 Some Generalizing Complexity Comments

At the beginning of our complexity considerations we limited ourselves to a family of valuation algebras whose elements are representable in tabular form. These formalisms share the property that the size of valuations grows exponentially with the size of their domains. Thus, the space of a valuation ϕ with domain $s = d(\phi)$ is bounded by $\mathcal{O}(d^{|s|})$ where d denotes the size of the largest variable frame. Similar statements were made about the time complexity of combination and projection. This view is too restrictive, since there are many other valuation algebra instances that do not keep with this estimation. On the other hand, there are also examples where it is unreasonable to measure time and space complexity by the same function as it has been done in equation (3.16). Examples of such formalisms will be given in Chapter 6. However, a general assumption we are allowed to make is that the space of a valuation and the execution time for the operations shrink under projection. Remember, the idea of local computation is to confine the domain size of intermediate factors during the computations. If this assumption could not be made, local computation would hardly be a reasonable approach.

Definition 3.3 *Let $\langle\Phi, D\rangle$ be a valuation algebra. A function $\omega : \Phi \rightarrow \mathbb{N}_0$ is called* weight function *if for all $\phi \in \Phi$ and $x \subseteq d(\phi)$ we have $\omega(\phi) \geq \omega(\phi^{\downarrow x})$.*

Weight functions are still too general to be used for complexity considerations. In many cases however, we can give a predictor for the weight function which is only based on the valuation's domain. Such *weight predictable* valuation algebras share the property that the weight of a valuation can be estimated from its domain only.

Definition 3.4 *A weight function ω of a valuation algebra $\langle\Phi, D\rangle$ is called* weight predictor *if there exists a function $f : D \rightarrow \mathbb{N}_0$ such that for all $\phi \in \Phi$*

$$\omega(\phi) \in \mathcal{O}\Big(f\big(d(\phi)\big)\Big).$$

We give some examples of weight functions and weight predictors for the valuation algebras introduced in Chapter 1:

Example 3.11

- *A weight predictor for the valuation algebra of arithmetic potentials that also applies to indicator functions and all formalisms of Section 5.3 has already been used in equation (3.16):*

$$\omega(\phi) \quad = \quad \prod_{X \in s} |\Omega_X| \;\in\; \mathcal{O}\Big(d^{|s|}\Big) \tag{3.17}$$

where $s = d(\phi)$ denotes the domain of valuation ϕ and

$$d \quad = \quad \max_{X \in d(\phi)} |\Omega_X|.$$

the largest variable frame. Observe that in this particular case, we could also use the weight function directly as weight predictor since it depends only on the valuation's domain.

- *A reasonable and often used weight function for relations is $\omega(\phi) = |d(\phi)| \cdot card(\phi)$, where $card(\phi)$ stands for the number of tuples in ϕ. Regrettably, this is not a weight predictor itself since the number of tuples cannot be deduced from the relation's domain.*

- *Finally, a weight predictor for the valuation algebra of set potentials and all formalisms of Section 5.7 is*

$$\omega(\phi) \in O\left(2^{2^{|s|}}\right). \tag{3.18}$$

Observe that we presuppose binary variables in this formula.

Subsequently, we will use weight predictors for the analysis of the time and space complexity of local computation. This allows us to make more general complexity statements that can then be specialized to a specific formalism by choosing an appropriate weight predictor. Further, we use two different weight predictors f and g for time and space complexity. This accounts for the observation that time and space complexity cannot necessarily be estimated by the same function, but both satisfy the fundamental property that they are non-increasing under projection. We obtain for the general time complexity of the fusion and bucket elimination algorithm:

$$O\Big((m + |d(\phi)|) \cdot f(\omega^* + 1)\Big). \tag{3.19}$$

This follows directly from equation (3.16). Remember that the assumption of valuations with a tabular representation allowed us to derive a lower space complexity in equation (3.16). This optimization cannot be performed for arbitrary valuation algebras. For the elimination of variable X_i we therefore need to compute the combinations first, which results in an intermediate factor with domain size $|\lambda(i)| \leq \omega^* + 1$. Then, the variable X_i can be eliminated and the result is added to the knowledgebase. Altogether, this gives us the following general space complexity of the fusion and bucket elimination algorithm:

$$O\Big(g(\omega^* + 1) + |d(\phi)| \cdot g(\omega^*)\Big). \tag{3.20}$$

Clearly, for the case of applying the fusion algorithm to arithmetic potentials, indicator functions and all other formalism of Section 5.3, we may choose the same weight predictor of equation (3.17) for both time and space complexity. Moreover, the space complexity can then be improved to the one given in equation (3.16). This shows the difficulties of making general complexity statements for local computation applied to unspecified valuation algebras.

3.2.6 Limitations of Fusion and Bucket Elimination

The fusion or bucket elimination algorithm is a generic local computation scheme to solve single-query inference problems in a simple and natural manner. Nevertheless, it has some drawbacks compared to other local computation procedures that will be introduced below. Here, we address two important points of criticism:

- The fusion or bucket elimination algorithm can be applied to each formalism that satisfies the valuation algebra axioms from Section 1.1. But we have also seen in Appendix A.2 of Chapter 1 that there exists a more general definition of the axiomatic framework which is based on general lattices instead of variable systems. Fusion and bucket elimination require variable elimination and can therefore not be generalized to valuation algebras on arbitrary lattices.

- Time and space complexity of the fusion algorithm are bounded by the treewidth of the inference problem, which makes the application difficult when join trees have large labels. Experiments from constraint programming (Larrosa & Morancho, 2003) show that the space requirement is particularly sensitive, for which reason inference is often combined with *search methods* that only require linear space. However, search methods need more structure than offered by the valuation framework. We refer to (Rossi *et al.*, 2006) for a survey of such trading or hybrid methods in constraint programming. Looking at equation (3.20), we observe that the function arguments of the two terms distinguish in a constant number. We will see later that this is a consequence of the particular join trees that emerge from the fusion and bucket elimination process. Further, we know that the first term vanishes when dealing with particular valuation algebras. It will be shown in Section 5.3 that this also includes constraint systems. To make local computation more space efficient for such formalisms, we therefore aim at reducing the argument of the second term in equation (3.20).

In the subsequent sections, we present an alternative picture of local computation that finally leads to a second algorithm for the solution of single-query inference problems that does not depend on variables anymore and which may also provide a better space performance. The essential difference with respect to the fusion algorithm is that the join tree construction is decoupled from the actual inference process. Instead, a suitable join tree for the current inference problem is initially constructed, and local computation is then described as a message-passing process where nodes act as virtual processors and collaborate by exchanging messages. The key advantage of this approach is that we no longer depend on the very particular join trees that are obtained from the fusion algorithm. To study the requirements for a join tree to serve as local computation base, we first introduce a specialized class of valuation algebras that provide neutral information pieces. It will later be shown in Section 3.10 that the existence of such neutral elements is not mandatory for the description of local computation as message-passing scheme. But if they exist, the discussion simplifies considerably. Moreover, neutral elements are also interesting in other contexts that will be discussed in later parts of this book.

3.3 VALUATION ALGEBRAS WITH NEUTRAL ELEMENTS

Following the view of a valuation algebra as a generic representation of knowledge or information, some instances may exist that contain pieces that express vacuous information. In this case, such a *neutral element* must exist for every set of questions $s \in D$ and must therefore be contained in every subsemigroup Φ_s of Φ. If we then combine those pieces with already existing knowledge of the same domain, we do not gain new information. Hence, there is an element $e_s \in \Phi_s$ for every subsemigroup Φ_s such that $\phi \otimes e_s = e_s \otimes \phi = \phi$ for all $\phi \in \Phi_s$. We further add a neutrality axiom to the valuation algebra system which states that a combination of neutral elements leads to a neutral element with respect to the union domain.

Definition 3.5 *A valuation algebra $\langle \Phi, D \rangle$ has neutral elements, if for every domain $s \in D$ there exists an element $e_s \in \Phi_s$ such that $\phi \otimes e_s = e_s \otimes \phi = \phi$ for all $\phi \in \Phi_s$. These elements must satisfy the following property:*

(A7) Neutrality: *For $x, y \in D$,*

$$e_x \otimes e_y \quad = \quad e_{x \cup y}. \tag{3.21}$$

If neutral elements exist, they are unique within the corresponding subsemigroup. In fact, suppose the existence of another element $e'_s \in \Phi_s$ with the identical property that $\phi \otimes e'_s = e'_s \otimes \phi = \phi$ for all $\phi \in \Phi_s$. Then, since e_s and e'_s behave neutrally among each other, $e_s = e_s \otimes e'_s = e'_s$. Furthermore, valuation algebras with neutral elements allow for a simplified version of the combination axiom that was already proposed in (Shenoy & Shafer, 1990).

Lemma 3.4 *In a valuation algebra with neutral elements, the combination axiom is equivalently expressed as:*

(A5) *Combination: For ϕ, $\psi \in \Phi$ with $d(\phi) = x$, $d(\psi) = y$,*

$$(\phi \otimes \psi)^{\downarrow x} \quad = \quad \phi \otimes \psi^{\downarrow x \cap y}. \tag{3.22}$$

Proof: Equation (3.22) follows directly from the combination axiom with $z = x$. On the other hand, let $x \subseteq z \subseteq x \cup y$. Since $z \cap (x \cup y) = z$ we derive from equation (3.22) and Axiom (A1) that

$$\begin{aligned}
(\phi \otimes \psi)^{\downarrow z} &= e_z \otimes (\phi \otimes \psi)^{\downarrow z} \\
&= (e_z \otimes \phi \otimes \psi)^{\downarrow z} \\
&= e_z \otimes \phi \otimes \psi^{\downarrow y \cap z} \\
&= \phi \otimes \psi^{\downarrow y \cap z}.
\end{aligned}$$

The last equality holds because

$$d(\phi \otimes \psi^{\downarrow y \cap z}) \ = \ x \cup (y \cap z) \ = \ (x \cup y) \cap (x \cup z) \ = \ z \ = \ d(e_z).$$

Note that this result presupposes the distributivity of the lattice D. ■

The following lemma shows that neutral elements also behave neutral with respect to valuations on larger domains:

Lemma 3.5 *For $\phi \in \Phi$ with $d(\phi) = x$ and $y \subseteq x$ it holds that*

$$\phi \otimes e_y \quad = \quad \phi. \tag{3.23}$$

Proof: From the associativity of combination and the neutrality axiom follows

$$\phi \otimes e_y \;=\; (\phi \otimes e_x) \otimes e_y \;=\; \phi \otimes (e_x \otimes e_y) \;=\; \phi \otimes e_x \;=\; \phi.$$

■

The presence of neutral elements allows furthermore to extend valuations to larger domains. This operation is called *vacuous extension* and may be seen as a dual operation to projection. For $\phi \in \Phi$ and $d(\phi) \subseteq y$ we define

$$\phi^{\uparrow y} \quad = \quad \phi \otimes e_y. \tag{3.24}$$

Before we consider some concrete examples of valuation algebras with and without neutral elements, we introduce another property called *stability* that is closely related to neutral elements. In fact, stability can either be seen as the algebraic property that enables undoing the operation of vacuous extension, or as a strengthening of the domain axiom (A6).

3.3.1 Stable Valuation Algebras

The neutrality axiom (A7) determines the behaviour of neutral elements under the operation of combination. It states that a combination of neutral elements will always result in a neutral element again. However, a similar property can also be requested for the operation of projection. In a *stable* valuation algebra, neutral elements always project to neutral elements again (Shafer, 1991). This property seems natural but there are indeed important examples which do not fulfill stability.

Definition 3.6 *A valuation algebra with neutral elements $\langle \Phi, D \rangle$ is called* stable *if the following property is satisfied for all $x, y \in D$ with $x \subseteq y$,*

(A8) Stability:

$$e_y^{\downarrow x} \quad = \quad e_x. \tag{3.25}$$

Stability is a very strong condition which also implies other valuation algebra properties: together with the combination axiom, it can, for example, be seen as a strengthening of the domain axiom. Also, it is possible to revoke vacuous extension under stability. These are the statements of the following lemma:

Lemma 3.6

1. *The property of stability with the combination axiom implies the domain axiom.*

2. *If stability holds, we have for $\phi \in \Phi$ with $d(\phi) = x \subseteq y$,*

$$(\phi^{\uparrow y})^{\downarrow x} \quad = \quad \phi. \tag{3.26}$$

Proof:

1. For $\phi \in \Phi$ with $d(\phi) = x$ we have

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

2. For $\phi \in \Phi$ with $d(\phi) = x \subseteq y$ we derive from the combination axiom

$$(\phi^{\uparrow y})^{\downarrow x} \quad = \quad (\phi \otimes e_y)^{\downarrow x} \quad = \quad \phi \otimes e_y^{\downarrow x} \quad = \quad \phi \otimes e_x \quad = \quad \phi.$$

■

■ 3.1 Neutral Elements and Indicator Functions

In the valuation algebras of indicator functions, the neutral element for the domain $s \in D$ is given by $e_s(\mathbf{x}) = 1$ for all $\mathbf{x} \in \Omega_s$. We then have for $\phi \in \Phi_s$

$$\phi \otimes e_s(\mathbf{x}) \quad = \quad \phi(\mathbf{x}) \cdot e_s(\mathbf{x}) \quad = \quad \phi(\mathbf{x}) \cdot 1 \quad = \quad \phi(\mathbf{x}).$$

Also, the neutrality axiom holds, since for $s, t \in D$ with $\mathbf{x} \in \Omega_{s \cup t}$ we have

$$e_t \otimes e_s(\mathbf{x}) \quad = \quad e_t(\mathbf{x}^{\downarrow t}) \cdot e_s(\mathbf{x}^{\downarrow s}) \quad = \quad 1 \cdot 1 \quad = \quad 1 \quad = \quad e_{s \cup t}(\mathbf{x}).$$

Finally, the valuation algebra of indicator functions is also stable. For $s, t \in D$ with $s \subseteq t$ and $\mathbf{x} \in \Omega_s$ it holds that

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

■ 3.2 Neutral Elements and Arithmetic Potentials

Arithmetic potentials and indicator functions share the same definition of combination. We may therefore conclude that arithmetic potentials also provide neutral elements. However, in contrast to indicator functions, arithmetic potentials are not stable. We have for $s, t \in D$ with $s \subset t$ and $\mathbf{x} \in \Omega_s$

$$e_t^{\downarrow s}(\mathbf{x}) \quad = \quad \sum_{\mathbf{y} \in \Omega_{t-s}} e_t(\mathbf{x}, \mathbf{y}) \quad = \quad |\Omega_{t-s}| \quad > \quad 1.$$

3.3 Neutral Elements and Set Potentials

In the valuation algebra of set potentials, the neutral element e_s for the domain $s \in D$ is given by $e_s(A) = 0$ for all proper subsets $A \subset \Omega_s$ and $e_s(\Omega_s) = 1$. Indeed, it holds for $\phi \in \Phi$ with $d(\phi) = s$ that

$$\phi \otimes e_s(A) \;=\; \sum_{B \bowtie C = A} \phi(B) \cdot e_s(C) \;=\; \phi(A) \cdot e(\Omega_s) \;=\; \phi(A).$$

The second equality follows since for all other values of C we have $e_s(C) = 0$. Also, the neutrality axiom holds which will be proved explicitly in Section 5.8.1. Further, set potentials are stable. For $s, t \in D$ and $s \subseteq t$ it holds that

$$e_t^{\downarrow s}(\Omega_s) \;=\; \sum_{A \subseteq \Omega_t : \pi_s(A) = \Omega_s} e_t(A) \;=\; e_t(\Omega_t) \;=\; 1.$$

The second equality holds because $e_t(\Omega_t) = 1$ is the only non-zero summand.

3.4 Neutral Elements and Relations

For $s \in D$ the neutral element in the valuation algebra of relations is given by $e_s = \Omega_s$. Similar to indicator functions these neutral elements are also stable. However, there is an important issue regarding neutral elements in the relational algebra. Since variable frames are often very large or even infinite (i.e. all possible strings), the neutral elements can sometimes not be expressed explicitly.

3.5 Neutral Elements and Density Functions

Equation (1.15) introduced the combination of density functions as simple multiplication. Therefore, the definition $e_s(\mathbf{x}) = 1$ for $\mathbf{x} \in \mathbb{R}^{|s|}$ would clearly meet the requirements for a neutral element. However, this function e_s is not a density since its integral will not be finite

$$\int_{-\infty}^{\infty} 1 \, d\mathbf{x} \;=\; \infty.$$

Hence, densities form a valuation algebra without neutral elements. This holds in particular also for Gaussian potentials in Instance 1.6.

Before we continue with the interpretation of the fusion algorithm as a message-passing scheme in Section 3.5, we introduce a second family of elements that may be contained in a valuation algebra. As neutral elements express neutral knowledge, *null elements* express contradictory knowledge with respect to their domain.

3.4 VALUATION ALGEBRAS WITH NULL ELEMENTS

Some valuation algebras contain elements that express incompatible, inconsistent or contradictory knowledge according to questions $s \in D$. Such valuations behave absorbingly for combination and are therefore called *absorbing elements* or *null elements*. Hence, in a valuation algebra with null elements there is an element $z_s \in \Phi_s$ such that $z_s \otimes \phi = \phi \otimes z_s = z_s$ for all $\phi \in \Phi_s$. We call a valuation $\phi \in \Phi_s$ *consistent*, if and only if $\phi \neq z_s$. It is furthermore a very natural claim that a projection of some consistent valuation produces again a consistent valuation. This requirement is captured by the following additional axiom.

(A9) *Nullity:* For $x, y \in D$, $x \subseteq y$ and $\phi \in \Phi_y$,

$$\phi^{\downarrow x} = z_x$$

if, and only if, $\phi = z_y$.

In other words, if some valuation projects to a null element, then it must itself be a null element. Thus, contradictions can only be derived from contradictions. We further observe that according to the above definition, null elements absorb only valuations of the same domain. In case of stable valuation algebras, however, this property also holds for any other valuation.

Lemma 3.7 *In a stable valuation algebra with null elements we have for all $\phi \in \Phi$ with $d(\phi) = x$ and $y \in D$*

$$\phi \otimes z_y = z_{x \cup y}. \tag{3.27}$$

Proof: We remark first that from equation (3.26) it follows

$$\left(z_y^{\uparrow x \cup y}\right)^{\downarrow y} = z_y.$$

Hence, we conclude from the nullity axiom that $z_y^{\uparrow x \cup y} = z_{x \cup y}$ and derive

$$\begin{aligned} \phi \otimes z_y &= (\phi \otimes e_{x \cup y}) \otimes (z_y \otimes e_{x \cup y}) = \phi^{\uparrow x \cup y} \otimes z_y^{\uparrow x \cup y} \\ &= \phi^{\uparrow x \cup y} \otimes z_{x \cup y} = z_{x \cup y}. \end{aligned}$$

■

Let us search for null elements in the valuation algebra instances of Chapter 1.

■ 3.6 Null Elements and Indicator Functions

In the valuation algebras of indicator functions the null element for the domain $s \in D$ is given by $z_s(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega_s$. We then have

$$\phi \otimes z_s(\mathbf{x}) = \phi(\mathbf{x}) \cdot z_s(\mathbf{x}) = \phi(\mathbf{x}) \cdot 0 = z_s(\mathbf{x}).$$

Since projection refers to maximization, it is easy to see that null elements, and only null elements project to null elements.

■ 3.7 Null Elements and Arithmetic Potentials

Again, since their combination rules are equal, arithmetic potentials and indicator functions share the same definition of null elements: for $s \in D$ we define $z_s(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega_s$. The absorption property for combination follows directly from Instance 3.6. Further, projection is defined as summation for arithmetic potentials and since the values are non-negative, it is again only the null element that projects to a null element.

■ 3.8 Null Elements and Relations

Since we have already identified the null elements in the valuation algebra of indicator functions, it is simple to give way to relations. Due to equation (1.9) null elements are simply empty relations with respect to their domain.

■ 3.9 Null Elements and Set Potentials

In the valuation algebra of set potentials, the null element z_s for the domain $s \subseteq r$ is defined by $z_s(A) = 0$ for all $A \subseteq \Omega_s$. These elements behave absorbingly for combination, and are the only elements that project to null elements, which follows again from their non-negative values. Although this is rather easy to see, we will give a formal proof in Section 5.8.2.

■ 3.10 Null Elements and Density Functions

In the valuation algebra of density functions, the null element for the domain s is given by $z_s(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega_s$. The properties of null elements follow again from the fact that density functions are non-negative.

■ 3.11 Null Elements and Gaussian Densities

In the introduction of density functions we also mentioned that the family of Gaussian densities forms a subalgebra of the valuation algebra of density functions. Although we are going to study this formalism extensively in Chapter 10, we already point out that Gaussian densities do not possess null elements due to the requirement for a positive definite concentration matrix.

Null elements play an important role in the semantics of inference problems. Imagine that we solve an inference problem with the empty set as single query. After the execution of local computation, we find a null element as answer to this query:

$$(\phi_1 \otimes \cdots \otimes \phi_n)^{\downarrow\emptyset} \quad = \quad z_\emptyset.$$

We therefore conclude from Axiom (A9) that the objective function ϕ must itself be a null element, or, in other words, that the knowledgebase is *inconsistent*. This is an important issue in constraint programming since it allows to check the satisfiability of a constraint system by local computation. A similar argumentation has, for example, been used in Instance 2.4.

This closes our intermezzo of special elements in a valuation algebra. We now return to the solution of single-query inference problems by local computation techniques and give an alternative picture of the fusion algorithm in terms of a message-passing scheme. This presupposes a valuation algebra with neutral elements, although it will later be shown in Section 3.10 that this assumption can be avoided.

3.5 LOCAL COMPUTATION AS MESSAGE-PASSING SCHEME

Let us reconsider the join tree resulting from the graphical representation of the fusion algorithm. We then observe that for all knowledgebase factor domains $d(\phi_i)$, there exists a node $v \in V$ in the join tree which covers $d(\phi_i)$, i.e. $d(\phi_i) \subseteq \lambda(v)$. This is a simple consequence of equation (3.12), which defines the label of the new nodes added during the elimination of a variable. More precisely, if $X_j \in d(\phi_i)$ is the first variable in the elimination sequence, then $d(\phi_i) \subseteq \lambda(j)$. This is a consequence of the particular node numbering defined in the fusion process. We may therefore assign factor ϕ_i to the join tree node $j \in V$. This process is repeated for all factors in the knowledgebase of the inference problem, which is illustrated in Example 3.12. Remember also that the query $x \subseteq d(\phi)$ of the inference problem always corresponds to the label of the root node since the latter contains all variables that have not been eliminated. A join tree that allows such a factor distribution will later be called a *covering join tree* in Definition 3.8. The process of distributing knowledgebase factors over join tree nodes may assign multiple factors to one node, whereas other nodes go away without an assigned valuation. In Example 3.12, only the root node does not contain a knowledgebase factor, but if we had eliminated more variables, then inner nodes also would exist that do not posses a knowledgebase factor. If we further assume a valuation algebra with neutral elements, we may assign the neutral element $e_{\lambda(i)}$ to all nodes $i \in V$ which do not hold a knowledgebase factor. On the other hand, if a node contains multiple knowledgebase factors, they are combined. The result of this process is a join tree where every node $i \in V$ contains exactly one valuation ψ_i with $d(\psi_i) \subseteq \lambda(i)$. This valuation either corresponds to a single knowledgebase factor, to a combination of multiple knowledgebase factors or to a neutral element, as shown in Example 3.12.

Example 3.12 *The knowledgebase factor domains in the medical inference problem of Instance 2.1 are: $d(\phi_1) = \{A\}$, $d(\phi_2) = \{A, T\}$, $d(\phi_3) = \{L, S\}$, $d(\phi_4) = \{B, S\}$, $d(\phi_5) = \{E, L, T\}$, $d(\phi_6) = \{E, X\}$, $d(\phi_7) = \{B, D, E\}$ and $d(\phi_8) = \{S\}$. The single query to compute is $x = \{A, B, D\}$. These factors are distributed over the nodes of the covering join tree in Figure 3.8 that results from the graphical fusion process (see Figure 3.1). The node factors of this join tree become: $\psi_1 = \phi_6$, $\psi_2 = \phi_3 \otimes \phi_4 \otimes \phi_8$, $\psi_3 = \phi_5$, $\psi_4 = \phi_1 \otimes \phi_2$, $\psi_5 = \phi_7$, $\psi_6 = e_{\{A,B,D\}}$.*

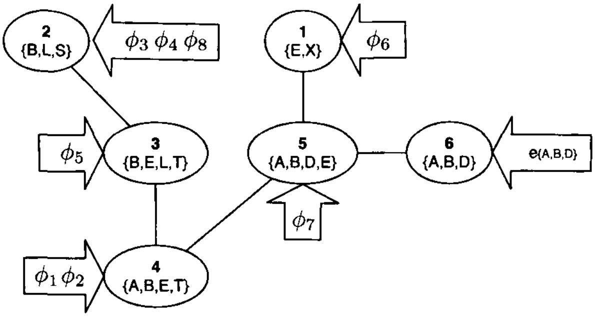


Figure 3.8 A covering join tree for the medical example of Instance 1.3.

A common characteristic of all local computation algorithms is their interpretation as message-passing schemes (Shenoy & Shafer, 1990). In this paradigm, nodes act as virtual processors that communicate by the exchange of messages. Nodes process incoming messages, compute new messages and send them to neighboring nodes of the join tree. Based on the new factorization constructed above, we now introduce the fusion algorithm from the viewpoint of message-passing: at the beginning, node 1 contains the valuation ψ_1. According to the fusion process, it eliminates the first variable X_1 of the elimination sequence from its node content and sends the result to node 5, which then combines the message to its own node content. Node 5 computes

$$\psi_5 \otimes \psi_1^{-X_1}$$

and replaces its node content with this result. More generally, if node i contains at step i of the fusion algorithm the valuation ν_i (which consists of its initial factor ψ_i combined with all messages sent to node i in the foregoing $i-1$ steps), then it computes ν^{-X_i} and sends the result to its unique neighbor with a higher node number. In Definition 3.11 this node is called *child node* and denoted by ch(i). The child node then computes

$$\nu_{ch(i)} \otimes \nu_i^{-X_i}$$

and sets its node content to the new value. This process is repeated for $i = 1, \ldots, |V|-1$. Then, at the end of the message-passing, the root node $|V|$ contains the result of the inference problem

$$\phi^{\downarrow x} \quad = \quad (\phi_1 \otimes \cdots \otimes \phi_m)^{\downarrow x}. \tag{3.28}$$

On the one hand, we may draw this conclusion directly since the message-passing procedure is just another view of the fusion algorithm. On the other hand, we deliver an explicit proof for the correctness of message-passing in a more general context in Theorem 3.6 and 3.7.

In the message-passing concept, nodes act as virtual processors with a private valuation storage that execute combinations and projections independently and communicate via the exchange of messages along the edges of the tree. Since all local computation methods we subsequently introduce adopt this interpretation, we may classify them as (virtually) distributed algorithms.

Example 3.13 *Figure 3.9 illustrates the message-passing view of the fusion algorithm. If $\mu_{i \rightarrow j}$ denotes the message sent from node i to node j at step i of the fusion algorithm, we have:*

- $\mu_{1\rightarrow 5} = \phi_6^{-X}$
- $\mu_{2\rightarrow 3} = (\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S}$
- $\mu_{3\rightarrow 4} = (\phi_5 \otimes \mu_{2\rightarrow 3})^{-L}$
- $\mu_{4\rightarrow 5} = (\phi_1 \otimes \phi_2 \otimes \mu_{3\rightarrow 4})^{-T}$
- $\mu_{5\rightarrow 6} = (\phi_7 \otimes \mu_{1\rightarrow 5} \otimes \mu_{4\rightarrow 5})^{-E}$

Thus, we obtain at the end of the message-passing in node 6:

$$e_{\{A,B,D\}} \otimes (\phi_6^{-X} \otimes \phi_7 \otimes (\phi_1 \otimes \phi_2 \otimes (\phi_5 \otimes (\phi_3 \otimes \phi_4 \otimes \phi_8)^{-S})^{-L})^{-T})^{-E}$$

which, by Lemma 3.5, corresponds to Example 3.1.

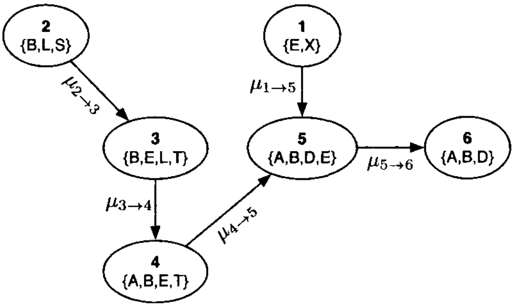


Figure 3.9 The fusion algorithm as message-passing scheme where arrows indicate the direction along which messages are sent.

3.5.1 The Complexity of Fusion as Message-Passing Scheme

The message-passing implementation of the fusion algorithm adopts the same time complexity but requires more space than actually necessary. When a variable X_i is eliminated in the fusion algorithm, all valuations that contain this variable in their domain are combined which leads to a result of domain s_i. In the message-passing scheme, this valuation is stored in node i, whereas in the fusion algorithm, variable X_i is first eliminated before the result is stored. Consequently, the message-passing implementation of the fusion algorithm has a space complexity of:

$$\mathcal{O}\Big(|d(\phi)| \cdot g(\omega^* + 1)\Big) \tag{3.29}$$

However, it is important to understand that this is just a consequence of its implementation as message-passing scheme and does not modify the statement about the general complexity of the fusion algorithm given in equation (3.20).

The reason why the space complexity of the fusion algorithm becomes worse when implemented as message-passing scheme is the very particular join tree that has been used. Remember, so far we obtained the join tree from a graphical representation of the fusion algorithm, and this procedure only creates very particular join trees. The fusion algorithm eliminates exactly one variable in each step and therefore, exactly one variable vanishes from the knowledgebase between node i and $i+1$. This is mirrored in the general space complexity of the fusion algorithm given in equation (3.20) where the two terms differ not surprisingly in one exponent. If we aim at an improvement of the space complexity, we must therefore get rid of this limiting property. In the following section, we develop a generalization of the message-passing conception by decoupling it from the actual join tree construction process. At the same time, we replace variable elimination again by projection such that the resulting algorithm can be applied to valuation algebras on arbitrary lattices.

3.6 COVERING JOIN TREES

The foregoing discussion identified the join tree as the major ingredient of local computation since it reflects the tree decomposition structure of an inference problem that bounds the domain of intermediate factors during the computations. If we aim at the decoupling of local computation from the join tree construction process, we have to ensure that the join tree taken for the computation is suitable for the current inference problem. This requirement is captured by the concept of a *covering join tree* that will be introduced below. Here, we directly give a general definition with regard to the processing of multiple queries in Chapter 4. In Figure 3.8 we assigned a knowledgebase factor to a join tree node if the domain of the factor was a subset of the node label. If such a node can be found for each knowledgebase factor, one speaks about a covering join tree for the factorization.

Definition 3.7 *Let $\mathcal{T} = (V, E, \lambda, D)$ be a join tree.*

- *$\mathcal{T}$ is said to* cover *the domains $s_1, \ldots, s_m \in D$ if there is for every s_i a node $j \in V$ with $s_i \subseteq \lambda(j)$.*
- *$\mathcal{T}$ is called* covering join tree for the factorization *$\phi_1 \otimes \cdots \otimes \phi_m$ if it covers the domains $s_i = d(\phi_i) \in D$ for $1 \leq i \leq m$.*

We furthermore say that a join tree covers an inference problem, if it covers all knowledgebase factors and queries. Its formal definition demands in particular that no *free variables* exist in the join tree (condition 3 below) . This means that each variable in the node labels must also occur somewhere in the knowledgebase.

Definition 3.8 *A join tree $\mathcal{T}$ is called* covering join tree for the inference problem

$$(\phi_1 \otimes \cdots \otimes \phi_m)^{\downarrow x_i}$$

with $x_i \in \{x_1, \ldots, x_s\}$, if the following conditions are satisfied:

1. *$\mathcal{T}$ is a covering join tree for the factorization $\phi_1 \otimes \cdots \otimes \phi_m$.*
2. *$\mathcal{T}$ covers the queries $\{x_1, \ldots, x_s\}$.*
3. *D corresponds to the power set $\mathcal{P}(d(\phi_1 \otimes \cdots \otimes \phi_m))$.*

A covering join tree corresponds to the concept of a *tree-decomposition* (Robertson & Seymour, 1984) that is widely used in the literature – also in many other contexts that are not directly related to valuation algebras.

Example 3.14 *In Section 3.5 we have already exploited the fact that the join tree of Figure 3.1 is a covering join tree for the medical inference problem of Instance 1.3. Another covering join tree for the same inference problem is shown in Figure 3.10.*

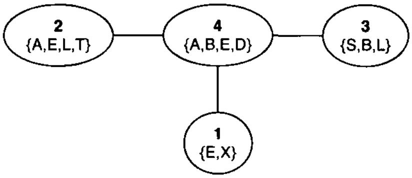


Figure 3.10 A further covering join tree for the medical example of Instance 2.1.

Covering join trees provide the required data structure for the introduction of more general and economical local computation schemes. We now follow the same procedure as in Section 3.5 and assign the knowledgebase factors to covering nodes. This defines an assignment mapping according to the following definition.

Definition 3.9 *Let (V, E, λ, D) be a covering join tree for the factorization*

$$\phi \quad = \quad \phi_1 \otimes \ldots \otimes \phi_m.$$

A function

$$a : \{1, \ldots, m\} \rightarrow V$$

is called an assignment mapping *for ϕ regarding V if for every $i \in \{1, \ldots, m\}$ we have $d(\phi_i) \subseteq \lambda(a(i))$.*

Example 3.15 *The assignment mapping of Figure 3.8 is* $a(1) = 4$, $a(2) = 4$, $a(3) = 2$, $a(4) = 2$, $a(5) = 3$, $a(6) = 1$, $a(7) = 5$, $a(8) = 2$. *Observe also that there are in general several possible assignment mappings.*

It was already pointed out that some nodes may go away empty-handed from this factor distribution process. Also, it may be that the domain of a factor assigned to a certain node does not fill its node label completely. We therefore initialize every join tree node with the neutral element of its label.

Definition 3.10 *Let* a *be an assignment mapping for a factorization* ϕ *regarding the nodes* V *of a covering join tree* (V, E, λ, D). *The* factor *assigned to node* $i \in V$ *is*

$$\psi_i \quad = \quad e_{\lambda(i)} \otimes \bigotimes_{j:a(j)=i} \phi_j. \tag{3.30}$$

This assures that for every node factor ψ_i we have $d(\psi_i) = \lambda(i)$. However, it will later be shown in Section 3.10 that we may in fact dispose of this artificial filling.

Example 3.16 *The join tree of Figure 3.11 is a covering join tree for the factorization of Instance 2.1 (but not for the inference problem since the query is not covered). The corresponding factor domains are given in Example 3.12. A possible factor assignment is:* $a(1) = a(2) = 1$, $a(3) = a(4) = a(8) = 3$, $a(5) = 2$, $a(6) = 7$ *and* $a(7) = 4$. *No factor is assigned to the nodes* 5 *and* 6. *This assignment creates the node factors:*

- $\psi_1 \;=\; e_{\{A,T\}} \otimes \phi_1 \otimes \phi_2 \;=\; \phi_1 \otimes \phi_2$
- $\psi_2 \;=\; e_{\{E,L,T\}} \otimes \phi_5 \;=\; \phi_5$
- $\psi_3 \;=\; e_{\{B,L,S\}} \otimes \phi_3 \otimes \phi_4 \otimes \phi_8 \;=\; \phi_3 \otimes \phi_4 \otimes \phi_8$
- $\psi_4 \;=\; e_{\{B,D,E\}} \otimes \phi_7 \;=\; \phi_7$
- $\psi_5 \;=\; e_{\{B,E,L\}}$
- $\psi_6 \;=\; e_{\{E,L\}}$
- $\psi_7 \;=\; e_{\{E,X\}} \otimes \phi_6 \;=\; \phi_6$

The factors ψ_i are usually called *join tree factors* and correspond either to (the vacuous extension of) a single factor of the original knowledgebase, a combination of multiple factors, or to a neutral element, if the factor set of the combination in equation (3.30) is empty. The following lemma shows that the join tree factors provide an alternative factorization to the knowledgebase of the inference problem:

Lemma 3.8 *It holds that*

$$\phi \;=\; \phi_1 \otimes \cdots \otimes \phi_m \;=\; \bigotimes_{i \in V} \psi_i. \tag{3.31}$$

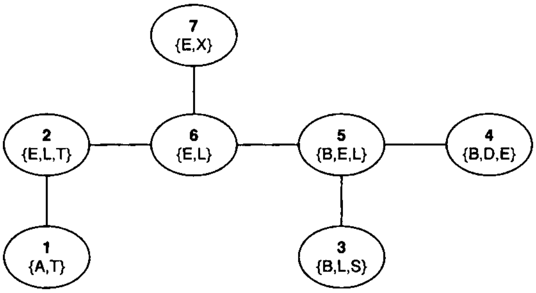


Figure 3.11 An covering join tree for the factorization of Instance 2.1.

Proof: The assignment mapping assigns every knowledgebase factor to exactly one join tree node. It therefore follows from the commutativity of combination and the neutrality axiom that

$$\begin{aligned}\bigotimes_{i \in V} \psi_i &= \phi_1 \otimes \ldots \otimes \phi_m \otimes \bigotimes_{i \in V} e_{\lambda(i)} \\ &= \phi_1 \otimes \ldots \otimes \phi_m \otimes e_{d(\phi)} = \phi \otimes e_{d(\phi)} = \phi.\end{aligned}$$

The second equality follows from the third requirement of Definition 3.8. ■

When a covering join tree is obtained from the graphical fusion process, its node numbering is determined by the variable elimination sequence, i.e. a node gets number $i \in \mathbb{N}$ if is was created during the elimination of the i-th variable in the fusion process. This particular numbering shaped up as very useful to identify the sending and receiving node during the message-passing process. However, if local computation is defined on arbitrary covering join trees, such a node numbering has to be introduced artificially. To do so, we first fix a node which covers the query as root node and assign the number $|V|$ to it. Then, by directing all edges towards the 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 $j > i$. Formally, let (V, E, λ, D) be a join tree. We determine a permutation π of the elements in V such that

- $\pi(k) = |V|$, if k is the root node;
- $\pi(j) > \pi(i)$ for every node $j \in V$ lying on the path between i and $|V|$.

The result is a renumbered join tree (V, E', λ, D) with edges $E' = \{(\pi(i), \pi(j)) : (i,j) \in E\}$ that is usually called a *directed join tree* towards the root node $|V|$. Note also that such a numbering is not unique. It is furthermore convenient to introduce the notions of *parents*, *child*, and *leaves* with respect to this node numbering:

Definition 3.11 *Let (V, E, λ, D) be a directed join tree towards root node $|V|$.*

- *The* parents $pa(i)$ *of a node i are defined by the set*

$$pa(i) = \{j : j < i \textit{ and } (i,j) \in E\}.$$

- *Nodes without parents are called* leaves.
- *The* child $ch(i)$ *of a node* $i < |V|$ *is the unique node* j *with* $j > i$ *and* $(i, j) \in E$.

In this numbering parents always have a smaller number than their child.

Definition 3.12 *Let* (V, E, λ, D) *be a directed join tree towards root node* $|V|$.

- *The* separator $sep(i)$ *of a node* $i < |V|$ *is defined by* $sep(i) = \lambda(i) \cap \lambda(ch(i))$.
- *The* eliminator $elim(i)$ *of a node* $i < |V|$ *is defined by* $elim(i) = \lambda(i) - sep(i)$.

Finally, the definition of a tree implies that whenever one of its edges $(i, ch(i))$ is removed, the tree splits into two separated trees where the one that contains the node i is called *subtree rooted to node i*, abbreviated by $\mathcal{T}_i$. Clearly, the running intersection property remains satisfied if join trees are cut in two.

Example 3.17 *Consider the join tree of Figure 3.15 which corresponds to the join tree of Figure 3.11 directed towards node* $\{B, D, E\}$ *and renumbered according to the above scheme. The parents of node* 5 *are* $pa(5) = \{2, 4\}$ *and its child is* $ch(5) = 6$. *Further, we have* $sep(5) = \{E, L\}$ *and* $elim(5) = \{B\}$. *Also, if we for example cut the edge between the nodes* 5 *and* 6, *we obtain two trees that both satisfy the running intersection property.*

3.7 JOIN TREE CONSTRUCTION

Before we actually describe how local computation can be performed on arbitrary covering join trees, we should first say a few words about how to build covering join trees for an inference problem. In fact, this topic is a broad research field by itself, which makes it impossible at this point to give all the important references. However, from our complexity studies of the fusion algorithm, we know the principal quality criterion of a join tree. The treewidth of the join tree determines the complexity of local computation and therefore, we focus on finding covering join trees whose treewidth is as small as possible. We further know that the treewidth of the inference problem provides a lower bound. Finding such an optimum join tree is NP-hard. Thus, if we want to achieve reasonable local computation complexity, we are forced to fall back on heuristics. There are a many different heuristics to find a suitable variable elimination sequence as for example (Rose, 1970; Bertele & Brioschi, 1972; Yannakakis, 1981; Kong, 1986; Almond & Kong, 1991; Haenni & Lehmann, 1999; Allen & Darwiche, 2002; Hopkins & Darwiche, 2002). An overview of recommended heuristics can be found in (Lehmann, 2001; Dechter, 2003) and a comparison is drawn by (Cano & Moral, 1995). These methods deliver a variable elimination sequence from which the covering join tree is constructed using some possibly improved version of the graphical fusion process of Section 3.2.1. The justification of this approach is the fact that no better join tree can be found if we do without variable elimination (Arnborg, 1985; Dechter & Pearl, 1989). However, building a join tree from a previously fixed elimination sequence requires a lot of knowledgebase rearrangement operations that in turn can lead to severe performance problems. A particularly efficient way has been proposed by (Lehmann, 2001) and uses a data-structure called *variable-valuation-linked-list*. It directly returns a covering join tree including an assignment mapping that assigns at most one knowledgebase factor to each join tree node. This is often a very advantageous property since it avoids that combinations are executed before the actual local computation algorithm is started. On the other hand, such join trees generally contain more nodes and therefore also require to store more factors and messages during local computation. To sum it up, covering join trees can be found by the graphical fusion process and some variable elimination sequence. As a complement, we skim over an alternative but equivalent method to construct join trees starting from the primal graph of the knowledgebase.

3.7.1 Join Tree Construction by Triangulation

Imagine the primal graph representation of a knowledgebase as introduced in Section 2.2. Here, the variables of the knowledgebase valuations correspond to the graph nodes, and two nodes are linked if the corresponding variables occur in the domain of the same valuation. Its associated dual graph has a node for each factor domain, and the nodes are connected if they share a common variable. Therefore, each valuation is naturally covered by some node of the dual graph. Finding a covering join tree therefore consists in removing edges from the dual graph until it is a tree that satisfies the running intersection property. The following theorem (Beeri *et al.*, 1983) states under which condition this is possible:

Theorem 3.4 *A graph is triangulated if, and only if, its dual graph has a join tree.*

A graph is called *triangulated* if each of its cycles of four or more nodes has an edge joining two nodes that are not adjacent in the cycle. Elsewhere, triangulated graphs are also called *chordal graphs* where the *chord* refers to the introduces edges during the triangulation process. Figure 3.12 shows a cycle of six nodes together with two possible triangulations.

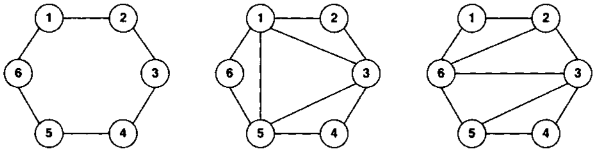


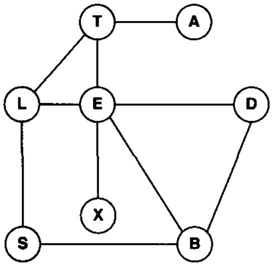
Figure 3.12 A graph with two possible triangulations.

Obviously, there are different possibilities to triangulate a graph. Various algorithmic aspects of this task are discussed in (Tarjan & Yannakakis, 1984). Here, we give a simple procedure from (Cano & Moral, 1995) that is based on a node elimination sequence $(X_1, \ldots, X_n)$. Applied to primal graphs, this clearly corresponds to a variable elimination sequence.

- For $i = 1, \ldots, n$:
    - Connect all pairs of nodes that are neighbors to X_i. Let L_i denote the set of added edges.
    - Remove node X_i and all edges connected to this node from the graph.
- To obtain the triangulation of the original graph, add all edges in $\bigcup_{i=1}^{n} L_i$.

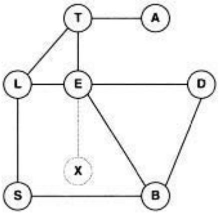
In fact, this algorithm corresponds exactly to the procedure of constructing an induced graph from a primal graph described in Section 3.2.4. We thus conclude that the induced graph of a primal graph ordered along a given variable elimination sequence is equal to the triangulated graph obtained from the same elimination sequence. This supports the above statement that constructing join trees via graph triangulation is just another interpretation of the methods introduce before.

Example 3.18 *Consider the knowledgebase of the medical example from Instance 2.1, whose domains are listed in Example 3.1, and draw up its primal graph:*

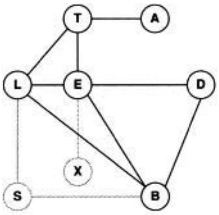


If we want to ensure that the join tree covers some specified query, then the query must also be added to the primal graph. This mirrors the insertion of a corresponding neutral element into the knowledgebase. We choose the elimination sequence (X, S, L, T, E, A, B, D) and execute the above algorithm:

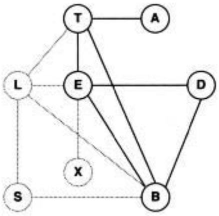
- *Elimination of variable X:*



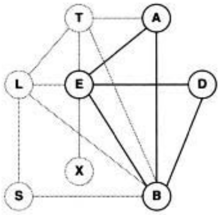
- *Elimination of variable S:*



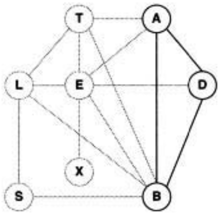
- *Elimination of variable L:*



- *Elimination of variable T:*



- *Elimination of variable E:*



At this point, the triangulation process can be stopped because the remaining nodes form a clique. Figure 3.13 shows the completed triangulation of the primal graph which is in fact equal to the induced graph that we obtain from the same elimination sequence by the procedure of Section 3.2.4, see Example 3.8. We further observe that the triangulated graph has the maximal cliques $\{B, L, S\}$, $\{B, E, L, T\}$, $\{A, E, T\}$, $\{A, B, E, T\}$, $\{A, B, D, E\}$ *and* $\{E, X\}$.

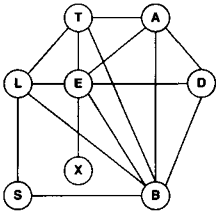


Figure 3.13 The triangulated primal graph of the medical example from Instance 2.1.

We now have a triangulated graph from which we can derive the join tree that was promised in Theorem 3.4. For this purpose, we identify all maximal cliques in the triangulated graph and build a new graph whose nodes are the maximal cliques. Each node is connected to all other nodes and each edge is labeled with the intersection of the corresponding cliques. Such a graph is called a *join graph*. Further, we refer to the *weight* of an edge in the join graph as the number of variables in the edge label. The following theorem is proved in (Jensen, 1988).

Theorem 3.5 *A spanning tree of a join graph is a join tree if, and only if, its sum of edge weights is maximal.*

An important advantage of this characterization is that standard enumeration algorithms for spanning trees (Broder & Mayr, 1997) can be used. Note also that several join trees with the same weight may exist, as it can easily be seen in the following example. These join trees clearly have the same nodes and thus the same treewidth, but they may distinguish in the separator widths.

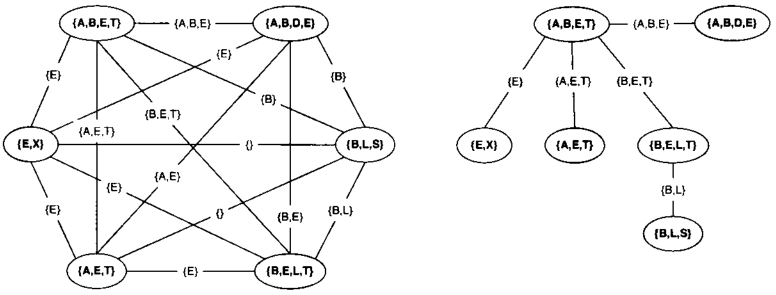


Figure 3.14 The join graph of the triangulation from Example 3.18 and a possible join tree derived as a spanning tree with maximal weight.

Example 3.19 *The left-hand side of Figure 3.14 shows the join graph obtained from the cliques identified in Example 3.18. There are* 6 *nodes which implies that a spanning tree must have* 5 *edges. A spanning tree with maximal weight can be found by including the three edges with weight* 3 *and choosing the other two edges with* *maximal weight among the remaining candidates. There are several join trees with maximal weight, one being shown on the right-hand side of Figure 3.14.*

We now abandon the widespread field of join tree construction and focus in the following sections on the solution of single-query inference problems based on a previously built covering join tree.

3.8 THE COLLECT ALGORITHM

The collect algorithm is the principal local computation scheme for the solution of single-query inference problems. It presupposes a covering join tree for the current inference problem that is directed and numbered according to the scheme at the end of Section 3.6 and that keeps a factor ψ_j on every node j which is defined according to equation (3.30). We therefore say that ψ_j represents the initial content of node j. The collect algorithm can then be described by three simple rules:

R1: Each node sends a message to its child when it has received all messages from its parents. This implies that leaves (i.e. nodes without parents) can send their messages right away.

R2: When a node is ready to send, it computes the message by projecting its current content to the separator and sends the message to its child.

R3: When a node receives a message, it updates its current content by combining it with the incoming message.

This procedure is repeated up to the root node. It follows from this first sketch that the content of the nodes may change during the algorithm's run. To incorporate this dynamic behavior, the following notation is introduced.

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

The particular way of numbering the nodes of the directed join tree implies that at step i, node i can send a message to its child. This allows the following specification of the collect algorithm.

- At step i, node i computes the message

  $$\mu_{i\rightarrow ch(i)} \quad = \quad \psi_i^{(i)\downarrow sep(i)}. \tag{3.32}$$

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

- The receiving node $ch(i)$ combines the message with its node content:

  $$\psi_{ch(i)}^{(i+1)} \quad = \quad \psi_{ch(i)}^{(i)} \otimes \mu_{i\rightarrow ch(i)}. \tag{3.33}$$

The content of all other nodes does not change at step i, i.e. for all $j \neq ch(i)$

$$\psi_j^{(i+1)} \quad = \quad \psi_j^{(i)}. \tag{3.34}$$

The justification of the collect algorithm is formulated by the following theorem. Remember that the root node has been chosen in such a way that it covers the query of the inference problem.

Theorem 3.6 *At the end of the collect algorithm, the root node $r = |V|$ contains the marginal of ϕ relative to $\lambda(r)$,*

$$\psi_r^{(r)} \quad = \quad \phi^{\downarrow\lambda(r)}. \tag{3.35}$$

The proof of this theorem can be found in the appendix of this chapter. The marginal $\phi^{\downarrow\lambda(r)}$ can now be used to solve the single-query inference problem by one last projection to the query x, because the latter is covered by the root node.

$$\phi^{\downarrow x} \quad = \quad \left(\phi^{\downarrow\lambda(r)}\right)^{\downarrow x}. \tag{3.36}$$

A summary of the collect algorithm is given in Algorithm 3.3. The input to this algorithm is a covering join tree (V, E, λ, D) where each node $i \in V$ contains a factor $\psi_i \in \Phi$. Also, the query x is given as input for the final projection of equation (3.36) performed in the statement.

Algorithm 3.3 The Collect Algorithm

```
input:  $(V, E, \lambda, D)$,  $x \subseteq d(\phi)$   output:  $(\phi_1 \otimes \cdots \otimes \phi_m)^{\downarrow x}$

begin
  for  $i = 1 \ldots |V| - 1$  do
    $\mu_{i \to ch(i)}$ := $\psi_i^{\downarrow\lambda(i)\cap\lambda(ch(i))}$;
    $\psi_{ch(i)}$ := $\psi_{ch(i)} \otimes \mu_{i \to ch(i)}$;
  end;
  return  $\psi_r^{\downarrow x}$;
end
```

The medical inference problem of Instance 2.1 is based on variables which allows us to express projection by variable elimination. Thus, if we illustrate the collect algorithm on the same join tree as the fusion algorithm in Example 3.13, we repeat exactly the same computations. We therefore consider a new query for the illustration of the collect algorithm in Example 3.20.

Example 3.20 *We reconsider the knowledgebase of the medical inference problem with the new query $x = \{B, E\}$ and use the join tree of Figure 3.11 for the computation. It has already been shown in Example 3.16 that this join tree covers our knowledgebase. Since node 4 of Figure 3.11 covers the new query, we directly have a covering join tree for the new inference problem. However, the node numbering has* *to be modified such that node 4 becomes the new root node. A possible renumbering is shown in Figure 3.15 and the new node factors are:* $\psi_1 = \phi_1 \otimes \phi_2$, $\psi_2 = \phi_5$, $\psi_3 = \phi_3 \otimes \phi_4 \otimes \phi_8$, $\psi_4 = \phi_6$, $\psi_5 = e_{\{E,L\}}$, $\psi_6 = e_{\{B,E,L\}}$, $\psi_7 = \phi_7$. *The messages sent during the collect algorithm are:*

- $\mu_{1\rightarrow 2} = \psi_1^{\downarrow\lambda(1)\cap\lambda(2)} = (\phi_1 \otimes \phi_2)^{\downarrow\{T\}}$
- $\mu_{2\rightarrow 5} = (\psi_2 \otimes \mu_{1\rightarrow 2})^{\downarrow\lambda(2)\cap\lambda(5)} = (\phi_5 \otimes \mu_{1\rightarrow 2})^{\downarrow\{E,L\}}$
- $\mu_{3\rightarrow 6} = \psi_3^{\downarrow\lambda(3)\cap\lambda(6)} = (\phi_3 \otimes \phi_4 \otimes \phi_8)^{\downarrow\{B,L\}}$
- $\mu_{4\rightarrow 5} = \psi_4^{\downarrow\lambda(4)\cap\lambda(5)} = \phi_6^{\downarrow\{E\}}$
- $\mu_{5\rightarrow 6} = (\psi_5 \otimes \mu_{2\rightarrow 5} \otimes \mu_{4\rightarrow 5})^{\downarrow\lambda(5)\cap\lambda(6)} = (e_{\{E,L\}} \otimes \mu_{2\rightarrow 5} \otimes \mu_{4\rightarrow 5})^{\downarrow\{E,L\}}$
- $\mu_{6\rightarrow 7} = (\psi_6 \otimes \mu_{3\rightarrow 6} \otimes \mu_{5\rightarrow 6})^{\downarrow\lambda(6)\cap\lambda(7)} = (e_{\{B,E,L\}} \otimes \mu_{3\rightarrow 6} \otimes \mu_{5\rightarrow 6})^{\downarrow\{B,E\}}$

At the end of the message-passing, the content of node 7 is:

$$\psi_7 \otimes \mu_{6\rightarrow 7} \quad = \quad \phi_6 \otimes \mu_{6\rightarrow 7}$$

and, according to Theorem 3.6, the query $\{B, E\}$ *is obtained by one last projection:*

$$\phi^{\downarrow\{B,E\}} \quad = \quad (\psi_7 \otimes \mu_{6\rightarrow 7})^{\downarrow\{B,E\}}.$$

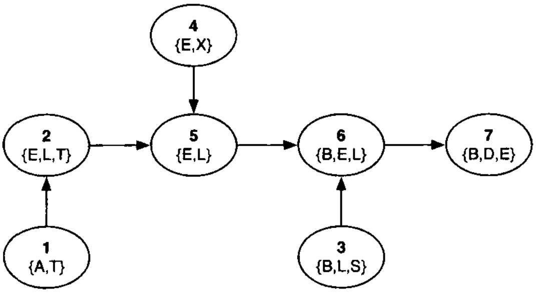


Figure 3.15 A complete run of the collect algorithm.

3.8.1 The Complexity of the Collect Algorithm

In the complexity study of the fusion algorithm, we brought forward the argument that all computations take place within the nodes of the join tree whose labels bound the domains of all intermediate results. Therefore, the treewidth of the inference problem, which reflects the minimum size of the largest node label over all possible covering join trees, became the decisive complexity factor. This argumentation was based on the interpretation of the fusion algorithm as message-passing scheme on a (particular) covering join tree. Exactly the same statement can also be made for the time complexity of the collect algorithm. However, the decoupling of the join tree construction process from the actual local computation gives us considerably more liberty in choosing an appropriate covering join tree for a given inference problem. One could therefore think that perhaps other methods exist that lead to better join trees than variable elimination can produce. The answer to this question has already been given in Section 3.7 since for a join tree of fixed treewidth, there always exists a variable elimination sequence producing a join tree of equal treewidth (Arnborg, 1985; Dechter & Pearl, 1989). The only difference with respect to the time complexity of fusion given in equation (3.19) is that the number of join tree nodes is no more proportional to the number of variables in the inference problem plus the number of knowledgebase factors. This results from the decoupling of join tree construction from local computation. Thus, recall that each node combines all incoming messages to its content. There are $|E| = |V| - 1$ edges, hence $|V| - 1$ combinations to execute. Additionally, each new message is obtained by projecting the current node content to the separator. This adds another $|V| - 1$ projections which altogether gives $2(|V| - 1)$ operations. We therefore obtain for the time complexity of the collect algorithm

$$\mathcal{O}\Big(|V| \cdot f(\omega^* + 1)\Big). \tag{3.37}$$

In the derivation of the time complexity for the fusion algorithm, the number of operations was bounded by $m + |d(\phi)|$ with m being the number of knowledgebase factors. It is fairly common for complexity considerations to assume a factor assignment mapping of the collect algorithm that assigns at most one knowledgebase factor to each join tree node. This ensures according to equation (3.30) that no combinations of knowledgebase factors are executed before the collect algorithm is started, since this would clearly falsify the complexity analysis of the latter. Therefore, similar assumptions will also be made for the complexity analysis of all subsequent local computation schemes. We then have $m \leq |V|$ and since $|V| \approx |d(\phi)|$ in the fusion algorithm, we conclude that both algorithms have the same time complexity.

Also, the space complexity is still similar to the message-passing implementation of the fusion algorithm given in equation (3.29), although the messages are generally much smaller. The reason is that each node combines incoming messages to its node content. Then, the message can be discarded that makes its smaller size irrelevant. Instead, we still keep one factor per node in memory whose size is bounded by the node label. Thus, we end with the following space complexity:

$$\mathcal{O}\left(|V| \cdot g(\omega^* + 1)\right). \tag{3.38}$$

3.8.2 Limitations of the Collect Algorithm

In Section 3.2.6 we addressed two important limitations of the fusion algorithm which should now be discussed from the viewpoint of collect. Collect is entirely based on projection and can therefore be applied to valuations taking domains from an arbitrary lattice. However, in such cases we need more general prerequisites since the definition of a join tree is still based on variables. The concepts generalizing join trees are *Markov trees* where the notion of *conditional independence* between elements of a general lattice replaces the running intersection property (Kohlas & Monney, 1995). We therefore conclude that the collect algorithm is, from this point of view, more generally applicable than the fusion algorithm.

The attentive reader may have noticed that we expressed the statement about the generality of the collect algorithm with great care. In fact, the version of the collect algorithm given here can be applied to valuation algebras on general lattices but, on the other hand, requires neutral elements for the initialization of join tree nodes. This in turn is not required for the fusion algorithm, and it is indeed an extremely limiting requirement. We have for example seen in Instance 3.4 that neutral elements in the valuation algebra of relations often correspond to infinite tables which can hardly be used to initialize join tree nodes. Even worse, the valuation algebra of density functions from Instance 3.5 does not provide neutral elements at all. This dependency of the collect algorithm on neutral elements was tacitly accepted for many years. It can, however, be avoided. A general solution that further increases the efficiency of the collect algorithm has been proposed by (Schneuwly *et al.*,2004). We know from Lemma 3.5 that neutral elements also behave neutral with respect to valuations of larger domains. Thus, the neutral element $e_\emptyset$ behaves neutral with respect to all valuations $\phi \in \Phi$ and could therefore be used to initialize join tree nodes. Instead of a neutral element per domain, we only need a single neutral element for the empty domain. Moreover, we will see in the following section that an element with this property can always be adjoined to any valuation algebra, if the latter does not provide neutral elements. This element is called *identity element* and its use finally leads to a general collect algorithm that can be applied to every valuation algebra without any restriction. A particular implementation of this algorithm will also exempt the message-passing conception from its inefficient space complexity.

3.9 ADJOINING AN IDENTITY ELEMENT

We are going to show in this section that a single identity element can be adjoined to any valuation algebra. This identity element then replaces neutral elements in the generalized version of the collect algorithm presented in Section 3.10. Let $\langle \Phi, D \rangle$

be a valuation algebra according to Definition 1.1. We add a new valuation e to Φ and denote the resulting system by $\langle \Phi', D \rangle$. Labeling, combination and projection are extended from Φ to Φ' in the following way:

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

If another element e' with the identical property that $e' \otimes \phi = \phi \otimes e' = \phi$ for all $\phi \in \Phi'$ already exists in Φ', there is no need to perform the extension. This is in particular the case if the valuation algebra provides a neutral element for the empty domain. We will next see that the proposed extension of $\langle \Phi, D \rangle$ conserves the properties of a valuation algebra. The simple proof of this lemma is given in (Schneuwly *et al.*,2004).

Lemma 3.9 *$\langle \Phi', D \rangle$ with extended operations d', $\otimes'$ and $\downarrow'$ is a valuation algebra.*

If there is no danger of confusion, we usually identify the operations in $\langle \Phi', D \rangle$ with their counterparts in $\langle \Phi, D \rangle$. Doing so, we subsequently write d for d', $\otimes$ for $\otimes'$ and $\downarrow$ for $\downarrow'$.

3.10 THE GENERALIZED COLLECT ALGORITHM

Finally, in this section, we derive the most general local computation scheme for the solution of single-query inference problems. It essentially corresponds to the collect algorithm of Section 3.8 but avoids the use of neutral elements. Instead, we assume a valuation algebra $\langle \Phi, D \rangle$ with an identity element $e \in \Phi$. The first occurrence of neutral elements in the collect algorithm is in equation (3.30) that defines the join tree factorization or, in other words, the initial node content for the message-passing. This equation can easily be rewritten using the identity element:

Definition 3.13 *Let a be an assignment mapping for a factorization ϕ regarding the nodes V of a covering join tree (V, E, λ, D). The* node factor *$i \in V$ assigned by a is*

$$\psi_i \quad = \quad e \otimes \bigotimes_{j:a(j)=i} \phi_j. \tag{3.39}$$

Also, the important statement of Lemma 3.8 still holds under this new setting:

$$\bigotimes_{i \in V} \psi_i = \phi_1 \otimes \ldots \otimes \phi_m \otimes \bigotimes_{i \in V} e = \phi_1 \otimes \ldots \otimes \phi_m \otimes e = \phi \otimes e = \phi.$$

In the subsequent sections, we always consider this modified factorization as join tree factorization. Each factor ψ_i corresponds either to a single factor of the original knowledgebase, a combination of some of them, or to the identity element if the factor set of the combination in equation (3.39) is empty. However, there is one essential difference between the two factorizations of equation (3.30) and (3.39). When neutral elements are used, it is always guaranteed that the domain of the factors covers the join tree node label entirely, i.e. we have $d(\psi_i) = \lambda(i)$. This is a consequence of the neutral element used in equation (3.30). On the other hand, if the identity element is used to initialize join tree nodes, we only have $d(\psi_i) \subseteq \lambda(i)$. It is therefore important to distinguish between the node label $\lambda(i)$ and the *node domain* $d(\psi_i)$: the node domain refers to the domain of the factor that is actually kept by the current node and this value may grow when incoming messages are combined to the node content. The label, on the other hand, represents the largest possible domain of a factor that would fit into this node and always remains constant.

Example 3.21 *The factors of Example 3.16 change only marginally under this modified definition. We now have* $\psi_5 = \psi_6 = e$.

The generalized collect algorithm for the solution of single-query inference problems without neutral elements can now be described by the following rules:

R1: Each node sends a message to its child when it has received all messages from its parents. This implies that leaves can send their messages right away.

R2: When a node is ready to send, it computes the message by projecting its current content to the intersection of its domain and its child's node label.

R3: When a node receives a message, it updates its current content by combining it with the incoming message.

One has to look closely to detect the difference to the collect algorithm given in Section 3.8. In fact, only Rule 2 changed in a small but important way. Remember, the node domain may now be smaller than the node label due to the use of identity elements. If node i would project its node factor to the separator $\lambda(i) \cap \lambda(ch(i))$, then this could lead to an undefined operation since $\lambda(i) \cap \lambda(ch(i)) \subseteq d(\psi_i)$ does not necessarily hold anymore. Instead, the factor is projected to $d(\psi_i) \cap \lambda(ch(i))$. In Section 3.8 we already introduced a notation to incorporate the dynamic behavior of the node factors during the message-passing. This notation must now be extended to take the growing node domains into consideration:

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

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

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

As before, the particular way of numbering the nodes of the directed join tree implies that at step i, node i can send a message to its child. This allows the following specification of the generalized collect algorithm.

- At step i, node i computes the message

$$\mu_{i\to ch(i)} \quad = \quad \psi_i^{(i)\downarrow\omega_i^{(i)}\cap\lambda(ch(i))}. \tag{3.40}$$

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

- The receiving node $ch(i)$ combines the message with its node content:

$$\psi_{ch(i)}^{(i+1)} \quad = \quad \psi_{ch(i)}^{(i)} \otimes \mu_{i\to ch(i)}. \tag{3.41}$$

  Its node domain changes to:

$$\omega_{ch(i)}^{(i+1)} \;=\; d(\psi_{ch(i)}^{(i+1)}) \;=\; \omega_{ch(i)}^{(i)} \cup \left(\omega_i^{(i)} \cap \lambda(ch(i))\right). \tag{3.42}$$

  The content of all other nodes does not change at step i,

$$\psi_j^{(i+1)} \quad = \quad \psi_j^{(i)} \tag{3.43}$$

  for all $j \neq ch(i)$, and the same holds for the node domains:

$$\omega_j^{(j+1)} \quad = \quad \omega_j^{(i)}. \tag{3.44}$$

The justification of the collect algorithm is formulated in the following theorem. Remember that in case of single-query inference problems, the root node is always chosen in such a way that it covers the query.

Theorem 3.7 *At the end of the generalized collect algorithm, the root node $r = |V|$ contains the marginal of ϕ relative to $\lambda(r)$,*

$$\psi_r^{(r)} \quad = \quad \phi^{\downarrow\lambda(r)}. \tag{3.45}$$

The proof of this theorem is given in the appendix of this chapter. We should also direct our attention to the implicit statement of this theorem that the node domain of the root node $d(\psi_r^{(r)})$ always corresponds to its node label $\lambda(r)$ at the end of the collect algorithm. This is again a consequence of the third requirement in Definition 3.8 as shown in the proof of the collect algorithm. We then obtain the solution of the single-query inference problem by one last projection:

$$\phi^{\downarrow x} \quad = \quad \left(\phi^{\downarrow\lambda(r)}\right)^{\downarrow x}. \tag{3.46}$$

Algorithm 3.4 The Generalized Collect Algorithm

```
input:  $(V, E, \lambda, D)$,  $x \subseteq d(\phi)$   output:  $(\phi_1 \otimes \ldots \otimes \phi_m)^{\downarrow x}$

begin
  for $i = 1 \ldots |V| - 1$ do
    $\mu_{i \rightarrow ch(i)} := \psi_i^{\downarrow d(\psi_i) \cap \lambda(ch(i))}$;
    $\psi_{ch(i)} := \psi_{ch(i)} \otimes \mu_{i \rightarrow ch(i)}$;
  end;
  return $\psi_r^{\downarrow x}$;
end
```

Example 3.22 *Example 3.21 gives a modified join tree factorization based on the use of identity elements instead of neutral elements. Since this concerns only the two factors ψ_5 and ψ_6, only the messages $\mu_{5 \rightarrow 6}$ and $\mu_{6 \rightarrow 7}$ change with respect to Example 3.20.*

$$\textbf{5:}\quad \mu_{5 \rightarrow 6} = (\psi_5 \otimes \mu_{2 \rightarrow 5} \otimes \mu_{4 \rightarrow 5})^{\downarrow \omega_5^{(5)} \cap \lambda(6)} = (e \otimes \mu_{2 \rightarrow 5} \otimes \mu_{4 \rightarrow 5})^{\downarrow \{E,L\}}$$

$$\textbf{6:}\quad \mu_{6 \rightarrow 7} = (\psi_6 \otimes \mu_{3 \rightarrow 6} \otimes \mu_{5 \rightarrow 6})^{\downarrow \omega_6^{(6)} \cap \lambda(7)} = (e \otimes \mu_{3 \rightarrow 6} \otimes \mu_{5 \rightarrow 6})^{\downarrow \{B,E\}}$$

The collect algorithm based on identity elements instead of neutral elements is the most general and fundamental local computation scheme for the solution of single-query inference problems. Moreover, this algorithm will play an important part in the following chapter in which local computation procedures for multi-query inference problems are studied. It turns out that many algorithmic insights concerning the computation of multiple queries can be derived from the correctness of the collect algorithm. We therefore list some further properties for later use, starting with a partial result of the generalized collect theorem for each subtree:

Lemma 3.10 *At the end of the generalized collect algorithm, node i contains*

$$\psi_i^{(i)} = \left(\bigotimes_{j \in \mathcal{T}_i} \psi_j \right)^{\downarrow \omega_i^{(i)}}. \tag{3.47}$$

The proof is given in the appendix of this chapter. Comparing this result with Theorem 3.7 indicates that only the label of the root node is guaranteed to be filled after the generalized collect algorithm, because it is not sure that all the variables in $\lambda(i)$ are covered by some factors in the current subtree. However, as the following lemma states, the node labels can be scaled down to coincide with the node domain and the resulting tree will again be a join tree.

Lemma 3.11 *At the end of the collect algorithm executed on a join tree $\mathcal{T} = (V, E, \lambda, D)$, the labeled tree $\mathcal{T}^* = (V, E, \lambda^*, D)$ with*

$$\lambda^*(i) = \omega_i^{(i)} = \omega_i^{(r)}$$

for $i = 1, \ldots, r$ is still a covering join tree for the projection problem.

Proof: It will be shown that the running intersection property is still satisfied between the nodes of the labeled tree $\mathcal{T}^*$. Let i and j be two nodes whose reduced labels contain X, i.e. $X \in \lambda^*(i) \cap \lambda^*(j)$. There exists a node k with $X \in \lambda(k)$ and $i, j \leq k$, since $X \in \lambda(i) \cap \lambda(j)$. By the same token, $X \in \lambda(ch(i))$. Then, from

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

follows that $X \in \lambda^*(ch(i))$ and by induction along the path from i to k, $X \in \lambda^*(k)$. The same argument applies to the nodes on the path from j to k and therefore the running intersection property holds for $\mathcal{T}^*$. The fact that $\mathcal{T}^*$ covers the original knowledgebase factors follows directly from $\omega_i \subseteq \omega_i^{(r)}$. ■

The last property highlights the relationship between node domains and labels:

Lemma 3.12 *It holds that*

$$\omega_i^{(r)} \cap \omega^{(r)}_{ch(i)} \quad = \quad \omega_i^{(r)} \cap \lambda(ch(i)). \tag{3.48}$$

Proof: The left-hand part of this equation is clearly contained in the right-hand part, because $\omega^{(r)}_{ch(i)} \subseteq \lambda(ch(i))$. Remembering that $\omega_i^{(i)} = \omega_i^{(r)}$ for $i = 1, \ldots, r$ and that $\omega_i^{(j)}$ are non-decreasing in $j = 1, 2 \ldots$ the reversed inclusion is derived as follows:

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

The last equality follows from $\omega^{(i)}_{ch(i)} \subseteq \omega^{(r)}_{ch(i)} \subseteq \lambda(ch(i))$. ■

3.10.1 Discussion of the Generalized Collect Algorithm

The generalized collect algorithm is the most general and fundamental local computation scheme for the solution of single-query inference problems. It depends neither on a particular lattice such as the fusion algorithm, nor on the existence of neutral elements. Consequently, this algorithm can be applied to any valuation algebra and it is therefore also qualified to serve as a starting point for the development of further local computation procedures in the following chapters. Besides its generality, there is a second argument for the use of the generalized collect algorithm. Using neutral elements to initialized join tree nodes blows the node content up until its domain agrees with the node label. All operations performed during the message-passing then take place on this maximum domain. On the other hand, using the identity element keeps the node domains as small as possible with the node label as upper bound. This makes the generalized collect algorithm more efficient, although its general complexity is not affected by this optimization. However, the real benefit of using the generalized collect algorithm becomes apparent when dealing with multi-query inference problems. We therefore refer to Section 4.1.3 in which the discussion about this performance gain is reconsidered and illustrated.

3.10.2 The Complexity of the Generalized Collect Algorithm

We have just mentioned that the use of the identity element instead of neutral elements for the initialization of join tree nodes does not change the overall complexity of the collect algorithm. Computations still take place on the factor domains that are bounded by the largest node label, and incoming messages are still combined to the node content. This repeats the two main arguments that lead to the complexity bounds given in equation (3.37) and (3.38). However, there is a technique to reduce the space complexity of the collect algorithm coming from the Shenoy-Shafer architecture that will be introduced in the following chapter. We first point out that the space requirement of the identity element can be neglected since it has been adjoined to the valuation algebra (Pouly & Kohlas, 2005). If we furthermore stress a factor distribution that assigns at most one knowledgebase factor to each join tree node, the node initialization process does not require any additional memory. This is again in contrast to the use of neutral elements for initialization. We next assume that each node has a separate *mailbox* for all its parents. Then, instead of combining a message directly to the node content, it is simply stored in the corresponding mailbox of the child node. Once all mailboxes are full, a node computes the message for its child node by combining its still original node content with all the messages of its mailboxes. Clearly, this is equal to the node content just before computing the message $\mu_{i \rightarrow ch(i)}$ in the above scheme. Finally, we project this valuation as proposed in equation (3.40) and send the result to the mailbox of the child node. It is clear that the statement of Theorem 3.7 (and Lemma 3.10) must be adapted under this optimization. At the end of the message-passing, the root node does not contain the projection of the objective function to its node label directly, but this result must first be computed by combining its still original node content with all the messages of its mailboxes. We pass on a formal proof for the correctness of this particular implementation of the generalized collect algorithm at this point, since it will be given in Section 4.1.1. Instead, we now analyze its impact on the space complexity: this procedure does not combine messages to the content of the receiving nodes but stores them in mailboxes and computes their combination only as an intermediate result of the message creation process, or to return the final query answer. Then, as soon as the message has been sent, this intermediate result can be discarded. Thus, instead of keeping $|V|$ node factors, we only store $|E| = |V| - 1$ messages. According to equation (3.40) the size of a message sent from node i to node $ch(i)$ is always bounded by the size of the corresponding separator $sep(i) = \lambda(i) \cap \lambda(ch(i))$. Consequently, the space of all messages in a complete run of the collect algorithm are bounded by the largest separator size in the join tree called its *separator width*. Similarly to the treewidth, we define the *separator width of an inference problem* by the smallest separator width over all possible covering join trees and write sep^* for this measure. This leads to the following bound for the space complexity of this implementation:

$$\mathcal{O}\Big(g(\omega^* + 1) + |V| \cdot g(sep^*)\Big). \tag{3.49}$$

We conclude from the definition of the separator that $sep^* \leq \omega^*$ or, in other words, that the separator width of an inference problem is not larger than its treewidth. Moreover, in practical applications the separator width is often much smaller than the treewidth. This generalizes the space complexity derived from the fusion algorithm in equation (3.20) to arbitrary covering join trees. It will be shown in Section 5.6 that for certain valuation algebras, we can omit the explicit computation of the node content and directly derive the message from the original node factor and the messages in the mailboxes. Then, the first term in the equation (3.49) vanishes, and the mailbox implementation of the generalized collect algorithm indeed provides a noteworthy improvement regarding space complexity.

It corresponds to the relevant literature (Pouly, 2008; Schneuwly, 2007; Schneuwly *et al.*,2004) to introduce the generalized collect algorithm in the above manner, although the improvement based on mailboxes is clearly preferable for computational purposes. On the other hand, combining messages directly to the node content often simplifies the algebraic reasoning about the collect algorithm and makes it possible to apply its correctness proof to other local computation schemes. In the following chapter, we will introduce the Shenoy-Shafer architecture as a local computation technique to solve multi-query inference problems. This architecture is entirely based on the mailbox scheme, and if we only consider some part of its message-passing, we naturally obtain the mailbox implementation of the generalized collect algorithm. This allows us to refer to equation (3.49) when talking about the space complexity of the generalized collect algorithm. Finally, since the collect algorithm with neutral elements does not have any advantage over its generalized version, we subsequently refer to the second approach as the collect algorithm.

3.11 AN APPLICATION: THE FAST FOURIER TRANSFORM

In this last section, we focus on the application of local computation to the single-query inference problem derived from the discrete Fourier transform in Instance 2.6. There are several reasons why this exact problem has been chosen for a case study: from a historic perspective, local computation methods were introduced to deal with problems of exponential (or even higher) complexity. But a direct computation of the discrete Fourier transform given in equation (2.8) only requires $O(N^2)$ operations, if N denotes the number of signal samplings. Moreover, a clever arrangement of the operations even leads to a $O(N \cdot \log N)$ complexity known as the *fast Fourier transform*. In fact, there is a whole family of algorithms that share this improved complexity, such that the notion of a fast Fourier transform does not refer to a single algorithm but rather to all algorithms that solve the discrete Fourier transform with a time complexity of $O(N \cdot \log N)$. Our case study thus shows the effect of applying local computation to a polynomial problem.

3.12 The Fast Fourier Transform

Instance 2.6 transformed the discrete Fourier transform into a single-query inference problem by the introduction of variables. The next step therefore consists in finding a covering join tree for the inference problem of equation (2.9). This however is not a difficult task due to the very structured factor domains in the inference problem. This structure is illustrated in Figure 2.11 on page 44. Figure 3.16 shows a possible join tree for the discrete Fourier transform. There are $m + 1$ nodes, and we quickly remark that the running intersection property is satisfied. It is furthermore a covering join tree: the root node corresponds to the query $\{K_0, \ldots, K_{m-1}\}$ and the knowledgebase factors can be distributed as shown in Figure 3.16. Thus, executing the generalized collect algorithm on this join tree computes a discrete Fourier transform.

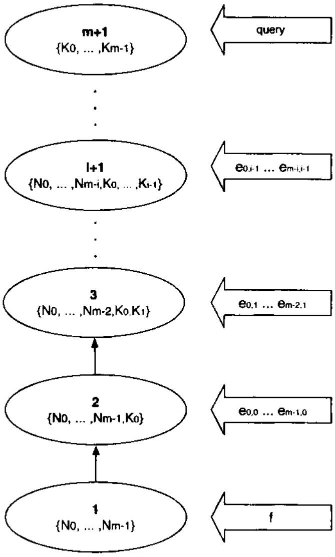


Figure 3.16 A possible covering join tree for the discrete Fourier transform.

Let us now analyse the time complexity of these computations. It is known from Section 3.10.2 that the time complexity of the generalized collect algorithm is bounded by

$$\mathcal{O}\left(|V| \cdot f(\omega^* + 1)\right).$$

We used the valuation algebra of (complex) arithmetic potentials in the derivation of the inference problem for the discrete Fourier transform. All variables are binary and we obtain a possible weight predictor from equation (3.17). Also, we have $|V| = m + 1$ nodes, and the largest node label has $m + 1$ variables. This specializes the complexity bound to:

$$\mathcal{O}\left(m \cdot 2^{m+1}\right).$$

In Instance 2.6 we chose $N = 2^m$, thus $m = \log N$ which finally results in

$$\mathcal{O}\left(N \cdot \log N\right).$$

This analysis shows that we obtain the time complexity of the fast Fourier transform by application of local computation. It is important to note that the local computation approach does not correspond to the classical scheme of J. Cooley and J. Tukey (Cooley & Tukey, 1965) that is based on the *divide-and-conquer* paradigm. Nevertheless, local computation reproduces the same complexity which makes it join the family of fast Fourier transforms. From a more general perspective, this result suggests that there may be good reasons to apply local computation techniques also to polynomial problems. This opens a new field of application for the theory in this book that has not yet been caught completely by the research community. Further applications of local computation to polynomial problems will be studied in Chapter 6 and 9 which both deal with formalism to model path problems in graphs.

3.12 CONCLUSION

This chapter was focused on the solution of single-query inference problems. We pointed out that the memory requirement of valuations as well as the time complexity of their operations generally increase with the size of the involved domains. This increase may be polynomial, but it is in many cases exponential or even worse which makes a direct computation of inference problems impossible. Instead, it is essential that algorithms confine in some way the size of all intermediate results that occur during the computations. This is the exact promise of local computation. The simplest local computation scheme is the fusion or bucket elimination algorithm. It is based on variable elimination instead of projection and is therefore only suitable for valuation algebras based on variable systems. The analysis of its time and space complexity identified the treewidth of the inference problem as the crucial complexity measure for local computation. It also turned out that especially the memory consumption of the fusion algorithm is often too high for practical purposes. A very promising approach to tackle these problems is to decouple join tree construction from the actual local computation process. We may then take an arbitrary covering join tree for the solution of a given inference problem and describe local computation as a message-passing scheme where nodes act as virtual processors that independently compute and transmit messages. Doing so, all local computation schemes become distributed algorithms. The decoupling process leads to the collect algorithm that is based on projection and therefore also applicable for valuations with domains from an arbitrary lattice. In its first occurrence, however, the collect algorithm presupposed the existence of neutral elements to initialize join tree nodes. In addition, it suffered from an even worse space complexity than the fusion algorithm. An algebraic solution for the dependence on neutral elements adjoins a unique identity element to the valuation algebra that is henceforth used for node initialization. Now, the collect algorithm can be applied to all formalisms that satisfy the valuation algebra axioms without any kind of restriction. Furthermore, computations generally take place on smaller domains and no unnecessary combinations with neutral elements are executed. The last addressed issue was the still unsatisfactory space complexity of the collect algorithm that comes from the algebraic description where each node combines incoming messages to its content. Alternatively, we proposed an implementation where messages are stored in mailboxes and only combined for the creation of the child message. Immediately after, the combination is again discarded. Consequently, the collect algorithm only stores messages that generally are much smaller than their combination as node content. The general space complexity reduces slightly under this setting, but for the valuation algebras we are going to study in Chapter 5, this reduction may be considerable. A similar caching policy based on mailboxes will take center stage in the following chapter where multi-query inference problems are treated. The last section of this chapter applied the collect algorithm to the inference problem of a discrete Fourier transform. Doing so, the treewidth complexity of local computation turns into the low polynomial complexity of the fast Fourier transform. This is a strong argument for our claim that local computation may also be useful to solve problems of polynomial nature.

Appendix: Proof of the Generalized Collect Algorithm

We prove Theorem 3.7 which is a generalization of Theorem 3.6. The following lemma will be useful for this undertaking:

Lemma C.13 *Define*

$$y_i \quad = \quad \bigcup_{j=i}^{r} \omega_j^{(i)} \qquad\qquad i = 1, \ldots, r. \tag{C.1}$$

Then, for $i = 1, \ldots, r-1$,

$$\left(\bigotimes_{j=i}^{r} \psi_j^{(i)}\right)^{\downarrow y_{i+1}} = \bigotimes_{j=i+1}^{r} \psi_j^{(i+1)} = \phi^{\downarrow y_{i+1}}. \tag{C.2}$$

Proof: First, it has to be ensured that $y_{i+1} \subseteq y_i$ holds in order to guarantee that the projection in equation (C.2) is well defined:

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

From equation (3.42) it follows that

$$\omega_{ch(i)}^{(i+1)} = \omega_{ch(i)}^{(i)} \cup (\omega_i^{(i)} \cap \lambda(ch(i))) \subseteq \omega_{ch(i)}^{(i)} \cup \omega_i^{(i)} \tag{C.3}$$

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

Next, the following property will be proved:

$$\omega_i^{(i)} \cap y_{i+1} = \omega_i^{(i)} \cap \lambda(ch(i)). \tag{C.4}$$

Assume first that $X \in \omega_i^{(i)} \cap \lambda(ch(i))$. Then, equation (C.3) implies that $X \in \omega_{ch(i)}^{(i+1)}$ and hence, by the definition of y_{i+1}, we have $X \in y_{i+1}$. Thus, $X \in \omega_i^{(i)} \cap y_{i+1}$. On the other hand, assume that $X \in \omega_i^{(i)} \cap y_{i+1}$. Then, by the running intersection property and the definition of y_{i+1}, $X \in \lambda(ch(i))$ and therefore $X \in \omega_i^{(i)} \cap \lambda(ch(i))$.

An immediate consequence of equation (3.42) and (3.43) is that

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

We therefore obtain by application of the combination axiom and Property (C.4):

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

This proves the first equality of (C.2). The second is shown by induction over i. For $i = 1$, the equation is satisfied since

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

Let us assume that the equation holds for i,

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

Then, by transitivity of marginalization,

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

which proves (C.2) for all i. ■

Theorem 3.7 can now be verified:

Proof of Theorem 3.7

Proof: By application of equation (C.2) for $i = r - 1$, it follows that

$$y_r = \omega_r^{(r)}. \tag{C.5}$$

It remains to prove that $\omega_r^{(r)} = \lambda(r)$. For this purpose, it is sufficient to show that if $X \in \lambda(r)$ then $X \in \omega_r^{(r)}$ since $\omega_r^{(r)} \subseteq \lambda(r)$. Let $X \in \lambda(r)$. Then, according to the definition of the covering join tree for an inference problem, there exists a factor ψ_j with $X \in d(\psi_j)$. ψ_j is assigned to node $p = a(j)$ and therefore $X \in \omega_p^{(p)}$. equation (3.41) implies that $X \in \omega_{ch(p)}^{(p+1)}$ and $X \in \omega_r^{(r)}$ follows by repeating this argument up to the root node r. ■

Proof of Lemma 3.10

Proof: Node i is the root of the subtree $\mathcal{T}_i$ and contains the marginal to $\omega_i^{(i)}$ of the factors associated with $\mathcal{T}_i$ due to equation (C.5). ■

PROBLEM SETS AND EXERCISES

C.1 ★ Propose a weight predictor for the valuation algebra of Gaussian potentials.

C.2 ★ Exercise B.1 in Chapter 2 asked to analyse the two inference problems from the discrete cosine transform and the inference problems from the discrete Fourier transform. Continue this analysis by showing that all three problems can be solved with the same covering join tree from Instance 3.12.

C.3 ★ Apply the collect algorithm to the inference problem of the Hadamard transform derived in Instance 2.5 and show that a clever choice of the covering join tree leads to the time complexity of the *fast Hadamard transform.* Indications are given in (Aji & McEliece, 2000; Gonzalez & Woods, 2006).

C.4 ★ If neutral elements are present in a valuation algebra $\langle \Phi, D \rangle$, we may introduce an operation of vacuous extension as shown in equation (3.24). Prove the following properties of vacuous extension:

1. if $x \subseteq y$ then $e_x^{\uparrow y} = e_y$;
2. if $d(\phi) = x$ then for all $y \in D$ we have $\phi \otimes e_y = \phi^{\uparrow x \cup y}$;
3. if $d(\phi) = x$ then $\phi^{\uparrow x} = \phi$;
4. if $d(\phi) = x$ and $y \subseteq x$, then $\phi \otimes e_y = \phi$;
5. if $d(\phi) = x$ and $x \subseteq y \subseteq z$, then $(\phi^{\uparrow y})^{\uparrow z} = \phi^{\uparrow z}$;
6. if $d(\phi) = x, d(\psi) = y$, and $x, y \subseteq z$, then $(\phi \otimes \psi)^{\uparrow z} = \phi^{\uparrow z} \otimes \psi^{\uparrow z}$;
7. if $d(\phi) = x, d(\psi) = y$, then $\phi \otimes \psi = \phi^{\uparrow x \cup y} \otimes \psi^{\uparrow x \cup y}$;
8. if $d(\phi) = x$ then $(\phi^{\uparrow x \cup y})^{\downarrow y} = (\phi^{\downarrow x \cap y})^{\uparrow y}$.

The solution to this exercise is given in (Kohlas, 2003), Lemma 3.1.

C.5 ★ In a valuation algebra with neutral element $\langle \Phi, D \rangle$ we may define a *transport operation* as follows: For $\phi \in \Phi$ with $d(\phi) = x$ and $y \in D$,

$$\phi^{\rightarrow y} = (\phi^{\uparrow x \cup y})^{\downarrow y}. \tag{C.6}$$

Due to Property 8 in Exercise C.4 we also have

$$\phi^{\rightarrow y} = (\phi^{\downarrow x \cap y})^{\uparrow y}. \tag{C.7}$$

Prove the following properties of the transport operation:

1. if $d(\phi) = x$ then $\phi^{\rightarrow x} = \phi$;
2. if $d(\phi) = x$ and $x, y, \subseteq z$ then $\phi^{\rightarrow y} = (\phi^{\uparrow z})^{\downarrow y}$;
3. if $d(\phi) = x$ and $x \subseteq y$ then $\phi^{\rightarrow y} = \phi^{\uparrow y}$;
4. if $d(\phi) = x$ and $x \supseteq y$ then $\phi^{\rightarrow y} = \phi^{\downarrow y}$;

5. if $y \subseteq z$ then $\phi^{\rightarrow y} = (\phi^{\rightarrow z})^{\rightarrow y}$;

6. For each ϕ and all x, y we have $(\phi^{\rightarrow x})^{\rightarrow y} = (\phi^{\rightarrow x \cap y})^{\rightarrow y}$;

7. if $d(\phi) = x$ and $d(\psi) = y$ then $(\phi \otimes \psi)^{\rightarrow x} = \phi \otimes \psi^{\rightarrow x}$.

The solution to this exercise is given in (Kohlas, 2003), Lemma 3.2.

C.6 ★ Let $\langle \Phi, D \rangle$ be a stable valuation algebra according to Definition 3.6. Based on the transport operation of equation (C.6), we further define the following relation: For $\phi, \psi \in \Phi$ with $d(\phi) = x$ and $d(\psi) = y$ we have

$$\phi \equiv \psi \quad \text{if, and only if,} \quad \phi^{\rightarrow y} = \psi \text{ and } \psi^{\rightarrow x} = \phi. \tag{C.8}$$

a) Prove that this relation is an *equivalence relation* satisfying the properties $\phi \equiv \phi$ (reflexivity), $\phi \equiv \psi$ implies $\psi \equiv \phi$ (symmetry) and $\phi \equiv \psi$ and $\psi \equiv \eta$ implies that $\phi \equiv \eta$ (transitivity) for $\phi, \psi, \eta \in \Phi$. The solution to this exercise is given in (Kohlas, 2003), Lemma 3.3.

b) Prove that this relation is further a *congruence relation* with respect to the operations of combination and transport, i.e. verify the following properties. The solution to this exercise is given in (Kohlas, 2003), Theorem 3.4.

1. if $\phi_1 \equiv \psi_1$ and $\phi_2 \equiv \psi_2$ then $\phi_1 \otimes \phi_2 \equiv \psi_1 \otimes \psi_2$;

2. if $\phi \equiv \psi$ then it holds for all $z \in D$ that $\phi^{\rightarrow z} \equiv \psi^{\rightarrow z}$.

c) Let r be a set of variables and $D = \mathcal{P}(r)$ its powerset. Prove that for $\phi, \psi \in \Phi$ we have $\phi \equiv \psi$ if, and only if,

$$\phi^{\uparrow r} = \psi^{\uparrow r}. \tag{C.9}$$

C.7 ★★ Let $\langle \Phi, D \rangle$ be a stable valuation algebra with the congruence relation of equation (C.8). We next consider the induced equivalence classes defined for $\phi \in \Phi$ with $d(\phi) = x$ by $[\phi] = \{\psi \in \Phi : \psi^{\rightarrow x} \equiv \phi\}$. Intuitively, this groups valuations that express the same information, even if they have different domains. As a consequence of equation (C.9) all valuations of the same equivalence class are equal, if they are vacuously extended to the complete variable universe. The elements $[\phi]$ are therefore called *domain-free valuations*. The set of all equivalence classes forms an algebra $\langle \Psi, D \rangle$, called *quotient algebra*, with

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

Since the relation is a congruence, we may define the following operations in $\langle \Psi, D \rangle$:

1. *Combination:* $[\phi] \otimes [\psi] = [\phi \otimes \psi]$;

2. *Transport:* $[\phi]^{\rightarrow x} = [\phi^{\rightarrow x}]$.

These operations are well-defined since they do not depend on the chosen representative for the equivalence classes. Verify the following properties in $\langle \Psi, D\rangle$:

1. for all $x, y \in D$ we have $([\phi]^{\Rightarrow x})^{\Rightarrow y} = [\phi]^{\Rightarrow x \cap y}$;
2. for all $x \in D$ we have $([\phi]^{\Rightarrow x} \otimes [\psi])^{\Rightarrow x} = [\phi]^{\Rightarrow x} \otimes [\psi]^{\Rightarrow x}$;
3. for all $x \in D$ we have $[e_s]^{\Rightarrow x} = [e_s]$;
4. for all $[\phi] \in \Psi$ we have $[\phi]^{\Rightarrow r} = [\phi]$.

The solution to this exercise is given in (Kohlas, 2003), Theorem 3.5. Structures like $\langle \Psi, D\rangle$ are called *domain-free valuation algebras* and posses a rich theory which takes center stage in algebraic information theory.

C.8 $\star\star$ We have seen in Exercise C.7 that a domain-free valuation algebra can be obtained for every stable valuation algebra. Here, we propose the converse direction to derive a valuation algebra starting from a domain-free valuation algebra. Let r be a set of variables and $D = \mathcal{P}(r)$ its powerset lattice. We assume an arbitrary domain-free valuation algebra $\langle \Psi, D\rangle$ defined by two operations

1. *Combination*: $\Psi \times \Psi \rightarrow \Psi; (\phi, \psi) \mapsto \phi \otimes \psi$;
2. *Focusing*: $\Psi \times D \rightarrow \Psi; (\psi, x) \mapsto \psi^{\Rightarrow x}$,

satisfying the following axioms:

(D1) *Semigroup:* Ψ is associative and commutative under combination. There is a neutral element $e \in \Psi$ such that $e \otimes \psi = \psi \otimes e = \psi$ for all $\psi \in \Psi$.

(D2) *Transitivity:* For $\psi \in \Psi$ and $x, y \in D$,

$$(\psi^{\Rightarrow x})^{\Rightarrow y} \quad = \quad \psi^{\Rightarrow x \cap y}.$$

(D3) *Combination:* For $\psi, \phi \in \Psi$ and $x \in D$

$$(\psi^{\Rightarrow x} \otimes \phi)^{\Rightarrow x} \quad = \quad \psi^{\Rightarrow x} \otimes \phi^{\Rightarrow x}.$$

(D4) *Neutrality:* For $x \in D$,

$$e^{\Rightarrow x} \quad = \quad e.$$

(D5) *Support:* For $\psi \in \Psi$,

$$\psi^{\Rightarrow r} \quad = \quad \psi.$$

If for $\psi \in \Psi$ and $x \in D$ we have $\psi^{\Rightarrow x} = \psi$, then x is called a *support* of ψ. Due to Axiom (D5), r is a support of every valuation algebra and, according to (D4), every domain is a support of the neutral element. We next consider the set of pairs

$$\Psi^* \quad = \quad \{(\psi, x) : \psi \in \Phi \text{ and } \psi^{\Rightarrow x} = \psi\}. \tag{C.10}$$

These pairs can be considered as valuations labeled by their supports. Therefore, we define the following operations on $\langle \Psi^*, D \rangle$:

1. *Labeling*: For $(\psi, x) \in \Psi^*$ we define

$$d(\psi, x) \quad = \quad x. \tag{C.11}$$

2. *Combination*: For $(\phi, x), (\psi, y) \in \Psi^*$ we define

$$(\phi, x) \otimes (\psi, y) \quad = \quad (\phi \otimes \psi, x \cup y) \tag{C.12}$$

3. *Projection*: For $(\psi, x) \in \Psi^*$ and $y \subseteq x$ we define

$$(\psi, x)^{\downarrow y} \quad = \quad (\psi^{\Rightarrow y}, y). \tag{C.13}$$

Prove that $\langle \Psi^*, D \rangle$ together with these operations satisfy the axioms of a stable valuation algebra. Indications can be found in (Kohlas, 2003), Section 3.2. Putting these things together, we may start from a stable valuation algebra and derive a domain-free valuation algebra, from which we may again derive a stable valuation algebra. This last algebra is isomorph to the valuation algebra at the beginning of the process as shown in (Kohlas, 2003). Conversely, we may start from a domain-free valuation algebra, derive a stable valuation algebra and then again a domain-free valuation algebra. Again, the algebras at the beginning and end of this process are isomorph.

CHAPTER 4

COMPUTING MULTIPLE QUERIES

The foregoing chapter introduced three local computation algorithms for the solution of single-query inference problems, among which the generalized collect algorithm was shown to be the most universal and efficient scheme. We therefore take this algorithm as the starting point for our study of the second class of local computation procedures that solve multi-query inference problems. Clearly, the most simple approach to compute multiple queries is to execute the collect algorithm repeatedly for each query. Since we already gave a general definition of a covering join tree that takes an arbitrary number of queries into account, it is not necessary to take a new join tree for the answering of a second query on the same knowledgebase. Instead, we redirect and renumber the join tree in such a way that its root node covers the current query and execute the collect algorithm. This procedure is repeated for each query. In Figure 3.15, arrows indicate the direction of the message-passing for a particular root node and therefore reflect the restrictions imposed on the node numbering. It is easy to see that changing the root node only affects the arrows between the old and the new root node. This is shown in Figure 4.1 where we also observe that a valid node numbering for the redirected join tree is obtained by inverting the numbering on the path between the old and the new root node. Hence, it is possible to solve a multi-query inference problem by repeated execution of the collect algorithm on the

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

same join tree with only a small preparation between two consecutive runs. Neither the join tree nor its factorization have to be modified.

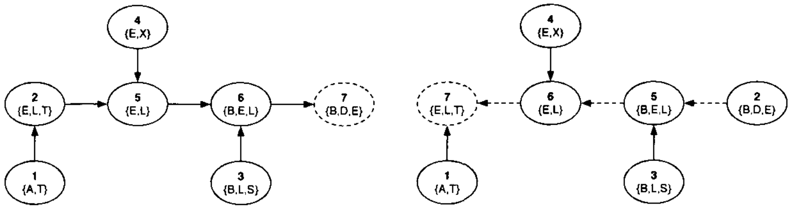


Figure 4.1 The left-hand join tree is rooted towards the node with label $\{B, D, E\}$. If then node $\{E, L, T\}$ is elected as new root node, only the dashed arrows between the old and the new root are affected. Their direction is inverted in the right-hand figure and the new node numbering is obtained by inverting the numbering between the old and the new root.

During a single run of the collect algorithm on a covering join tree (V, E, λ, D), exactly $|E|$ messages are computed and transmitted. Consequently, if a multi-query inference problem with $n \in \mathbb{N}$ queries is solved by repeated application of collect on the same join tree, $n|E|$ messages are computed and exchanged. It will be illustrated in a short while that most of these messages are identical and their computation can therefore be avoided by an appropriate caching policy. The arising algorithm is called *Shenoy-Shafer architecture* and will be presented in Section 4.1. It reduces the number of computed messages to $2|E|$ for an arbitrary number of queries. However, the drawback of this approach is that all messages have to be stored until the end of the total message-passing. This was the main reason for the development of more sophisticated algorithms that aim at a shorter lifetime of messages. These algorithms are called *Lauritzen-Spiegelhalter architecture*, *HUGIN architecture* and *idempotent architecture* and will be introduced in Section 4.3 to Section 4.5. But these alternative architectures require more structure in the underlying valuation algebra and are therefore less generally applicable than the Shenoy-Shafer architecture. More concretely, they presuppose some concept of *division* that will be defined in Section 4.2. Essentially, there are three requirements for introducing a division operator, namely either *separativity*, *regularity* or *idempotency*. These considerations require more profound algebraic insights that are not necessary for the understanding of division-based local computation algorithms. We therefore delay their discussion to the appendix of this chapter. A particular application of division in the context of valuation algebras is *scaling* or *normalization*. If, for example, we want to interpret arithmetic potentials from Instance 1.3 as discrete probability distributions, or set potentials from Instance 1.4 as Dempster-Shafer belief functions, normalization becomes an important issue. Based on the division operator, scaling or normalization can be introduced on a generic, algebraic level as explored in Section 4.7. Moreover, it is then also required that the solution of inference problems gives scaled results. We therefore focus in Section 4.8 on how the local computation architectures of this chapter can be adapted to deliver scaled results directly.

4.1 THE SHENOY-SHAFER ARCHITECTURE

The initial situation for the Shenoy-Shafer architecture and all other local computation schemes of this chapter is a covering join tree for the multi-query inference problem according to Definition 3.8. Join tree nodes are initialized using the identity element, and the knowledgebase factors are combined to covering nodes as specified by equation (3.39). Further, we shall see that local computation algorithms for the solution of multi-query inference problems exchange messages in both directions of the edges. It is therefore no longer necessary to direct join trees. Let us now reconsider the above idea of executing the collect algorithm repeatedly for each query on the same join tree. Since only the path from the old to the new root node changes between two consecutive runs, only the messages of these region are affected. All other messages do not change but are nevertheless recomputed.

Example 4.1 *Assume that the collect algorithm has been executed on the left-hand join tree of Figure 4.1. Then, the root node is changed and collect is restarted for the right-hand join tree. It is easy to see that the messages $\mu_{1\rightarrow 7}$, $\mu_{4\rightarrow 6}$ and $\mu_{3\rightarrow 5}$ already existed in the first run of the collect algorithm. Only the node numbering changed but not their content.*

A technique to benefit from already computed messages is to store them for later reuse. The Shenoy-Shafer architecture (Shenoy & Shafer, 1990) organises this caching by installing *mailboxes* between neighboring nodes which store the exchanged messages. In fact, this is similar to the technique applied in Section 3.10.2 to improve the space complexity of the collect algorithm. This time however, we assume two mailboxes between every pair of neighboring nodes since messages are sent in both directions. Figure 4.2 schematically illustrates this concept.

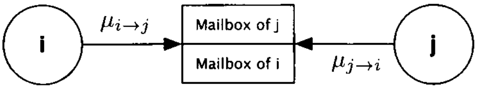


Figure 4.2 The Shenoy-Shafer architecture assumes mailboxes between neighboring nodes to store the exchanged messages for later reuse.

Then, the Shenoy-Shafer algorithm can be described by the following rules:

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

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

The algorithm stops when every node has received all messages from its neighbors. In order to specify this algorithm formally, a new notation is introduced that determines the domain of the valuation described in Rule 2. If i and j are neighbors, the domain of node i at the time where it sends a message to j is given by

$$\omega_{i\rightarrow j} \quad = \quad \omega_i \cup \bigcup_{k\in ne(i),j\neq k} d(\mu_{k\rightarrow i}), \tag{4.1}$$

where $d(\psi_i) = \omega_i$ is the domain of the node content of node i.

- The message sent from node i to node j is

$$\mu_{i\rightarrow j} \quad = \quad \left(\psi_i \otimes \bigotimes_{k\in ne(i),j\neq k} \mu_{k\rightarrow i}\right)^{\downarrow\omega_{i\rightarrow j}\cap\lambda(j)}. \tag{4.2}$$

Theorem 4.1 *At the end of the message-passing in the Shenoy-Shafer architecture, node i can compute*

$$\phi^{\downarrow\lambda(i)} \quad = \quad \psi_i \otimes \bigotimes_{j\in ne(i)} \mu_{j\rightarrow i}. \tag{4.3}$$

Proof: The important point is that the messages $\mu_{k\rightarrow j}$ do not depend on the actual schedule used to compute them. Due to this fact, any node $i \in V$ can be selected as root node. Then, the edges are directed towards this root and the nodes are renumbered. The message-passing towards the root corresponds to the collect algorithm and the proposition for node i follows from Theorem 3.7. ∎

Answering the queries of the inference problem from this last result demands one additional projection per query. Since each query x_i is covered by some node $j \in V$, we obtain the answer for this query by computing:

$$\phi^{\downarrow x_i} \quad = \quad \left(\phi^{\downarrow\lambda(j)}\right)^{\downarrow x_i}. \tag{4.4}$$

Example 4.2 *We execute the Shenoy-Shafer architecture on the join tree of Figure 4.3. There are 6 edges, thus 12 messages to be sent and stored in mailboxes. These messages are:*

- $\mu_{1\rightarrow 2} = \psi_1^{\downarrow\omega_{1\rightarrow 2}}$
- $\mu_{3\rightarrow 6} = \psi_3^{\downarrow\omega_{3\rightarrow 6}}$
- $\mu_{4\rightarrow 5} = \psi_4^{\downarrow\omega_{4\rightarrow 5}}$
- $\mu_{7\rightarrow 6} = \psi_7^{\downarrow\omega_{7\rightarrow 6}}$
- $\mu_{2\rightarrow 1} = (\psi_2 \otimes \mu_{5\rightarrow 2})^{\downarrow\omega_{2\rightarrow 1}}$
- $\mu_{6\rightarrow 3} = (\psi_6 \otimes \mu_{5\rightarrow 6} \otimes \mu_{7\rightarrow 6})^{\downarrow\omega_{6\rightarrow 3}}$
- $\mu_{5\rightarrow 4} = (\psi_5 \otimes \mu_{2\rightarrow 5} \otimes \mu_{6\rightarrow 5})^{\downarrow\omega_{5\rightarrow 4}}$

- $\mu_{6\to7} = (\psi_6 \otimes \mu_{3\to6} \otimes \mu_{5\to6})^{\downarrow\omega_{6\to7}}$
- $\mu_{2\to5} = (\psi_2 \otimes \mu_{1\to2})^{\downarrow\omega_{2\to5}}$
- $\mu_{5\to2} = (\psi_5 \otimes \mu_{4\to5} \otimes \mu_{6\to5})^{\downarrow\omega_{5\to2}}$
- $\mu_{5\to6} = (\psi_5 \otimes \mu_{4\to5} \otimes \mu_{2\to5})^{\downarrow\omega_{5\to6}}$
- $\mu_{6\to5} = (\psi_6 \otimes \mu_{7\to6} \otimes \mu_{3\to6})^{\downarrow\omega_{6\to5}}$

At then end of the message-passing, the nodes compute:

- $\phi^{\downarrow\{A,T\}} = \psi_1 \otimes \mu_{2\to1}$
- $\phi^{\downarrow\{E,L,T\}} = \psi_2 \otimes \mu_{1\to2} \otimes \mu_{5\to2}$
- $\phi^{\downarrow\{B,L,S\}} = \psi_3 \otimes \mu_{6\to3}$
- $\phi^{\downarrow\{E,X\}} = \psi_4 \otimes \mu_{5\to4}$
- $\phi^{\downarrow\{E,L\}} = \psi_5 \otimes \mu_{2\to5} \otimes \mu_{4\to5} \otimes \mu_{6\to5}$
- $\phi^{\downarrow\{B,E,L\}} = \psi_6 \otimes \mu_{5\to6} \otimes \mu_{3\to6} \otimes \mu_{7\to6}$
- $\phi^{\downarrow\{B,D,E\}} = \psi_7 \otimes \mu_{6\to7}$

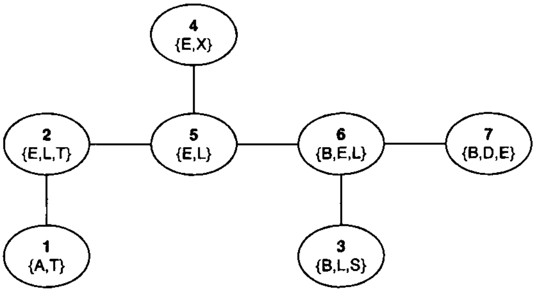


Figure 4.3 Since messages are sent in all directions in the Shenoy-Shafer architecture, join trees do not need to be rooted anymore.

Since the Shenoy-Shafer architecture accepts arbitrary valuation algebras, we may in particular apply it to formalisms based on variable elimination. This specialization is sometimes called *cluster-tree elimination*. In this context, a *cluster* refers to a join tree node together with the knowledgebase factors that are assigned to it. Further, we could use the join tree from the graphical representation of the fusion algorithm if we are, for example, only interested in queries that consist of single variables. Under this even more restrictive setting, the Shenoy-Shafer architecture is also called *bucket-tree elimination* (Kask *et al.*, 2001). Answering more general queries with the bucket-tree elimination algorithm is again possible if neutral elements are present. However, we continue to focus our studies on the more general and efficient version of the Shenoy-Shafer architecture introduced above.

4.1.1 Collect & Distribute Phase

For a previously fixed node numbering, Theorem 4.1 implies that

$$\phi^{\downarrow\lambda(i)} = \psi_i \otimes \bigotimes_{j \in pa(i)} \mu_{j\rightarrow i} \otimes \mu_{ch(i)\rightarrow i} = \psi_i^{(r)} \otimes \mu_{ch(i)\rightarrow i}. \tag{4.5}$$

This shows that the collect algorithm is extended by a single message coming from the child node in order to obtain the Shenoy-Shafer architecture. In fact, it is always possible to schedule a first part of the messages in such a way that their sequence corresponds to the execution of the collect algorithm. The root node $r \in V$ will then be the first node which has received the messages of all its neighbors. It suggests itself to name this first phase of the Shenoy-Shafer architecture *collect phase* or *inward phase* since the messages are propagated from the leaves towards the root node. It furthermore holds that

$$\omega_{i\rightarrow ch(i)} = \omega_i^{(i)}$$

for this particular scheduling. This finally delivers the proof of correctness for the particular implementation proposed in Section 3.10.2 to improve the space complexity of the collect algorithm. In fact, the collect phase of the Shenoy-Shafer architecture behaves exactly in this way. Messages are no more combined to the node content but stored in mailboxes, and the combination is only computed to obtain the message for the child node. Then, the result is again discarded, which guarantees the lower space complexity of equation (3.49). The messages that remain to be sent after the collect phase of the Shenoy-Shafer architecture constitute the *distribute phase*. Clearly, the root node will be the only node that may initiate this phase since it possesses all necessary messages after the collect phase. Next, the parents of the root node will be able to send their messages and this propagation continues until the leaves are reached. In fact, nodes can send their messages in the inverse order of their numbering. Therefore, the distribute phase is also called *outward phase*.

Lemma 4.1 *It holds that*

$$\lambda(i) = \omega_i^{(r)} \cup (\lambda(i) \cap \lambda(ch(i))). \tag{4.6}$$

Proof: We have

$$\begin{aligned}
\lambda(i) &= \lambda(i) \cup (\lambda(i) \cap \lambda(ch(i))) \\
&= \omega_i^{(r)} \cup \omega_{ch(i)\rightarrow i} \cup (\lambda(i) \cap \lambda(ch(i))) \\
&= \omega_i^{(r)} \cup (\lambda(i) \cap \lambda(ch(i))).
\end{aligned}$$

The second equality follows from equation (4.5). ■

The next lemma states that the eliminator $elim(i) = \lambda(i) - (\lambda(i) \cap \lambda(ch(i)))$ (see Definition 3.12) of each node $i \in V$ will be filled after the collect phase:

Lemma 4.2 *It holds that*

$$\omega_i^{(r)} - (\omega_i^{(r)} \cap \lambda(ch(i))) \quad = \quad \lambda(i) - (\lambda(i) \cap \lambda(ch(i))). \tag{4.7}$$

Proof: Assume X contained in the right-hand part of the equation. Thus, $X \in \lambda(i)$ but $X \notin \lambda(i) \cap \lambda(ch(i))$. From equation (4.6) can be deduced that $X \in \omega_i^{(r)}$. Since

$$\omega_i^{(r)} \cap \lambda(ch(i)) \quad \subseteq \quad \lambda(i) \cap \lambda(ch(i)),$$

$X \notin \omega_i^{(r)} \cap \lambda(ch(i))$. This proves that

$$\omega_i^{(r)} - (\omega_i^{(r)} \cap \lambda(ch(i))) \quad \supseteq \quad \lambda(i) - (\lambda(i) \cap \lambda(ch(i))).$$

Assume that X is contained in the left-hand part. So, $X \in \omega_i^{(r)} \subseteq \lambda(i)$ but $X \notin \omega_i^{(r)} \cap \lambda(ch(i))$ which in turn implies that $X \notin \lambda(ch(i))$ and consequently $X \notin \lambda(i) \cap \lambda(ch(i))$. Thus, $X \in \lambda(i) - (\lambda(i) \cap \lambda(ch(i)))$ and equality must hold. ■

The particular scheduling into a collect and distribute phase allows us to describe the intrinsically distributed Shenoy-Shafer architecture by a sequential algorithm. First, Algorithm 4.1 performs the complete message-passing. It does not return any value but ensures that all $\mu_{i \to ch(i)}$ and $\mu_{ch(i) \to i}$ are stored in the corresponding mailboxes. To answer a query $x \subseteq d(\phi)$ we then call Algorithm 4.2 which presupposes that the query is covered by some node in the join tree. The message-passing procedure can therefore be seen as a preparation step to the actual query answering.

Algorithm 4.1 Shenoy-Shafer Architecture: Message-Passing

```
input:  $(V, E, \lambda, D)$   output:  -

begin
  for $i = 1 \ldots |V| - 1$ do                                          // Collect Phase:
    $\omega$ := $d(\psi_i) \cup \bigcup_{j \in pa(i)} d(\mu_{j \to i})$;
    $\mu_{i \to ch(i)}$ := $(\psi_i \otimes \bigotimes_{j \in pa(i)} \mu_{j \to i})^{\downarrow \omega \cap \lambda(ch(i))}$;          // in mailbox
  end;
  for $i = |V| - 1 \ldots 1$ do                                      // Distribute Phase:
    $\omega$ := $d(\psi_i) \cup \bigcup_{j \in ne(ch(i)) \wedge j \neq i} d(\mu_{j \to i})$;
    $\mu_{ch(i) \to i}$ := $(\psi_{ch(i)} \otimes \bigotimes_{j \in ne(ch(i)) \wedge j \neq i} \mu_{j \to i})^{\downarrow \omega \cap \lambda(i)}$;     // in mailbox
  end;
end
```

Algorithm 4.2 Shenoy-Shafer Architecture: Query Answering

```
input:  $(V, E, \lambda, D)$,  $x \subseteq d(\phi)$   output:  $\phi^{\downarrow x}$

begin
  for $i = 1 \ldots |V|$ do
    if $x \subseteq \lambda(i)$ do
      return $(\psi_i \otimes \bigotimes_{j \in ne(i)} \mu_{j \to i})^{\downarrow x}$;
    end;
end
```

The implementation of Algorithm 4.2 corresponds to Theorem 4.1, but can easily be changed using equation (4.5). The combination of the node content with all parent messages has already been computed for the creation of the child message in the collect phase. If this partial result is still available, it is sufficient to combine it with the message obtained from the child node. Doing so, Algorithm 4.2 executes at most one combination. Clearly, under this modification, the collect phase becomes again equal to the original description of the collect algorithm in Section 3.10 since we again store the combination of the parent messages with the original node content for each node. This saves a lot of redundant computation time in the query-answering procedure since the combination of the original node factor with all arrived messages is already available, for the prize of an additional factor per node whose space is again bounded by the node label. Hence, we generally refer to such a situation as a *space-for-time trade*. As memory seems to be the more sensitive resource in many practical applications, it always depends on the context if a space-for-time trade is also a good deal. In the following, we discuss some further optimization issues.

4.1.2 The Binary Shenoy-Shafer Architecture

In a *binary join tree*, each node has at most three neighbors or, if we reconsider directed join trees, at most two parents. (Shenoy, 1997) remarked that binary join trees generally allow better performance for the Shenoy-Shafer architecture. The reason is that nodes with more than three neighbors compute a lot of redundant combinations. Figure 4.4 shows a non-binary join tree that covers the same knowledgebase factors and queries as the join tree of Figure 4.3. Further, this join tree has one node less such that only 10 messages instead of 12 are sent during the Shenoy-Shafer architecture. We list the messages sent by node 5:

$$\begin{aligned}
\mu_{5\to 2} &= (\psi_5 \otimes \mu_{4\to 5} \otimes \mu_{6\to 5} \otimes \mu_{3\to 5})^{\downarrow \omega_{5\to 2}}\\
\mu_{5\to 3} &= (\psi_5 \otimes \mu_{2\to 5} \otimes \mu_{4\to 5} \otimes \mu_{6\to 5})^{\downarrow \omega_{5\to 3}}\\
\mu_{5\to 4} &= (\psi_5 \otimes \mu_{2\to 5} \otimes \mu_{6\to 5} \otimes \mu_{3\to 5})^{\downarrow \omega_{5\to 4}}\\
\mu_{5\to 6} &= (\psi_5 \otimes \mu_{2\to 5} \otimes \mu_{4\to 5} \otimes \mu_{3\to 5})^{\downarrow \omega_{5\to 6}}
\end{aligned}$$

It is easy to see that some combinations of messages must be computed more than once. For comparison, let us also list the messages sent by node 5 in Figure 4.3 that only has three neighbors. Here, no combination is computed more than once:

$$\begin{aligned}
\mu_{5\to 2} &= (\psi_5 \otimes \mu_{4\to 5} \otimes \mu_{6\to 5})^{\downarrow \omega_{5\to 2}}\\
\mu_{5\to 4} &= (\psi_5 \otimes \mu_{2\to 5} \otimes \mu_{6\to 5})^{\downarrow \omega_{5\to 4}}\\
\mu_{5\to 6} &= (\psi_5 \otimes \mu_{2\to 5} \otimes \mu_{4\to 5})^{\downarrow \omega_{5\to 6}}
\end{aligned}$$

A second reason for dismissing non-binary join trees is that some computations may take place on larger domains than actually necessary. Comparing the two join trees in Figure 4.5, we observe that the treewidth of the binary join tree is smaller which naturally leads to a better time complexity. On the other hand, the binary join tree contains more nodes that again increases the space requirement. But this increase

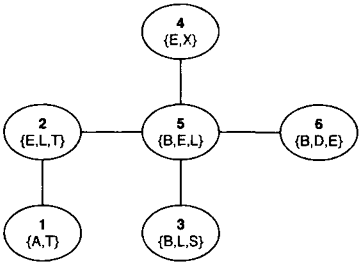


Figure 4.4 Using non-binary join trees creates redundant combinations.

is only linear since the node separators will not grow. It is therefore generally a good idea to use binary join trees for the Shenoy-Shafer architecture. Finally, we state that any join tree can be transformed into a binary join tree by adding a sufficient number of new nodes. A corresponding join tree binarization algorithm is given in (Lehmann, 2001). Further ways to improve the performance of local computation architectures by aiming to cut down on messages during the propagation are proposed by (Schmidt & Shenoy, 1998) and (Haenni, 2004).

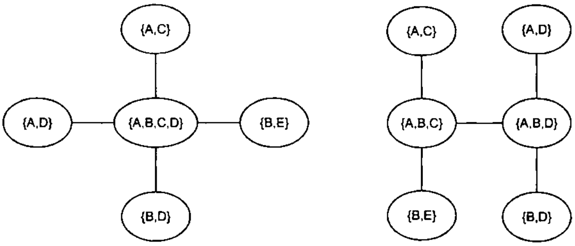


Figure 4.5 The treewidth of non-binary join trees is sometimes larger than necessary.

4.1.3 Performance Gains due to the Identity Element

It was foreshadowed multiple times in the context of the generalized collect algorithm that initializing join tree nodes with the identity element instead of neutral elements also increases the efficiency of local computation. In fact, the improvement is absolute and concerns both time and space resources. This effect also occurs when dealing with multiple queries. To solve a multi-query inference problem, we first ensure that each query is covered by some node in the join tree. In most cases, however, queries are very different from the knowledgebase factor domains. Therefore, their covering nodes often still contain the identity element after the knowledgebase factor distribution. In such cases, the performance gain caused by the identity element becomes important. Let us reconsider the knowledgebase of the medical example from Instance 2.1 and assume the query set $\{\{A,B,S,T\},\{D,S,X\}\}$. We solve this multi-query inference problem using the Shenoy-Shafer architecture and first build a join tree that covers the eight knowledgebase factors and the two queries. Further, we want to ensure that all computations take place during the run of the Shenoy-Shafer architecture, which requires that each join tree node holds at most one knowledgebase factor. Otherwise, a non-trivial combination would be executed prior to the Shenoy-Shafer architecture as a result of equation (3.39). A binary join tree that fulfills all these requirements is shown in Figure 4.6. Each colored node holds one of the eight knowledgebase factors, and the other (white) nodes store the identity element. We observe that the domain of each factor directly corresponds to the corresponding node label in this particular example.

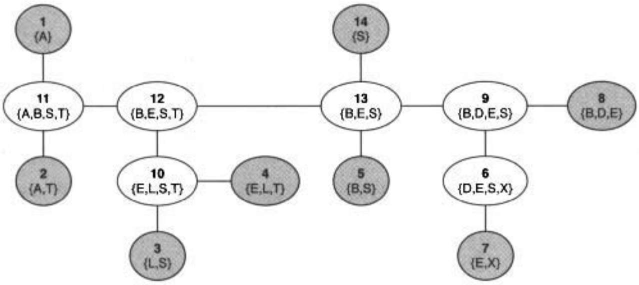


Figure 4.6 A binary join tree for the medical inference problem with queries $\{A, B, S, T\}$ and $\{D, S, X\}$. The colored nodes indicate the residence of knowledgebase factors.

We now execute the collect phase of the Shenoy-Shafer architecture and send messages towards node 14. In doing so, we are not interested in the actual value of the messages but only in the domain size of the total combination in equation (4.2) before the projection is computed. Since domains are bounded by the node labels, we color each node where the total label has been reached. The result of this process is shown in Figure 4.7. Interestingly, only two further nodes compute on their maximum domain size. All others process valuations with smaller domains. For example, the message sent by node 11 consists of its node content (i.e. the identity element), the message from node 1 (i.e. a valuation with domain $\{A\}$) and the message from node 2 (i.e. a valuation of domain $\{A, T\}$). Combining these three factors leads to a valuation of domain $\{A, T\} \subset \{A, B, S, T\}$. Therefore, this node remains uncolored in Figure 4.7. If alternatively neutral elements were used for initialization, each node content would be blown up to the label size, or, in other words, all nodes would be colored.

4.1.4 Complexity of the Shenoy-Shafer Architecture

The particular message scheduling that separates the Shenoy-Shafer architecture into a collect and a distribute phase suggests that executing the message-passing in the Shenoy-Shafer architecture doubles the effort of the (improved) collect algorithm. We may therefore conclude that the Shenoy-Shafer architecture adopts a similar

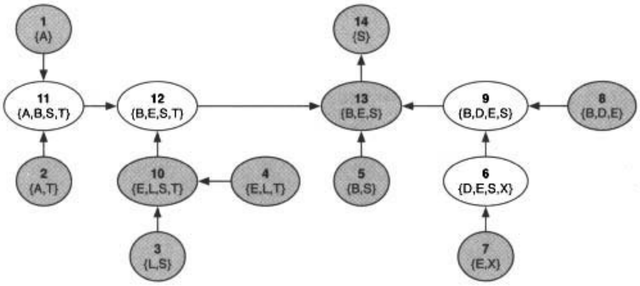


Figure 4.7 The behavior of node domains during the collect phase. Only the colored nodes deal with valuations of maximum domain size.

time and space complexity as the collect algorithm in Section 3.10.2. There are $2|E| = 2(|V| - 1)$ messages exchanged in the Shenoy-Shafer architecture. Each message requires one projection and consists of the combination of the original node content with all messages obtained from all other neighbors. This sums up to $|ne(i)| - 1$ combinations for each message. Clearly, the number of neighbors is always bounded by the degree of the join tree, i.e. we have $|ne(i)| - 1 \leq deg - 1$. In total, the number of operations executed in the message-passing of the Shenoy-Shafer architecture is therefore bounded by

$$2(|V| - 1)(1 + (deg - 1)) \quad \leq \quad 2|V|deg.$$

Thus, we obtain the following time complexity bound for the message-passing:

$$\mathcal{O}\Big(|V| \cdot deg \cdot f(\omega^* + 1)\Big). \tag{4.8}$$

It is clear that we additionally require $|ne(i)| - 1 \leq deg$ combinations and one projection to obtain the answer for a given query after the message-passing. But since every query must be covered by some node in the join tree, it is reasonable to bound the number of queries by the number of join tree nodes. The additional effort for query answering does therefore not change the above time complexity. Note also that the factor deg is a constant when dealing with binary join trees and disappears from equation (4.8), possibly at the expense of a higher number of nodes.

Regarding space complexity, we recall that the number of messages sent in the Shenoy-Shafer architecture equals two times the number of messages in the improved collect algorithm. Therefore, both algorithms have the same space complexity

$$\mathcal{O}\Big(g(\omega^* + 1) + |V| \cdot g(sep^*)\Big). \tag{4.9}$$

4.1.5 Discussion of the Shenoy-Shafer Architecture

Regarding time complexity, the improvement brought by the Shenoy-Shafer architecture compared to the repeated application of the collect algorithm as proposed in the introduction of this chapter is obvious. If we compute projections to all node labels in the join tree, we need $|V|$ executions of the collect algorithm, which result in an overall complexity of

$$\mathcal{O}\Big(|V|^2 \cdot f(\omega^* + 1)\Big).$$

Since the degree of a join tree is generally much smaller than its number of nodes, the Shenoy-Shafer architecture provides a considerable speedup. Both approaches further share the same asymptotic space complexity.

The Shenoy-Shafer architecture is the most general local computation scheme for the solution of multi-query inference problems since it can be applied to arbitrary valuation algebras without restrictions of any kind. Further, it provides the best possible asymptotic space complexity one can expect for a local computation procedure. Concerning the time complexity, further architectures for the solution of multi-query inference problems have been proposed that eliminate the degree from equation (4.8), even if the join tree is not binary. However, it was shown in (Lepar & Shenoy, 1998) that the runtime gain of these architectures compared to the binary Shenoy-Shafer architecture is often insignificant. Moreover, these alternative architectures generally have a worse space complexity, which recommends the application of the Shenoy-Shafer architecture whenever memory is the more problematic resource. On the other hand, the computation of messages becomes much easier in these architectures, and they also provide more efficient methods for query answering.

4.1.6 The Super-Cluster Architecture

A possible improvement that applies to the Shenoy-Shafer architecture consists in a *time-for-space trade* called *super-cluster scheme* or, if applied to variable systems, *super-bucket scheme* (Dechter, 2006; Kask *et al.*, 2001). If we recall that the space complexity of the Shenoy-Shafer architecture is determined by the largest separator in the join tree, we may simply merge those neighboring nodes that have a large separator in-between. Doing so, the node labels grow but large separators disappear as shown in Example 4.3. Clearly, it is again subject to the concrete complexity predictors f and g if this time-for-space trade is also a good deal.

Example 4.3 *The join tree shown on the left-hand side of Figure 4.8 has treewidth* 5 *and separator width* 3. *The separators are shown as edge labels. The largest separator* $\{B, C, D\}$ *is between node* 2 *and* 4. *Merging the two nodes leads to the join tree on the right-hand side of this of Figure 4.8. Now, the treewidth became* 6 *but the separator width is only* 2.

The subsequent studies of alternative local computation schemes are essentially based on the assumption that time is the more sensitive resource. These additional

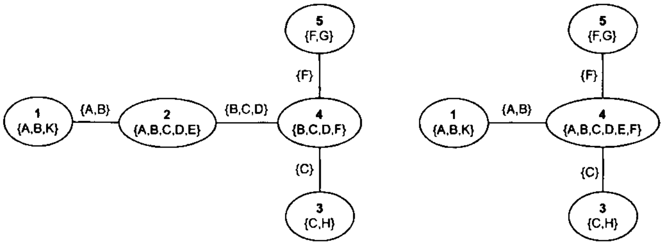


Figure 4.8 Merging neighboring nodes with a large separator in-between leads to better space complexity at the expense of time complexity.

architectures then simplify the message-passing by exploiting a division operator in the underlying valuation algebra which therefore requires more algebraic structure. The background for the existence of a division operation will next be examined.

4.2 VALUATION ALGEBRAS WITH INVERSE ELEMENTS

A particular important class of valuation algebras are those that contain an *inverse element* for every valuation $\phi \in \Phi$. Hence, these valuation algebras posses the notion of *division*, which will later be exploited for a more efficient message caching policy during local computation. Let us first give a formal definition of inverse elements in a valuation algebra $\langle \Phi, D \rangle$, based on the corresponding notion in semigroup theory.

Definition 4.1 *Valuations $\phi, \psi \in \Phi$ are called* inverses, *if*

$$\phi \otimes \psi \otimes \phi = \phi, \quad \text{and} \quad \psi \otimes \phi \otimes \psi = \psi. \tag{4.10}$$

We directly conclude from this definition that inverses necessarily have the same domain since the first condition implies that $d(\psi) \subseteq d(\phi)$ and from the second we obtain $d(\phi) \subseteq d(\psi)$. Therefore, $d(\phi) = d(\psi)$ must hold. It is furthermore suggested that inverse elements are not necessarily unique.

In Appendix D.1 of this chapter, we are going to study under which conditions valuation algebras provide inverse elements. In fact, the pure existence of inverse elements according to the above definition is not yet sufficient for the introduction of division-based local computation architectures since it only imposes a condition of combination but not on projection. For this purpose, (Kohlas, 2003) distinguishes three stronger algebraic properties that all imply the existence of inverses. *Separativity* is the weakest requirement among them. It allows the valuation algebra to be embedded into a union of groups that naturally include inverse elements. However, the price we pay is that full projection gets lost in the embedding algebra (see Appendix A.3 of Chapter 1 for valuation algebras with partial projection). Alternatively, *regularity* is a stronger condition than separativity and allows the valuation algebra to be decomposed into groups directly such that full projection is conserved. Finally, if the third and strongest requirement called *idempotency* holds, every valuation turns out to be the inverse of itself. The algebraic study of division demands notions from semigroup theory, although the understanding of these semigroup constructions is not absolutely necessary, neither for computational purposes nor to retrace concrete examples of valuation algebras with inverses. We continue with a first example of a (regular) valuation algebra that provides inverse elements. Many further instances with division can be found in the appendix of this chapter.

■ 4.1 Inverse Arithmetic Potentials

The valuation algebra of arithmetic potentials provides inverse elements. For $p \in \Phi$ with $d(p) = s$ and $\mathbf{x} \in \Omega_s$ we define

$$p^{-1}(\mathbf{x}) \quad = \quad \begin{cases} \frac{1}{p(\mathbf{x})} & \text{if } p(\mathbf{x}) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that p and p^{-1} satisfy equation (4.10) and therefore are inverses. In fact, we learn in the appendix of this chapter that arithmetic potentials form a regular valuation algebra.

Idempotency is a very strong and important algebraic condition under which each valuation becomes the inverse of itself. Thus, division becomes trivial in the case of an idempotent valuation algebra. Idempotency can be seen as a particular form of regularity, but it also has an interest in itself. There is a special local computation architecture discussed in Section 4.5 that applies only to idempotent valuation algebras but not to formalisms with a weaker notion of division. Finally, idempotent valuation algebras have many interesting properties that become important in later sections.

4.2.1 Idempotent Valuation Algebras

The property of idempotency is defined as follows:

Definition 4.2 *A valuation algebra $\langle \Phi, D \rangle$ is called idempotent if for all $\phi \in \Phi$ and $t \subseteq d(\phi)$, it holds that*

$$\phi \otimes \phi^{\downarrow t} \quad = \quad \phi. \tag{4.11}$$

In particular, it follows from this definition that $\phi \otimes \phi = \phi$. Thus, choosing $\psi = \phi$ satisfies the requirement for inverses in equation (4.10) which means that each element becomes the inverse of itself. Comparing the two definitions 4.1 and 4.2, we remark that the latter also includes a statement about projection, which makes it considerably stronger. A similar condition will occur in the definition of separativity and regularity given in the appendix of this chapter that finally enables division-based local computation.

Lemma 4.3 *The neutrality axiom (A7) always holds in a stable and idempotent valuation algebra.*

Proof: From stability and idempotency we derive

$$e_x \otimes e_{x\cup y} \;=\; e_{x\cup y}^{\downarrow x} \otimes e_{x\cup y} \;=\; e_{x\cup y}.$$

Then, by the labeling axiom and the definition of neutral elements,

$$e_x \otimes e_y \;=\; e_x \otimes e_y \otimes e_{x\cup y} \;=\; e_x \otimes e_{x\cup y} \;=\; e_{x\cup y},$$

which proves the neutrality axiom. ■

■ 4.2 Indicator Functions and Idempotency

The valuation algebra of indicator functions is idempotent. Indeed, for $i \in \Phi$ with $d(i) = s$, $t \subseteq s$ and $\mathbf{x} \in \Omega_s$ we have

$$(i \otimes i^{\downarrow t})(\mathbf{x}) \;=\; i(\mathbf{x}) \cdot i^{\downarrow t}(\mathbf{x}^{\downarrow t}) \;=\; i(\mathbf{x}) \cdot \max_{\mathbf{y}\in\Omega_{s-t}} i(\mathbf{x}^{\downarrow t}, \mathbf{y}) \;=\; i(\mathbf{x}).$$

This holds because $i(\mathbf{x}) \leq \max_{\mathbf{y}\in\Omega_{s-t}} i(\mathbf{x}^{\downarrow t}, \mathbf{y})$.

■ 4.3 Relations and Idempotency

Is is not surprising that also the valuation algebra of relations is idempotent. For $R \in \Phi$ with $d(R) = s$ and $t \subseteq s$ we have

$$R \otimes R^{\downarrow t} \;=\; R \bowtie \pi_t(R) \;=\; R \bowtie \{\mathbf{x}^{\downarrow t} : \mathbf{x} \in R\} \;=\; R.$$

Besides their computational importance addressed in this book, idempotent valuation algebras are fundamental in *algebraic information theory*. An idempotent valuation algebra with neutral and null elements is called *information algebra* (Kohlas, 2003) and allows the introduction of a partial order between valuations to express that some knowledge pieces are *more informative* than others. This establishes the basis of an algebraic theory of information that is treated exhaustively in (Kohlas, 2003). See also Exercise D.4 at the end of this chapter.

Definition 4.3 *An idempotent valuation algebra with neutral and null elements is called information algebra.*

We now focus on local computation schemes which exploit division to improve the message scheduling and time complexity of the Shenoy-Shafer architecture.

4.3 THE LAURITZEN-SPIEGELHALTER ARCHITECTURE

The Shenoy-Shafer architecture answers multi-query inference problems on any valuation algebra and therefore represents the most general local computation scheme.

Alternatively, research in the field of Bayesian networks has led to another architecture called the *Lauritzen-Spiegelhalter architecture* (Lauritzen & Spiegelhalter, 1988), which can be applied if the valuation algebra provides a division operator. Thus, the starting point of this section is a multi-query inference problem with factors taken from a separative valuation algebra. As explained above, separativity is the weakest algebraic condition that allows the introduction of a division operator in the valuation algebra. Essentially, a separative valuation algebra $\langle \Phi, D \rangle$ can be embedded into a union of groups $\langle \Phi^*, D \rangle$ where each element of Φ^* has a well-defined inverse. On the other hand, it is shown in Appendix D.1.1 that projection is only partially defined in this embedding. We therefore need to impose a further condition on the inference problem to avoid non-defined projections during the local computation process. The knowledgebase factors $\{\phi_1, \ldots, \phi_n\}$ may be elements of Φ^* but the objective function $\phi = \phi_1 \otimes \cdots \otimes \phi_n$ must be element of Φ which guarantees that ϕ can be projected to any possible query. The same additional constraint is also assumed for the HUGIN architecture (Jensen *et al.*, 1990) of Section 4.4.

Remember, the inward phase of the Shenoy-Shafer architecture is almost equal to the collect algorithm with the only difference that incoming messages are not combined with the node content but kept in mailboxes. This is indispensable for the correctness of the Shenoy-Shafer architecture since otherwise, node i would get back its own message sent during the inward phase as part of the message obtained in the outward phase. In other words, some knowledge would be considered twice in every node, and this would falsify the final result. In case of a division operation, we can divide this doubly treated message out, and this idea is exploited by the Lauritzen-Spiegelhalter architecture and by the HUGIN architecture.

The Lauritzen-Spiegelhalter architecture starts executing the collect algorithm towards root node m. We define by ψ' the content of node i just before sending its message to the child node. According to Section 3.10 we have

$$\psi'_i \;=\; \psi_i^{(r)} \;=\; \psi_i^{(i)} \;=\; \psi_i \otimes \bigotimes_{j \in pa(i)} \mu_{j \to i}. \tag{4.12}$$

Then, the message for the child node $ch(i)$ is computed as

$$\mu_{i \to ch(i)} \;=\; (\psi'_i)^{\downarrow \omega_i \cap \lambda(ch(i))}.$$

This is the point where division comes into play. As soon as node i has sent its message, it divides the message out of its own content:

$$\psi'_i \otimes \mu^{-1}_{i \to ch(i)}. \tag{4.13}$$

This is repeated up to the root node. Thereupon, the outward propagation proceeds in a similar way. A node sends a message to all its parents when it has received a message from its child. In contrast, however, no division is performed during the outward phase. So, if node j is ready to send a message towards i and its current content is ψ'_j, the message is

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

which is combined directly with the content of the receiving node i.

Theorem 4.2 *At the end of the Lauritzen-Spiegelhalter architecture, node i contains $\phi^{\downarrow \lambda(i)}$ provided that all messages during an execution of the Shenoy-Shafer architecture can be computed.*

The proof of this theorem is given in Appendix D.2.1. It is important to note that the correctness of the Lauritzen-Spiegelhalter architecture is conditioned on the existence of the messages in the Shenoy-Shafer architecture, since the computations take place in the separative embedding Φ^* where projection is only partially defined. However, (Schneuwly, 2007) weakens this requirement by proving that the existence of the inward messages is in fact sufficient to make the Shenoy-Shafer architecture and therefore also the Lauritzen-Spiegelhalter architecture work. Furthermore, in case of a regular valuation algebra, all factors are elements of Φ and consequently all projections exist. Theorem 4.2 can therefore be simplified by dropping this assumption. A further interesting issue with respect to this theorem concerns the messages sent in the distribute phase. Namely, we may directly conclude that

$$\mu_{ch(i) \to i} \quad = \quad \phi^{\downarrow \lambda(i) \cap \lambda(ch(i))}. \tag{4.14}$$

We again summarize this local computation scheme in the shape of a pseudo-code algorithm. As in the Shenoy-Shafer architecture, we separate the message-passing procedure from the actual query answering process.

Algorithm 4.3 Lauritzen-Spiegelhalter Architecture: Message-Passing

```
input:  $(V, E, \lambda, D)$  output:  -

begin
  for $i = 1 \ldots |V| - 1$ do                              // Collect Phase:
    $\mu_{i \to ch(i)} := \psi_i^{\downarrow d(\psi_i) \cap \lambda(ch(i))}$
    $\psi_{ch(i)} := \psi_{ch(i)} \otimes \mu_{i \to ch(i)}$;
    $\psi_i := \psi_i \otimes \mu_{i \to ch(i)}^{-1}$;
  end;
  for $i = |V| - 1 \ldots 1$ do                              // Distribute Phase:
    $\mu_{ch(i) \to i} := \psi_{ch(i)}^{\downarrow \lambda(i) \cap \lambda(ch(i))}$;
    $\psi_i := \psi_i \otimes \mu_{ch(i) \to i}$;
  end;
end
```

Algorithm 4.4 Lauritzen-Spiegelhalter Architecture: Query Answering

```
input:  $(V, E, \lambda, D)$,  $x \subseteq d(\phi)$  output:  $\phi^{\downarrow x}$

begin
  for $i = 1 \ldots |V|$ do
    if $x \subseteq \lambda(i)$ do
      return $\psi_i^{\downarrow x}$;
    end;
end
```

We next illustrate the Lauritzen-Spiegelhalter architecture using the medical knowledgebase. In Instance 2.1 we expressed the conditional probability tables of the Bayesian network that underlies this example in the formalism of arithmetic potentials. It is known from Instance 4.1 that arithmetic potentials form a regular valuation algebra. We are therefore not only allowed to apply this architecture, but we do not even need to care about the existence of messages as explained before.

Example 4.4 *Reconsider the covering join tree for the medical knowledgebase of Figure 3.15. During the collect phase, the Lauritzen-Spiegelhalter sends exactly the same messages as the generalized collect algorithm. These messages are given in the Examples 3.20 and 3.22. Whenever a node has sent a message, it divides the latter out of its node content. Thus, the join tree nodes hold the following factors at the end of the inward phase:*

- $\psi_1' \;=\; \psi_1 \otimes \mu_{1\to2}^{-1}$
- $\psi_2' \;=\; \psi_2 \otimes \mu_{1\to2} \otimes \mu_{2\to5}^{-1}$
- $\psi_3' \;=\; \psi_3 \otimes \mu_{3\to6}^{-1}$
- $\psi_4' \;=\; \psi_4 \otimes \mu_{4\to5}^{-1}$
- $\psi_5' \;=\; \psi_5 \otimes \mu_{4\to5} \otimes \mu_{2\to5} \otimes \mu_{5\to6}^{-1}$
- $\psi_6' \;=\; \psi_6 \otimes \mu_{5\to6} \otimes \mu_{6\to7}^{-1}$
- $\psi_7' \;=\; \psi_7 \otimes \mu_{6\to7}$

The root node does not perform a division since there is no message to divide out. As a consequence of the collect algorithm, it holds that

$$\psi_7' \;=\; \psi_7 \otimes \mu_{6\to7} \;=\; \phi^{\downarrow\{B,D,E\}}.$$

We then enter the distribute phase and send messages from the root down to the leaves. These messages are always combined to the node content of the receiver:

- *Node* 6 *obtains the message*

$$\mu_{7\to6} \;=\; (\psi_7 \otimes \mu_{6\to7})^{\downarrow\{B,E\}} \;=\; \phi^{\downarrow\{B,E\}}$$

and updates its node content to

$$\begin{aligned}\psi_6' \otimes \mu_{7\rightarrow 6} &= \psi_6 \otimes \mu_{5\rightarrow 6} \otimes \mu_{6\rightarrow 7}^{-1} \otimes (\psi_7 \otimes \mu_{6\rightarrow 7})^{\downarrow\{B,E\}} \\ &= \left(\psi_6 \otimes \mu_{5\rightarrow 6} \otimes \mu_{6\rightarrow 7}^{-1} \otimes \psi_7 \otimes \mu_{6\rightarrow 7}\right)^{\downarrow\{B,E,L\}} \\ &= (\psi_6 \otimes \mu_{5\rightarrow 6} \otimes \psi_7)^{\downarrow\{B,E,L\}} \;=\; \phi^{\downarrow\{B,E,L\}}.\end{aligned}$$

The second equality follows from the combination axiom. For the remaining nodes, we only give the messages that are obtained in the distribute phase.

- *Node* 5 *obtains the message*

$$\mu_{6\rightarrow 5} = (\psi_6 \otimes \mu_{3\rightarrow 6} \otimes \mu_{5\rightarrow 6} \otimes \mu_{6\rightarrow 7}^{-1} \otimes \mu_{7\rightarrow 6})^{\downarrow\{E,L\}} = \phi^{\downarrow\{E,L\}}$$

- *Node* 4 *obtains the message*

$$\mu_{5\rightarrow 4} = (\psi_5 \otimes \mu_{2\rightarrow 5} \otimes \mu_{4\rightarrow 5} \otimes \mu_{5\rightarrow 6}^{-1} \otimes \mu_{6\rightarrow 5})^{\downarrow\{E\}} = \phi^{\downarrow\{E\}}$$

- *Node* 3 *obtains the message*

$$\mu_{6\rightarrow 3} = (\psi_6 \otimes \mu_{3\rightarrow 6} \otimes \mu_{5\rightarrow 6} \otimes \mu_{6\rightarrow 7}^{-1} \otimes \mu_{7\rightarrow 6})^{\downarrow\{B,L\}} = \phi^{\downarrow\{B,L\}}$$

- *Node* 2 *obtains the message*

$$\mu_{5\rightarrow 2} = (\psi_5 \otimes \mu_{2\rightarrow 5} \otimes \mu_{4\rightarrow 5} \otimes \mu_{5\rightarrow 6}^{-1} \otimes \mu_{6\rightarrow 5})^{\downarrow\{E,L\}} = \phi^{\downarrow\{E,L\}}$$

- *Node* 1 *obtains the message*

$$\mu_{2\rightarrow 1} = (\psi_2 \otimes \mu_{1\rightarrow 2} \otimes \mu_{2\rightarrow 5}^{-1} \otimes \mu_{5\rightarrow 2})^{\downarrow\{T\}} = \phi^{\downarrow\{T\}}$$

At the end of the message-passing, each node directly contains the projection of the objective function ϕ to its own node label. This example is well suited to once more highlight the idea of division. Consider node 4 *of Figure 3.15: During the collect phase, it sends a message to node* 5 *that consists of its initial node content. Later, in the distribute phase, it receives a message that consists of the relevant information from all nodes in the join tree, including its own content sent during the collect phase. It is therefore necessary that this particular piece of information has been divided out since otherwise, it would be considered twice in the final result. Dealing with arithmetic potentials, the reader may easily convince himself that combining a potential with itself leads to a different result.*

4.3.1 Complexity of the Lauritzen-Spiegelhalter Architecture

The collect phase of the Lauritzen-Spiegelhalter architecture is almost identical to the unimproved collect algorithm. In a nutshell, there are $|V|-1$ transmitted messages and each message is combined to the content of the receiving node. Further, one projection is required to obtain a message that altogether gives $2(|V|-1)$ operations. In addition to the collect algorithm, the collect phase of the Lauritzen-Spiegelhalter architecture computes one division in each node. We therefore obtain $3(|V|-1)$ operations that all take place on the node labels. To compute a message in the distribute phase, we again need one projection and, since messages are again combined to the node content, we end with an overall number of $5(|V|-1)$ operations for the message-passing in the Lauritzen-Spiegelhalter architecture. Finally, query answering is simple and just requires a single projection per query. Thus, the time complexity of the Lauritzen-Spiegelhalter architecture is:

$$\mathcal{O}\Big(|V| \cdot f(\omega^* + 1)\Big). \tag{4.15}$$

In contrast to the Shenoy-Shafer architecture, messages do not need to be stored until the end of the complete message-passing, but can be discarded once they have been sent and divided out of the node content. But we again keep one factor in each node whose size is bounded by the node label. This gives us the following space complexity that is equal to the unimproved version of the collect algorithm:

$$\mathcal{O}\Big(|V| \cdot g(\omega^* + 1)\Big). \tag{4.16}$$

The Lauritzen-Spiegelhalter architecture performs division during its collect phase. One can easily imagine that a similar effect can be achieved by delaying the execution of division to the distribute phase. This essentially is the idea of a further multi-query local computation scheme called *HUGIN architecture*. It is then possible to perform division on the separators between the join tree nodes that increases the performance of this operation. The drawback is that we again need to remember the collect messages until the end of the algorithm.

4.4 THE HUGIN ARCHITECTURE

The HUGIN architecture (Jensen *et al.*, 1990) is a modification of Lauritzen-Spiegelhalter to make the divisions less costly. It postpones division to the distribute phase such that the collect phase again corresponds to the collect algorithm with the only difference that every message $\mu_{i \to j}$ is stored in the *separator* between the neighboring nodes i and j. These separators have the label $sep(i) = \lambda(i) \cap \lambda(j)$ according to Definition 3.12. They can be seen as components between join tree nodes, comparable to the mailboxes in the Shenoy-Shafer architecture. A graphical illustration is shown in Figure 4.9. Subsequently, we denote by S the set of all separators. In the following distribute phase, the messages are computed as in the Lauritzen-Spiegelhalter architecture, but have to pass through the separator lying between the sending and receiving nodes. The separator becomes activated by the crossing message and holds it back in order to divide out its current content. Finally, the modified message is delivered to the destination, and the original message is stored in the separator.

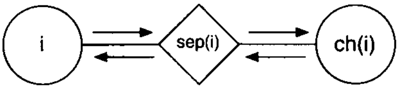


Figure 4.9 Separator in the HUGIN architecture.

Formally, the message sent from node i to $ch(i)$ during the collect phase is

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

This message is stored in the separator $sep(i)$. In the distribute phase, node $ch(i)$ then sends the message

$$\mu'_{ch(i) \to i} \quad = \quad \left(\psi_{ch(i)} \otimes \bigotimes_{k \in ne(i)} \mu_{k \to ch(i)} \right)^{\downarrow sep(i)} \quad = \quad \phi^{\downarrow sep(i)}$$

towards i. This message arrives at the separator where it is altered to

$$\mu_{ch(i) \to i} \quad = \quad \mu'_{ch(i) \to i} \otimes \mu^{-1}_{i \to ch(i)}. \tag{4.17}$$

The message $\mu_{ch(i) \to i}$ is sent to node i and combined with the node content. This formula also shows the advantage of the HUGIN architecture over the Lauritzen-Spiegelhalter architecture. Divisions are performed in the separators exclusively and these have in general smaller labels than the join tree nodes. In equation (4.14) we have seen that the distribute messages in the Lauritzen-Spiegelhalter architecture correspond already to the projection of the objective function to the separator. This is also the case for the messages that are sent in the distribute phase of the HUGIN architecture and that are stored in the separators. However, the message emitted by the separator does not have this property anymore since division has been performed.

Theorem 4.3 *At the end of the HUGIN architecture, each node $i \in V$ stores $\phi^{\downarrow \lambda(i)}$ and every separator between i and j the projection $\phi^{\downarrow \lambda(i) \cap \lambda(j)}$ provided that all messages during an execution of the Shenoy-Shafer architecture can be computed.*

The HUGIN algorithm imposes the identical restrictions on the valuation algebra as the Lauritzen-Spiegelhalter architecture and has furthermore been identified as an improvement of the latter that makes the division operation less costly. It is therefore not surprising that the above theorem underlies the same existential condition regarding the messages of the Shenoy-Shafer architecture as Theorem 4.2. The proof of this theorem can be found in Appendix D.2.2.

Algorithm 4.5 HUGIN Architecture: Message-Passing

```
input:   $(V, E, \lambda, D)$   output:   -

begin
  for $i = 1 \ldots |V| - 1$ do                                   // Collect Phase:
    $\mu_{i \rightarrow ch(i)}$ := $\psi_i^{\downarrow d(\psi_i) \cap \lambda(ch(i))}$
    $\psi_{sep(i)}$ := $\mu_{i \rightarrow ch(i)}$;
    $\psi_{ch(i)}$ := $\psi_{ch(i)} \otimes \mu_{i \rightarrow ch(i)}$;
  end;
  for $i = |V| - 1 \ldots 1$ do                                 // Distribute Phase:
    $\mu'_{ch(i) \rightarrow i}$ := $\psi_{ch(i)}^{\downarrow \lambda(i) \cap \lambda(ch(i))}$;
    $\mu_{ch(i) \rightarrow i}$ := $\mu'_{ch(i) \rightarrow i} \otimes \psi_{sep}^{-1}$;
    $\psi_{sep(i)}$ := $\mu'_{i \rightarrow ch(i)}$;
    $\psi_i$ := $\psi_i \otimes \mu_{ch(i) \rightarrow i}$;
  end;
end
```

We again summarize this local computation scheme by a pseudo-code algorithm. Here, the factor $\psi_{sep(i)}$ refers to the content of the separator between node i and $ch(i)$. However, since both theorems 4.2 and 4.3 make equal statements about the result at the end of the message-passing, we can reuse Algorithm 4.4 for the query answering in the HUGIN architecture and just redefine the message-passing procedure:

Example 4.5 *We illustrate the HUGIN architecture in the same setting as Lauritzen-Spiegelhalter architecture in Example 4.4. However, since the collect phase of the HUGIN architecture is identical to the generalized collect algorithm, we directly take the messages from Examples 3.20 and 3.22. The node contents after the collect phase corresponds to Example 4.4 but without the inverses of the messages. Instead, all messages sent during the collect phase are assumed to be stored in the separators. In the distribute phase, messages are sent from the root node down to the leaves. When a message traverses a separator that stores the upward message from the collect phase, the latter is divided out of the traversing message and the result is forwarded to the receiving node. Let us consider these forwarded messages in detail:*

- *Node* 6 *obtains the message*

$$\mu_{7 \rightarrow 6} \quad = \quad (\psi_7 \otimes \mu_{6 \rightarrow 7})^{\downarrow \{B,E\}} \otimes \mu_{6 \rightarrow 7}^{-1}.$$

Note that the division was performed while traversing the separator node. Then, node 6 *updates its content to*

$$\begin{aligned} \psi_6 \otimes \mu_{5 \rightarrow 6} \otimes \mu_{7 \rightarrow 6} \quad &= \quad \psi_6 \otimes \mu_{5 \rightarrow 6} \otimes (\psi_7 \otimes \mu_{6 \rightarrow 7})^{\downarrow \{B,E\}} \otimes \mu_{6 \rightarrow 7}^{-1} \\ &= \quad \left(\psi_6 \otimes \mu_{5 \rightarrow 6} \otimes \psi_7 \otimes \mu_{6 \rightarrow 7} \otimes \mu_{6 \rightarrow 7}^{-1}\right)^{\downarrow \{B,E,L\}} \\ &= \quad (\psi_6 \otimes \mu_{5 \rightarrow 6} \otimes \psi_7)^{\downarrow \{B,E,L\}} \quad = \quad \phi^{\downarrow \{B,E,L\}}. \end{aligned}$$

The second equality follows from the combination axiom. For the remaining nodes, we only give the messages that are obtained in the distribute phase.

- *Node* 5 *obtains the message*

$$\mu_{6\to 5} \;=\; (\psi_6 \otimes \mu_{3\to 6} \otimes \mu_{5\to 6} \otimes \mu_{7\to 6})^{\downarrow\{E,L\}} \otimes \mu_{5\to 6}^{-1}$$

- *Node* 4 *obtains the message*

$$\mu_{5\to 4} \;=\; (\psi_5 \otimes \mu_{2\to 5} \otimes \mu_{4\to 5} \otimes \mu_{6\to 5})^{\downarrow\{E\}} \otimes \mu_{4\to 5}^{-1}$$

- *Node* 3 *obtains the message*

$$\mu_{6\to 3} \;=\; (\psi_6 \otimes \mu_{3\to 6} \otimes \mu_{5\to 6} \otimes \mu_{7\to 6})^{\downarrow\{B,L\}} \otimes \mu_{3\to 6}^{-1}$$

- *Node* 2 *obtains the message*

$$\mu_{5\to 2} \;=\; (\psi_5 \otimes \mu_{2\to 5} \otimes \mu_{4\to 5} \otimes \mu_{6\to 5})^{\downarrow\{E,L\}} \otimes \mu_{2\to 5}^{-1}$$

- *Node* 1 *obtains the message*

$$\mu_{2\to 1} \;=\; (\psi_2 \otimes \mu_{1\to 2} \otimes \mu_{5\to 2})^{\downarrow\{T\}} \otimes \mu_{1\to 2}^{-1}$$

These messages consist of the messages from Example 4.4 from which the collect message is divided out. We therefore see that the division only takes place on the separator size and not on the node label itself.

4.4.1 Complexity of the HUGIN Architecture

Division is more efficient in the HUGIN architecture since it takes place on the separators which are generally smaller than the node labels. On the other hand, we additionally have to keep the inward messages in the memory that is not necessary in the Lauritzen-Spiegelhalter architecture. Both changes do not affect the overall complexity such that HUGIN is subject to the same time and space complexity bounds as derived for the Lauritzen-Spiegelhalter architecture in Section 4.3.1. From this point of view, the HUGIN architecture is generally preferred over Lauritzen-Spiegelhalter. An analysis for the time-space trade-off between the HUGIN and Shenoy-Shafer architecture based on Bayesian networks has been drawn by (Allen & Darwiche, 2003), and we again refer to (Lepar & Shenoy, 1998) where all three multi-query architectures are juxtaposed. We are now going to discuss one last local computation scheme for the solution of multi-query inference problems that applies to idempotent valuation algebras. In some sense, and this already foreshadows its proof, this scheme is identical to the Lauritzen-Spiegelhalter architecture where division is simply ignored because it has no effect in an idempotent valuation algebra.

4.5 THE IDEMPOTENT ARCHITECTURE

Section 4.2.1 introduces idempotency as a strong algebraic property under which division becomes trivial. Thus, the Lauritzen-Spiegelhalter as well as the HUGIN architecture can both be applied to inference problems with factors from an idempotent valuation algebra. Clearly, if division in these architectures does not provoke any effect, we can ignore its execution altogether. This leads to the most simple local computation scheme called *idempotent architecture* (Kohlas, 2003).

In the Lauritzen-Spiegelhalter architecture, the message sent from i to $ch(i)$ during the collect phase is divided out of the node content. Since idempotent valuations are their own inverses, this reduces to a simple combination. Hence, instead of dividing out the emitted message, it is simply combined to the content of the sending node. By idempotency, this combination has no effect. Thus, consider the collect message

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

This message is divided out of the node of i:

$$\begin{aligned}\psi_i \otimes \bigotimes_{j\in pa(i)} \mu_{j\to i} \otimes \mu_{i\to ch(i)}^{-1} &= \psi_i \otimes \bigotimes_{j\in pa(i)} \mu_{j\to i} \otimes \mu_{i\to ch(i)} \\ = \left(\psi_i \otimes \bigotimes_{j\in pa(i)} \mu_{j\to i}\right) \otimes \left(\psi_i \otimes \bigotimes_{j\in pa(i)} \mu_{j\to i}\right)^{\downarrow\omega_i\cap\lambda(ch(i))} \\ = \psi_i \otimes \bigotimes_{j\in pa(i)} \mu_{j\to i}.\end{aligned}$$

The last equality follows from idempotency. Consequently, the idempotent architecture consists in the execution of the collect algorithm towards a previously fixed root node m followed by a simple distribute phase where incoming messages are combined to the node content without any further action. In particular, no division needs to be performed. This derives the idempotent architecture from Lauritzen-Spiegelhalter, but it also follows in a similar way from the HUGIN architecture (Pouly, 2008). To sum it up, the architectures of Lauritzen-Spiegelhalter and HUGIN both simplify to the same scheme where no division needs to be done.

Theorem 4.4 *At the end of the idempotent architecture, node i contains $\phi^{\downarrow\lambda(i)}$.*

This theorem follows from the correctness of the Lauritzen-Spiegelhalter architecture and the above considerations. Further, there is no need to condition the statement on the existence of the Shenoy-Shafer messages since idempotent valuation algebras are always regular. This is shown in Appendix D.1.3. For the pseudo-code algorithm, we may again restrict ourselves to the message-passing part and refer to Algorithm 4.4 for the query answering procedure.

Algorithm 4.6 Idempotent Architecture: Message-Passing

```
input:  $(V, E, \lambda, D)$  output:  -

begin
  for $i = 1 \ldots |V| - 1$  do                                   // Collect Phase:
    $\mu_{i \to ch(i)} := \psi_i^{\downarrow d(\psi_i) \cap \lambda(ch(i))}$
    $\psi_{ch(i)} := \psi_{ch(i)} \otimes \mu_{i \to ch(i)}$;
  end;
  for $i = |V| - 1 \ldots 1$  do                                   // Distribute Phase:
    $\mu_{ch(i) \to i} := \psi_{ch(i)}^{\downarrow \lambda(i) \cap \lambda(ch(i))}$;
    $\psi_i := \psi_i \otimes \mu_{ch(i) \to i}$;
  end;
end
```

Example 4.6 *If we again call in the same setting as for the Lauritzen-Spiegelhalter and HUGIN architecture in Example 4.4 and 4.5 for the illustration of the idempotent architecture, things become particularly simple (as it is often the case with the idempotent architecture). The collect phase executed the generalized collect algorithm whose messages can be found in Example 3.20 and 3.22. Then, the distribute phase just continues in exactly the same way:*

- *Node* 6 *obtains the message*

$$\mu_{7 \to 6} = (\psi_7 \otimes \mu_{6 \to 7})^{\downarrow \{B,E\}}$$

and updates its node content to

$$\begin{aligned}
\psi_6 \otimes \mu_{5 \to 6} \otimes \mu_{7 \to 6} &= \psi_6 \otimes \mu_{5 \to 6} \otimes (\psi_7 \otimes \mu_{6 \to 7})^{\downarrow \{B,E\}} \\
&= (\psi_6 \otimes \mu_{5 \to 6} \otimes \psi_7 \otimes \mu_{6 \to 7})^{\downarrow \{B,E,L\}} \\
&= (\psi_6 \otimes \mu_{5 \to 6} \otimes \psi_7)^{\downarrow \{B,E,L\}} = \phi^{\downarrow \{B,E,L\}}.
\end{aligned}$$

The second equality follows from the combination axiom. Since $\mu_{6 \to 7}$ corresponds to a projection of $\psi_6 \otimes \mu_{5 \to 6}$ it follows from idempotency that

$$\psi_6 \otimes \mu_{5 \to 6} \otimes \mu_{6 \to 7} = \psi_6 \otimes \mu_{5 \to 6}.$$

This explains the third equality. For the remaining nodes, we only give the messages that are obtained in the distribute phase.

- *Node* 5 *obtains the message*

$$\mu_{6 \to 5} = (\psi_6 \otimes \mu_{3 \to 6} \otimes \mu_{5 \to 6} \otimes \mu_{7 \to 6})^{\downarrow \{E,L\}}$$

- *Node* 4 *obtains the message*

$$\mu_{5 \to 4} = (\psi_5 \otimes \mu_{2 \to 5} \otimes \mu_{4 \to 5} \otimes \mu_{6 \to 5})^{\downarrow \{E\}}$$

- *Node* 3 *obtains the message*

$$\mu_{6\to 3} \;=\; (\psi_6 \otimes \mu_{3\to 6} \otimes \mu_{5\to 6} \otimes \mu_{7\to 6})^{\downarrow\{B,L\}}$$

- *Node* 2 *obtains the message*

$$\mu_{5\to 2} \;=\; (\psi_5 \otimes \mu_{2\to 5} \otimes \mu_{4\to 5} \otimes \mu_{6\to 5})^{\downarrow\{E,L\}}$$

- *Node* 1 *obtains the message*

$$\mu_{2\to 1} \;=\; (\psi_2 \otimes \mu_{1\to 2} \otimes \mu_{5\to 2})^{\downarrow\{T\}}$$

Observe that no division has been performed anywhere.

4.5.1 Complexity of the Idempotent Architecture

The idempotent architecture is absolutely symmetric concerning the number of executed operations. There are $2(|V|-1)$ exchanged messages, thus $2(|V|-1)$ projections and $2(|V|-1)$ combinations to perform. No division is performed anywhere in this architecture. Further, each message can directly be discarded once it has been combined to the node content of its receiver. Thus, although we obtain the same bounds for the time and space complexity as in the Lauritzen-Spiegelhalter and HUGIN architecture, the idempotent architecture can be considered as getting the best of both worlds. Since no division is performed, the idempotent architecture even outperforms the HUGIN scheme and does not need to remember any messages. Thus, if time is the more sensitive resource and idempotency holds, the idempotent architecture becomes the best choice to solve multi-query inference problems by local computation. Moreover, if we are dealing with a valuation algebra where both complexities are polynomial, the idempotent architecture provides a specialized query answering procedure for queries that are not covered by the join tree. This procedure is presented in the following section.

4.6 ANSWERING UNCOVERED QUERIES

The concept of a covering join tree ensures that all queries from the original inference problem can be answered after the completed distributed phase. But if later a new query arrives that is not covered by any of the join tree nodes, it is generally necessary to construct a new join tree and to execute the local computation architecture from scratch. Alternatively, there are *updating methods* described in (Schneuwly, 2007) that modify the join tree to cover the new query. Then, only a part of the complete message-passing has to be re-processed. However, a third option that is especially suited for formalisms with a polynomial complexity exists in the idempotent case. This method is a consequence of an interesting property of the idempotent architecture that is concerned with a particular factorization of the objective function. Initially, the original factorization of the objective function is given in the definition of the inference problem. Then, a second factorization is obtained from Lemma 3.8 with the property that the domain of each factor is covered by some specific node of the join tree. In case of idempotency, we obtain a third factorization of the objective function from the node contents after local computation:

Lemma 4.4 *In an idempotent valuation algebra we have*

$$\phi \quad = \quad \bigotimes_{i=1}^{r} \phi^{\downarrow\lambda(i)}. \tag{4.18}$$

Proof: The proof of this lemma is based on the correctness of the Lauritzen-Spiegelhalter architecture, which can always be executed when idempotency holds. At the beginning, we clearly have

$$\phi \quad = \quad \bigotimes_{i=1}^{r} \psi_i.$$

During the collect phase, each node computes the message for its child node and divides it out of its node content. Then, the message is sent to the child node where it is combined to the node content. Thus, each message that is divided out of some node content is later combined to the content of another node. We therefore conclude that the above equation still holds at the end of the collect phase in the Lauritzen-Spiegelhalter architecture. Formally, equation (4.13) defines the node content at the end of the collect phase in the Lauritzen-Spiegelhalter architecture and we thus have

$$\phi \quad = \quad \phi^{\downarrow\lambda(r)} \otimes \bigotimes_{i=1}^{r-1} \psi_i' \otimes \mu_{i\rightarrow ch(i)}^{-1}.$$

By idempotency it further holds that

$$\phi \quad = \quad \phi \otimes \bigotimes_{i=1}^{r-1} \phi^{\downarrow\lambda(i)\cap\lambda(ch(i))}$$

and therefore

$$\begin{aligned}\phi \quad &= \quad \phi^{\downarrow\lambda(r)} \otimes \bigotimes_{i=1}^{r-1} \phi^{\downarrow\lambda(i)\cap\lambda(ch(i))} \otimes \bigotimes_{i=1}^{r-1} \psi_i' \otimes \mu_{i\rightarrow ch(i)}^{-1} \\ &= \quad \phi^{\downarrow\lambda(r)} \otimes \bigotimes_{i=1}^{r-1} \psi_i' \otimes \mu_{i\rightarrow ch(i)}^{-1} \otimes \phi^{\downarrow\lambda(i)\cap\lambda(ch(i))}.\end{aligned}$$

But the marginal of ϕ with respect to $\lambda(i) \cap \lambda(ch(i))$ corresponds exactly to the message sent from node $ch(i)$ to node i during the distribute phase of the Lauritzen-Spiegelhalter architecture, see equation (4.14). We therefore have by the correctness of the Lauritzen-Spiegelhalter architecture

$$
\begin{aligned}
\phi &= \phi^{\downarrow\lambda(r)} \otimes \bigotimes_{i=1}^{r-1} \psi_i' \otimes \mu_{i\rightarrow ch(i)}^{-1} \otimes \phi^{\downarrow\lambda(i)\cap\lambda(ch(i))} \\
&= \phi^{\downarrow\lambda(r)} \otimes \bigotimes_{i=1}^{r-1} \phi^{\downarrow\lambda(i)} = \bigotimes_{i=1}^{r} \phi^{\downarrow\lambda(i)}.
\end{aligned}
$$

■

As a consequence of this result, we may consider local computation as a transformation of the join tree factorization of Lemma 3.8 into a new factorization where each factor corresponds to the projection of the objective function to some node label. The following theorem provides a further generalization of this statement.

Theorem 4.5 *For* $i = 1, \ldots, r$ *and*

$$
y_i = \bigcup_{j=i}^{r} \lambda(j) \tag{4.19}
$$

it holds for an idempotent valuation algebra that

$$
\phi^{\downarrow y_i} = \bigotimes_{j=i}^{r} \phi^{\downarrow\lambda(j)}. \tag{4.20}
$$

Proof: We proceed by induction: For $i = 1$ the statement follows from Lemma 4.4 and from the domain axiom. For $i + 1$ we observe that $y_{i+1} \subseteq y_i$ and obtain using the induction hypothesis and the combination axiom

$$
\begin{aligned}
\phi^{\downarrow y_{i+1}} &= \left(\phi^{\downarrow y_i}\right)^{\downarrow y_{i+1}} = \left(\bigotimes_{j=i}^{r} \phi^{\downarrow\lambda(j)}\right)^{\downarrow y_{i+1}} \\
&= \left(\phi^{\downarrow\lambda(i)} \otimes \bigotimes_{j=i+1}^{r} \phi^{\downarrow\lambda(j)}\right)^{\downarrow y_{i+1}} = \phi^{\downarrow\lambda(i)\cap y_{i+1}} \otimes \bigotimes_{j=i+1}^{r} \phi^{\downarrow\lambda(j)}.
\end{aligned}
$$

From the running intersection property we further derive $\lambda(i)\cap\lambda(ch(i)) = \lambda(i)\cap y_{i+1}$ and obtain by idempotency

$$
\begin{aligned}
\phi^{\downarrow y_{i+1}} &= \phi^{\downarrow\lambda(i)\cap\lambda(ch(i))} \otimes \phi^{\downarrow\lambda(ch(i))} \otimes \bigotimes_{j=i+1, j\neq ch(i)}^{r} \phi^{\downarrow\lambda(j)} \\
&= \phi^{\downarrow\lambda(ch(i))} \otimes \bigotimes_{j=i+1, j\neq ch(i)}^{r} \phi^{\downarrow\lambda(j)} = \bigotimes_{j=i+1}^{r} \phi^{\downarrow\lambda(j)}.
\end{aligned}
$$

■

We next observe that combining the node content of two neighboring nodes always results in the objective function projected to their common domain.

Lemma 4.5 *For $1 \leq i, j \leq r$ and $j \in ne(i)$ we have*

$$\phi^{\downarrow\lambda(i)\cup\lambda(j)} \quad = \quad \phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(j)}.$$

Proof: We remark that if i and j are neighbors, then either $i = ch(j)$ or $j = ch(i)$. It is therefore sufficient to prove the following statement:

$$\phi^{\downarrow\lambda(i)\cup\lambda(ch(i))} \quad = \quad \phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(ch(i))}. \tag{4.21}$$

It follows from Theorem 4.5, idempotency and the combination axiom that

$$\begin{aligned}
\phi^{\downarrow\lambda(i)\cup\lambda(ch(i))} &= \left(\phi^{\downarrow y_i}\right)^{\downarrow\lambda(i)\cup\lambda(ch(i))} = \left(\bigotimes_{j=i}^{r} \phi^{\downarrow\lambda(j)}\right)^{\downarrow\lambda(i)\cup\lambda(ch(i))} \\
&= \left(\phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(ch(i))} \otimes \bigotimes_{j=i+1}^{r} \phi^{\downarrow\lambda(j)}\right)^{\downarrow\lambda(i)\cup\lambda(ch(i))} \\
&= \phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(ch(i))} \otimes \left(\bigotimes_{j=i+1}^{r} \phi^{\downarrow\lambda(j)}\right)^{\downarrow y_{i+1}\cap(\lambda(i)\cup\lambda(ch(i)))} \\
&= \phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(ch(i))} \otimes \left(\phi^{\downarrow y_{i+1}}\right)^{\downarrow y_{i+1}\cap(\lambda(i)\cup\lambda(ch(i)))}.
\end{aligned}$$

We further conclude from equation (4.19) that

$$\begin{aligned}
y_{i+1} \cap (\lambda(i) \cup \lambda(ch(i))) &= (y_{i+1} \cap \lambda(i)) \cup (y_{i+1} \cap \lambda(ch(i))) \\
&= (\lambda(i) \cap \lambda(ch(i))) \cup \lambda(ch(i)) = \lambda(ch(i))
\end{aligned}$$

and finally obtain

$$\begin{aligned}
\phi^{\downarrow\lambda(i)\cup\lambda(ch(i))} &= \phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(ch(i))} \otimes \left(\phi^{\downarrow y_{i+1}}\right)^{\downarrow\lambda(ch(i))} \\
&= \phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(ch(i))} \otimes \phi^{\downarrow\lambda(ch(i))} = \phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(ch(i))}.
\end{aligned}$$

■

This lemma states that if a new query $x \subseteq d(\phi)$ arrives that is not covered by the join tree, it can nevertheless be answered if we find two neighboring nodes whose union label covers the query, i.e. $x \subseteq \lambda(i) \cup \lambda(j)$. We then compute

$$\phi^{\downarrow x} = \left(\phi^{\downarrow\lambda(i)\cup\lambda(j)}\right)^{\downarrow x} = \left(\phi^{\downarrow\lambda(i)} \otimes \phi^{\downarrow\lambda(j)}\right)^{\downarrow x}$$

Thus, we neither need a new join tree nor a new propagation to answer this query. However, it is admittedly a rare case that two neighboring nodes can be found that cover some new query. The following lemma therefore generalizes this procedure to paths connecting two arbitrary nodes in the join tree.

Lemma 4.6 *Let* $(p_1, \ldots, p_k)$ *be a path from node* p_1 *to node* p_k *with* $p_i \in V$ *for* $1 \leq i \leq k$. *We then have for an idempotent valuation algebra*

$$\phi^{\downarrow \lambda(p_1) \cup \ldots \cup \lambda(p_k)} \quad = \quad \bigotimes_{i=1}^{k} \phi^{\downarrow \lambda(p_i)}. \tag{4.22}$$

Proof: We proceed by induction over the length of the path: For $i = 2$ we have $p_1 \in ne(p_2)$ and the statement follows from Lemma 4.5. For a path $(p_1, \ldots, p_{k+1})$ we merge the nodes $\{p_1, \ldots, p_k\}$ to a single node called z to which all neighbors of p_1 to p_k are connected. The label of node z is $\lambda(z) = \lambda(p_1) \cup \ldots \cup \lambda(p_k)$. Clearly, the running intersection property is still satisfied in the new tree. Since $p_{k+1} \in ne(z)$ we may again apply Lemma 4.5 and obtain,

$$\phi^{\downarrow \lambda(z) \cup \lambda(p_{k+1})} \;=\; \phi^{\downarrow \lambda(z)} \otimes \phi^{\downarrow \lambda(p_{k+1})} \;=\; \left(\phi^{\downarrow \lambda(p_1)} \otimes \ldots \otimes \phi^{\downarrow \lambda(p_k)}\right) \otimes \phi^{\downarrow \lambda(p_{k+1})}$$

The last equality follows from the induction hypothesis. ■

It then follows immediately from this lemma and the transitivity of projection that

$$\phi^{\downarrow \lambda(p_1) \cup \lambda(p_k)} \quad = \quad \left(\bigotimes_{i=1}^{k} \phi^{\downarrow \lambda(p_i)} \right)^{\downarrow \lambda(p_1) \cup \lambda(p_k)} \tag{4.23}$$

This allows us to express the following yet rather inefficient procedure to answer a query $x \subseteq d(\phi)$ that is covered by the union label of two arbitrary join tree nodes:

1. Find two nodes $i, j \in V$ such that $x \subseteq \lambda(i) \cup \lambda(j)$.

2. Combine all node contents on the unique path between i and j.

3. Project the result to x.

We here assume for simplicity that the query is split over only two nodes. If more than two nodes are necessary to cover the query, it is easily possible to generalize Lemma 4.6 to arbitrary subtrees of the join tree instead of only paths. Clearly, this simple procedure is very inefficient and almost amounts to a naive computation of the objective function itself, due to the combination of all node factors on the path between i and j in the second step. However, there is a more intelligent way to perform these computations that only requires combining the content of two neighboring nodes. This is the statement of the following lemma:

Lemma 4.7 *In an idempotent valuation algebra, it holds that*

$$\phi^{\downarrow \lambda(p_1) \cup \lambda(p_k)} \quad = \quad \phi^{\downarrow \lambda(p_k)} \otimes \phi^{\downarrow \lambda(p_1) \cup (\lambda(p_{k-1}) \cap \lambda(p_k))}.$$

Proof: It follows from the transitivity axiom, Lemma 4.6, the combination axiom and the running intersection property that

$$\begin{aligned}
\phi^{\downarrow\lambda(p_1)\cup\lambda(p_k)} &= \left(\phi^{\downarrow\lambda(p_1)\cup\ldots\cup\lambda(p_k)}\right)^{\downarrow\lambda(p_1)\cup\lambda(p_k)} \\
&= \left(\phi^{\downarrow\lambda(p_k)} \otimes \phi^{\downarrow\lambda(p_1)\cup\ldots\cup\lambda(p_{k-1})}\right)^{\downarrow\lambda(p_1)\cup\lambda(p_k)} \\
&= \phi^{\downarrow\lambda(p_k)} \otimes \phi^{\downarrow(\lambda(p_1)\cup\ldots\cup\lambda(p_{k-1}))\cap(\lambda(p_1)\cup\lambda(p_k))} \\
&= \phi^{\downarrow\lambda(p_k)} \otimes \phi^{\downarrow\lambda(p_1)\cup(\lambda(p_{k-1})\cap\lambda(p_k))}.
\end{aligned}$$

■

We now give an algorithm similar to the above procedure that exploits Lemma 4.7 to avoid the combination of all node factors on the path between node p_1 and p_k.

Algorithm 4.7 Specialized Query Answering

```
input:  $(V, E, \lambda, D)$,  $p_1, p_k \in V$,  $x \subseteq \lambda(p_1) \cup \lambda(p_k)$  output:  $\phi^{\downarrow x}$

begin
  $\eta$ := $\phi^{\downarrow\lambda(p_1)}$;
  find a path $(p_1, \ldots, p_k)$;
  for $i = 1 \ldots k-1$ do
    $\eta$ := $\phi^{\downarrow\lambda(p_{i+1})} \otimes \eta^{\downarrow\lambda(p_1)\cup(\lambda(p_i)\cap\lambda(p_{i+1}))}$;
  end;
  return $\eta^{\downarrow x}$;
end
```

Theorem 4.6 *Algorithm 4.7 outputs $\eta^{\downarrow x} = \phi^{\downarrow x}$ for $x \subseteq \lambda(p_1) \cup \lambda(p_k)$.*

Proof: Let us first show that the projection of η in the loop is well-defined. After repetition i of the loop we have:

$$d(\eta) = \lambda(p_{i+1}) \cup \lambda(p_1) \cup (\lambda(p_i) \cap \lambda(p_{i+1})) = \lambda(p_1) \cup \lambda(p_{i+1}). \qquad (4.24)$$

In loop $i + 1$, η is projected to $\lambda(p_1) \cup (\lambda(p_{i+1}) \cap \lambda(p_{i+2})) \subseteq \lambda(p_1) \cup \lambda(p_{i+1})$. Hence, this projection is feasible. We next prove that

$$\eta = \phi^{\downarrow\lambda(p_1)\cup\lambda(p_k)} \qquad (4.25)$$

holds at the end of the loop. The statement of the theorem then follows by the transitivity of projection. For $k = 2$ it follows from Lemma 4.6 that

$$\begin{aligned}
\phi^{\downarrow\lambda(p_1)\cup\lambda(p_2)} &= \phi^{\downarrow\lambda(p_2)} \otimes \phi^{\downarrow\lambda(p_1)} \\
&= \phi^{\downarrow\lambda(p_2)} \otimes \eta = \phi^{\downarrow\lambda(p_2)} \otimes \eta^{\downarrow\lambda(p_1)\cup(\lambda(p_1)\cap\lambda(p_2))}.
\end{aligned}$$

The second equality follows by initialization and the third from the domain axiom. This proves the correctness of the algorithm for paths of length 2. We now assume that equation (4.25) holds for paths of length k. For a path of length $k+1$, the last repetition of the loop computes

$$\begin{aligned} \eta &= \phi^{\downarrow\lambda(p_{k+1})} \otimes \left(\phi^{\downarrow\lambda(p_1)\cup\lambda(p_k)}\right)^{\downarrow\lambda(p_1)\cup(\lambda(p_k)\cap\lambda(p_{k+1}))} \\ &= \phi^{\downarrow\lambda(p_{k+1})} \otimes \phi^{\downarrow\lambda(p_1)\cup(\lambda(p_k)\cap\lambda(p_{k+1}))} \\ &= \phi^{\downarrow\lambda(p_1)\cup\lambda(p_{k+1})} \end{aligned}$$

The first equality follows from the induction hypothesis, the second from transitivity and the third from lemma 4.7. This proves the correctness of equation (4.25 and the statement of the theorem follows by the transitivity of projection. ∎

Example 4.7 *We want to answer the new query $x = \{A, B\}$ in the join tree of Figure 4.3 after a complete run of the idempotent architecture. We immediately see that this query is not covered by the join tree, but it is for example covered by the union label of the two nodes* 1 *and* 6*, i.e. $x \subseteq \lambda(1) \cup \lambda(6) = \{A, B, E, L, T\}$. The path between these two nodes has already been shown in Figure 4.10. The following three steps are executed by Algorithm 4.7:*

Step 1: $\phi^{\downarrow\lambda(1)\cup\lambda(2)} = \phi^{\downarrow\lambda(2)} \otimes \phi^{\downarrow\lambda(1)}$

Step 2: $\phi^{\downarrow\lambda(1)\cup\lambda(5)} = \phi^{\downarrow\lambda(5)} \otimes \phi^{\downarrow\lambda(1)\cup(\lambda(2)\cap\lambda(5))}$

Step 3: $\phi^{\downarrow\lambda(1)\cup\lambda(6)} = \phi^{\downarrow\lambda(6)} \otimes \phi^{\downarrow\lambda(1)\cup(\lambda(5)\cap\lambda(6))}$

Finally, the algorithm returns

$$\left(\phi^{\downarrow\lambda(1)\cup\lambda(6)}\right)^{\downarrow\{A,B\}} = \left(\phi^{\downarrow\{A,B,E,L,T\}}\right)^{\downarrow\{A,B\}} = \phi^{\downarrow\{A,B\}}.$$

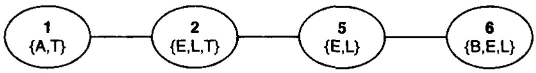


Figure 4.10 The path between node 1 and node 6 in the join tree of Figure 4.3.

A closer look on Algorithm 4.7 or more precisely on equation (4.24) shows that we may extract a lot more information from the intermediate results

$$\eta = \phi^{\downarrow\lambda(p_1)\cup\lambda(p_i)}$$

for $1 \leq i \leq k$. In fact, it is not only possible to compute the final query $x \subseteq \lambda(p_1) \cup \lambda(p_k)$ as stated in Theorem 4.6, but we actually obtain all possible queries that are subsets of $\lambda(p_1) \cup \lambda(p_i)$ by just one additional projection per query. This is the statement of the following complement to Theorem 4.6.

Lemma 4.8 *Let $(p_1, \ldots, p_k)$ be a path from node p_1 to node p_k. Algorithm 4.7 can be adapted to compute $\phi^{\downarrow x}$ for all $x \subseteq \lambda(p_1) \cup \lambda(p_i)$ for $i = 2, \ldots, k$.*

Example 4.8 *From the intermediate results of Example 4.7 we may compute all possible queries that are subsets of $\{A, T, E, L, T\}$ and $\{A, T, B, E, L\}$ by just one additional projection per query. This includes in particular the binary queries $\{A, E\}, \{A, L\}$ and $\{T, B\}$ which are not covered by the join tree in Figure 4.3.*

A further extension of Algorithm 4.7 provides for a previously fixed node $i \in V$ a pre-compilation of the join tree such that all possible queries, that are subsets of $\lambda(i) \cup \lambda(j)$ for all nodes $j \in V$, can be computed by just one additional projection per query. To do so, we start with the selected node i and compute for each neighbor $k \in ne(i)$ the projection of ϕ to $\lambda(i) \cup \lambda(k)$. This intermediate result is stored in the neighbor node k. Then, the process continues for all neighbors of node k that have not yet been considered. If $l \in ne(k)$ is such a neighbor of k, we compute the projection to $\lambda(i) \cup \lambda(l)$ by reusing the intermediate result stored in k. This is possible due to Lemma 4.7:

$$\phi^{\downarrow\lambda(i)\cup\lambda(l)} \quad = \quad \phi^{\downarrow\lambda(l)} \otimes \phi^{\downarrow\lambda(i)\cup(\lambda(k)\cap\lambda(l))}.$$

At the end of this process, each node $j \in V$ contains $\phi^{\downarrow\lambda(i)\cup\lambda(j)}$ which allows us to answer all queries that are subsets of $\lambda(i) \cup \lambda(j)$ by just one additional projection. These projections setup another factorization of the objective function ϕ as a consequence of Lemma 4.6.

Algorithm 4.8 Compilation for Specialized Query Answering

```
input:  $(V, E, \lambda, D)$,  $i \in V$   output:  -

begin
  $\eta_i$ := $\phi^{\downarrow\lambda(i)}$;
  for each $k \in ne(i)$ do
    visit(i, j);
  end;
end

function visit(k,l)
begin
  $\eta_l$ := $\phi^{\downarrow\lambda(l)} \otimes \eta_k^{\downarrow\lambda(i)\cup(\lambda(k)\cap\lambda(l))}$;
  for each $j \in ne(k) - \{l\}$ do
    visit(l,j);
  end;
end
```

4.6.1 The Complexity of Answering Uncovered Queries

We first point out that finding the path between two nodes in a directed tree is particularly easy. We simply compute the intersection of the two paths that connect both nodes with the root node. Due to equation (4.24) the loop statement of Algorithm 4.7 computes a factor of domain $\lambda(p_1) \cup \lambda(p_i)$. Its number of repetitions is bounded by the longest possible path in the join tree, which has at most $|V|$ nodes. We therefore conclude that Algorithm 4.7 adopts a time complexity of

$$\mathcal{O}\left(|V| \cdot f\left(2(\omega^* + 1)\right)\right). \tag{4.26}$$

Since only one combination of two node contents is kept in memory at every time, we obtain for the space complexity

$$\mathcal{O}\left(g\left(2(\omega^* + 1)\right)\right). \tag{4.27}$$

Algorithm 4.8 combines the node content of each node with the content of one of its neighbors. This clearly results in the same time complexity. But in contrast, we store one such factor in each node, which gives us a space complexity of

$$\mathcal{O}\left(|V| \cdot g\left(2(\omega^* + 1)\right)\right) \tag{4.28}$$

for Algorithm 4.8. At first glance, the complexities of these algorithms seem very bad because they are controlled by the double of the treewidth. Especially for formalism with exponential time or space behaviour, it could therefore be more efficient to construct a new join tree that also covers the new queries and execute the complete local computation process from scratch. But whenever we add new queries to the inference problem, we risk obtaining a join tree with a larger treewidth. This makes it difficult to directly compare the two approaches. The definition of a covering join tree ensures that every query is covered by the label of a join tree node. Here, we only require that every query is covered by the union of two arbitrary node labels. We will see in Chapter 9 that important applications exist where this is always satisfied. Imagine for example that our query set consists of all possible queries of two variables $\{X, Y\}$ with $X, Y \in d(\phi)$ and $X \neq Y$. Building a covering join tree for such an inference problem always leads a join tree whose largest node label contains all variables. We then compute the objective function directly and the effect of local computation disappears. On the other hand, it is always guaranteed that such queries are covered by the union label of two join tree nodes. We can therefore ignore the query set in the join tree construction process and obtain the best possible join tree that can be found for the current knowledgebase. Later, queries are answered by the above procedures which clearly is more efficient than computing the objective function directly. Chapter 9 shows that such particular query sets frequently occur when local computation is used for the solution of path problems in sparse networks. Then, the query $\{X, Y\}$ for example represents the shortest distance between two network hosts X and Y, and the query set of all possible pairs of variables models the so-called *all-pairs shortest path problem*.

We have now seen several important concepts that are all related to a division operator in the valuation algebra: namely, two local computation architectures that directly exploit division and the particularly simple idempotent architecture, including a specialized procedure to answer uncovered queries. In addition, the presence of inverse elements allows us to introduce the notion of *scaling* or *normalization* on an algebraic level and also to derive variations of local computation architectures that directly compute scaled results. This is the subject of the next section.

4.7 SCALING AND NORMALIZATION

Scaling or normalization is an important notion in a couple of valuation algebra instances. If, for example, we want to use arithmetic potentials to represent discrete probability distributions, it is semantically important that all values sum up to one. Only in this case are we allowed to talk about probabilities. Similarly, normalized set potentials from Instance 1.4 adopt the semantics of *belief functions* (Shafer, 1976). As mentioned above, the algebraic background for the introduction of a generic scaling operator is a separative valuation algebra. It is shown in this section how scaling can be introduced potentially in every valuation algebra that fulfills this mathematical property. Nevertheless, we emphasize that there must also be a semantical reason that demands a scaling operator, and this cannot be treated on a purely algebraic level.

Following (Kohlas, 2003), we define the scaling or normalization of $\phi \in \Phi$ by

$$\phi^{\downarrow} \quad = \quad \phi \otimes \left(\phi^{\downarrow\emptyset}\right)^{-1}. \tag{4.29}$$

It is important to note that because separativity is assumed, scaled valuations do not necessarily belong to Φ but only to the separative embedding Φ^*. For simplicity and because this holds for all instances presented in this book, we subsequently assume that $\phi^{\downarrow} \in \Phi$ for all $\phi \in \Phi$. This ensures that all projections of $\phi^{\downarrow}$ to $t \subseteq d(\phi)$ are defined, although we are dealing with a valuation algebra with partial projection. We refer to (Kohlas, 2003) for a more general study of scaling without this additional assumption. However, to gain further insights into scalable valuation algebras, we require some results derived for separative valuation algebras in Appendix D.1.1. This is necessary to obtain the following equation and for the proof of Lemma 4.9. Nevertheless, it is possible to understand these statements without knowledge of separative valuation algebras.

Combining both sides of equation (4.29) with $\phi^{\downarrow\emptyset}$ leads to

$$\phi \quad = \quad \phi^{\downarrow} \otimes \phi^{\downarrow\emptyset} \tag{4.30}$$

according to the theory of separative valuation algebras. A valuation and its scale are contained in the same group. The following lemma lists some further elementary properties of scaling. It is important to note that these properties only apply if the scale $\phi^{\downarrow}$ of a valuation ϕ is again contained in the original algebra Φ. This is required since scaling is only defined for elements in Φ.

Lemma 4.9

1. *For $\phi \in \Phi$ with $\phi^{\downarrow} \in \Phi$ we have*

$$(\phi^{\downarrow})^{\downarrow} = \phi^{\downarrow}. \tag{4.31}$$

2. *For $\phi, \psi \in \Phi$ with $\phi^{\downarrow}, \psi^{\downarrow} \in \Phi$ we have*

$$(\phi \otimes \psi)^{\downarrow} = (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow}. \tag{4.32}$$

3. *For $\phi \in \Phi$ with $\phi^{\downarrow} \in \Phi$ and $t \subseteq d(\phi)$ we have*

$$(\phi^{\downarrow})^{\downarrow t} = (\phi^{\downarrow t})^{\downarrow}. \tag{4.33}$$

The first property of Lemma 4.9 states that scaling an already scaled valuation has no effect. Second, the combination of scaled valuations does not generally lead to a scaled valuation anymore. But computing a scaled combination does not require scaling the factors of the combination. Finally, the third statement states that projection and scaling are interchangeable. The proof can be found an Appendix D.3

In a separative valuation algebra with null elements, every null element z_s forms itself a group that is a consequence of cancellativity. Therefore, null elements are inverse to themselves and consequently

$$z_s^{\downarrow} = z_s \otimes \left(z_s^{\downarrow\emptyset}\right)^{-1} = z_s \otimes z_{\emptyset}^{-1} = z_s \otimes z_{\emptyset} = z_s. \tag{4.34}$$

If the separative valuation algebra has neutral elements according to Section 3.3, their scale is generally not a neutral element anymore. This property however is guaranteed if the valuation algebra is stable. It follows from (4.30) that

$$e_s = e_s^{\downarrow} \otimes e_s^{\downarrow\emptyset} = e_s^{\downarrow} \otimes e_{\emptyset} = e_s^{\downarrow}. \tag{4.35}$$

Let $\Phi^{\downarrow}$ be the set of all scaled valuations, i.e.

$$\Phi^{\downarrow} = \{\phi^{\downarrow} : \phi \in \Phi\}. \tag{4.36}$$

The operation of labeling is well-defined in $\Phi^{\downarrow}$. If we assume furthermore that all scales are element of Φ, i.e. $\Phi^{\downarrow} \subseteq \Phi$, we know from Lemma 4.9 that $\Phi^{\downarrow}$ is also closed under projection. However, since a combination of scaled valuations does generally not yield a scaled valuation anymore, we define

$$\phi^{\downarrow} \oplus \psi^{\downarrow} = (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow}. \tag{4.37}$$

Note that Lemma 4.9 also implies

$$\phi^{\downarrow} \oplus \psi^{\downarrow} = (\phi \otimes \psi)^{\downarrow}. \tag{4.38}$$

The requirement that all scales are elements of Φ holds in particular if the valuation algebra is regular. If even idempotency holds, every valuation is the scale of itself,

$$\phi^{\downarrow} = \phi \otimes \left(\phi^{\downarrow\emptyset}\right)^{-1} = \phi \otimes \left(\phi^{\downarrow\emptyset}\right) = \phi. \tag{4.39}$$

Consequently, we have $\Phi^{\downarrow} = \Phi$ in case of idempotency.

Theorem 4.7 *Assume $\Phi^{\downarrow} \subseteq \Phi$. Then, $\langle\Phi^{\downarrow}, D\rangle$ with the operations of labeling d, combination $\oplus$, and projection $\downarrow$ is a valuation algebra.*

Proof: We verify the axioms from Section 1.1 on $\langle\Phi^{\downarrow}, D\rangle$:

(A1) *Commutative Semigroup:* Commutativity of $\oplus$ follows directly from (4.37). Associativity is derived as follows:

$$\begin{aligned}(\phi^{\downarrow} \oplus \psi^{\downarrow}) \oplus \chi^{\downarrow} &= (\phi \otimes \psi)^{\downarrow} \oplus \chi^{\downarrow} = ((\phi \otimes \psi) \otimes \chi)^{\downarrow} \\ &= (\phi \otimes (\psi \otimes \chi))^{\downarrow} = \phi^{\downarrow} \oplus (\psi \otimes \chi)^{\downarrow} \\ &= \phi^{\downarrow} \oplus (\psi^{\downarrow} \oplus \chi^{\downarrow}).\end{aligned}$$

(A2) *Labeling:* Since a valuation and its scale both have the same domain,

$$d(\phi^{\downarrow} \oplus \psi^{\downarrow}) = d((\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow}) = d(\phi^{\downarrow} \otimes \psi^{\downarrow}) = d(\phi^{\downarrow}) \cup d(\psi^{\downarrow}).$$

(A3) *Projection:* For $\phi^{\downarrow} \in \Phi^{\downarrow}$ with $d(\phi^{\downarrow}) = x$ we have

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

(A4) *Transitivity:* For $y \subseteq x \subseteq d(\phi^{\downarrow})$ it follows that

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

(A5) *Combination:* For $\phi^{\downarrow}$, $\psi^{\downarrow} \in \Phi^{\downarrow}$ with $d(\phi^{\downarrow}) = x$, $d(\psi^{\downarrow}) = y$ and $z \in D$ such that $x \subseteq z \subseteq x \cup y$, we have

$$\begin{aligned}(\phi^{\downarrow} \oplus \psi^{\downarrow})^{\downarrow z} &= ((\phi \otimes \psi)^{\downarrow})^{\downarrow z} = ((\phi \otimes \psi)^{\downarrow z})^{\downarrow} \\ &= (\phi \otimes \psi^{\downarrow y \cap z})^{\downarrow} = \phi^{\downarrow} \oplus (\psi^{\downarrow y \cap z})^{\downarrow} = \phi^{\downarrow} \oplus (\psi^{\downarrow})^{\downarrow y \cap z}.\end{aligned}$$

(A6) *Domain:* For $\phi^{\downarrow} \in \Phi^{\downarrow}$ and $d(\phi^{\downarrow}) = x$,

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

■

Accepting the requirement that $\Phi^{\downarrow} \subseteq \Phi$, the mapping $\phi \mapsto \phi^{\downarrow}$ is a homomorphism from $\langle\Phi, D\rangle$ to $\langle\Phi^{\downarrow}, D\rangle$. The latter is called *scaled valuation algebra* associated with $\langle\Phi, D\rangle$. Finally, in a valuation algebra $\langle\Phi', D\rangle$ with adjoined identity element e, this element is not affected from scaling. From $e = e^{-1}$ follows that

$$e^{\downarrow} = e \otimes \left(e^{\downarrow\emptyset}\right)^{-1} = e \otimes e^{-1} = e \otimes e = e. \tag{4.40}$$

Let us now consider two examples of separative valuation algebras with a semantical requirement for scaling. A third example can be found in Appendix D.3 of this chapter and in Appendix E.5 of Chapter 5.

4.4 Probability Potentials

Normalized arithmetic potentials often take a preferred position because they adopt the interpretation of *discrete probability distributions*. An arithmetic potential p with domain s is normalized if its values sum up to one, i.e. if

$$\sum_{\mathbf{x}\in\Omega_s} p(\mathbf{x}) \quad = \quad 1.$$

This is achieved by applying the generic scaling operation of Definition 4.29:

$$p^{\downarrow}(\mathbf{x}) \quad = \quad p(\mathbf{x}) \cdot \left(p^{\downarrow\emptyset}\right)^{-1}(\diamond) \quad = \quad \frac{p(\mathbf{x})}{\sum_{\mathbf{y}\in\Omega_s} p(\mathbf{y})}. \tag{4.41}$$

We illustrate this process by applying the scaling operation to the arithmetic potential p_3 from Instance 1.3. Since $p_3^{\downarrow\emptyset}(\diamond) = 2$ we obtain

$p_3 =$

A	B	C	
a	b	c	0.12
a	b	$\overline{c}$	0.48
a	$\overline{b}$	c	0.36
a	$\overline{b}$	$\overline{c}$	0.04
$\overline{a}$	b	c	0.06
$\overline{a}$	b	$\overline{c}$	0.24
$\overline{a}$	$\overline{b}$	c	0.63
$\overline{a}$	$\overline{b}$	$\overline{c}$	0.07

$p_3^{\downarrow} =$

A	B	C	
a	b	c	0.060
a	b	$\overline{c}$	0.240
a	$\overline{b}$	c	0.180
a	$\overline{b}$	$\overline{c}$	0.020
$\overline{a}$	b	c	0.030
$\overline{a}$	b	$\overline{c}$	0.120
$\overline{a}$	$\overline{b}$	c	0.315
$\overline{a}$	$\overline{b}$	$\overline{c}$	0.035

4.5 Probability Density Functions

Continuous probability distributions are a particularly important class of density functions and motivate our interest in normalized densities. According to equation (4.29), the scale of a density function $f \in \Phi$ with $d(f) = s$ is

$$f^{\downarrow}(\mathbf{x}) \quad = \quad f(\mathbf{x}) \otimes \left(f^{\downarrow\emptyset}\right)^{-1}(\diamond) \quad = \quad \frac{f(\mathbf{x})}{\int f(\mathbf{x})d\mathbf{x}} \tag{4.42}$$

for $\mathbf{x} \in \Omega_s$. Consequently, we have for a scaled density

$$\int f^{\downarrow}(\mathbf{x})d\mathbf{x} \quad = \quad 1.$$

Obtaining scaled results from the computation of an inference problem is an essential requirement for many applications. We may deduce from the second property of Lemma 4.9 that the result of computing a query of an inference problem will not necessarily lead to a scaled valuation again, even though the knowledgebase of the inference problem consists of only scaled valuations. This problem will be analyzed in the following section. To do so, we should also be aware of the computational effort of executing a scaling operation on some valuation $\phi \in \Phi^*$. We know from equation (4.29) that the scale of ϕ is obtained by combining ϕ with the inverse of its projection to the empty domain. Scaling therefore requires executing one projection and one combination. The inverse is computed on the empty domain for all valuations and can therefore be neglected, i.e. this effort is constant for all valuations.

4.8 LOCAL COMPUTATION WITH SCALING

Let us now focus on the task of computing scaled queries $\{x_1, \ldots, x_s\}$ from a knowledgebase $\{\phi_1, \ldots, \phi_n\}$ where $x_i \subseteq d(\phi)$ and $\phi = \phi_1 \otimes \ldots \otimes \phi_n$. We derive from Lemma 4.9 that

$$\begin{aligned}\left(\phi^{\downarrow x_i}\right)^{\downarrow} = \left(\phi^{\downarrow}\right)^{\downarrow x_i} &= \left((\phi_1 \otimes \cdots \otimes \phi_m)^{\downarrow}\right)^{\downarrow x_i} \\ &= \left((\phi_1^{\downarrow} \otimes \cdots \otimes \phi_m^{\downarrow})^{\downarrow}\right)^{\downarrow x_i} = \left(\phi_1^{\downarrow} \oplus \cdots \oplus \phi_m^{\downarrow}\right)^{\downarrow x_i}.\end{aligned}$$

This derivation provides different possibilities. The first expression is the straightforward approach: we compute each query using some local computation architecture and perform a scaling operation on the result. This requires $|x|$ scaling operations, thus $|x|$ combinations but only one projection since the inverse of the objective function projected to the empty domain is identical for all queries. The second expression performs only a single scaling operation on the total objective function. However, we directly refuse this approach since it requires explicitly building the objective function ϕ. A third approach follows from the last expression if we additionally suppose that all scales are contained in Φ, i.e. that $\Phi^{\downarrow} \subseteq \Phi$. We then know from Theorem 4.7 that we may directly compute in the valuation algebra $\langle \Phi^{\downarrow}, D\rangle$. However, this is again worse than the straightforward approach since it executes one scaling operation per combination during the local computation process as a result of equation (4.38).

A far more promising approach arises if we integrate scaling directly into the local computation architectures. Since the underlying valuation algebra is assumed to be separative, we can potentially use the Shenoy-Shafer, Lauritzen-Spiegelhalter or HUGIN architecture. Furthermore, if the valuation algebra is idempotent, we do not need to care about scaling at all because idempotent valuations are always scaled. Thus, we are next going to reconsider the three architectures above and show that only a single execution of scaling in the root node of the join tree is sufficient in each of them to scale all query answers. This is well-known in the context of Bayesian networks, and for general valuation algebras it was shown by (Kohlas, 2003).

4.8.1 The Scaled Shenoy-Shafer Architecture

The *scaled Shenoy-Shafer architecture* first executes the collect phase so that the projection of the objective function ϕ to the root node label $\lambda(r)$ can be computed in the root node $r \in V$. Then, the root content ψ_r is replaced by

$$\psi_r \otimes \left(\phi^{\downarrow\emptyset}\right)^{-1} \quad = \quad \psi_r \otimes \left((\phi^{\downarrow\lambda(r)})^{\downarrow\emptyset}\right)^{-1}. \tag{4.43}$$

When the distribute phase starts, the outgoing messages from the root node are computed from this modified node content. The result of this process is stated in the following theorem.

Theorem 4.8 *At the end of the message-passing in the scaled Shenoy-Shafer architecture, we obtain at node i*

$$\left(\phi^{\downarrow\lambda(i)}\right)^{\downarrow} \;=\; \left(\phi^{\downarrow}\right)^{\downarrow\lambda(i)} \;=\; \psi_i \otimes \bigotimes_{j \in ne(i)} \mu'_{j\to i}, \tag{4.44}$$

where $\mu'_{j\to i}$ are the modified messages from the scaled architecture.

Proof: The messages of the collect phase do not change, i.e. $\mu_{i\to ch(i)} = \mu'_{i\to ch(i)}$. For the distribute messages, we show that

$$\mu'_{ch(i)\to i} \quad = \quad \mu_{ch(i)\to i} \otimes \left(\phi^{\downarrow\emptyset}\right)^{-1}. \tag{4.45}$$

The distribute message sent by the root node to parent i is

$$\begin{aligned}
\mu'_{r\to i} &= \left(\psi_r \otimes \left(\phi^{\downarrow\emptyset}\right)^{-1} \otimes \bigotimes_{j\in ne(r), j\neq i} \mu_{j\to r}\right)^{\downarrow\omega_{r\to i}\cap\lambda(i)} \\
&= \left(\psi_r \otimes \bigotimes_{j\in ne(r), j\neq i} \mu_{j\to r}\right)^{\downarrow\omega_{r\to i}\cap\lambda(i)} \otimes \left(\phi^{\downarrow\emptyset}\right)^{-1} \\
&= \mu_{r\to i} \otimes \left(\phi^{\downarrow\emptyset}\right)^{-1}.
\end{aligned}$$

This follows from the combination axiom. Hence, equation (4.45) holds for all messages that are emitted by the root node. We proceed by induction and assume that the proposition holds for $\mu_{ch(i)\to i}$. Then, by application of the combination axiom, we have for all $j \in pa(i)$

$$\begin{aligned}
\mu'_{i \to j} &= \left(\psi_i \otimes \bigotimes_{k \in ne(i), k \neq j} \mu'_{k \to i}\right)^{\downarrow \omega_{i \to j} \cap \lambda(j)} \\
&= \left(\psi_i \otimes \mu'_{ch(i) \to i} \otimes \bigotimes_{k \in pa(i), k \neq j} \mu_{k \to i}\right)^{\downarrow \omega_{i \to j} \cap \lambda(j)} \\
&= \left(\psi_i \otimes \mu_{ch(i) \to i} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \otimes \bigotimes_{k \in pa(i), k \neq j} \mu_{k \to i}\right)^{\downarrow \omega_{i \to j} \cap \lambda(j)} \\
&= \left(\psi_i \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \otimes \bigotimes_{k \in ne(i), k \neq j} \mu_{k \to i}\right)^{\downarrow \omega_{i \to j} \cap \lambda(j)} \\
&= \left(\psi_i \otimes \bigotimes_{k \in ne(i), k \neq j} \mu_{k \to i}\right)^{\downarrow \omega_{i \to j} \cap \lambda(j)} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} = \mu_{i \to j} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1}.
\end{aligned}$$

By Theorem 4.1 we finally conclude that

$$\begin{aligned}
\psi_i \otimes \bigotimes_{j \in ne(i)} \mu'_{j \to i} &= \psi_i \otimes \mu'_{ch(i) \to i} \bigotimes_{j \in pa(i)} \mu_{j \to i} \\
&= \psi_i \otimes \mu_{ch(i) \to i} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \otimes \bigotimes_{j \in pa(i)} \mu_{j \to i} \\
&= \psi_i \otimes \bigotimes_{j \in ne(i)} \mu_{j \to i} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \\
&= \phi^{\downarrow \lambda(i)} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} = \left(\phi^{\downarrow \lambda(i)}\right)^{\downarrow} = \left(\phi^{\downarrow}\right)^{\downarrow \lambda(i)}.
\end{aligned}$$

■

To sum it up, the execution of one single scaling operation in the root node scales all projections of the objective function in the Shenoy-Shafer architecture. We summarize this modified Shenoy-Shafer architecture in terms of a pseudo-code algorithm. Since the modification only concerns the message-passing, only Algorithm 4.1 must be adapted.

Algorithm 4.9 Scaled Shenoy-Shafer Architecture: Message-Passing

```
input:  $(V, E, \lambda, D)$   output:  -

begin
  for $i = 1 \ldots |V| - 1$ do                                  // Collect Phase:
    $\omega$ := $d(\psi_i) \cup \bigcup_{j \in pa(i)} d(\mu_{j \to i})$;
    $\mu_{i \rightarrow ch(i)}$ := $(\psi_i \otimes \bigotimes_{j \in pa(i)} \mu_{j \rightarrow i})^{\downarrow \omega \cap \lambda(ch(i))}$;        // in mailbox
  end;
  $\phi^{\downarrow \emptyset}$ := $(\psi_r \otimes \bigotimes_{j \in pa(r)} \mu_{j \rightarrow r})^{\downarrow \emptyset}$;        // Scaling:
  $\psi_r$ := $\psi_r \otimes (\phi^{\downarrow \emptyset})^{-1}$;
  for $i = |V| - 1 \ldots 1$ do        // Distribute Phase:
    $\omega$ := $d(\psi_i) \cup \bigcup_{j \in ne(ch(i)) \wedge j \neq i} d(\mu_{j \rightarrow i})$;
    $\mu_{ch(i) \rightarrow i}$ := $(\psi_{ch(i)} \otimes \bigotimes_{j \in ne(ch(i)) \wedge j \neq i} \mu_{j \rightarrow i})^{\downarrow \omega \cap \lambda(i)}$;        // in mailbox
  end;
end
```

4.8.2 The Scaled Lauritzen-Spiegelhalter Architecture

The Lauritzen-Spiegelhalter architecture first executes the collect algorithm with the extension that emitted messages are divided out of the node content. At the end of this collect phase, the root node directly contains the projection of the objective function ϕ to the root node label $\lambda(r)$. We pursue a similar strategy as in the scaled Shenoy-Shafer architecture and replace the root content by its scale

$$(\phi^{\downarrow})^{\downarrow \lambda(r)} \quad = \quad \phi^{\downarrow \lambda(r)} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1}. \tag{4.46}$$

Then, the distribute phase proceeds as usually.

Theorem 4.9 *At the end of the scaled Lauritzen-Spiegelhalter architecture, node i contains $(\phi^{\downarrow \lambda(i)})^{\downarrow} = (\phi^{\downarrow})^{\downarrow \lambda(i)}$ provided that all messages during an execution of the Shenoy-Shafer architecture can be computed.*

Proof: By equation (4.46), the theorem is satisfied for the root node. We proceed by induction and assume that the proposition holds for the node $ch(i)$ which therefore contains $(\phi^{\downarrow})^{\downarrow \lambda(ch(i))}$. The content of node i in turn is given by equation (4.13). Applying Theorem 4.2 gives

$$\begin{aligned}
& \psi_i' \otimes \mu_{i \rightarrow ch(i)}^{-1} \otimes (\phi^{\downarrow})^{\downarrow \lambda(i) \cap \lambda(ch(i))} \\
= \; & \psi_i' \otimes \mu_{i \rightarrow ch(i)}^{-1} \otimes \phi^{\downarrow \lambda(i) \cap \lambda(ch(i))} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \\
= \; & \phi^{\downarrow \lambda(i)} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \\
= \; & \left(\phi^{\downarrow \lambda(i)}\right)^{\downarrow} = \left(\phi^{\downarrow}\right)^{\downarrow \lambda(i)}.
\end{aligned}$$

■

Again, a single execution of scaling in the root node scales all results in the Lauritzen-Spiegelhalter architecture. We again give the required modification of Algorithm 4.3 explicitly:

Algorithm 4.10 Scaled Lauritzen-Spiegelhalter Architecture: Message-Passing

```
input:  $(V, E, \lambda, D)$  output:  -

begin
  for $i = 1 \ldots |V| - 1$ do                              // Collect Phase:
    $\mu_{i \to ch(i)} := \psi_i^{\downarrow d(\psi_i) \cap \lambda(ch(i))}$
    $\psi_{ch(i)} := \psi_{ch(i)} \otimes \mu_{i \to ch(i)}$;
    $\psi_i := \psi_i \otimes \mu_{i \to ch(i)}^{-1}$;
  end;
  $\psi_r := \psi_r \otimes (\psi_r^{\downarrow \emptyset})^{-1}$;                        // Scaling:
  for $i = |V| - 1 \ldots 1$ do                            // Distribute Phase:
    $\mu_{ch(i) \to i} := \psi_{ch(i)}^{\downarrow \lambda(i) \cap \lambda(ch(i))}$;
    $\psi_i := \psi_i \otimes \mu_{ch(i) \to i}$;
  end;
end
```

4.8.3 The Scaled HUGIN Architecture

Scaling in the HUGIN architecture is equal to Lauritzen-Spiegelhalter. We again execute the collect phase and modify the root content according to equation (4.46). Then, the outward phase starts in the usual way.

Theorem 4.10 *At the end of the computations in the HUGIN architecture, each node $i \in V$ stores $(\phi^{\downarrow \lambda(i)})^{\downarrow} = (\phi^{\downarrow})^{\downarrow \lambda(i)}$ and every separator between i and j the projection $(\phi^{\downarrow \lambda(i) \cap \lambda(j)})^{\downarrow} = (\phi^{\downarrow})^{\downarrow \lambda(i) \cap \lambda(j)}$ provided that all messages during an execution of the Shenoy-Shafer architecture can be computed.*

Proof: By equation (4.46), the theorem is satisfied for the root node. We proceed by induction and assume that the proposition holds for node $ch(i)$. Then, the separator lying between i and $ch(i)$ will update its stored content to

$$(\phi^{\downarrow})^{\downarrow \lambda(i) \cap \lambda(ch(i))} \otimes \mu_{i \to ch(i)}^{-1} \quad = \quad \phi^{\downarrow \lambda(i) \cap \lambda(ch(i))} \otimes \mu_{i \to ch(i)}^{-1} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1}.$$

Node i contains ψ_i' when receiving the forwarded separator content and computes

$$\psi_i' \otimes \phi^{\downarrow \lambda(i) \cap \lambda(ch(i))} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \otimes \mu_{i \to ch(i)}^{-1} \quad =$$

$$\phi^{\downarrow \lambda(i)} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \quad = \quad \left(\phi^{\downarrow}\right)^{\downarrow \lambda(i)} \quad = \quad \left(\phi^{\downarrow \lambda(i)}\right)^{\downarrow}.$$

■

Once more, a single execution of scaling in the root node scales all results in the HUGIN architecture. It follows the corresponding adaption of Algorithm 4.5.

Algorithm 4.11 Scaled HUGIN Architecture: Message-Passing

```
input:  $(V, E, \lambda, D)$  output:  -

begin
  for $i = 1 \ldots |V| - 1$ do                                    // Collect Phase:
    $\mu_{i \rightarrow ch(i)} := \psi_i^{\downarrow d(\psi_i) \cap \lambda(ch(i))}$
    $\psi_{sep(i)} := \mu_{i \rightarrow ch(i)}$;
    $\psi_{ch(i)} := \psi_{ch(i)} \otimes \mu_{i \rightarrow ch(i)}$;
  end;
  $\psi_r := \psi_r \otimes (\psi_r^{\downarrow \emptyset})^{-1}$;                                        // Scaling:
  for $i = |V| - 1 \ldots 1$ do                                 // Distribute Phase:
    $\mu'_{ch(i) \rightarrow i} := \psi_{ch(i)}^{\downarrow \lambda(i) \cap \lambda(ch(i))}$;
    $\mu_{ch(i) \rightarrow i} := \mu'_{ch(i) \rightarrow i} \otimes \psi_{sep(i)}^{-1}$;
    $\psi_{sep(i)} := \mu'_{i \rightarrow ch(i)}$;
    $\psi_i := \psi_i \otimes \mu_{ch(i) \rightarrow i}$;
  end;
end
```

These adaptions for scaling close the discussion of local computation. Further local computation schemes but for an extended computational task will be introduced in Chapters 9 and 8.

4.9 CONCLUSION

The topic of this chapter was the solution of inference problems where multiple queries have to be computed. We quickly remarked that a repeated application of the generalized collect algorithm for each query leads to a lot of redundant computations. On the other hand, a more detailed analysis has shown that two such runs differ only in the messages sent between the two root nodes. This fact is exploited by the Shenoy-Shafer architecture that solves a multi-query inference problem by only the double number of messages used for the computation of a single query. In addition, since the Shenoy-Shafer architecture stores messages in mailboxes instead of combining them to the node content, we obtain the best possible space complexity one can expect for a general local computation scheme. However, the prize we have to pay is that a lot of redundant operations are still performed, unless particular join trees such as binary join trees are used. This was the key motivation for the introduction of further local computation procedure with a more sophisticated caching policy. These architectures benefit from a division operation that is in general not present in a valuation algebra. We therefore discussed three different, algebraic properties in the appendix of this chapter that allow the introduction of division: if the valuation algebra is separative, it can be embedded into a union of groups which naturally provide inverse elements. This is the most general property and leads to an extension of the original valuation algebra where division is possible but where projection is only partially defined. Alternatively, if the valuation algebra is regular, it decomposes directly into a union of groups and full projection is conserved. The third and strongest property is idempotency. In this case, every element is the inverse of itself and division becomes trivial. Back to local computation, we developed two specialized architectures that apply to separative (and hence regular) valuation algebras. The Lauritzen-Spiegelhalter architecture executes the collect algorithm in its inward phase but always divides the message for the child node out of each node content. This ensures that no information is combined a second time when the distribute message arrives. Here, the division takes place on the node content and all messages can be discarded as soon as they have been combined to some node content. On the other hand, we need to store one factor per node whose size is bounded by the node label. Thus, we avoid the redundant computations of the Shenoy-Shafer architecture to the expense of a considerably higher space requirement. The second scheme called HUGIN architecture delays division to the distribute phase where it only takes place on the separators. This makes division less costly but requires to store the collect messages until the end of the message-passing. A fourth and last local computation scheme called idempotent architecture can be applied if idempotency holds in the valuation algebra. Essentially, it is equal to the Lauritzen-Spiegelhalter architecture but passes on the execution of trivial division. Thus, no additional effort is spent in division and no messages must be cached. The very strong property of idempotency further enables a specialized procedure to answer queries which are not covered by the join tree. This procedure becomes especially worthwhile when a large number of queries has to be computed. Finally, we pointed out that scaling or normalization is an important issue for many applications. From the algebraic point of view, we propose a generic definition of scaling when division is present, i.e. if the valuation algebra is at least separative. However, there must also be a semantical motivation for a scaling operation in a valuation algebra which cannot be justified on a purely algebraic level. If this requirement exists, we are naturally interested in computing scaled queries and it turned out that a slight modification of local computation architectures allows computing an arbitrary number of scaled queries with only one single execution of the scaling operation in the root node of the join tree. Figure 4.11 summarizes the local computation procedures and their algebraic requirements.

Appendix: Valuation Algebras with Division

In the first part of the appendix, we will study under which conditions valuation algebras provide a division operation. It has already been mentioned that the pure existence of inverse elements according to Definition 4.1 is not sufficient since this only refers to combination. But in order to apply division in local computation architectures, we also need a property for the projection operation. However, the definition of inverse elements is nevertheless a good starting point for the understanding of division in the context of valuation algebras, and it turns out that all properties studied below imply their existence. (Kohlas, 2003) distinguishes three different cases: *Separativity* is the weakest requirement. It allows the valuation algebra to be embedded into a union of groups that naturally include inverse elements. However, the price we

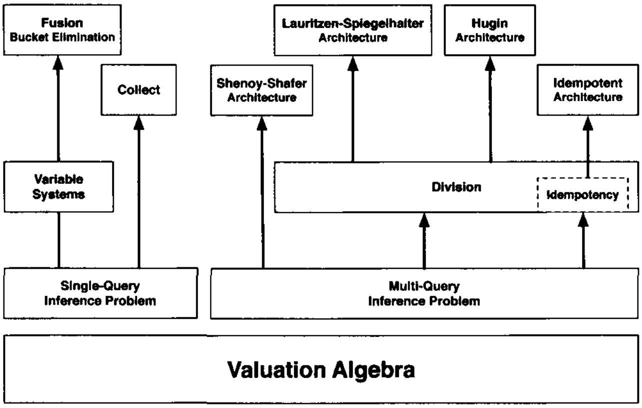


Figure 4.11 A graphical summary of local computation architectures.

pay is that full projection gets lost and we therefore recommend reading Appendix A.3 before entering the study of separative valuation algebras. Alternatively, *regularity* is a stronger condition. In this case, the valuation algebra itself is decomposed into groups, such that full projection is conserved. Finally, if the third and strongest requirement called *idempotency* holds, every valuation turns out to be the inverse of itself. This has already been shown in Section 4.2.1 but here we are going to treat idempotency as a special case of regularity and thus separativity.

D.1 PROPERTIES FOR THE INTRODUCTION OF DIVISION

Equivalence relations play an important role in all three cases since they will be used for the decomposition of Φ. An equivalence relation γ is a*reflexive*, *symmetric* and *transitive* relation. Thus, we have for $\phi, \psi, \eta \in \Phi$

1. *Reflexivity:* $\phi \equiv \phi \pmod{\gamma}$,
2. *Symmetry:* $\phi \equiv \psi \pmod{\gamma}$ implies $\psi \equiv \phi \pmod{\gamma}$,
3. *Transitivity:* $\phi \equiv \psi \pmod{\gamma}$ and $\psi \equiv \eta \pmod{\gamma}$ imply $\phi \equiv \eta \pmod{\gamma}$.

Since we want to partition a valuation algebra, equivalence relations that are compatible with the operations in Φ are of particular importance. Such relations are called *congruences* and satisfy the following properties:

1. $\phi \equiv \psi \pmod{\gamma}$ implies $d(\phi) = d(\psi)$.
2. $\phi \equiv \psi \pmod{\gamma}$ implies $\phi^{\downarrow x} \equiv \psi^{\downarrow x} \pmod{\gamma}$ if $x \subseteq d(\phi) = d(\psi)$.
3. $\phi_1 \equiv \psi_1 \pmod{\gamma}$ and $\phi_2 \equiv \psi_2 \pmod{\gamma}$ imply $\phi_1 \otimes \phi_2 \equiv \psi_1 \otimes \psi_2 \pmod{\gamma}$.

An intuitive motivation for the introduction of congruences in valuation algebras is that different valuations sometimes represent the same knowledge. Thus, they are pooled in equivalence classes that are induced by such a congruence relation γ.

D.1.1 Separative Valuation Algebras

The mathematical notion of a *group* involves a set with a binary operation that is associative, has a unique identity and each element has an inverse. Thus, we may obtain inverse elements in a valuation algebra if the latter can be embedded into a union of groups. It is known from semigroup theory (Clifford & Preston, 1967) that the property of *cancellativity* is sufficient to embed a semigroup into a union of groups, and it is therefore reasonable to apply this technique also for valuation algebras which, according to Axiom (A1), form a semigroup under combination (Lauritzen & Jensen, 1997). See also Exercise A.4 in Chapter 1. However, (Kohlas, 2003) remarked that a more restrictive requirement is needed for valuation algebras that also accounts for the operation of projection. These prerequisites are given in the following definition of *separativity* where we assume a congruence γ that divides Φ into disjoint equivalence classes $[\phi]_\gamma$ given by

$$[\phi]_\gamma \quad = \quad \{\psi \in \Phi : \psi \equiv \phi \pmod{\gamma}\}. \tag{D.1}$$

Definition D.4 *A valuation algebra* $\langle \Phi, D \rangle$ *is called* separative, *if there is a congruence* γ *such that*

- *for all* $\psi, \psi' \in [\phi]_\gamma$ *with* $\phi \otimes \psi = \phi \otimes \psi'$*, we have* $\psi = \psi'$*,*
- *for all* $\phi \in \Phi$ *and* $t \subseteq d(\phi)$*;*

$$\phi^{\downarrow t} \otimes \phi \quad \equiv \quad \phi \pmod{\gamma}. \tag{D.2}$$

From the properties of a congruence, it follows that all elements of an equivalence class have the same domain. Further, the equivalence classes are closed under combination. For $\phi, \psi \in [\phi]_\gamma$ and thus $\phi \equiv \psi \pmod{\gamma}$, we conclude from (D.2)

$$\phi \otimes \psi \;\equiv\; \phi \otimes \phi \;\equiv\; \phi \pmod{\gamma}.$$

Additionally, since all $[\phi]_\gamma$ are subsets of Φ, combination within an equivalence class is both associative and commutative. Hence, Φ decomposes into a family of disjoint, *commutative semigroups*

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

Semigroups obeying the first property in the definition of separativity are called *cancellative* (Clifford & Preston, 1967) and it follows from semigroup theory that every cancellative and commutative semigroup $[\phi]_\gamma$ can be embedded into a group. These groups are denoted by $\gamma(\phi)$ and contain pairs (ϕ, ψ) of elements from $[\phi]_\gamma$

(Croisot, 1953; Tamura & Kimura, 1954). This is similar to the introduction of positive, rational numbers as pairs (numerator and denominator) of natural numbers. Two group elements (ϕ, ψ) and (ϕ', ψ') are identified if $\phi \otimes \psi' = \phi' \otimes \psi$. Further, combination within $\gamma(\phi)$ is defined by

$$(\phi, \psi) \otimes (\phi', \psi') \quad = \quad (\phi \otimes \phi', \psi \otimes \psi'). \tag{D.3}$$

Let Φ^* be the union of those groups $\gamma(\phi)$, i.e.

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

We define the combination $\otimes^*$ of elements $(\phi, \psi), (\phi', \psi') \in \Phi^*$ by

$$(\phi, \psi) \otimes^* (\phi', \psi') \quad = \quad (\phi \otimes \phi', \psi \otimes \psi'). \tag{D.4}$$

It can easily be shown that this combination is well-defined, associative and commutative (Kohlas, 2003). Φ^* is therefore a semigroup under $\otimes^*$. Moreover, the mapping from Φ to Φ^* defined by

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

is a semigroup homomorphism which is furthermore one-to-one due to the cancellativity of the semigroup $[\phi]_\gamma$. In this way, Φ is embedded as a semigroup into Φ^*. Subsequently, we identify $\phi \in \Phi$ with its counterpart $(\phi \otimes \phi, \phi)$ in Φ^*. Since $\gamma(\phi)$ are groups, they contain both inverses and an identity element. We therefore write

$$\phi^{-1} \quad = \quad (\phi, \phi \otimes \phi)$$

for the inverse of group element ϕ, and denote the identity element within $\gamma(\phi)$ as $f_{\gamma(\phi)}$. Note that neither inverses nor identity elements necessarily belong to Φ but only to Φ^*. The embedding of Φ into a union of groups Φ^* via its decomposition into commutative semigroups is illustrated in Figure D.1.

We next introduce a relation between the semigroups $[\phi]_\gamma$. Since γ is a congruence, we may define the combination between congruence classes as

$$[\psi]_\gamma \otimes [\phi]_\gamma = [\phi \otimes \psi]_\gamma, \tag{D.5}$$

and say that

$$[\psi]_\gamma \leq [\phi]_\gamma \quad \text{if} \quad [\psi \otimes \phi]_\gamma = [\phi]_\gamma. \tag{D.6}$$

This relation is a partial order, i.e. reflexive, transitive and antisymmetric according to Definition A.2, as shown in (Kohlas, 2003). It can further be carried over to $\gamma(\phi)$ by defining

$$\gamma(\psi) \leq \gamma(\phi) \quad \text{if} \quad [\psi]_\gamma \leq [\phi]_\gamma.$$

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

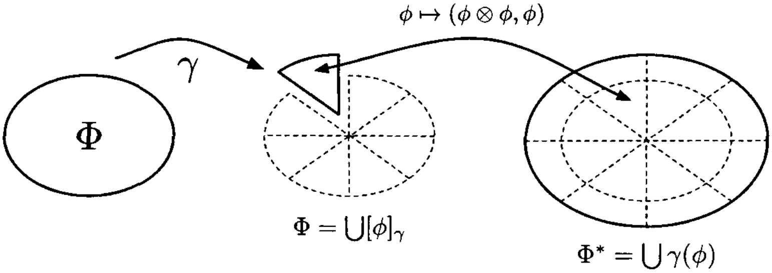

$\phi \mapsto (\phi \otimes \phi, \phi)$
γ
Φ
$\Phi = \bigcup [\phi]_\gamma$
$\Phi^* = \bigcup \gamma(\phi)$

Figure D.1 In a separative valuation algebra Φ is decomposed into disjoint equivalence classes, which are cancellative semigroups, by the congruence γ and finally embedded into a union of groups Φ^*.

Lemma D.10

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

It is now possible to extend (partially) the operations of labeling, projection and combination from $\langle \Phi, D \rangle$ to $\langle \Phi^*, D \rangle$:

- Labeling d^* for elements in Φ^* is defined for $\eta \in \Phi^*$ by $d^*(\eta) = d(\psi)$ for some $\psi \in \Phi$, if $\eta \in \gamma(\psi)$. Since γ is a congruence, d^* does not depend on the representative ψ of the group $\gamma(\psi)$. Therefore, d^* is well-defined. Further, the label for an element $\phi \in \Phi$ is $d^*(\phi) = d(\phi)$. In other words, d^* is an extension of d, and since η and η^{-1} are contained in the same group, we directly get $d^*(\eta) = d^*(\eta^{-1})$.

- We have already seen in equation (D.4) how the combination is carried out among elements in Φ^*. This operator also extends $\otimes$, since

$$\phi \otimes^* \psi \; = \; (\phi \otimes \phi, \phi) \otimes^* (\psi \otimes \psi, \psi) \; = \; (\phi \otimes \psi \otimes \phi \otimes \psi, \phi \otimes \psi) \; = \; \phi \otimes \psi$$

  for elements $\phi, \psi \in \Phi$, which are identified by $(\phi \otimes \phi, \phi) \in \Phi^*$ and $(\psi \otimes \psi, \psi) \in \Phi^*$ respectively. With this extension of combination, we further remark that ϕ and ϕ^{-1} are indeed inverses according to Definition 4.1, since

$$\phi \otimes^* \phi^{-1} \otimes^* \phi \; = \; \phi \otimes^* f_{\gamma(\phi)} \; = \; \phi \tag{D.7}$$

  and

$$\phi^{-1} \otimes^* \phi \otimes^* \phi^{-1} \; = \; \phi^{-1} \otimes^* f_{\gamma(\phi)} \; = \; \phi^{-1}. \tag{D.8}$$

We therefore conclude that separativity is a sufficient property to guarantee the existence of inverse elements.

- Finally, we partially extend the operator of projection. Given an element $\eta \in \Phi^*$, the projection $\downarrow^*$ of η to a domain s is defined by

$$\eta^{\downarrow * s} \quad = \quad \phi^{\downarrow s} \otimes^* \psi^{-1}, \tag{D.9}$$

if there are $\phi, \psi \in \Phi$ with $d(\psi) \subseteq s \subseteq d(\phi)$ and $\gamma(\psi) \leq \gamma(\phi)$ such that

$$\eta \quad = \quad \phi \otimes^* \psi^{-1}.$$

This definition will be justified using the following lemma.

Lemma D.11 *Let $\eta = (\phi, \psi)$ be the pair representation of an element $\eta \in \Phi^*$ with $\phi, \psi \in \Phi$ and $d(\phi) = d(\psi)$. Then η can be written as*

$$\eta \quad = \quad \phi \otimes^* \psi^{-1}.$$

Proof: Because γ is a congruence, we have $d^*(\eta) = d(\phi) = d(\psi)$ and further

$$\begin{aligned}(\phi, \psi) &= (\phi \otimes \phi \otimes \psi, \psi \otimes \phi \otimes \psi) \\ &= (\phi \otimes \phi, \phi) \otimes^* (\psi, \psi \otimes \psi) \\ &= \phi \otimes^* \psi^{-1}.\end{aligned}$$

■

Note again the similarity to positive, rational numbers represented as pairs of naturals. So, any element $\eta \in \Phi^*$ can at least be projected to its own domain. It will next be shown that $\downarrow^*$ is well-defined. Take two representations of the same element $\eta_1, \eta_2 \in \Phi^*$ with $\eta_1 = (\phi, \psi)$, $\eta_2 = (\phi', \psi')$ and $\phi \otimes \psi' = \psi \otimes \phi'$ such that $\eta_1 = \eta_2$. Assume that both can be projected to a domain s, i.e. there are $\phi_1, \phi_2, \psi_1, \psi_2 \in \Phi$, $\gamma(\psi_1) \leq \gamma(\phi_1)$, $\gamma(\psi_2) \leq \gamma(\phi_2)$ and $d(\psi_1) \subseteq s \subseteq d(\phi_1)$, $d(\psi_2) \subseteq s \subseteq d(\phi_2)$ such that $\eta_1 = \phi_1 \otimes^* \psi_1^{-1}$ and $\eta_2 = \phi_2 \otimes^* \psi_2^{-1}$. We show that

$$\eta_1^{\downarrow * s} \quad = \quad \eta_2^{\downarrow * s}. \tag{D.10}$$

First, we conclude from $\eta_1 = \eta_2$ that

$$\phi_1 \otimes^* \psi_1^{-1} \quad = \quad \phi_2 \otimes^* \psi_2^{-1}.$$

We multiply both sides with $\psi_1 \otimes^* \psi_2$ and obtain

$$\phi_1 \otimes^* \psi_1^{-1} \otimes^* \psi_1 \otimes^* \psi_2 \quad = \quad \phi_2 \otimes^* \psi_2^{-1} \otimes^* \psi_2 \otimes^* \psi_1,$$

thus

$$\phi_1 \otimes^* f_{\gamma(\psi_1)} \otimes^* \psi_2 \quad = \quad \phi_2 \otimes^* f_{\gamma(\psi_2)} \otimes^* \psi_1.$$

Since $\gamma(\psi_1) \leq \gamma(\phi_1)$ and $\gamma(\psi_2) \leq \gamma(\phi_2)$, Lemma D.10 implies

$$\phi_1 \otimes^* \psi_2 \quad = \quad \phi_2 \otimes^* \psi_1,$$

and because $\otimes^*$ extends the combination in Φ, $\phi_1 \otimes \psi_2 = \phi_2 \otimes \psi_1$ must hold. By application of the combination axiom in $\langle \Phi, D \rangle$

$$\phi_1^{\downarrow s} \otimes \psi_2 \;=\; (\phi_1 \otimes \psi_2)^{\downarrow s} \;=\; (\phi_2 \otimes \psi_1)^{\downarrow s} \;=\; \phi_2^{\downarrow s} \otimes \psi_1. \tag{D.11}$$

Next, it will be shown that

$$\gamma(\psi_1) \;\leq\; \gamma(\phi_2^{\downarrow s}) \quad \text{and} \quad \gamma(\psi_2) \;\leq\; \gamma(\phi_1^{\downarrow s}). \tag{D.12}$$

Since $\gamma(\psi_1) = \gamma(\psi_1^{-1}) \leq \gamma(\phi_1)$, equation (D.6) implies that $\gamma(\phi_1) = \gamma(\phi_1 \otimes \psi_1) = \gamma(\phi_1 \otimes \psi_1^{-1})$ and therefore

$$\gamma(\phi_1) \;=\; \gamma(\phi_1 \otimes^* \psi_1^{-1}) \;=\; \gamma(\phi_2 \otimes^* \psi_2^{-1}) \;=\; \gamma(\phi_2). \tag{D.13}$$

On the other hand, the combination axiom in $\langle \Phi, D \rangle$ yields

$$\gamma(\phi_1^{\downarrow s}) \;=\; \gamma((\phi_1 \otimes \psi_1)^{\downarrow s}) \;=\; \gamma(\phi_1^{\downarrow s} \otimes \psi_1).$$

Therefore $\gamma(\psi_1) \leq \gamma(\phi_1^{\downarrow s})$ and using equation (D.13) we obtain $\gamma(\psi_1) \leq \gamma(\phi_2^{\downarrow s})$. This proves the first relation of (D.12) and the second is obtained by symmetry. Finally, we deduce from multiplying both sides of (D.11) with $\psi_1^{-1} \otimes^* \psi_2^{-1}$

$$\phi_1^{\downarrow s} \otimes^* \psi_2 \otimes^* \psi_1^{-1} \otimes^* \psi_2^{-1} \;=\; \phi_2^{\downarrow s} \otimes^* \psi_1 \otimes^* \psi_1^{-1} \otimes^* \psi_2^{-1},$$

hence

$$\phi_1^{\downarrow s} \otimes^* \psi_1^{-1} \otimes^* f_{\gamma(\psi_2)} \;=\; \phi_2^{\downarrow s} \otimes^* \psi_2^{-1} \otimes^* f_{\gamma(\psi_1)}$$

and due to (D.12) and Lemma D.10 we finally obtain

$$\eta_1^{\downarrow * s} \;=\; \phi_1^{\downarrow s} \otimes^* \psi_1^{-1} \;=\; \phi_2^{\downarrow s} \otimes^* \psi_2^{-1} \;=\; \eta_2^{\downarrow * s}.$$

This proves that projection in $\langle \Phi^*, D \rangle$ is well-defined.

It remains to be verified that $\downarrow^*$ is really an extension of $\downarrow$. Let $\phi \in \Phi$. In the first step we have to guarantee that $\phi^{\downarrow * s}$ is defined for all $s \subseteq d(\phi)$. The valuation ϕ can be written as

$$\phi \quad = \quad (\phi \otimes \phi^{\downarrow \emptyset}) \otimes^* (\phi^{\downarrow \emptyset})^{-1},$$

since

$$\phi^{\downarrow \emptyset} \otimes^* (\phi^{\downarrow \emptyset})^{-1} \quad = \quad f_{\phi^{\downarrow \emptyset}}$$

and $\gamma(\phi) \geq \gamma(\phi^{\downarrow\emptyset})$. According to Lemma D.10 we have $\gamma(\phi^{\downarrow\emptyset}) \leq \gamma(\phi \otimes \phi^{\downarrow\emptyset})$ and $d(\phi^{\downarrow\emptyset}) = \emptyset \subseteq d(\phi \otimes \phi^{\downarrow\emptyset})$. Therefore, the projection $\downarrow^*$ of ϕ to any domain $s \subseteq d(\phi \otimes \phi^{\downarrow\emptyset}) = d(\phi)$ is defined. Now, using the combination axiom in $\langle \Phi, D \rangle$,

$$\begin{aligned} \phi^{\downarrow * s} &= (\phi \otimes \phi^{\downarrow\emptyset})^{\downarrow s} \otimes^* (\phi^{\downarrow\emptyset})^{-1} \\ &= (\phi^{\downarrow s} \otimes \phi^{\downarrow\emptyset}) \otimes^* (\phi^{\downarrow\emptyset})^{-1} \\ &= \phi^{\downarrow s}. \end{aligned}$$

Note that the extended domain operator $\mathcal{M}^*$ is directly induced by the definition of the new projection operator, i.e. for $\eta \in \Phi^*$

$$\begin{aligned} \mathcal{M}^*(\eta) \quad = \quad & \{s : \exists \phi, \psi \in \Phi \text{ such that } d(\psi) \subseteq s \subseteq d(\phi), \\ & \gamma(\psi) \leq \gamma(\phi) \text{ and } \eta = \phi \otimes^* \psi^{-1}\}. \end{aligned} \tag{D.14}$$

Theorem D.11 *$\langle \Phi^*, D \rangle$ with the operations of labeling d^*, combination $\otimes^*$, domain $\mathcal{M}^*$ and projection $\downarrow^*$ is a valuation algebra with partial projection.*

Proof: We verify the axioms on $\langle \Phi^*, D \rangle$ given in Appendix A.3:

(A1$''$) *Commutative Semigroup:* We have already seen that Φ^* is a commutative semigroup under $\otimes^*$.

(A2$''$) *Labeling:* Consider two elements $\eta_1, \eta_2 \in \Phi^*$ with $\eta_1 \in \gamma(\phi)$ and $\eta_2 \in \gamma(\psi)$ for $\phi, \psi \in \Phi$. It follows from the definition of the labeling operator that $d^*(\eta_1) = d(\phi)$ and $d^*(\eta_2) = d(\psi)$. We conclude from equation (D.4) that $\eta_1 \otimes^* \eta_2 \in \gamma(\phi \otimes \psi)$. Indeed, η_1 and η_2 can be written as pairs of elements of $[\phi]_\gamma$ and $[\psi]_\gamma$ respectively and therefore $\eta_1 \otimes^* \eta_2$ as pairs of elements of $[\phi]_\gamma \otimes [\psi]_\gamma = [\phi \otimes \psi]_\gamma$. Then, it follows that

$$d^*(\eta_1 \otimes^* \eta_2) \;=\; d(\phi \otimes \psi) \;=\; d(\phi) \cup d(\psi) \;=\; d^*(\eta_1) \cup d^*(\eta_2).$$

(A3$''$) *Projection:* If $\eta \in \Phi^*$ can be projected to s, we have

$$\eta^{\downarrow * s} \;=\; \phi^{\downarrow s} \otimes^* \psi^{-1}$$

with $d(\psi) \subseteq s$. It follows from the labeling axiom

$$d^*(\eta^{\downarrow * s}) \;=\; d^*(\phi^{\downarrow s} \otimes^* \psi^{-1}) \;=\; d(\phi^{\downarrow s}) \cup d^*(\psi^{-1}) \;=\; d(\phi^{\downarrow s}) \;=\; s.$$

(A4$''$) *Transitivity:* Let $\eta \in \Phi^*$, $t \subseteq s \subseteq d(\eta)$ such that $t \in \mathcal{M}^*(\eta)$. We have

$$\eta \;=\; \phi \otimes^* \psi^{-1}$$

with $\phi, \psi \in \Phi$, $d(\psi) \subseteq t$ and $\gamma(\psi) \leq \gamma(\phi)$. Since $d(\psi) \subseteq t \subseteq s \subseteq d(\phi)$ it follows $s \in \mathcal{M}^*(\eta)$. The projection of η to the domain s yields

$$\eta^{\downarrow * s} \;=\; \phi^{\downarrow s} \otimes^* \psi^{-1}.$$

From Lemma D.10 and $\gamma(\psi) \leq \gamma(\phi)$ we derive $\gamma(\phi) = \gamma(\phi \otimes \psi)$ and obtain by application of the combination axiom in $\langle \Phi, D \rangle$:

$$\gamma(\phi^{\downarrow s}) \;=\; \gamma((\phi \otimes \psi)^{\downarrow s}) \;=\; \gamma(\phi^{\downarrow s} \otimes \psi).$$

Therefore $\gamma(\psi) \leq \gamma(\phi^{\downarrow s})$ holds. Since $d(\psi) \subseteq t$ we have shown that $t \in \mathcal{M}^*(\eta^{\downarrow * s})$. By the transitivity axiom in $\langle \Phi, D \rangle$ we finally get

$$(\eta^{\downarrow * s})^{\downarrow t} \;=\; (\phi^{\downarrow s})^{\downarrow t} \otimes^* \psi^{-1} \;=\; \phi^{\downarrow t} \otimes^* \psi^{-1} \;=\; \eta^{\downarrow * t}.$$

(A5$''$) *Combination:* Let $\eta_1, \eta_2 \in \Phi^*$ with $s = d(\eta_1)$, $t = d(\eta_2)$ and $s \subseteq z \subseteq s \cup t$. We claim that $z \cap t \in \mathcal{M}^*(\eta_2)$ implies $z \in \mathcal{M}^*(\eta_1 \otimes^* \eta_2)$. Assume that $z \cap t \in \mathcal{M}^*(\eta_2)$, i.e.

$$\eta_2 \;=\; \phi_2 \otimes^* \psi_2^{-1}$$

with $\phi_2, \psi_2 \in \Phi$, $d(\psi_2) \subseteq z \cap t$ and $\gamma(\psi_2) \leq \gamma(\phi_2)$. By Lemma D.11 there are $\phi_1, \psi_1 \in \Phi$ with $\gamma(\phi_1) = \gamma(\psi_1)$ and $d(\phi_1) = d(\psi_1) = s$ such that

$$\eta_1 \;=\; \phi_1 \otimes^* \psi_1^{-1}.$$

We then get

$$\eta_1 \otimes^* \eta_2 \;=\; (\phi_1 \otimes \phi_2) \otimes^* (\psi_1 \otimes \psi_2)^{-1}$$

with $d(\psi_1 \otimes \psi_2) = d(\psi_1) \cup d(\psi_2) \subseteq z$. Further

$$\gamma(\phi_1 \otimes \phi_2) \;=\; \gamma(\phi_1 \otimes \psi_1 \otimes \phi_2 \otimes \psi_2)$$

and it follows $\gamma(\psi_1 \otimes \psi_2) \leq \gamma(\phi_1 \otimes \phi_2)$. So $z \in \mathcal{M}^*(\eta_1 \otimes^* \eta_2)$. We finally get by the combination axiom in $\langle \Phi, D \rangle$

$$\begin{aligned} \eta_1 \otimes^* \eta_2^{\downarrow z \cap t} &= (\phi_1 \otimes \phi_2^{\downarrow z \cap t}) \otimes^* \psi_1^{-1} \otimes^* \psi_2^{-1} \\ &= (\phi_1 \otimes \phi_2)^{\downarrow z} \otimes^* (\psi_1 \otimes \psi_2)^{-1} \\ &= (\eta_1 \otimes \eta_2)^{\downarrow * z}. \end{aligned}$$

This holds because $d(\phi_1 \otimes \phi_2) = s \cup t$ and $\gamma(\psi_1 \otimes \psi_2) \leq \gamma(\phi_1 \otimes \phi_2)$.

(A6$''$) *Domain:* Let $\eta \in \Phi^*$ with $t = d^*(\eta)$. By Lemma D.11, $t \in \mathcal{M}^*(\eta)$, that is $\eta = \phi \otimes^* \psi^{-1}$ with $d(\psi) \subseteq t$ and $\gamma(\psi) \leq \gamma(\phi)$. Then, as a consequence of the domain axiom in $\langle \Phi, D \rangle$, we get

$$\eta^{\downarrow * t} \;=\; \phi^{\downarrow t} \otimes^* \psi^{-1} \;=\; \phi \otimes^* \psi^{-1} \;=\; \eta.$$

■

This shows that the property of separativity is sufficient to provide a valuation algebra with a division operation. Again, we point out that we loose the property of full projection by this construction. Therefore, we end with a valuation algebra with partial projection as stated by the above theorem. Before listing some instances of separative valuation algebras, we remark that cases exist where we do not need to worry about the existence of the projections in the combination axiom.

Lemma D.12 *Let $\eta \in \Phi^*$ and $\psi \in \Phi$ with $s = d^*(\eta)$ and $t = d(\psi)$. For every domain $z \in D$ such that $s \subseteq z \subseteq s \cup t$ we have $z \in \mathcal{M}(\eta \otimes \psi)$.*

Proof: Since ψ can be projected to any domain contained in $d(\psi)$, especially to $z \cap t$, it follows from the combination axiom that $z \in \mathcal{M}(\eta \otimes \psi)$. ■

It is shown in (Pouly, 2008) that an identity element e can also be adjoined to a separative valuation algebra without affecting separativity. It always holds that $e = e^{-1}$ that allows us to derive division-based local computation architectures from the Shenoy-Shafer architecture of Section 4.1 that uses this particular element.

D.6 Set Potentials and Separativity

In order to prove that set potentials are separative, we first introduce some alternative representations. We define for a set potential m with domain $d(m) = s$ and $A \subseteq \Omega_s$

$$b_m(A) = \sum_{B \subseteq A} m(B) \quad \text{and} \quad q_m(A) = \sum_{B \supseteq A} m(B). \tag{D.15}$$

Clearly, both functions b_m and q_m are themselves set potentials. It is furthermore shown in (Shafer, 1976) that the two transformation rules are one-to-one with the following inverse transformations

$$m(A) = \sum_{B \subseteq A} (-1)^{|A-B|} \, b_m(B) \tag{D.16}$$

and

$$m(A) = \sum_{B \supseteq A} (-1)^{|B-A|} \, q_m(B). \tag{D.17}$$

Consequently, the operations of combination and projection can be carried over from m-functions to b-functions and q-functions:

$$\begin{aligned} b_{m_1} \otimes b_{m_2} &= b_{m_1 \otimes m_2}, & b_m^{\downarrow t} &= b_{m^{\downarrow t}}, \\ q_{m_1} \otimes q_{m_2} &= q_{m_1 \otimes m_2}, & q_m^{\downarrow t} &= q_{m^{\downarrow t}}. \end{aligned}$$

To sum it up, m-functions, b-functions and q-functions describe the same system, and since m-functions build a valuation algebra, the same holds for the two other representations. Additionally, we also deduce that all properties that hold for one of these representations naturally hold for the two others.

It turns out that the operations of combination and projection simplify considerably if they are expressed either in the system of q-functions or b-functions. More concretely, combination reduces to simple multiplication in the system of q-functions and projection does not need any computation at all in the system of b-functions. This is the statement of the following theorem which is for example proved in (Kohlas, 2003).

Theorem D.12

1. *For q_{m_1} with domain s and q_{m_2} with domain t we have for all $A \subseteq \Omega_{s \cup t}$*

$$q_{m_1} \otimes q_{m_2}(A) \quad = \quad q_{m_1}(A^{\downarrow s}) \cdot q_{m_2}(A^{\downarrow t}). \tag{D.18}$$

2. *For b_m with domain s and $t \subseteq s$ we have for all $A \subseteq \Omega_t$*

$$b_m^{\downarrow t}(A) \quad = \quad b_m(A^{\uparrow s}). \tag{D.19}$$

Now, we are able to show that set potentials are separative. For this purpose, we change into the system of q-functions. Then, a congruence γ is needed that divides Φ into cancellative equivalence classes. Let us therefore introduce the *support* of a q-function $q \in \Phi$ with $d(q) = s$ by

$$supp(q) \quad = \quad \{A \subseteq \Omega_s : q(A) > 0\}.$$

q_1 and q_2 with equal domain are equivalent if they have the same support, i.e.

$$q_1 \equiv q_2 \pmod{\gamma} \quad \text{if} \quad d(q_1) = d(q_2) \text{ and } supp(q_1) = supp(q_2).$$

In particular, we have $q \equiv q \otimes q^{\downarrow t} \pmod{\gamma}$ since

$$\begin{aligned} supp(q \otimes q^{\downarrow t}) &= \{A \subseteq \Omega_s : q(A) \cdot q^{\downarrow t}(A^{\downarrow t}) > 0\} \\ &= \{A \subseteq \Omega_s : q(A) > 0\} \\ &= supp(q). \end{aligned}$$

The second equality holds because m-functions are non-negative, projection of m-functions is defined by a sum, and q-functions are defined by a sum over the values of an m-function. Altogether, we have $q^{\downarrow t}(A^{\downarrow t}) = 0$ implies $q(A) = 0$ because the non-negative terms of a sum giving zero must themselves be zero. Hence, Φ is divided into disjoint equivalence classes of q-functions with equal domain and support. These classes are cancellative since

$$q(A) \cdot q_1(A) \quad = \quad q(A) \cdot q_2(A)$$

implies that $q_1(A) = q_2(A)$ if q, q_1, q_2 all have the same domain and support. This fulfills all requirements for separativity given in Definition D.4.

It should be mentioned that not every b-function or q-function transforms into a set potential m using either equation (D.16) or (D.17). This is because both transformations can create negative values which shows that Φ^* is really an extension of the system of q-functions.

D.7 Density Functions and Separativity

In order to show that density functions are separative, we proceed similarly to the foregoing example and define the *support* of a density $f \in \Phi_s$ by

$$supp(f) \quad = \quad \{\mathbf{x} \in \Omega_s : f(\mathbf{x}) > 0\}.$$

Then, we again say that two densities f and g with equal domain are equivalent if they have the same support. We conclude $f \equiv f \otimes f^{\downarrow t} \pmod{\gamma}$ since

$$\begin{aligned} supp(f \otimes f^{\downarrow t}) &= \{\mathbf{x} \in \Omega_s : f(\mathbf{x}) \cdot f^{\downarrow t}(\mathbf{x}^{\downarrow t}) > 0\} \\ &= \{\mathbf{x} \in \Omega_s : f(\mathbf{x}) > 0\} \\ &= supp(f). \end{aligned}$$

The second equality holds because densities are non-negative and therefore $f^{\downarrow t}(\mathbf{x}^{\downarrow t}) = 0$ implies $f(\mathbf{x}) = 0$. Hence, Φ divides into disjoint equivalence classes of densities with equal domain and support. These classes are cancellative which fulfills all requirements for separativity given in Definition D.4.

D.1.2 Regular Valuation Algebras

We have just seen that a separative valuation algebra decomposes into disjoint semigroups which in turn can be embedded into groups. Thus, we obtain an extended valuation algebra where every element has an inverse. However, (Kohlas, 2003) remarked that in some cases, valuation algebras can directly be decomposed into groups instead of only semigroups. This makes life much easier since we can avoid the rather complicated embedding and, moreover, full projection is conserved. A sufficient condition for this simplification is the property of *regularity* stated in the following definition. Note also that it again extends the notion of regularity from semigroup theory to incorporate the operation of projection.

Definition D.5

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

$$\phi \quad = \quad \phi^{\downarrow t} \otimes \chi \otimes \phi. \tag{D.20}$$

- *A valuation algebra $\langle \Phi, D \rangle$ is called* regular, *if all its elements are regular.*

In contrast to the more general case of separativity, we are now looking for a congruence that decomposes Φ directly into a union of groups. For this purpose, we introduce the *Green relation* (Green, 1951) between valuations:

$$\phi \equiv \psi \pmod{\gamma} \quad \text{if} \quad \phi \otimes \Phi \;=\; \psi \otimes \Phi. \tag{D.21}$$

$\phi \otimes \Phi$ denotes the set of valuations $\{\phi \otimes \eta : \eta \in \Phi\}$, i.e. the *principal ideal* generated by ϕ. (Kohlas, 2003) proves that the Green relation is actually a congruence in a regular valuation algebra and that the corresponding equivalence classes $[\phi]_\gamma$ are directly groups which therefore provide inverse elements.

Lemma D.13 *If ϕ is regular with $\phi = \phi \otimes \chi \otimes \phi$, then ϕ and $\chi \otimes \phi \otimes \chi$ are inverses.*

Proof: From Definition 4.1 we obtain

$$\begin{aligned}\phi \otimes (\chi \otimes \phi \otimes \chi) \otimes \phi &= \phi \otimes \chi \otimes (\phi \otimes \chi \otimes \phi)\\ &= \phi \otimes \chi \otimes \phi\\ &= \phi\end{aligned}$$

and

$$\begin{aligned}(\chi \otimes \phi \otimes \chi) \otimes \phi \otimes (\chi \otimes \phi \otimes \chi) &= \chi \otimes (\phi \otimes \chi \otimes \phi) \otimes \chi \otimes \phi \otimes \chi\\ &= \chi \otimes (\phi \otimes \chi \otimes \phi) \otimes \chi\\ &= \chi \otimes \phi \otimes \chi.\end{aligned}$$

■

The equivalence classes $[\phi]_\gamma$ are clearly cancellative since they are groups and therefore contain inverse elements. Further, the Green relation clearly satisfies equation (D.2), which makes a regular valuation algebra also separative. From this point of view, regular valuation algebras are special cases of separative valuation algebras and since Φ does not need to be extended, full projection is preserved. This construction is illustrated in Figure D.2.

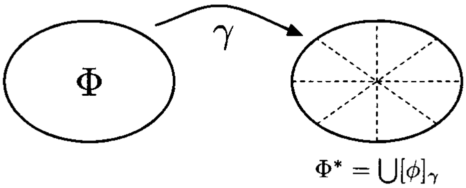


Figure D.2 A regular valuation algebra Φ decomposes directly into a union of groups Φ^* by the Green relation.

Idempotent elements play an important role in regular valuation algebras. These are elements $f \in \Phi$ such that $f \otimes f = f$. According to Definition 4.1 we may therefore say that idempotent elements are inverse to themselves. Essentially, idempotents are obtained by combining inverses. Thus, if $\phi, \psi \in \Phi$ are inverses, $f = \phi \otimes \psi$ is idempotent since

$$f \otimes f = (\phi \otimes \psi) \otimes (\phi \otimes \psi) = (\phi \otimes \psi \otimes \phi) \otimes \psi = \phi \otimes \psi = f.$$

These idempotents behave neutral with respect to ϕ and ψ. We have

$$f \otimes \phi \;=\; (\phi \otimes \psi) \otimes \phi \;=\; \phi,$$

and the same holds equally for ψ. The following important lemma states that every principal ideal $\phi \otimes \Phi$ is generated by a unique idempotent valuation.

Lemma D.14 *In a regular valuation algebra there exists for all $\phi \in \Phi$ a unique idempotent $f \in \Phi$ with $\phi \otimes \Phi = f \otimes \Phi$.*

Proof: Since ϕ is regular there is a χ such that $\phi = \phi \otimes \chi \otimes \phi$ and we know from Lemma D.13 that ϕ and $\chi \otimes \phi \otimes \chi$ are inverses. Thus, $f = \phi \otimes (\chi \otimes \phi \otimes \chi) = \phi \otimes \chi$ is an idempotent such that $f \otimes \phi = \phi$. Therefore, $\phi \otimes \psi = f \otimes (\phi \otimes \psi) \in f \otimes \Phi$ and $f \otimes \psi = \phi \otimes (\chi \otimes \psi) \in \phi \otimes \Phi$. Consequently, $\phi \otimes \Phi = f \otimes \Phi$ must hold, and it remains to prove that f is unique. Suppose that $f_1 \otimes \Phi = f_2 \otimes \Phi$. Then, there exists $\chi \in \Phi$ such that $f_2 = f_1 \otimes \chi$. This implies that $f_1 \otimes f_2 = f_1 \otimes (f_1 \otimes \chi) = f_1 \otimes \chi = f_2$. Similarly, we derive $f_1 \otimes f_2 = f_2$ and therefore it follows that $f_1 = f_2$. ■

We again point out that we may also adjoin an identity element to a regular valuation algebra without loosing this property. An explicit proof can be found in (Pouly, 2008). Here, we now focus on an instance of a regular valuation algebra. More examples will be given in Appendix E.2 of Chapter 5.

■ D.8 Arithmetic Potentials and Regularity

In Instance 4.1 we used arithmetic potentials to give a simple example of inverse elements in a valuation algebra. There, we also mentioned that arithmetic potentials are in fact regular which induces these inverse elements. Now, we are able to verify this claim. Let p be an arithmetic potential with domain s and $t \subseteq s$. Then, the definition of regularity may be written as:

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

So, defining

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

leads naturally to a solution of this equation. Hence, the inverse of a potential p with domain s and $\mathbf{x} \in \Omega_s$ is given by

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

D.1.3 Idempotent Valuation Algebras

The property of regularity allows the decomposition of a valuation algebra into groups such that inverses exist within Φ directly. We also learned that every such group is generated from a unique idempotent element. Therefore, the last simplifying condition for the introduction of division identifies valuation algebras where every element is idempotent. This has been proposed in (Kohlas, 2003) and leads to a decomposition where every valuation forms its own group. The property of idempotency has already been defined in Definition 4.2. By choosing $\chi = \phi$ in Definition D.5, we directly remark that every idempotent valuation algebra is regular too. Then, since principal ideals are spanned by a unique idempotent, each element of an idempotent valuation algebra generates its own principal ideal. Consequently, all groups $[\phi]_\gamma$ consist of the single element ϕ which is therefore also the inverse of itself.

It is not surprising that also the property of idempotency remains conserved if an identity element is adjoined to an idempotent valuation algebra. We again refer to (Pouly, 2008) for the proof of this statement. Some examples of idempotent valuation algebras have already been given in Section 4.2.1, others follow in Appendix E.4.

D.2 PROOFS OF DIVISION-BASED ARCHITECTURES

The proofs of the two division-based local computation schemes called Lauritzen-Spiegelhalter architecture and HUGIN architectures were retained in Chapter 4 since they are based on properties derived in Appendix D.1.

D.2.1 Proof of the Lauritzen-Spiegelhalter Architecture

The proof of the Lauritzen-Spiegelhalter architecture is based on the messages used in the Shenoy-Shafer architecture. If they exist, Lauritzen-Spiegelhalter gives the correct results because the inward messages are identical in both architectures. Further, the Shenoy-Shafer messages are needed in equation (D.23) below for the application of the combination axiom. However, (Schneuwly, 2007) weakens this requirement by proving that the existence of the inward messages is in fact sufficient to make the Shenoy-Shafer architecture and therefore also Lauritzen-Spiegelhalter work.

Proof of Theorem 4.2

Proof: Let μ' denote the messages sent during an execution of the Shenoy-Shafer architecture. We know that

$$\mu_{i\to j} \quad = \quad \mu'_{i\to j}$$

during the collect propagation and the theorem holds for the root node as a result of the collect algorithm. We prove that it is correct for all nodes by induction over the outward propagation phase using the correctness of Shenoy-Shafer. For this purpose, we schedule the distribute phase by taking the reverse node numbering. When a node j is ready to send a message towards i, node i stores

$$\psi_i \otimes (\mu'_{i\to j})^{-1} \otimes \bigotimes_{k\in ne(i),k\neq j} \mu'_{k\to i}. \tag{D.22}$$

By the induction hypothesis, the incoming message at step i is

$$\begin{aligned}
\mu_{j\to i} &= \phi^{\downarrow\lambda(j)\cap\lambda(i)} \\
&= \left(\psi_j \otimes \bigotimes_{k\in ne(j)} \mu'_{k\to j}\right)^{\downarrow\lambda(j)\cap\lambda(i)} \\
&= \left(\psi_j \otimes \bigotimes_{k\in ne(j),k\neq i} \mu'_{k\to j}\right)^{\downarrow\omega_{j\to i}\cap\lambda(i)} \otimes \mu'_{i\to j} \\
&= \mu'_{j\to i} \otimes \mu'_{i\to j} = \mu'_{j\to i} \otimes \mu_{i\to j},
\end{aligned} \tag{D.23}$$

The third equality follows from the combination axiom, and the assumption is needed to ensure that $\mu'_{j\to i}$ exists. So we obtain at node i, when the incoming message $\mu_{j\to i}$ is combined with the actual content and using Theorem 4.1,

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

It follows from Lemma D.10 that

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

hence

$$\phi^{\downarrow\lambda(i)} \otimes f_{\gamma(\mu'_{i\to j})} = \phi^{\downarrow\lambda(i)}.$$

■

D.2.2 Proof of the HUGIN Architecture

Proof of Theorem 4.3

Proof: The proof is based on the correctness of the Lauritzen-Spiegelhalter architecture. First we consider the separators in the join tree (V, E, λ, D) as real nodes. Thus, let (V', E', λ', D) be such a modified join tree. We then adapt the assignment mapping a making it again surjective. We simply assign to every separator node in $V' - V$ the identity element e and name the extended assignment mapping a'. Now, the execution of the Lauritzen-Spiegelhalter architecture is started using $(V', E', \lambda', D, a')$.

Take a node $i \in V$ which is not a separator in the original tree. It sends a message $\mu_{i \rightarrow j}$ in the collect phase of Lauritzen-Spiegelhalter and divides it out of its current content which we abbreviate with η_i. By the construction of (V', E', λ', D), the receiving node $j = ch(i)$ is a separator in (V, E, λ, D), that is $j \in (V' - V)$. Node i contains $\eta_i \otimes (\mu_{i \rightarrow j})^{-1}$ and j stores $e \otimes \mu_{i \rightarrow j} = \mu_{i \rightarrow j}$ after this step. Then node j is ready to send a message towards node $k = ch(j)$. But we clearly have $\mu_{i \rightarrow j} = \mu_{j \rightarrow k}$. Since every emitted message is divided out of the store, the content of node j becomes

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

We continue with Lauritzen-Spiegelhalter and assume that node k is ready to send the message for node j during the distribute phase. This message equals $\phi^{\downarrow \lambda(k) \cap \lambda(j)}$ due to Theorem 4.2 and also becomes the new content of j according to equation (D.23). The message sent from j towards i is finally $\phi^{\downarrow \lambda(j) \cap \lambda(i)} = \phi^{\downarrow \lambda(k) \cap \lambda(i)}$ so that we get there

$$\eta_i \otimes (\mu_{i \rightarrow j})^{-1} \otimes \phi^{\downarrow \lambda(i) \cap \lambda(k)} \quad = \quad \phi^{\downarrow \lambda(i)}.$$

This follows again from the correctness of Lauritzen-Spiegelhalter. But

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

is also the message from k towards i in the HUGIN architecture using (V, E, λ, D, a), which has passed already through the separator j. ■

The messages used throughout the proof correspond to the Lauritzen-Spiegelhalter messages. Therefore, we may conclude that the existence of the collect messages is also a sufficient condition for Theorem 4.3. If, on the other hand, the valuation algebra is regular, this condition can again be dropped.

D.3 PROOF FOR SCALING IN VALUATION ALGEBRAS

Proof of Lemma 4.9

Proof:

1. By application of the combination axiom

$$(\phi^{\downarrow})^{\downarrow \emptyset} \;=\; \left(\phi \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1}\right)^{\downarrow \emptyset} \;=\; \phi^{\downarrow \emptyset} \otimes \left(\phi^{\downarrow \emptyset}\right)^{-1} \;=\; f_{\gamma(\phi^{\downarrow \emptyset})}.$$

Hence

$$(\phi^{\downarrow})^{\downarrow} \;=\; \phi^{\downarrow} \otimes \left((\phi^{\downarrow})^{\downarrow \emptyset}\right)^{-1} \;=\; \phi^{\downarrow} \otimes \left(f_{\gamma(\phi^{\downarrow \emptyset})}\right)^{-1} \;=\; \phi^{\downarrow}$$

since $\gamma(\phi^{\downarrow}) = \gamma(\phi) \geq \gamma(\phi^{\downarrow \emptyset})$.

2. We first remark that $\phi = \phi^{\downarrow} \otimes \phi^{\downarrow\emptyset}$ since $\gamma(\phi) \geq \gamma(\phi^{\downarrow\emptyset})$ and using the combination axiom and equation (4.30) we have

$$\begin{aligned}
\phi \otimes \psi &= (\phi^{\downarrow} \otimes \phi^{\downarrow\emptyset}) \otimes (\psi^{\downarrow} \otimes \psi^{\downarrow\emptyset}) \\
&= (\phi^{\downarrow} \otimes \psi^{\downarrow}) \otimes (\phi^{\downarrow\emptyset} \otimes \psi^{\downarrow\emptyset}) \\
&= (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow} \otimes (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow\emptyset} \otimes (\phi^{\downarrow\emptyset} \otimes \psi^{\downarrow\emptyset}) \\
&= (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow} \otimes ((\phi^{\downarrow\emptyset} \otimes \psi^{\downarrow\emptyset}) \otimes (\phi^{\downarrow} \otimes \psi^{\downarrow}))^{\downarrow\emptyset} \\
&= (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow} \otimes ((\phi^{\downarrow} \otimes \phi^{\downarrow\emptyset}) \otimes (\psi^{\downarrow} \otimes \psi^{\downarrow\emptyset}))^{\downarrow\emptyset} \\
&= (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow} \otimes (\phi \otimes \psi)^{\downarrow\emptyset}.
\end{aligned}$$

From this we conclude that

$$\begin{aligned}
(\phi \otimes \psi)^{\downarrow} &= (\phi \otimes \psi) \otimes \left((\phi \otimes \psi)^{\downarrow\emptyset}\right)^{-1} \\
&= (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow} \otimes (\phi \otimes \psi)^{\downarrow\emptyset} \otimes \left((\phi \otimes \psi)^{\downarrow\emptyset}\right)^{-1} \\
&= (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow} \otimes f_{\gamma((\phi \otimes \psi)^{\downarrow\emptyset})} \\
&= (\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow}
\end{aligned}$$

because again $\gamma((\phi^{\downarrow} \otimes \psi^{\downarrow})^{\downarrow}) \geq \gamma((\phi \otimes \psi)^{\downarrow\emptyset})$.

3. From equation (4.30) we conclude on the one hand

$$\phi^{\downarrow t} = \left(\phi^{\downarrow t}\right)^{\downarrow} \otimes \left(\phi^{\downarrow t}\right)^{\downarrow\emptyset} = \left(\phi^{\downarrow t}\right)^{\downarrow} \otimes \phi^{\downarrow\emptyset},$$

and on the other hand

$$\phi^{\downarrow t} = \left(\phi^{\downarrow} \otimes \phi^{\downarrow\emptyset}\right)^{\downarrow t} = \left(\phi^{\downarrow}\right)^{\downarrow t} \otimes \phi^{\downarrow\emptyset}.$$

Hence

$$\left(\phi^{\downarrow t}\right)^{\downarrow} \otimes \phi^{\downarrow\emptyset} = \left(\phi^{\downarrow}\right)^{\downarrow t} \otimes \phi^{\downarrow\emptyset}.$$

Since $\gamma(\phi^{\downarrow\emptyset}) \leq \gamma((\phi^{\downarrow})^{\downarrow t}), \gamma((\phi^{\downarrow t})^{\downarrow})$ it follows that

$$\left(\phi^{\downarrow t}\right)^{\downarrow} = \left(\phi^{\downarrow}\right)^{\downarrow t} \otimes \phi^{\downarrow\emptyset} \otimes \left(\phi^{\downarrow\emptyset}\right)^{-1} = \left(\phi^{\downarrow}\right)^{\downarrow t}.$$

■

D.9 Scaling of Belief Functions

In Instance D.6 we derive the two alternative representations of set potentials as b-functions and q-functions. These systems are isomorphic, which allows us to conclude that they all form valuation algebras and satisfy the same additional properties. Since combination corresponds to simple multiplication in the case of q-functions, it is often convenient to study related algebraic properties in this system. Thus, we obtain by applying the definition of scaling to a q-function:

$$q^{\downarrow} \quad = \quad q \otimes \left(q^{\downarrow\emptyset}\right)^{-1}.$$

It is important to note that $q^{\downarrow\emptyset}$ consists of two values, $q^{\downarrow\emptyset}(\emptyset)$ and $q^{\downarrow\emptyset}(\{\diamond\})$. Since the empty set can never be obtained from projecting a non-empty configuration set, the formula can be written for the case where $A \neq \emptyset$ and $q^{\downarrow\emptyset}(\{\diamond\}) \neq 0$ as

$$q^{\downarrow}(A) \quad = \quad \frac{q(A)}{q^{\downarrow\emptyset}(\{\diamond\})}. \tag{D.24}$$

We then obtain for the denominator

$$q^{\downarrow\emptyset}(\{\diamond\}) \quad = \quad \sum_{A\neq\emptyset} m(A) \quad = \quad m^{\downarrow\emptyset}(\{\diamond\}).$$

Since

$$\begin{aligned} q^{\downarrow\emptyset}(\emptyset) \quad &= \quad \sum_{A\subseteq\Omega_\emptyset} m^{\downarrow\emptyset}(A) \quad = \quad m^{\downarrow\emptyset}(\{\diamond\}) + m^{\downarrow\emptyset}(\emptyset) \\ &= \quad \sum_{A\neq\emptyset} m(A) + m(\emptyset) \quad = \quad \sum_{A} m(A) \quad = \quad q(\emptyset), \end{aligned}$$

we have for the empty set

$$q^{\downarrow}(\emptyset) \;=\; \frac{q(\emptyset)}{q^{\downarrow\emptyset}(\emptyset)} \;=\; \frac{q(\emptyset)}{q(\emptyset)} \;=\; 1. \tag{D.25}$$

So, the scale of a q-function is obtained by equation (D.24) and (D.25), and it is common to refer to non-negative, scaled q-functions as *commonality functions*.

We next translate the scaling operation for q-functions to the system of m-functions using equation (D.17) and obtain

$$\begin{aligned} m^{\downarrow}(A) \quad &= \quad \sum_{B\supseteq A} (-1)^{|B-A|}\, q^{\downarrow}(B) \qquad\qquad (D.26) \\ &= \quad \frac{\sum_{B\supseteq A}(-1)^{|B-A|}\, q(B)}{\sum_{B\neq\emptyset} m(B)} \;=\; \frac{m(A)}{\sum_{B\neq\emptyset} m(B)} \;=\; \frac{m(A)}{m^{\downarrow\emptyset}(\{\diamond\})}, \end{aligned}$$

if $A \neq \emptyset$ and $m^{\downarrow\emptyset}(\{\diamond\}) \neq 0$. Finally, we obtain for $m^{\downarrow}(\emptyset)$

$$\begin{aligned} m^{\downarrow}(\emptyset) &= \sum_{A}(-1)^{|A|}\, q^{\downarrow}(A) \\ &= \frac{\sum_{A\neq\emptyset}(-1)^{|A|}\, q(A)}{\sum_{A\neq\emptyset} m(A)} + 1 \\ &= \frac{\sum_{A}(-1)^{|A|}\, q(A) - q(\emptyset)}{\sum_{A\neq\emptyset} m(A)} + 1 \\ &= \frac{m(\emptyset)}{\sum_{A\neq\emptyset} m(A)} - \frac{m(\emptyset) + \sum_{A\neq\emptyset} m(A)}{\sum_{A\neq\emptyset} m(A)} + 1 \;=\; 0. \end{aligned}$$

Thus, the scale of a set potential is given by equation (D.26) if $A \neq \emptyset$ and $m^{\downarrow\emptyset}(\{\diamond\}) \neq 0$, and $m^{\downarrow}(\emptyset) = 0$ otherwise. This satisfies the two properties:

$$m^{\downarrow}(\emptyset) = 0 \quad \text{and} \quad \sum_{A\subseteq\Omega_s} m^{\downarrow}(A) = 1. \tag{D.27}$$

In the theory of evidence, such potentials are called *basic probability assignments* or *mass functions*. Intuitively, $m(A)$ represents the part of our belief that the actual world (i.e. some configuration) belongs to A – without supporting any more specific subset, by lack of adequate information (Smets, 2000; Smets & Kennes, 1994). Under this interpretation, the two normalization conditions imply that no belief is held in the empty configuration set (which corresponds to a closed world assumption) and that the total belief has measure 1. In a similar way, the operation of scaling can also be translated to the system of b-functions using equation (D.15). It is then common to refer to non-negative, scaled b-functions as *belief functions*. A comprehensive theory of mass, belief and commonality functions is contained in (Shafer, 1976). We also pointed out in Lemma 4.9 that the combination of two normalized set potentials (mass functions) does generally not lead to a mass function again. However, this can be achieved using equation (4.32). Assume two mass functions m_1 and m_2 with domains s and t. We have for $\emptyset \subset A \subseteq \Omega_{s\cup t}$

$$\begin{aligned} m_1 \oplus m_2(A) &= (m_1 \otimes m_2)^{\downarrow}(A) \\ &= \frac{m_1 \otimes m_2(A)}{(m_1 \otimes m_2)^{\downarrow\emptyset}(\{\diamond\})} \;=\; \frac{1}{K}\, m_1 \otimes m_2(A), \end{aligned}$$

where

$$\begin{aligned} K &= (m_1 \otimes m_2)^{\downarrow\emptyset}(\{\diamond\}) \\ &= \sum_{B\neq\emptyset} m_1 \otimes m_2(B) \;=\; \sum_{A_1^{\uparrow s\cup t}\cap A_2^{\uparrow s\cup t}\neq\emptyset} m_1(A_1)\cdot m_2(A_2). \end{aligned}$$

We further define $m_1 \oplus m_2(\emptyset) = 0$ and, if $K = 0$, we set $m_1 \oplus m_2(A) = 0$. This operation is called *Dempster's rule of combination* (Dempster, 1968).

We complete the study of scaling for set potentials and our excursion to *Dempster-Shafer theory* by a small example of normalizing a set potential. Let $r = \{A, B\}$ be a set of variables with finite frames $\Omega_A = \{a, \overline{a}\}$ and $\Omega_B = \{b, \overline{b}\}$, and m a set potentials with domain $d(m) = \{A, B\}$ defined as:

$$m \quad = \quad \begin{array}{|c||c|} \hline \emptyset & 0.6 \\ \{(a,b)\} & 0.1 \\ \{(\overline{a},b),(a,\overline{b})\} & 0.1 \\ \{(a,b),(\overline{a},\overline{b})\} & 0.2 \\ \hline \end{array}$$

We then compute

$$m^{\downarrow\emptyset} \quad = \quad \begin{array}{|c||c|} \hline \emptyset & 0.6 \\ \{\diamond\} & 0.4 \\ \hline \end{array}$$

and obtain for its associated mass function

$$m^{\downarrow} \quad = \quad \begin{array}{|c||c|} \hline \emptyset & 0 \\ \{(a,b)\} & 0.25 \\ \{(\overline{a},b),(a,\overline{b})\} & 0.25 \\ \{(a,b),(\overline{a},\overline{b})\} & 0.50 \\ \hline \end{array}$$

PROBLEM SETS AND EXERCISES

D.1 ★ Exercise B.4 in Chapter 2 asked to transform the fundamental computational problems of filtering, prediction and smoothing in hidden Markov chains into an inference problem. Show that the filtering and smoothing problem can simultaneously be solved by a complete run of the Shenoy-Shafer, Lauritzen-Spiegelhalter or HUGIN architecture, i.e. that filtering corresponds to the collect phase and smoothing to the distribute phase in the three local computation architectures.

D.2 ★ Section 4.5 derives the idempotent architecture from the correctness of the Lauritzen-Spiegelhalter architecture.

- **a)** Provide an alternative proof by starting from the HUGIN architecture. Indication: consider the separators in the HUGIN architecture as ordinary join tree nodes and observe that HUGIN becomes identical to Lauritzen-Spiegelhalter.
- **b)** Derive the idempotent architecture directly from the generalized collect algorithm in Section 3.10 without reference to another division-based architecture. This is similar to the proof of the Shenoy-Shafer architecture.

D.3 ★★ Consider a join tree (V, E, λ, D) with $|V| = r$ that is fully propagated by the Shenoy-Shafer architecture, i.e. for each node $i \in V$ we have

$$\phi^{\downarrow\lambda(i)} = (\psi_1 \otimes \cdots \otimes \psi_r)^{\downarrow\lambda(i)} = \psi_i \otimes \bigotimes_{j \in ne(i)} \mu_{j \to i}.$$

as stated in Theorem 4.1. Assume now that a new valuation $\eta \in \Phi$ is combined to some covering node $k \in V$ with $d(\eta) = \lambda(k)$. This is called a *consistent update*. If the Shenoy-Shafer architecture is repeated from scratch, we obtain at the end of the message-passing for each node $i \in V$

$$(\phi \otimes \eta)^{\downarrow\lambda(i)} = (\psi_1 \otimes \cdots \otimes \psi_r \otimes \eta)^{\downarrow\lambda(i)} = \psi_i' \otimes \bigotimes_{j \in ne(i)} \mu_{j \to i}'$$

where $\psi_i' = \psi_i$ for $i \neq k$ and $\psi_k' = \psi_k \otimes \eta$. However, instead of recomputing all messages $\mu_{i \to j}$, it is more appropriate to reuse the messages that do not change by this updating process. Prove that during the collect phase, only the messages between the node k and the root node change. In the distribute phase, all messages must be recomputed. Perform the same analysis for the Lauritzen-Spiegelhalter, HUGIN and idempotent architecture. The solution can be found in Chapter 6 of (Schneuwly, 2007).

D.4 ★ Let $\langle \Phi, D \rangle$ be an idempotent valuation algebra. For $\phi, \psi \in \Phi$ we define

$$\phi \geq \psi \quad \text{if, and only if,} \quad \phi \otimes \psi = \phi. \tag{D.28}$$

This expresses that the information piece ϕ is *more informative* than ψ.

a) Prove that this relation is a partial order, i.e. verify the axioms of Definition A.2 in the appendix of Chapter 1.

b) We further assume that $\langle \Phi, D \rangle$ is an information algebra, i.e. that neutral and null elements are present, see Definition 4.3. We then also have the operation of vacuous extension given in equation (3.24) of Chapter 3. Prove the following properties of relation (D.28) for $\phi, \psi \in \Phi$ and $x, y \in D$:

1. if $\phi \leq \psi$ then $d(\phi) \subseteq d(\psi)$;
2. if $x \subseteq y$ then $e_x \leq e_y$ and $z_x \leq z_y$;
3. if $x \subseteq d(\phi)$ then $e_x \leq \phi$ and, if $d(\phi) \subseteq x$, then $\phi \leq z_x$;
4. $\phi, \psi \leq \phi \otimes \psi$;
5. $\phi \otimes \psi = \sup\{\phi, \psi\}$;
6. if $x \subseteq d(\phi)$ then $\phi^{\downarrow x} \leq \phi$;
7. if $d(\phi) \subseteq y$ then $\phi \leq \phi^{\uparrow y}$;
8. if $x \subseteq y = d(\phi)$ then $(\phi^{\downarrow x})^{\uparrow y} \leq \phi$;

9. $\phi_1 \leq \phi_2$ and $\psi_1 \leq \psi_2$ imply $\phi_1 \otimes \psi_1 \leq \phi_2 \otimes \psi_2$;

10. if $x \subseteq d(\phi) \cap d(\psi)$ then $\phi^{\downarrow x} \otimes \psi^{\downarrow x} \leq (\phi \otimes \psi)^{\downarrow x}$;

11. if $x \subseteq d(\phi)$ then $\phi \leq \psi$ implies $\phi^{\downarrow x} \leq \psi^{\downarrow x}$;

12. if $d(\phi) \subseteq y$ then $\phi \leq \psi$ implies $\phi^{\uparrow y} \leq \psi^{\uparrow y}$;

13. if $d(\phi) \subseteq x \subseteq d(\psi)$ then $\phi \leq \psi$ implies $\phi \leq \psi^{\downarrow x}$.

The partial order together with Property 5 implies that Φ is a *semilattice*. Indications for this exercise can be found in Lemma 6.3 of (Kohlas, 2003).

D.5 ★★ Let $\langle \Phi, D\rangle$ be an information algebra and suppose that the information content of an element $\phi \in \Phi$ has been asserted. Then, all information pieces which are less informative than ϕ should also be true. The sets

$$I(\phi) \quad = \quad \{\psi \in \Phi : \phi \geq \psi\}. \tag{D.29}$$

are called *principal ideals*. Observe that principal ideals are closed under combination, i.e. if $\psi, \nu \in I(\phi)$ then $\psi \otimes \nu \in I(\phi)$. We further define

$$I_\Phi \quad = \quad \{I(\phi) : \phi \in \Phi\}$$

and introduce the following operations in $\langle I_\Phi, D\rangle$:

1. *Labeling*: For $I(\phi) \in I_\Phi$ we define $d(I(\phi)) = d(\phi)$.

2. *Combination*: For $I(\phi_1), I(\phi_2) \in I_\Phi$ we define

$$I_1 \otimes I_2 \quad = \quad \{\nu \in \Phi : \phi_1 \otimes \phi_2 \geq \nu\}$$

3. *Projection*: For $I(\phi) \in I_\Phi$ and $x \subseteq d(I(\phi))$ we define

$$I^{\downarrow x} \quad = \quad \{\psi \in \Phi : \phi^{\downarrow x} \geq \psi\}.$$

Prove that the set of principal ideals I_Φ is closed under combination and projection, and show that $\langle I_\Phi, D\rangle$ satisfies the axioms of an information algebra. Indications can be found in (Kohlas, 2003), Section 6.2.

D.6 ★★ Let $s, t, u \in D$ be disjoint sets of variables and assume that we want to define a valuation $\phi \in \Phi$ with $d(\phi) = s \cup t \cup u$ by specifying its projections ψ_1 and ψ_2 relative to $s \cup u$ and $t \cup u$. Of course, the projections must be consistent such that $\psi_1^{\downarrow u} = \psi_2^{\downarrow u}$ holds. The problem of finding such a valuation $\phi = \psi_1 \otimes \psi_2$ is called *marginal problem*. Intuitively, this is similar to the specification of a building by its floor, body and sheer plan. Prove that if $\langle \Phi, D\rangle$ is a regular valuation algebra according to Definition D.5, then there exists $\phi \in \Phi$ with $d(\phi) = s \cup t \cup u$ such that

$$\phi^{\downarrow s \cup u} = \psi_1 \quad \text{and} \quad \phi^{\downarrow t \cup u} = \psi_2.$$

The solution to this exercise can be found in (Kohlas, 2003), Theorem 5.15. Marginal problems are closely related to the notion of conditional independence in valuation algebras that is treated in Exercise D.7.

D.7 ★★ Let $\langle \Phi, D \rangle$ be a valuation algebra and $\phi \in \Phi$. If s, t, u are disjoint subsets of $d(\phi)$, we say that s is *conditionally independent* of t given u with respect to ϕ, if there exist $\psi_1, \psi_2 \in \Phi$ such that $d(\psi_1) = s \cup u, d(\psi_2) = t \cup u$ and

$$\phi^{\downarrow s \cup t \cup u} = \psi_1 \otimes \psi_2. \tag{D.30}$$

If s is conditionally independent of t given u, we write $s \bot_\phi t | u$.

a) Show that conditional independence in valuation algebras coincides with *stochastic conditional independence* in probability theory when applied to the instances of probability potentials, density functions and Gaussian potentials presented in Chapter 1.

b) Prove the following properties of conditional independence. Let $\phi \in \Phi$ and assume $s, t, u, v \subseteq d(\phi)$ to be disjoint sets of variables. Then

1. *Symmetry*: $s \bot_\phi t | u$ implies $t \bot_\phi s | u$. (G1)
2. *Decomposition*: $s \bot_\phi t \cup v | u$ implies $s \bot_\phi t | u$. (G2)

If furthermore the valuation algebra has neutral elements, we have

3. *Weak Union*: $s \bot_\phi t \cup v | u$ implies $s \bot_\phi t | u \cup v$. (G3)

The solution to these exercises can be found in Section 5.1 of (Kohlas, 2003).

D.8 ★★ Consider the definition of conditional independence in Exercise D.7. For $u = \emptyset$ we use the short notation $s \bot_\phi t$ and say that s is *independent* of t with respect to $\phi \in \Phi$. Let $\langle \Phi, D \rangle$ be a regular valuation algebra, $\phi \in \Phi$ and $s, t \in D$ disjoint subsets of $d(\phi)$. Then, the valuation

$$\phi_{s|t} = (\phi^{\downarrow t})^{-1} \otimes \phi^{\downarrow s \cup t} \tag{D.31}$$

is called the *conditional* of ϕ for s given t.

a) Prove that the factors $\psi_1 = \phi_{s|t}$ and $\psi_2 = \phi^{\downarrow t}$ satisfy equation (D.30) and identify $\phi_{s|t}$ in the valuation algebra or probability potentials.

b) Conditionals in regular valuation algebras have many properties that are well-known from conditional probabilities. Prove the following identities and rewrite them in the formalism of probability potentials:

1. If $\phi \in \Phi$ and $s, t \subseteq d(\phi)$ are disjoint, then

$$\phi_{s|t}^{\downarrow t} = f_{\gamma(\phi^{\downarrow t})}.$$

2. If $\phi \in \Phi$ and $s, t, u \subseteq d(\phi)$ are disjoint, then

$$\phi_{s \cup t | u} = \phi_{s | t \cup u} \otimes \phi_{t|u}.$$

3. If $\phi \in \Phi$, $s, t \subseteq d(\phi)$ are disjoint and $u \subseteq s$, then

$$\phi_{s|t}^{\downarrow t \cup u} = \phi_{u|t}.$$

4. If $\phi \in \Phi$ and $s, t, u \subseteq d(\phi)$ are disjoint, then

$$(\phi_{u|s \cup t} \otimes \phi_{s|t})^{\downarrow t \cup u} = \phi_{u|t}.$$

5. If $\phi, \psi \in \Phi$, $s, t \subseteq d(\phi)$ are disjoint, and $d(\psi) = t$, then

$$(\phi^{\downarrow s \cup t} \otimes \psi)_{s|t} = \phi_{s|t} \otimes f_{\gamma(\psi)}.$$

c) Prove that if the valuation algebra is regular, Property (G3) holds even if no neutral elements are present.

d) Prove that conditional independence in regular valuation algebras satisfies:

4. *Contraction*: $s \bot_\phi t | u$ and $s \bot_\phi v | t \cup u$ imply $s \bot_\phi t \cup v | u$. (G4)

The properties (G1) to (G4) may be considered as a system of axioms for an abstract calculus of conditional independence.

The solution to these exercises can be found in Section 5.1 of (Kohlas, 2003).

D.9 ★★ Exercise D.8 can be generalized to separative valuation algebras, see Appendix D.1.1. The same definition of conditionals applies in this case, but in contrast, the conditionals do generally not belong to Φ but only to the separative embedding Φ^*. Also, the properties listed in Exercise D.8.b still hold in the separative case, although we must pay attention to partial projection in the proofs. Identify the conditional $\phi_{s|t}$ in the valuation algebras of density functions, Gaussian potentials and set potentials. The solution to this exercises can be found in Section 5.3 of (Kohlas, 2003).

PART II

GENERIC CONSTRUCTIONS

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas
Copyright © 2011 John Wiley & Sons, Inc.

CHAPTER 5

SEMIRING VALUATION ALGEBRAS

The mathematical framework of valuation algebras introduced in Chapter 1 provides sufficient structure for the application of generic local computation methods. In the process, the numerous advantages of such a general framework became clear, and we have seen for ourselves that different formalisms are unified under this common perspective. As a fundamental requirement for this generality, we did not assume any further knowledge about the structure of valuations. They have only been considered as mathematical objects that possess a domain and that can further be combined and projected according to certain axioms. For certain applications, however, it is useful to have additional knowledge about the structure of valuations. This amounts to the identification of families of valuation algebra instances which share some common structure. A first example of such a family of valuation algebras grows out of the obvious similar structure of indicator functions and arithmetic potentials introduced as Instances 1.1 and 1.3. On the one hand, both instances are obtained by assigning values to tuples out of a finite set of tuples, but on the other hand, they differ in their definitions of combination and projection. Yet, there is a common ground between them. We will learn in this chapter that both examples belong to a family of instances, called *semiring valuation algebras*. A *commutative semiring* is by itself a fundamental algebraic structure that comprises a set of values and two operations

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

called addition and multiplication. Then, the values assigned to tuples are those of the semiring, and the operations of combination and projection can both be expressed using the two semiring operations. In other words, every commutative semiring induces a new valuation algebra, and the two examples of indicator functions and arithmetic potentials are both members of this large family. The reasons why we are interested in semiring valuation algebras are manifold. First and foremost, semiring instances are very large in number and, because each commutative semiring gives rise to a valuation algebra, we naturally obtain as many new valuation algebras as commutative semirings exist. Moreover, if we interpret valuation algebras as formalisms for knowledge representation, we are not even able to explain for some of these instances what kind of knowledge they model. Nevertheless, we have efficient algorithms for their processing thanks to the local computation framework. Second, we will see that a single verification proof of the valuation algebra axioms covers all formalisms that adopt this common structure, and we are also relieved from searching each formalism separately for neutral elements, null elements and the presence of a division operator. Finally, semiring valuations also stimulate new applications of local computation techniques that go beyond the pure computation of inference. Solution construction in constraint systems is an example of such an application that will be discussed in Chapter 8. Besides semiring valuation algebras, there are other families of formalisms derived in a similar way from other algebraic structures. This powerful technique is called *generic construction* and takes center stage in the second part of this book.

This chapter starts with a general introduction to semiring theory, accompanied by a large catalogue of instances to convince the reader of the richness of semiring examples. Section 5.2 then studies different classes of semirings that arise from the properties of a canonical order relation. Following (Kohlas & Wilson, 2008), we then show in Section 5.3 how semirings produce valuation algebras and we give an extensive member list of this family of valuation algebras in Section 5.4. Section 5.5 deals with the algebraic properties of semiring valuations and show which properties of the semiring guarantee the existence of neutral and null elements. The study of division is again postponed to the chapter appendix. A second family of valuation algebras, which is closely related to semiring valuations, is introduced in Section 5.7, and its algebraic properties are analyzed in Section 5.8.

5.1 SEMIRINGS

A semiring is an algebraic structure consisting of a set provided with two operations, called addition and multiplication, which satisfy the distributive law. In recent years, the importance of semirings for computational purposes has grown significantly and the distributive law is in fact the reason for this increasing popularity. From the simple formula $a \times (b + c) = a \times b + a \times c$, we can directly observe that the left-hand side is computationally more efficient since we perform only one multiplication. The distributive law is the cause of efficiency of local computation, since it induces the combination axiom. However, let us start with a short introduction to semiring theory.

Definition 5.1 *A tuple* $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *with binary operations* $+$ *and* $\times$ *is called* semiring *if the following properties hold:*

- $+$ *and* $\times$ *are both associative;* *(S1)*
- $+$ *is commutative;* *(S2)*
- *for* $a, b, c \in A$*:* $a \times (b + c) = a \times b + a \times c$*;* *(S3)*
- *for* $a, b, c \in A$*:* $(a + b) \times c = a \times c + b \times c$*;* *(S4)*
- $+$ *has a neutral element* $\mathbf{0}$*, i.e.* $a + \mathbf{0} = a$ *for all* $a \in A$*;* *(S5)*
- $\times$ *has a neutral element* $\mathbf{1}$*, i.e.* $a \times \mathbf{1} = \mathbf{1} \times a = a$ *for all* $a \in A$*;* *(S6)*
- $a \times \mathbf{0} = \mathbf{0} \times a = \mathbf{0}$ *for all* $a \in A$. *(S7)*

There are different definitions of semirings in the literature. This definition is taken from (Golan, 1999) and assumes the existence of two neutral elements. The neutral element $\mathbf{0} \in A$ with respect to addition is sometimes called *zero element*. It is a simple consequence of (S5) that a zero element is always unique. Moreover, if an algebraic structure provides all properties of a semiring except the presence of a zero element then the latter can always be adjoined artificially. The neutral element $\mathbf{1} \in A$ with respect to multiplication is called *unit element*. It is also unique but in contrast to the zero element, it can only be adjoined if the operation of addition is idempotent (see Definition 5.2 below). The corresponding constructions are shown in (Kohlas & Wilson, 2008).

Definition 5.2 *Let* $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *be a semiring:*

- *it is called* commutative *if* $a \times b = b \times a$ *for all* $a, b \in A$*;*
- *it is called* idempotent *if* $a + a = a$ *for all* $a \in A$*;*
- *it is called* positive *if* $a + b = \mathbf{0}$ *implies that* $a = b = \mathbf{0}$ *for all* $a, b \in A$*;*

Let us consider some examples of semirings:

Example 5.1 (Arithmetic Semirings) *Consider the set of non-negative real numbers* $\mathbb{R}_{\geq 0}$ *with* $+$ *and* $\times$ *designating the usual operations of addition and multiplication. This is clearly a positive, commutative semiring with the number* 0 *as zero element and the number* 1 *as unit element. In the same way, we could also take the fields of complex, real or rational numbers, or alternatively only non-negative integers or non-negative rationals. In the former three cases, the semiring would not be positive anymore, whereas ordinary addition and multiplication on non-negative integers and rationals yield again a positive semiring.*

Example 5.2 (Boolean Semiring) *Take the set* $\mathbb{B} = \{0, 1\}$ *with the intention that* 0 *designates the truth value "false" and* 1*"true". Addition is defined as* $a + b =$

$\max\{a,b\}$ *and represents the logical disjunction. Similarly, multiplication is defined as* $a \times b = \min\{a,b\}$ *which stands for the logical conjunction. This is a commutative, positive and idempotent semiring with zero element* 0 *and unit element* 1.

Example 5.3 (Bottleneck Semiring) *A generalization of the Boolean 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 remains commutative, positive and idempotent.*

Example 5.4 (Tropical Semiring) *An important semiring is defined over the set of non-negative integers* $\mathbb{N} \cup \{0, \infty\}$ *with* $a+b = \min\{a,b\}$ *and the usual integer addition* $+_{\mathbb{N}}$ *for* $\times$ *with the convention that* $a +_{\mathbb{N}} \infty = \infty$. *This semiring is commutative, positive and idempotent,* ∞ *is the zero element and the integer* 0 *is the unit element.*

Example 5.5 (Arctic Semiring) *The arctic semiring takes* max *for addition over the set of real numbers* $\mathbb{R} \cup \{-\infty\}$. *Multiplication is* $+_{\mathbb{R}}$ *with* $a +_{\mathbb{R}} (-\infty) = -\infty$. *This semiring is commutative, positive and idempotent, has* $-\infty$ *as zero element and* 0 *as unit element.*

Example 5.6 (Truncation Semiring) *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.* $a \times b = \min\{a +_{\mathbb{N}} b, k\}$. *This is a commutative, positive and idempotent semiring with k being the zero element and* 0 *the unit.*

Example 5.7 (Semiring of Formal Languages) *A* string *is a finite sequence of symbols from a countable alphabet* Σ *and a set of strings is called* language. *In particular, we refer to the language of all possible strings over the alphabet* Σ *as* Σ^*. *For* $A, B \subseteq \Sigma^*$ *we define* $A + B = A \cup B$ *and* $A \times B = \{ab \,|a \in A \text{ and } b \in B\}$. *This semiring of formal languages is idempotent with zero element* $\emptyset$ *and unit element* $\{\epsilon\}$. *It is also positive but not commutative.*

Example 5.8 (Triangular Norm Semiring) Triangular norms *(*t-norms*) were originally introduced in the context of probabilistic metric spaces (Menger, 1942; Schweizer & Sklar, 1960). They represent binary operations on the unit interval* $[0,1]$ *which are commutative, associative, 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* $T(a,0) = T(0,a) = 0$.

T-norms are used in fuzzy set theory and possibility theory. In order to obtain a semiring, we define the operation $\times$ *on the unit interval by a t-norm and* $+$ *as* max. *This is a commutative, positive and idempotent semiring with the number* 0 *as zero element and* 1 *as unit. Here are some 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:* $T(a,1) = T(1,a) = a$ *and* $T(a,b) = 0$ *in all other cases.*

The semiring induced by the product t-norm is also called probabilistic semiring. *Instead of maximization, we may also take minimization for addition, which only reverses the definition of zero and unit element.*

In addition to these examples, we may also define structures to derive semirings from other semirings. Vectors and matrices with semiring values are typical examples that form themselves a semiring.

5.1.1 Multidimensional Semirings

Take n possibly different semirings $\langle A_i, +_i, \times_i, \mathbf{0}_i, \mathbf{1}_i\rangle$ for $i = 1, \ldots, n$ and define

$$A \quad = \quad A_1 \times \cdots \times A_n$$

with corresponding, component-wise operations

$$\begin{aligned}(a_1, \ldots, a_n) + (b_1, \ldots, b_n) &= (a_1 +_1 b_1, \ldots, a_n +_n b_n) \\ (a_1, \ldots, a_n) \times (b_1, \ldots, b_n) &= (a_1 \times_1 b_1, \ldots, a_n \times_n b_n).\end{aligned}$$

These operations inherit associativity, commutativity and distributivity from their components. Hence, $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ becomes itself a semiring with zero element $\mathbf{0} = (\mathbf{0}_1, \ldots, \mathbf{0}_n)$ and unit $\mathbf{1} = (\mathbf{1}_1, \ldots, \mathbf{1}_n)$. If all A_i are commutative, positive or idempotent, then so is A.

5.1.2 Semiring Matrices

Given a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ and $n \in \mathbb{N}$, we consider the set $\mathcal{M}(A,n)$ of $n \times n$ matrices with elements in A. Addition and multiplication of semiring matrices is defined in the usual way:

$$(\mathbf{M}_1 + \mathbf{M}_2)(i,j) \quad = \quad \mathbf{M}_1(i,j) + \mathbf{M}_2(i,j) \tag{5.1}$$

and

$$(\mathbf{M}_1 \times \mathbf{M}_2)(i,j) \quad = \quad \sum_{k=1}^{n} \mathbf{M}_1(i,k) \times \mathbf{M}_2(k,j) \tag{5.2}$$

for $1 \leq i,j \leq n$. It is easy to see that square matrices over a semiring themselves form a semiring. Also, we obtain the zero element for semiring matrices by $\mathbf{O}(i,j) = \mathbf{0}$ for $1 \leq i,j \leq n$ and likewise, we obtain the unit element for the algebra of semiring matrices by $\mathbf{I}(i,j) = \mathbf{1}$ if $i = j$ and $\mathbf{I}(i,j) = \mathbf{0}$ otherwise. Finally, if $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ is positive or idempotent, then so are matrices over this semiring, but this does not hold for commutativity. Note in particular that ordinary real-valued, square matrices of the same order form a semiring.

5.2 SEMIRINGS AND ORDER

The substructure $\langle A, +\rangle$ is a commutative *monoid* with neutral element, which makes it always possible (Gondran & Minoux, 2008) to introduce a *canonical preorder* (see Definition A.2 in the appendix of Chapter 1). We define for $a, b \in A$:

$$a \leq b \quad \text{if, and only if} \quad \exists c \in A : a + c = b. \tag{5.3}$$

Lemma 5.1 *The Relation (5.3) is a preorder, i.e. reflexive and transitive.*

Proof: Reflexivity follows from the existence of a neutral element. Since $a + \mathbf{0} = a$ we have $a \leq a$ for all $a \in A$. To prove transitivity, let us assume that $a \leq b$ and $b \leq d$. Consequently, there exist $c, c' \in A$ such that $a + c = b$ and $b + c' = d$. We have $a + c + c' = d$ and therefore $a \leq d$. ■

The following lemma ensures that the canonical preorder is compatible with both semiring operations.

Lemma 5.2 *The canonical preorder of a semiring* $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *satisfies:*

1. $a \leq b$ *implies that* $a + c \leq b + c$ *for all* $a, b, c \in A$; *(SP1)*

2. $a \leq b$ *implies that* $a \times c \leq b \times c$ *and* $c \times a \leq c \times b$ *for all* $a, b, c \in A$; *(SP2)*

3. $\mathbf{0} \leq a, b \leq a + b$ *for all* $a, b \in A$. *(SP3)*

Proof:

1. Assume $a \leq b$, i.e. there exists $x \in A$ such that $a + x = b$. Then, $(a + c) + x = b + c$ and therefore $a + c \leq b + c$.

2. Assume $a \leq b$, i.e. there exists $x \in A$ such that $a + x = b$. Then $(a + x) \times c = (a \times c) + (x \times c) = b \times c$ and therefore $a \times c \leq b \times c$. The proof of the second statement is symmetric.

3. Since $\mathbf{0} + b = b$ we clearly have $\mathbf{0} \leq b$. Using Property (SP1) we further derive $a + \mathbf{0} \leq a + b$. Hence, $a \leq a + b$ and a similar argument shows $b \leq a + b$.

■

The canonical preorder of a semiring is in general not antisymmetric. Take for example the arithmetic semiring of integers $\langle \mathbb{Z}, +, \cdot, 0, 1\rangle$ and observe that $a \leq b$ and $b \leq a \nRightarrow a = b$. This is a consequence of the existence of additive inverses. Antisymmetry is therefore compatible with the existence of additive inverses or, in other words, with the ring structure of $\langle A, +\rangle$. This is also indicated by Property (SP3) of the above lemma. However, if the canonical preorder is antisymmetric, it is called a *partial order* (see Definition A.2 in the appendix of Chapter 1) and the semiring becomes a *dioid* (Gondran & Minoux, 2008). A sufficient condition is *idempotency*.

Lemma 5.3 *If $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ is an idempotent semiring, we may characterize the canonical preorder (5.3) as*

$$a \leq b \quad \textit{if, and only if} \quad a + b = b. \tag{5.4}$$

Proof: It is sufficient to show that $a \leq b$ according to (5.3) implies $a + b = b$. Suppose $a \leq b$, i.e. there exists $c \in A$ such that $a + c = b$. We then have by idempotency $a + b = a + a + c = a + c = b$. ■

Lemma 5.4 *The Relation (5.4) is a partial order.*

Proof: Since the two relations (5.3) and (5.4) are equivalent in case of an idempotent semiring, we obtain reflexivity and transitivity from Lemma 5.1. To prove antisymmetry, we assume that $a \leq b$ and $b \leq a$, i.e. $a + b = b$ and $b + a = a$. Consequently, $a = a + b = b$ and therefore $a = b$. ■

An idempotent semiring is therefore always a dioid and we refer to Relation (5.4) as its canonical partial order.

Example 5.9 *We already pointed out that the canonical preorder in the arithmetic semiring $\langle \mathbb{Z}, +, \cdot, 0, 1\rangle$ is not antisymmetric and therefore not a partial order. Restricting this semiring to non-negative integers $\langle \mathbb{N} \cup \{0\}, +, \cdot, 0, 1\rangle$ turns the canonical preorder into a partial order. This is an example of a dioid that is not idempotent and shows that idempotency is indeed only a sufficient condition for a canonical partial order. The examples 5.2 to 5.8 are all idempotent and thus dioids. However, it is important to remark that the canonical semiring order does not necessarily agree with the natural order between number. Take for example the tropical semiring $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0\rangle$ where the relation (5.4) becomes $a \leq b$ if, and only if, $\min\{a, b\} = b$. Here, the canonical semiring order corresponds to the inverse of the natural order between natural numbers.*

Two further properties of idempotent semirings are listed in the following lemma. In particular, we learn from (SP5) why all idempotent semirings in the example catalogue of Section 5.1 are positive.

Lemma 5.5 *Let $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ be an idempotent semiring.*

1. *We have $a + b = \sup\{a, b\}$ with respect to the canonical order.* *(SP4)*

2. *The semiring is positive.* *(SP5)*

Proof:

1. According to Property (SP3) $a, b \leq a + b$. Let $c \in A$ be another upper bound for a and b, i.e. $a \leq c$ and $b \leq c$. We conclude from $a+c = c$ and $b+c = c$ that $(a+c)+(b+c) = c+c$ and by idempotency $(a+b)+c = c$. Consequently, $a + b \leq c$ which implies that $a + b$ is the least upper bound of a and b.

2. Suppose that $a+b = \mathbf{0}$. Applying Property (SP3) we obtain $\mathbf{0} \leq a \leq a+b = \mathbf{0}$ and by transitivity and antisymmetry we conclude that $a = \mathbf{0}$. Similarly, we prove $b = \mathbf{0}$.

■

Definition 5.3 *If a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ satisfies $a + \mathbf{1} = \mathbf{1}$ for all $a \in A$, then it is called* bounded semiring. *If also commutativity holds, the semiring is called* c-semiring *(*constraint semiring*).*

This definition comes from (Mohri, 2002; Bistarelli *et al.*, 2002) and characterizes bounded semirings by the fact that the unit element is absorbing with respect to addition. Also, bounded semirings are close to *simple semirings* in (Lehmann, 1976). We first remark that bounded semirings are idempotent, since $\mathbf{1} + \mathbf{1} = \mathbf{1}$ implies that $a + a = a$ for all $a \in A$. This follows from distributivity: $a + a = \mathbf{1} \times (a + a) = (\mathbf{1} \times a) + (\mathbf{1} \times a) = (\mathbf{1} + \mathbf{1}) \times a = \mathbf{1} \times a = a$. Consequently, the relation $\leq$ is a partial order which furthermore satisfies the following properties.

Lemma 5.6 *Let $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ be a bounded semiring.*

1. *For all $a, b \in A$ we have $\mathbf{0} \leq a \times b \leq a, b \leq a + b \leq \mathbf{1}$.* *(SP6)*

2. *If $\times$ is idempotent, $a \times b = \inf\{a, b\}$.* *(SP7)*

Proof:

1. From Definition 5.3 follows that $a \leq \mathbf{1}$ for all $a \in A$. Because Property (SP3) still holds, it remains to be proved that $a \times b \leq a$ for all $a, b \in A$. This claim results from the distributive law since $a + (a \times b) = (a \times \mathbf{1}) + (a \times b) = a \times (\mathbf{1} + b) = a \times \mathbf{1} = a$.

2. By Property (SP6) we have $a \times b \leq a, b$. Let c be another lower bound of a and b, i.e. $c \leq a$ and $c \leq b$. Then, by (SP2) $c \times b \leq a \times b$. Similarly, we derive from $c \leq b$ that $c = c \times c \leq c \times b$ and therefore by transitivity $c \leq a \times b$. Thus, $a \times b$ is the greatest lower bound.

■

With supremum and infimum according to (SP4) and (SP7), c-semirings with idempotent multiplication adopt the structure of a lattice according to Definition A.5 in the appendix of Chapter 1. Moreover, the following theorem states that it is even distributive. This has been remarked by (Bistarelli *et al.*, 1997), but in contrast to this reference it is not necessarily complete since we do not assume infinite summation in the definition of a c-semiring.

Theorem 5.1 *A c-semiring with idempotent multiplication is a* bounded, distributive lattice *with* $a + b = \sup\{a, b\} = a \vee b$ *and* $a \times b = \inf\{a, b\} = a \wedge b$.

Proof: It remains to prove that the two operations + and × distribute over each other in the case of an idempotent c-semiring. By definition, × distributes over +. On the other hand, we have for $a, b, c \in A$

$$\begin{aligned}(a+b) \times (a+c) &= a \times (a+b) + c \times (a+b)\\ &= (a \times a) + (a \times b) + (a \times c) + (b \times c)\\ &= a + a \times (b+c) + (b \times c)\\ &= a \times (\mathbf{1} + (b+c)) + (b \times c)\\ &= (a \times \mathbf{1}) + (b \times c) \;=\; a + (b \times c).\end{aligned}$$

We therefore have a distributive lattice. Since $a + \mathbf{1} = \mathbf{1}$ and $a \times \mathbf{0} = \mathbf{0}$ for all $a \in A$ the lattice is bounded with bottom element $\mathbf{0}$ and top element $\mathbf{1}$. ■

Example 5.10 *The Boolean semiring of Example 5.2, the bottleneck semiring of Example 5.3, the tropical semiring of Example 5.4, the truncation semiring of Example 5.6 and all t-norm semirings of Example 5.8 are c-semirings and therefore also bounded. Among them, multiplication is idempotent in the Boolean semiring and in the bottleneck semiring which therefore become bounded, distributive lattices. Note also that if all semirings are c-semirings, then so is the induced multidimensional semiring. A similar statement does not hold for the semiring of matrices.*

Example 5.11 (Semiring of Boolean Functions) *Consider a set of* $r \in \mathbb{N}$ *propositional variables. Then, the set of all Boolean functions* $f : \{0,1\}^r \rightarrow \{0,1\}$ *forms a semiring with addition* $f + g = \max\{f, g\}$ *and multiplication* $f \times g = \min\{f, g\}$*, both being evaluated point-wise. If* f_0 *denotes the constant mapping to* 0 *and* f_1 *the constant mapping to* 1*, then* f_0 *is the zero element and* f_1 *the unit element of the semiring. The semiring is a c-semiring and since multiplication is also idempotent, this semiring is a distributive lattice. In particular, the semiring of Boolean functions with* $r = 0$ *corresponds to the Boolean semiring of Example 5.2 where the two and only elements* f_0 *and* f_1 *are identified with their values* 0 *and* 1.

Conversely to Theorem 5.1, we note that every bounded, distributive lattice (see Definition A.6 in the appendix of Chapter 1) is an idempotent semiring with *join* for + and *meet* for ×. The bottom element ⊥ of the lattice becomes the zero element and the top element ⊤ becomes the unit element. Thus, this semiring is even a c-semiring with idempotent multiplication. We therefore have an equivalence between the two structures and conclude that the powerset lattice of Example A.2 and the division lattice of Example A.3 are both c-semirings with idempotent multiplication. There are naturally many other semiring examples that have not appeared in this chapter. For example, we mention that every (unit) *ring* is a semiring with the additional property that inverse additive elements exist. In the same breath, *fields* are rings with multiplicative inverses. These remarks lead to further semiring examples such as the *ring of polynomials* or the *Galois field*. We also refer to (Davey & Priestley, 1990) for a broad listing of further examples of distributive lattices and to the comprehensive literature about semirings (Golan, 1999; Golan, 2003; Gondran & Minoux, 2008).

5.3 SEMIRING VALUATION ALGEBRAS

Equipped with this catalogue of semiring examples, we will now come to the main part of this chapter and show how semirings induce valuation algebras by a mapping from tuples to semiring values. This theory was developed in (Kohlas, 2004; Kohlas & Wilson, 2006; Kohlas & Wilson, 2008), who also substantiated for the first time the relationship between semiring properties and the attributes of their induced valuation algebras. In particular, we will discover that formalisms for constraint modelling are an important subgroup of semiring valuation algebras. For this reason, we subsequently prefer the term configuration instead of tuple which is more common in constraint literature. In this context, a similar framework to abstract constraint satisfaction problems was introduced by (Bistarelli *et al.*, 1997; Bistarelli *et al.*, 2002). The importance of such formalisms outside the field of constraint satisfaction was furthermore explored by (Aji, 1999; Aji & McEliece, 2000), who also proved the applicability of the Shenoy-Shafer architecture. However, this is only one of many conclusions to which we come by showing that semiring-based formalisms satisfy the valuation algebra axioms.

To start with, consider a *commutative semiring* $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ and a set r of variables with finite frames. A *semiring valuation* ϕ with domain $s \subseteq r$ is defined to be a function that associates a value from A with each configuration $\mathbf{x} \in \Omega_s$,

$$\phi : \Omega_s \rightarrow A.$$

Remember that $\Omega_\emptyset = \{\diamond\}$ such that a semiring valuation on the empty domain is $\phi(\diamond) \in A$. We subsequently denote the set of all semiring valuations with domain s by Φ_s and use Φ for all semiring valuations whose domains belong to the powerset lattice $D = \mathcal{P}(r)$. Next, the following operations in $\langle\Phi, D\rangle$ are introduced:

1. *Labeling:* $\Phi \rightarrow D$: $d(\phi) = s$ if $\phi \in \Phi_s$.

2. *Combination:* $\Phi \times \Phi \rightarrow \Phi$: for $\phi, \psi \in \Phi$ and $\mathbf{x} \in \Omega_{d(\phi)\cup d(\psi)}$ we define

$$(\phi \otimes \psi)(\mathbf{x}) \quad = \quad \phi(\mathbf{x}^{\downarrow d(\phi)}) \times \psi(\mathbf{x}^{\downarrow d(\psi)}). \tag{5.5}$$

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

$$\phi^{\downarrow t}(\mathbf{x}) \quad = \quad \sum_{\mathbf{y} \in \Omega_{s-t}} \phi(\mathbf{x}, \mathbf{y}). \tag{5.6}$$

Note that the definition of projection is well defined due to the associativity and commutativity of semiring addition and the finiteness of the domains. We now arrive at the central theorem of this chapter which states that we obtain a valuation algebra from every commutative semiring through the above generic construction:

Theorem 5.2 *A system of semiring valuations $\langle\Phi, D\rangle$ with respect to a commutative semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ with labeling, combination and projection as defined above, satisfies the axioms of a valuation algebra.*

Proof: We verify the Axioms (A1) to (A6) of a valuation algebra given in Section 1.1. Observe that the labeling (A2), projection (A3) and domain (A6) properties are immediate consequences of the above definitions.

(A1) *Commutative Semigroup:* The commutativity of combination follows directly from the commutativity of the × operation in the semiring A and the definition of combination. To prove associativity, assume that ϕ, ψ and η are valuations with domains $d(\phi) = s$, $d(\psi) = t$ and $d(\eta) = u$, then for $\mathbf{x} \in \Omega_{s \cup t \cup u}$

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

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

(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 $t \subseteq s \subseteq t \cup u$. Using the distributivity of semiring multiplication over addition, we obtain for $\mathbf{x} \in \Omega_s$,

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

■

To sum it up, a simple mapping from configurations to the values of a commutative semiring provides sufficient structure to give rise to a valuation algebra. The listing of semirings given in the two foregoing sections served to exemplify the semiring concepts introduced beforehand. We are next going to reconsider these semirings in order to show which valuation algebras they concretely induce. We will meet familiar instances such as arithmetic potentials or indicator functions, but also many new instances that considerably extend the valuation algebra catalogue of Chapter 1.

5.4 EXAMPLES OF SEMIRING VALUATION ALGEBRAS

If we consider the arithmetic semiring $\langle \mathbb{R}_{\geq 0}, +, \cdot, 0, 1 \rangle$ of non-negative real numbers presented in Section 5.1, we come across the valuation algebra of arithmetic potentials introduced as Instance 1.3. It is easy to see that both operations defined for semiring valuations correspond exactly to the operations for arithmetic potentials in equation (1.10) and (1.11). Therefore, the proof of Theorem 5.2 also delivers the promised argument that arithmetic potentials indeed satisfy the valuation algebra axioms. Moreover, since the arithmetic semiring can be built on complex numbers, we also come to the conclusion that the formalism used for the discrete Fourier transform in Instance 2.6 forms a valuation algebra. Similar statements hold for the valuation algebras of indicator functions, Boolean functions, crisp constraints or the relational algebra that are induced by the Boolean semiring $\langle \{0, 1\}, \max, \min, 0, 1 \rangle$. Again, if we replace semiring addition and multiplication in equation (5.5) and (5.6) by the corresponding operations max and min, we obtain the combination and projection rules given in Instance 1.1.

■ 5.1 Weighted Constraints - Spohn Potentials - GAI Preferences

Besides crisp constraints, alternative constraint systems may be derived by examining other semirings: the tropical semiring $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0 \rangle$ of Example 5.4 induces the valuation algebra of *weighted constraints* (Bistarelli *et al.*, 1997; Bistarelli *et al.*, 1999). This formalism also corresponds to *Spohn potentials* (Spohn, 1988) which have been proposed as a dynamic theory of graded belief states based on ordinary numbers. (Kohlas, 2003) delivers the explicit proof that weighted constraints satisfy the valuation algebra axioms. But in our context of semiring valuation algebras, this insight follows naturally from Theorem 5.2. No further proof is necessary. We again refer to later sections for concrete applications based on weighted constraints that in turn give birth to further inference problems. Here, we content ourselves with a small example on how to compute with weighted constraints.

Let $r = \{A, B, C\}$ be a set of three variables with finite frames $\Omega_A = \{a, \overline{a}\}$, $\Omega_B = \{b, \overline{b}\}$ and $\Omega_C = \{c, \overline{c}\}$. We define two weighted constraints c_1 and c_2 with domain $d(c_1) = \{A, B\}$ and $d(c_2) = \{B, C\}$:

$c_1 =$

A	B	
a	b	4
a	$\bar{b}$	8
$\bar{a}$	b	0
$\bar{a}$	$\bar{b}$	2

$c_2 =$

B	C	
b	c	∞
b	$\bar{c}$	1
$\bar{b}$	c	1
$\bar{b}$	$\bar{c}$	7

We combine c_1 and c_2 and project the result to $\{A, C\}$

$c_3 = c_1 \otimes c_2 =$

A	B	C	
a	b	c	∞
a	b	$\bar{c}$	5
a	$\bar{b}$	c	9
a	$\bar{b}$	$\bar{c}$	15
$\bar{a}$	b	c	∞
$\bar{a}$	b	$\bar{c}$	1
$\bar{a}$	$\bar{b}$	c	3
$\bar{a}$	$\bar{b}$	$\bar{c}$	9

$c_3^{\downarrow\{A,C\}} =$

A	C	
a	c	∞
a	$\bar{c}$	5
$\bar{a}$	c	3
$\bar{a}$	$\bar{c}$	1

Alternatively to the tropical semiring, we may also take the arctic semiring $\langle \mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0\rangle$ of Example 5.5 to build weighted constraints, which then corresponds to the formalism of *generalized additive independent preferences (GAI preferences)* (Fishburn, 1974; Bacchus & Adam, 1995). The modification of the above example to weighted constraints induced by the arctic semiring is left to the reader.

5.2 Possibility Potentials - Probabilistic Constraints - Fuzzy Sets

The very popular valuation algebra of *possibility potentials* is induced by the triangular norm semirings $\langle [0,1], \max, \text{t-norm}, 0, 1\rangle$ of Example 5.8. Historically, possibility theory was proposed by (Zadeh, 1978) as an alternative approach to probability theory, and (Shenoy, 1992a) furnished the explicit proof that this formalism indeed satisfies the valuation algebra axioms. This was limited to some specific t-norms and (Kohlas, 2003) generalized the proof to arbitrary t-norms. However, thanks to the generic construction of semiring valuations, the general statement that possibility potentials form a valuation algebra under any t-norm follows immediately from Theorem 5.2. We also refer to (Schiex, 1992) which unified this and the foregoing instance to *possibilistic constraints*.

To give an example of how to compute with possibility potentials, we settle for the use of the Lukasiewicz t-norm defined as $a \times b = \max\{a + b - 1, 0\}$ for all $a \in [0, 1]$. Again, let $r = \{A, B, C\}$ be a set of three variables with finite frames $\Omega_A = \{a, \bar{a}\}$, $\Omega_B = \{b, \bar{b}\}$ and $\Omega_C = \{c, \bar{c}\}$. We define two possibility potentials p_1 and p_2 with domain $d(p_1) = \{A, B\}$ and $d(p_2) = \{B, C\}$:

$p_1 =$

A	B	
a	b	0.6
a	$\bar{b}$	0.4
$\bar{a}$	b	0.3
$\bar{a}$	$\bar{b}$	0.7

$p_2 =$

B	C	
b	c	0.2
b	$\bar{c}$	0.8
$\bar{b}$	c	0.9
$\bar{b}$	$\bar{c}$	0.1

We combine p_1 and p_2 and project the result to $\{A, C\}$

$p_3 = p_1 \otimes p_2 =$

A	B	C	
a	b	c	0
a	b	$\bar{c}$	0.2
a	$\bar{b}$	c	0
a	$\bar{b}$	$\bar{c}$	0.2
$\bar{a}$	b	c	0
$\bar{a}$	b	$\bar{c}$	0.1
$\bar{a}$	$\bar{b}$	c	0.6
$\bar{a}$	$\bar{b}$	$\bar{c}$	0

$p_3^{\downarrow\{A,C\}} =$

A	C	
a	c	0
a	$\bar{c}$	0.2
$\bar{a}$	c	0.6
$\bar{a}$	$\bar{c}$	0.1

Particularly important among these t-norms is ordinary multiplication. The valuation algebra induced by the corresponding probabilistic semiring of Example 5.8 is known as the formalism of *probabilistic constraints* (Bistarelli & Rossi, 2008) or *fuzzy subsets* (Zadeh, 1978). An example can easily be obtained from the arithmetic potentials presented as Instance 1.3. Combination is identical for both instances such that only projection, which now consists of maximization instead of summation, has to be recomputed.

5.3 Set-based Constraints - Assumption-based Constraints

The valuation algebra induced by the powerset lattice of Example A.2 in the appendix of Chapter 1 corresponds to the formalism of *set-based constraints* (Bistarelli *et al.*, 1997). Imagine two propositional variables A_1 and A_2. We then build the powerset lattice from their configuration set $\mathcal{P}(\{(0,0), \ldots, (1,1)\})$ and obtain a complete, distributive lattice according to Example A.2. To introduce valuations that assign values from this semiring, let $r = \{A, B, C\}$ be a set of three variables with finite frames $\Omega_A = \{a, \bar{a}\}$, $\Omega_B = \{b, \bar{b}\}$ and $\Omega_C = \{c, \bar{c}\}$. We define two set-based constraint c_1 and c_2 with domain $d(c_1) = \{A, B\}$ and $d(c_2) = \{B, C\}$:

$c_1 =$

A	B	
a	b	$\{(0,0), (1,0)\}$
a	$\bar{b}$	$\emptyset$
$\bar{a}$	b	$\{(1,1)\}$
$\bar{a}$	$\bar{b}$	$\{(1,0), (0,1)\}$

$c_2 =$

B	C	
b	c	$\{(0,0), (0,1), (1,0)\}$
b	$\bar{c}$	$\{(0,1)\}$
$\bar{b}$	c	$\emptyset$
$\bar{b}$	$\bar{c}$	$\{(0,0), (0,1)\}$

We combine c_1 and c_2 and project the result to $\{A, C\}$

$c_3 =$

A	B	C	
a	b	c	$\{(0,0),(1,0)\}$
a	b	$\bar{c}$	$\emptyset$
a	$\bar{b}$	c	$\emptyset$
a	$\bar{b}$	$\bar{c}$	$\emptyset$
$\bar{a}$	b	c	$\emptyset$
$\bar{a}$	b	$\bar{c}$	$\emptyset$
$\bar{a}$	$\bar{b}$	c	$\emptyset$
$\bar{a}$	$\bar{b}$	$\bar{c}$	$\{(0,1)\}$

$c_3^{\downarrow\{A,C\}} =$

A	C	
a	c	$\{(0,0),(1,0)\}$
a	$\bar{c}$	$\emptyset$
$\bar{a}$	c	$\emptyset$
$\bar{a}$	$\bar{c}$	$\{(0,1)\}$

Let us give a particular interpretation to this example: The variables A and B are considered as *assumptions*. Now, observing that $(1,0) \in c_1(a,b)$, we may say that the configuration (a,b) is possible under the assumption that A holds but not B. Since $(0,0) \in c_1(a,b)$ too, the configuration (a,b) is still possible if neither A nor B holds. The constraint c_2 specifies that (b,c) is possible if at most one of the assumptions is true. Combining the two constraints c_1 and c_2 gives $\{(0,0),(1,0)\}$ for the configuration (a,b,c). The assignment $(0,1)$ is missing since c_1 does not hold under this assumption. So, a set-based constraint can also be understood as an *assumption-based constraint* which relates this formalism to *assumption-based reasoning* (de Kleer *et al.*, 1986).

The number of valuation algebras obtained via the construction of semiring valuations is enormous. In fact, we obtain a different valuation algebra from every commutative semiring and the instances presented just above are only the tip of the iceberg. Generic constructions also produce formalisms for which we not even have a slight idea of possible application fields. Nevertheless, we understand their algebraic properties and also possess efficient algorithms for their processing thanks to the local computation framework. One such example could be the valuation algebra induced by the division lattice from Example A.3.

5.5 PROPERTIES OF SEMIRING VALUATION ALGEBRAS

Throughout the two foregoing chapters about local computation, we introduced additional properties of valuation algebras which are interesting for computational and semantical purposes. The potential presence of these properties had to be verified for each formalism separately. A further gain of generic constructions is that such properties can now be verified for whole families of valuation algebras instances. Following this guideline, we are now going to investigate which mathematical attributes are needed in the semiring to guarantee the presence of neutral and null elements in the induced valuation algebra. Again, the more technical discussion of division for semiring valuations is postponed to the appendix of this chapter.

5.5.1 Semiring Valuation Algebras with Neutral Elements

Section 3.3 introduced neutral elements as a (rather inefficient) possibility to initialize join tree nodes or, in other words, to derive a join tree factorization from a given knowledgebase. On the other hand, neutral elements represent an important semantical aspect in a valuation algebra by expressing neutral knowledge with respect to some domain. Since we presuppose the existence of a unit element in the underlying semiring, we always get a neutral elements in the induced valuation algebra by the definition $e_s(\mathbf{x}) = \mathbf{1}$ for all $\mathbf{x} \in \Omega_s$ and $s \in D$. This is the neutral valuation in the semigroup Φ_s with respect to combination, i.e. for all $\phi \in \Phi_s$ we have

$$e_s \otimes \phi = e_s(\mathbf{x}) \times \phi(\mathbf{x}) \;=\; \mathbf{1} \times \phi(\mathbf{x}) \;=\; \phi.$$

These neutral elements satisfy Property (A7) of Section 3.3:

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

$$(e_s \otimes e_t)(\mathbf{x}) \;=\; e_s(\mathbf{x}^{\downarrow s}) \times e_t(\mathbf{x}^{\downarrow t}) \;=\; \mathbf{1} \times \mathbf{1} \;=\; \mathbf{1}. \tag{5.7}$$

5.5.2 Stable Semiring Valuation Algebras

Although all semiring valuation algebras provide neutral elements, they generally are not stable, i.e. neutral semiring valuations do not necessarily project to neutral elements again. This has already been remarked in Instance 3.2 in Section 3.3.1 where it is shown that the valuation algebra induced by the arithmetic semiring of Example 5.1 is not stable. However, it turns out that idempotent addition is a sufficient semiring property to induce a stable valuation algebra.

(A8) *Stability:* If the semiring is idempotent, we have $e_s^{\downarrow t} = e_t$ for $t \subseteq s$ and $\mathbf{x} \in \Omega_s$

$$e_s^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y} \in \Omega_{s-t}} e_s(\mathbf{x}, \mathbf{y}) \;=\; \sum_{\mathbf{y} \in \Omega_{s-t}} \mathbf{1} \;=\; \mathbf{1}. \tag{5.8}$$

Let us summarize the insights of this section:

Theorem 5.3 *All semiring valuation algebras provide neutral elements. If furthermore the semiring is idempotent, then the induced valuation algebra is stable.*

In particular, all elements induced by c-semirings (or bounded, distributive lattices) are stable since these properties always imply idempotency. From this new perspective, we easily confirm the properties related to neutral elements of indicator functions and arithmetic potentials that have been derived in Instance 3.1 and 3.2. Let us consider some more instances:

■ 5.4 Weighted Constraints and Neutral Elements

The valuation algebra of weighted constraints from Instance 5.1 is induced by the tropical semiring of Example 5.4 which has the number 0 as unit element.

The neutral weighted constraint e_s for the domain $s \in D$ and $\mathbf{x} \in \Omega_s$ is therefore given by $e_s(\mathbf{x}) = 0$. Further, this semiring is idempotent, which directly implies stability in the valuation algebra of weighted constraints.

■ 5.5 Probabilistic Constraints and Neutral Elements

The valuation algebra of probabilistic constraints from Instance 5.2 is induced by the multiplicative t-norm semiring of Example 5.8 which has the number 1 as unit element. The neutral probabilistic constraint e_s for the domain $s \in D$ and $\mathbf{x} \in \Omega_s$ is therefore given by $e_s(\mathbf{x}) = 1$. Further, this semiring is idempotent, which again implies stability in this valuation algebra.

This gives a first indication of how easily the algebraic properties of a formalism can be analysed through the perspective of generic construction. A second interesting valuation algebra property that we are now going to study from the semiring perspective is the existence of null elements.

5.5.3 Semiring Valuation Algebras with Null Elements

Null elements have been introduced in Section 3.4 for pure semantical purposes. They represent contradictory information with respect to a given domain and are significant to interpret the possible results of inference problems. Since every semiring contains a zero element, we can directly advise the candidate for a null semiring valuation in Φ_s. This is $z_s(\mathbf{x}) = \mathbf{0}$ for all $\mathbf{x} \in \Omega_s$, and it clearly holds for $\phi \in \Phi_s$ that

$$\phi \otimes z_s(\mathbf{x}) \;=\; \phi(\mathbf{x}) \times z_s(\mathbf{x}) \;=\; \mathbf{0}$$

and therefore $\phi \otimes z_s = z_s$. But this is not yet sufficient because the nullity axiom (A9) must also be satisfied, and this comprises two requirements. First, remark that null elements always project to null elements. More involved is the second requirement which claims that only null elements project to null elements. Positivity of the semiring is a sufficient condition to fulfill this axiom.

(A9) *Nullity:* In a positive semiring $\phi^{\downarrow t} = z_t$ always implies that $\phi = z_s$. Indeed, let $\mathbf{x} \in \Omega_t$ with $t \subseteq s = d(\phi)$. Then,

$$z_t(\mathbf{x}) \;=\; \mathbf{0} \;=\; \phi^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y} \in \Omega_{s-t}} \phi(\mathbf{x},\mathbf{y}) \tag{5.9}$$

implies that $\phi(\mathbf{x},\mathbf{y}) = \mathbf{0}$ for all $\mathbf{y} \in \Omega_{s-t}$ and consequently, $\phi = z_s$.

To summarize:

Theorem 5.4 *Semiring valuation algebras induced by commutative, positive semirings provide null elements.*

Remember that due to Property (SP5), idempotent semirings are always positive and therefore induce valuation algebras with null elements. This again confirms what we have found out in Section 3.4: indicator functions are induced by the Boolean semiring of Example 5.2 and therefore provide null elements. The valuation algebra of arithmetic potentials is induced by the arithmetic semirings of Example 5.1 that are, in certain cases, positive and, in others, not. In the latter case, no null elements will be present. Let us add some further examples of valuation algebras with null elements:

■ 5.6 Weighted Constraints and Null Elements

The valuation algebra of weighted constraints from Instance 5.1 is induced by the tropical semiring of Example 5.4 which has ∞ as zero element. This semiring is idempotent, thus positive and therefore induces the null element $z_s(\mathbf{x}) = \infty$ for the domain $s \in D$ and $\mathbf{x} \in \Omega_s$.

■ 5.7 Possibility Potentials and Null Elements

Independently of the chosen t-norm, all t-norm semirings of Example 5.8 are idempotent and therefore positive. Consequently, they all provide the same null element $z_s(\mathbf{x}) = 0$ for the domain $s \in D$ and $\mathbf{x} \in \Omega_s$.

■ 5.8 Set-based Constraints and Null Elements

The valuation algebra of set-based constraints from Instance 5.3 is induced by the powerset lattice of Example A.2. This semiring is again positive with the empty set as zero element. It therefore induces the null element $z_s(\mathbf{x}) = \emptyset$ for the domain $s \in D$ and $\mathbf{x} \in \Omega_s$.

In the appendix of this chapter, we provide a detailed analysis of division and identify the semiring properties that either lead to separative, regular or idempotent valuation algebras. Also, we revisit normalization or scaling which is an important application of division in valuation algebras. Figure 5.1 summarizes the properties related to neutral and null elements for each semiring valuation algebra studied in this chapter. The properties related to division are shown in Figure E.4 in the appendix.

5.6 SOME COMPUTATIONAL ASPECTS

Examples of inference problems based on semiring valuations have already been considered in Instance 2.1 and Instance 2.4. Further examples from constraint reasoning will be listed in Chapter 8. Given a multi-query inference problem with a knowledgebase of semiring valuations, we may always apply the Shenoy-Shafer architecture for the computation of the queries. The applicability of the other architectures from Chapter 4 naturally depends on the presence of a division operator in the semiring valuation algebra. Let us consider the complexity of the Shenoy-Shafer architecture for semiring valuations in more detail. A possible weight predictor for semiring val-

	Instance	Neutral Elements	Stability	Null Elements
1.1	Indicator Functions	√	√	√
1.3	Arithmetic Potentials on $\mathbb{R}_{\geq 0}$	√	○	√
1.3	Arithmetic Potentials on $\mathbb{R}, \mathbb{C}$	√	○	○
5.1	Weighted Constraints	√	√	√
5.2	Possibility Potentials	√	√	√
5.3	Set-based Constraints	√	√	√

Figure 5.1 Semiring valuation algebras with neutral and null elements.

uations was already given in equation (3.17). Inserted into equation (4.8), we obtain for the time complexity of the Shenoy-Shafer architecture

$$\mathcal{O}\left(|V| \cdot deg \cdot d^{\omega^*+1}\right), \tag{5.10}$$

where d denotes the size of the largest variable frame. Concerning the space complexity, there is an important issue with respect to the general bound given in equation (4.9). In the Shenoy-Shafer architecture, the message sent from node i to neighbor $j \in ne(i)$ is obtained by first combining the node content of i with all messages received from all other neighbors of i except j. Then, the result of this combination is projected to the intersection of its domain and the node label of neighbor j. For arbitrary valuation algebras, we must assume that the complete combination has to be computed before the projection can be evaluated. This creates an intermediate factor whose domain is bounded by the node label $\lambda(i)$. In the general space complexity of equation (4.9) this corresponds to the first term. However, when dealing with semiring valuations, the computation of the complete combination can be omitted. For illustration, assume a set of semiring valuations $\{\phi_1, \ldots, \phi_n\} \subseteq \Phi$ with $s = d(\phi_1) \cup \ldots \cup d(\phi_n)$, $t \subseteq s$ and $\mathbf{x} \in \Omega_s$, then the projection

$$\psi(\mathbf{x}^{\downarrow t}) \;=\; (\phi_1 \otimes \ldots \otimes \phi_n)^{\downarrow t}(\mathbf{x}^{\downarrow t}) \;=\; \sum_{\mathbf{z} \in \Omega_{s-t}} (\phi_1 \otimes \ldots \otimes \phi_n)(\mathbf{x}^{\downarrow t}, \mathbf{z})$$

can be computed as follows: We initialize $\psi(\mathbf{y}) = \mathbf{0}$ for all $\mathbf{y} \in \Omega_t$ and compute for each configuration $\mathbf{x} \in \Omega_s$

$$\phi(\mathbf{x}) \;=\; \phi_1(\mathbf{x}^{\downarrow d(\phi_1)}) \times \ldots \times \phi_n(\mathbf{x}^{\downarrow d(\phi_n)}).$$

Then, the semiring value $\phi(\mathbf{x})$ is added to $\psi(\mathbf{x}^{\downarrow s})$. Clearly, this procedure determines ψ completely. Since only one value $\phi(\mathbf{x})$ exists at a time, the space complexity is bounded by the domain $t = d(\psi)$. Applying this technique in the Shenoy-Shafer architecture therefore reduces the space complexity to the domain of the messages, which in turn are bounded by the separator width sep^*. Altogether, we obtain for the space complexity of the Shenoy-Shafer architecture applied to semiring valuations

$$\mathcal{O}\Big(|V| \cdot d^{sep^*}\Big). \tag{5.11}$$

Note also that the time complexity is not affected by this procedure.

This brings the study of semiring valuation algebras to a first end. Semiring valuations will again take center stage in Chapter 8 where it is shown that the inference problem turns into an optimization task when dealing with valuation algebras induced by idempotent semirings. In the following section, we focus on a second generic construction that is closely related to semiring valuation algebras. But instead of mapping configurations to semiring values, we consider sets of configurations that are mapped to semiring values. An already known member of this new family of valuation algebras is the formalism of set potentials from Instance 1.4 in Chapter 1.

5.7 SET-BASED SEMIRING VALUATION ALGEBRAS

The family of semiring valuation algebras is certainly extensive, but there are nevertheless important formalisms that do not admit this particular structure. Some of them have already been mentioned in Section 1; for example, densities or set potentials. The last formalism is of particular interest in this context. Set potentials map configuration sets on non-negative real numbers, whereas both valuation algebra operations reduce to addition and multiplication. This is remarkably close to the buildup of semiring valuation algebras, if we envisage a generalization from configurations to configuration sets. In doing so, we hit upon a second family of valuation algebra instances that covers such important formalisms as set potentials or possibility measures. This theory again produces a multiplicity of new valuation algebra instances and also puts belief functions into a more general context. Here, we confine ourselves to a short treatment of set-based semiring valuations, leaving out an inspection of the more complex topic of division as performed for semiring valuations in the appendix of this chapter.

Let us again consider a commutative semiring $\langle E, +, \times, \mathbf{0}, \mathbf{1}\rangle$ and a countable set r of variables with finite frames. A *set-based semiring valuation* ϕ with finite domain $s \subseteq r$ is defined to be a function that associates a semiring value from E with each configuration subset of Ω_s,

$$\phi : \mathcal{P}(\Omega_s) \rightarrow E.$$

The set of all set-based semiring valuations with domain s will subsequently be denoted by Φ_s and we use Φ for all possible set-based semiring valuations whose domain belongs to the lattice $D = \mathcal{P}(r)$. Next, the following operations are introduced:

1. *Labeling:* $\Phi \rightarrow D$: $d(\phi) = s$ if $\phi \in \Phi_s$.

2. *Combination:* $\Phi \times \Phi \rightarrow \Phi$: for $\phi, \psi \in \Phi$ and $A \subseteq \Omega_{d(\phi) \cup d(\psi)}$ we define

$$(\phi \otimes \psi)(A) \quad = \quad \sum_{B \bowtie C = A} \phi(B) \times \psi(C). \tag{5.12}$$

3. *Projection:* $\Phi \times D \rightarrow \Phi$: for $\phi \in \Phi$, $t \subseteq d(\phi)$ and $A \subseteq \Omega_t$ we define

$$\phi^{\downarrow t}(A) \quad = \quad \sum_{\pi_t(B) = A} \phi(B). \tag{5.13}$$

Similar to Instance 1.4 we again use the operations from the relational algebra (i.e. projection and natural join) to deal with configuration sets. The advantages become apparent in the proof of the following theorem. Since relations are known to form a valuation algebra, we may benefit from their algebraic properties to verify the valuation algebra axioms for set-based semiring valuations. This makes the following proof particularly elegant.

Theorem 5.5 *A system of set-based semiring valuations $\langle \Phi, D \rangle$ with respect to a commutative semiring $\langle E, +, \times, \mathbf{0}, \mathbf{1} \rangle$ with labeling, combination and projection as defined above, satisfies the axioms of a valuation algebra.*

Proof: We verify the valuation algebra axioms given in Section 1.1. The labeling (A2) and projection axioms (A3) are direct consequences of the above definitions.

(A1) *Commutative Semigroup:* Commutativity of combination follows directly from the commutativity of semiring multiplication and natural join. To prove associativity we assume that $\phi \in \Phi_s$, $\psi \in \Phi_t$, $\nu \in \Phi_u$ and $A \subseteq \Omega_{s \cup t \cup u}$

$$\begin{aligned}
(\phi \otimes (\psi \otimes \nu))(A) &= \sum_{B \bowtie E = A} \phi(B) \times (\psi \otimes \nu)(E) \\
&= \sum_{B \bowtie E = A} \phi(B) \times \sum_{C \bowtie D = E} \psi(C) \times \nu(D) \\
&= \sum_{B \bowtie E = A} \sum_{C \bowtie D = E} \phi(B) \times \psi(C) \times \nu(D) \\
&= \sum_{B \bowtie C \bowtie D = A} \phi(B) \times \psi(C) \times \nu(D).
\end{aligned}$$

The same result is obtained in exactly the same way for $((\phi \otimes \psi) \otimes \nu)(A)$ which proves associativity of combination.

(A4) *Transitivity:* For $\phi \in \Phi$ with $s \subseteq t \subseteq d(\phi)$ we have

$$\begin{aligned}
(\phi^{\downarrow t})^{\downarrow s}(A) &= \sum_{\pi_s(B) = A} \phi^{\downarrow t}(B) = \sum_{\pi_s(B) = A} \sum_{\pi_t(C) = B} \phi(C) \\
&= \sum_{\pi_s(\pi_t(C)) = A} \phi(C) = \sum_{\pi_s(C) = A} \phi(C) = \phi^{\downarrow s}(A).
\end{aligned}$$

Observe that we used the transitivity of projection for relations.

(A5) *Combination:* Suppose that $\phi \in \Phi_s$, $\psi \in \Phi_t$ and $A \subseteq \Omega_z$, where $s \subseteq z \subseteq s \cup t$. Then, using the combination property of the relational algebra we obtain

$$\begin{aligned}
(\phi \otimes \psi)^{\downarrow z}(A) &= \sum_{\pi_z(B)=A} \phi \otimes \psi(B) \;=\; \sum_{\pi_z(B)=A} \; \sum_{C \bowtie D = B} \phi(C) \times \psi(D) \\
&= \sum_{\pi_z(C \bowtie D)=A} \phi(C) \times \psi(D) \;=\; \sum_{C \bowtie \pi_{t \cap z}(D)=A} \phi(C) \times \psi(D) \\
&= \sum_{C \bowtie E = A} \phi(C) \times \sum_{\pi_{t \cap z}(D)=E} \psi(D) \\
&= \sum_{C \bowtie E = A} \phi(C) \times \psi^{\downarrow t \cap z}(E) \;=\; \phi \otimes \psi^{\downarrow t \cap z}(A).
\end{aligned}$$

(A6) *Domain:* For $\phi \in \Phi$ and $x = d(\phi)$ we have

$$\phi^{\downarrow x}(A) \;=\; \sum_{B=A} \phi(B) \;=\; \phi(A).$$

■

Giving a first summary, commutative semirings possess enough structure to afford this second family of valuation algebras. The best known example of such an algebra are set potentials from Instance 1.4 (including their normalized variant called mass functions and belief functions), and it is indeed interesting to see them as a member of a more comprehensive family of formalisms. We list some further examples:

■ 5.9 Possibility Measures

(Zadeh, 1979) originally introduced *possibility measures* compatible to our framework of set-based semiring valuations over the product t-norm semiring of Example 5.8. But it turned out that possibility functions are completely specified by their values assigned to singleton configuration sets. Based on this insight, (Shenoy, 1992a) derived the valuation algebra of possibility potentials discussed as Instance 5.2. Working with possibility functions is therefore a bit unusual but we can deduce from the above theorem that they nevertheless form a valuation algebra themselves.

■ 5.10 Disbelief Functions

A *disbelief function* according to (Spohn, 1988; Spohn, 1990) is again compatible to our framework of set-based semiring valuations over the tropical semiring of Example 5.4. But similar to the foregoing instance, it was shown by (Shenoy, 1992b) that disbelief functions are completely specified by their values assigned to singleton configuration sets. This corresponds to the valuation algebra of weighted constraints discussed in Instance 5.1. It is therefore again not usual to work with disbelief functions in practice, although they also form a valuation algebra.

There are further attempts to connect belief functions with fuzzy set theory that results in further instances of set-based semiring valuations over different t-norm semirings. See for example (Biacino, 2007; Pichon & Denoeux, 2008).

5.8 PROPERTIES OF SET-BASED SEMIRING VALUATION ALGEBRAS

We are next going to investigate the necessary requirements for the underlying semiring to guarantee neutral and null elements in the induced valuation algebra.

5.8.1 Neutral and Stable Set-Based Semiring Valuations

Derived from Instance 3.3 we identify the neutral element e_s for the domain $s \in D$:

$$e_s(A) \quad = \quad \begin{cases} \mathbf{1}, & \text{if } A = \Omega_s, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{5.14}$$

Indeed, it holds for $\phi \in \Phi$ with $d(\phi) = s$ that

$$\begin{aligned} \phi \otimes e_s(A) &= \sum_{B \bowtie C = A} \phi(B) \times e_s(C) \\ &= \phi(A) \times e(\Omega_s) \; = \; \phi(A) \times \mathbf{1} \; = \; \phi(A). \end{aligned}$$

The second equality follows since for all other values of C we have $e_s(C) = \mathbf{0}$. These elements also satisfy property (A7):

(A7) *Neutrality:* On the one hand we have

$$e_s \otimes e_t(\Omega_{s \cup t}) \; = \; \sum_{A \bowtie B = \Omega_{s \cup t}} e_s(A) \times e_t(B) \; = \; e_s(\Omega_s) \times e_t(\Omega_t) \; = \; \mathbf{1}.$$

On the other hand, if $A \bowtie B \subset \Omega_{s \cup t}$, then either $A \subset \Omega_s$ or $B \subset \Omega_t$. So, at least one factor corresponds to the zero element of the semiring and therefore $e_s \otimes e_t(C) = \mathbf{0}$ for all $C \subset \Omega_{s \cup t}$.

Set-based semiring valuation algebras are always stable.

(A8) *Stability:* On the one hand we have

$$e_s^{\downarrow t}(\Omega_t) \; = \; \sum_{\pi_t(A) = \Omega_t} e_s(A) \; = \; e_s(\Omega_s) \; = \; \mathbf{1}.$$

The second equality holds because $e_s(\Omega_s) = \mathbf{1}$ is the only non-zero term within this sum. On the other hand, we have $e_s^{\downarrow t}(A) = \mathbf{0}$ for all $A \subset \Omega_t$ because $e_s(\Omega_s)$ does not occur in the sum of the projection.

These result are summarized in the following theorem:

Theorem 5.6 *Set-based semiring valuation algebras provide neutral elements and are always stable.*

5.8.2 Null Set-Based Semiring Valuations

Because all semirings possess a zero element, we have for every domain $s \in D$ a valuation z_s such that $\phi \otimes z_s = z_s \otimes \phi = z_s$. This element is defined as $z_s(A) = \mathbf{0}$ for all $A \subseteq \Omega_s$. Indeed, we have for $\phi \in \Phi_s$,

$$\phi \otimes z_s(A) \;=\; \sum_{B \bowtie C = A} \phi(B) \times z_s(C) \;=\; \mathbf{0}.$$

These candidates z_s must additionally satisfy the nullity axiom that requires two properties. The first condition that null elements project to null elements is clearly satisfied. More involved is the second condition that only null elements project to null elements. Positivity is again a sufficient condition.

(A9) *Nullity:* For $\phi \in \Phi_s$ and $t \subseteq s$, we have

$$\mathbf{0} \;=\; \phi^{\downarrow t}(A) \;=\; \sum_{\pi_t(B)=A} \phi(B)$$

implies that $\phi(B) = \mathbf{0}$ for all B with $\pi_t(B) = A$. Hence, $\phi = z_s$.

Theorem 5.7 *Set-based semiring valuation algebras induced by commutative, positive semirings provide null elements.*

A question unanswered up to now concerns the relationship between traditional semiring valuations and set-based semiring valuations. On the one hand, semiring valuations might be seen as special cases of set-based semiring valuations where only singleton configuration sets are allowed to have non-zero values. However, we nevertheless desist from saying that set-based semiring valuations include the family of traditional semiring valuations. The main reason for this is the inconsistency of the definition of neutral elements in both formalisms. This is also underlined by the fact that a semiring valuation algebra requires an idempotent semiring for stability, whereas all set-based semiring valuation algebras with neutral elements are naturally stable. We thus prefer to consider the two semiring related formalisms studied in this chapter as disjoint subfamilies of valuation algebras.

5.9 CONCLUSION

A generic construction identifies a family of valuation algebra instances that share a common structure. On the one hand, this relieves us from verifying the axiomatic system and algebraic properties individually for each member of such a family. On the other hand, it also helps to search for new valuation algebra instances. This chapter introduced two generic constructions related to semirings. Semiring valuations are mappings from configurations to values of a commutative semiring. This allows directly to assimilate the many important formalisms used in soft constraint reasoning into the valuation algebra framework and to immediately conclude that they all qualify for the application of local computation. Typical examples of such formalisms are crisp constraints, weighted constraints, probabilistic constraints, possibilistic constraint or assumption-based constraint, but it also includes other formalisms that are not related to constraint systems such as probability potentials. A closer inspection of semiring valuation algebras in general identified the sufficient properties of a semiring to induce valuation algebras with neutral and null elements or with a division and scaling operator (see appendix). This again discharges us from analysing each formalism individually. The second family of valuation algebras studies in this chapter are set-based semiring valuations, obtained from mapping sets of configurations to semiring values. Its best known member is the formalism of set potentials.

Appendix: Semiring Valuation Algebras with Division

The presence of inverse valuations is of particular interest because they allow the application of specialized local computation architectures. In Appendix D.1 we identified three different conditions for the existence of inverse valuations in general. Namely, these conditions are separativity, regularity and idempotency. Following (Kohlas & Wilson, 2006), we renew these considerations and investigate the requirements for a semiring to induce a valuation algebra with inverse elements. More precisely, it is effectual to identify the semiring properties that either induce separative, regular or idempotent valuation algebras. Then, the theory developed in Section 4.2 can be applied to identify the inverse valuations. We start again with the most general requirement called separativity.

E.1 SEPARATIVE SEMIRING VALUATION ALGEBRAS

According to (Hewitt & Zuckerman, 1956), a commutative semigroup A with operation $\times$ can be embedded into a semigroup which is a union of disjoint groups if, and only if, it is *separative*. This means that for all $a, b \in A$,

$$a \times b \;=\; a \times a \;=\; b \times b \tag{E.1}$$

implies $a = b$. Thus, let $\{G_\alpha : \alpha \in Y\}$ be such a family of disjoint groups with index set Y, whose union

$$G \;=\; \bigcup_{\alpha \in Y} G_\alpha$$

is a semigroup into which the commutative semigroup A is embedded. Hence, there exists an injective mapping $h : A \to G$ such that

$$h(a \times b) \quad = \quad h(a) \times h(b).$$

Note that the left-hand multiplication refers to the semigroup operation in A whereas the right-hand operation stands for the semigroup operation in G. If we identify every semigroup element $a \in A$ with its image $h(a) \in G$, we may assume without loss of generality that $A \subseteq G$.

Every group G_α contains a unit element, which we denote by f_α. These units are idempotent since $f_\alpha \times f_\alpha = f_\alpha$. Let $f \in G$ be an arbitrary idempotent element. Then, f must belong to some group G_α. Consequently, $f \times f = f \times f_\alpha$, which implies that $f = f_\alpha$ due to equation (E.1). Thus, the group units are the only idempotent elements in G.

Next, it is clear that $f_\alpha \times f_\beta$ is also an idempotent element and consequently the unit of some group G_γ, i.e. $f_\alpha \times f_\beta = f_\gamma$. We define $\alpha \leq \beta$ if

$$f_\alpha \times f_\beta \quad = \quad f_\alpha.$$

This relation is clearly reflexive, antisymmetric and transitive, i.e. a partial order between the elements of Y. Now, if $f_\alpha \times f_\beta = f_\gamma$, then it follows that $\gamma \leq \alpha, \beta$. Let $\delta \in Y$ be another lower bound of α and β. We have $f_\alpha \times f_\delta = f_\delta$ and $f_\beta \times f_\delta = f_\delta$. Then, $f_\gamma \times f_\delta = f_\alpha \times f_\beta \times f_\delta = f_\alpha \times f_\delta = f_\delta$. So $\delta \leq \gamma$ and γ is therefore the greatest lower bound of α and β. We write $\gamma = \alpha \wedge \beta$ and hence

$$f_\alpha \times f_\beta \quad = \quad f_{\alpha \wedge \beta}.$$

To sum it up, Y forms a *semilattice*, a partially ordered set where the infimum exists between any pair of elements.

Subsequently, we denote the inverse of a group element $a \in G_\alpha$ by a^{-1}. Suppose $a, a^{-1} \in G_\alpha$ and $b, b^{-1} \in G_\beta$, then $(a \times b) \times (a^{-1} \times b^{-1}) = f_\alpha \times f_\beta = f_{\alpha \wedge \beta}$, and it follows that

$$(a \times b)^{-1} \quad = \quad a^{-1} \times b^{-1}.$$

Suppose now $a \times b \in G_\gamma$. Thus $(a \times b)^{-1} \in G_\gamma$ and $(a \times b) \times (a \times b)^{-1} = f_\gamma$. But as we have just seen $f_\gamma = f_{\alpha \wedge \beta}$, hence $\gamma = \alpha \wedge \beta$ and $a \times b, a^{-1} \times b^{-1} \in G_{\alpha \wedge \beta}$.

We next introduce an equivalence relation between semigroup elements of A and say that $a \equiv b$ if a and b belong to the same group G_α. This is a congruence relation with respect to $\times$, since $a \equiv a'$ and $b \equiv b'$ imply that $a \times b \equiv a' \times b'$, which in turn implies that the congruence classes are semigroups. Consequently, A decomposes into a family of disjoint semigroups,

$$A \quad = \quad \bigcup_{a \in A} [a].$$

Also, the partial order of Y carries over to equivalence classes by defining $[a] \leq [b]$ if, and only if, $[a \times b] = [a]$. Further, we have $[a \times b] = [a] \wedge [b]$ for all $a, b \in A$ which shows that the semigroups $[a]$ form a semilattice, isomorph to Y. We call the equivalence class $[a]$ the *support* of a. Observe also that $[\mathbf{0}] = \{\mathbf{0}\}$. This holds because if $a \in [\mathbf{0}] = G_\gamma$ then $a \equiv \mathbf{0}$ and $f_\gamma \equiv \mathbf{0}$. Hence, $a = a \times f_\gamma = \mathbf{0} \times \mathbf{0} = \mathbf{0}$.

The result of the support decomposition of A is summarized in Figure E.1. The following definition given in (Kohlas & Wilson, 2006) is crucial since it summarizes all requirements for a semiring to give rise to a separative valuation algebra. Together with the requirement of a separative semigroup, it demands a decomposition that is monotonic under addition. This is needed to make allowance for the projection in equation (D.2), as shown beneath.

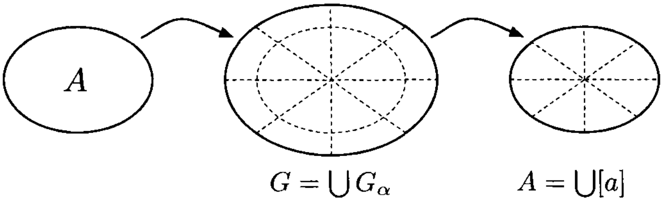


Figure E.1 A separative semigroup A is embedded into a semigroup G that consists of disjoint groups G_α. The support decomposition of A is then derived by defining two semigroup elements as equivalent if they belong to the same group G_α.

Definition E.4 *A semiring* $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ *is called* separative, *if*

- *its multiplicative semigroup is separative,*
- *there is an embedding into a union of groups such that for all* $a, b \in A$

$$[a] \quad \leq \quad [a + b]. \qquad \text{(E.2)}$$

The second condition expresses a kind of strengthening of positivity. This is the statement of the following lemma.

Lemma E.7 *A separative semiring is positive.*

Proof: From equation (E.2) we conclude $[\mathbf{0}] \leq [a]$ for all $a \in A$. Then, assume $a + b = \mathbf{0}$. Hence, $[\mathbf{0}] \leq [a], [b] \leq [a + b] = [\mathbf{0}]$ and consequently, $a = b = \mathbf{0}$. ■

The following theorem states that a separative semiring is sufficient to guarantee that the induced valuation algebra is separative in the sense of Definition D.4.

Theorem E.8 *Let* $\langle \Phi, D \rangle$ *be a valuation algebra induced by a separative semiring. Then* $\langle \Phi, D \rangle$ *is separative.*

Proof: Since the semiring is separative, the same holds for the combination semigroup of Φ, i.e. $\phi\otimes\psi = \phi\otimes\phi = \psi\otimes\psi$ implies $\phi = \psi$. Consequently, this semigroup can also be embedded into a semigroup that is a union of disjoint groups. In fact, the decomposition which is interesting for our purposes is the one induced by the decomposition of the underlying semiring. We say that $\phi \equiv \psi$, if

- $d(\phi) = d(\psi) = s$, and
- $\phi(\mathbf{x}) \equiv \psi(\mathbf{x})$ for all $\mathbf{x} \in \Omega_s$.

This is clearly an equivalence relation in Φ. Let $\phi, \psi, \eta \in \Phi$ and $\psi \equiv \eta$ with $d(\phi) = s$ and $d(\psi) = d(\eta) = t$. Then, it follows that $d(\phi \otimes \psi) = d(\phi \otimes \eta) = s \cup t$ and for all $\mathbf{x} \in \Omega_{s\cup t}$ we have

$$\phi(\mathbf{x}^{\downarrow s}) \times \psi(\mathbf{x}^{\downarrow t}) \quad \equiv \quad \phi(\mathbf{x}^{\downarrow s}) \times \eta(\mathbf{x}^{\downarrow t}).$$

We therefore have a combination congruence in Φ. It then follows that the equivalence classes $[\phi]$ are subsemigroups of the combination semigroup of Φ.

Next, we define for a valuation $\phi \in \Phi_s$ the mapping $sp_{[\phi]} : \Omega_s \rightarrow Y$ by

$$sp_{[\phi]}(\mathbf{x}) = \alpha \quad \text{if} \quad \phi(\mathbf{x}) \in G_\alpha,$$

where Y is the semilattice of the group decomposition of the separative semiring. This mapping is well-defined since $sp_{[\phi]} = sp_{[\psi]}$ if $[\phi] = [\psi]$. We define for a valuation ϕ with $d(\phi) = s$

$$G_{[\phi]} \quad = \quad \{g : \Omega_s \rightarrow G : \forall \mathbf{x} \in \Omega_s,\ g(\mathbf{x}) \in G_{sp_{[\phi]}(\mathbf{x})}\}.$$

It follows that $G_{[\phi]}$ is a group, if we define $g \times f$ by $g \times f(\mathbf{x}) = g(\mathbf{x}) \times f(\mathbf{x})$, and the semigroup $[\phi]$ is embedded in it. The unit element $f_{[\phi]}$ of $G_{[\phi]}$ is given by $f_{[\phi]}(\mathbf{x}) = f_{sp_{[\phi]}(\mathbf{x})}$ and the inverse of ϕ is defined by $\phi^{-1}(\mathbf{x}) = (\phi(\mathbf{x}))^{-1}$. This induces again a partial order $[\phi] \leq [\psi]$ if $f_{[\phi]}(\mathbf{x}) \leq f_{[\psi]}(\mathbf{x})$ for all $\mathbf{x} \in \Omega_s$, or if $[\phi \otimes \psi] = [\phi]$. It is even a semilattice with $f_{[\phi\otimes\psi]} = f_{[\phi]} \wedge f_{[\psi]}$.

The union of these groups

$$G^* \quad = \quad \bigcup_{\phi\in\Phi} G_{[\phi]}$$

is a commutative semigroup because, if $g_1 \in G_{[\phi]}$ and $g_2 \in G_{[\psi]}$, then $g_1 \otimes g_2$ is defined for $\mathbf{x} \in \Omega_{s\cup t}$, $d(\phi) = s$ and $d(\psi) = t$ by

$$g_1 \otimes g_2(\mathbf{x}) \quad = \quad g_1(\mathbf{x}^{\downarrow s}) \times g_2(\mathbf{x}^{\downarrow t})$$

and belongs to $G_{[\phi\otimes\psi]}$ and is commutative as well as associative.

We have the equivalence $\phi \otimes \phi \equiv \phi$ because $[\phi]$ is closed under combination. From equation (E.2) it follows that for $t \subseteq d(\phi)$ and all $\mathbf{x} \in \Omega_s$,

$$[\phi(\mathbf{x})] \quad \leq \quad [\phi^{\downarrow t}(\mathbf{x}^{\downarrow t})].$$

This means that $[\phi] \leq [\phi^{\downarrow t}]$ or also

$$\phi^{\downarrow t} \otimes \phi \quad \equiv \quad \phi.$$

We thus derived the second requirement for a separative valuation algebra given in Definition D.4 for the congruence induced by the separative semiring. It remains to show that the cancellativity property holds in every equivalence class $[\phi]$. Thus, assume $\psi, \psi' \in [\phi]$, $d(\phi) = s$ and for all $\mathbf{x} \in \Omega_s$

$$\phi(\mathbf{x}) \times \psi(\mathbf{x}) \quad = \quad \phi(\mathbf{x}) \times \psi'(\mathbf{x}).$$

Since all elements $\phi(\mathbf{x}), \psi(\mathbf{x})$ and $\psi'(\mathbf{x})$ are contained in the same group, it follows that $\psi(\mathbf{x}) = \psi'(\mathbf{x})$ by multiplication with $\phi(\mathbf{x})^{-1}$. Therefore, $\psi = \psi'$ which proves cancellativity of $[\phi]$. ■

Let us consider an example of a separative semiring and its induced valuation algebra. Subsequently, we then show that although separativity is again the weakest condition to allow for a division operation in the induced valuation algebra, there are still formalisms that do not even possess this structure.

■ E.11 Arithmetic Potentials and Separativity

The particular arithmetic semiring of non-negative integers $\mathbb{N} \cup \{0\}$ is separative and decomposes into the semigroups $\{0\}$ and $\mathbb{N}$. The first is already a trivial group, whereas $\mathbb{N}$ is embedded into the group of positive rational numbers. Note also that $[0 \times a] = [0]$, hence $[0] \leq [a]$ and $[0] \leq [0 + a] \leq [a]$. So, equation (E.2) holds. Thus, it induces a particular subalgebra of the valuation algebra of arithmetic potentials that is separative.

■ E.12 Possibility Potentials and Separativity

The Lukasiewicz and drastic product t-norm semirings of Example 5.8 are not separative. Consequently, we cannot deduce anything about a possible division operation in their induced valuation algebras. This, however, does not hold for all t-norm semirings as shown in the following section.

E.2 REGULAR SEMIRING VALUATION ALGEBRAS

In the previous case, we exploited the fact that the multiplicative semigroup of a separative semiring can be embedded into a semigroup consisting of a union of disjoint groups. This allows introduction of a particular equivalence relation between semiring elements which in turn leads to a decomposition of the induced valuation algebra into cancellative semigroups with the corresponding congruence relation satisfying the requirement of a separative valuation algebra. In this section, we start with a semiring whose multiplicative semigroup decomposes directly into a union of groups. The mathematical requirement is captured by the following definition.

Definition E.5 *A semigroup A with an operation $\times$ is called* regular *if for all $a \in A$ there is an element $b \in A$ such that*

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

Appendix D.1.2 introduced the Green relation in the context of a regular valuation algebra and we learned that the corresponding congruence classes are directly groups. This technique can be generalized to regular semigroups. We define

$$a \equiv b \quad \text{if, and only if,} \quad a \times A = b \times A,$$

and obtain a decomposition of A into disjoint congruence classes,

$$A \quad = \quad \bigcup_{a \in A} [a].$$

These classes are commutative groups under $\times$, where every element a has a unique inverse denoted by a^{-1}. Further, we write $f_{[a]}$ for the identity element in the group $[a]$ and take up the partial order defined by $f_{[a]} \leq f_{[b]}$ if, and only if, $f_{[a]} \times f_{[b]} = f_{[a]}$. This is again a semilattice as we know from Section E.1. Finally, we obtain an induced partial order between the decomposition groups by $[a] \leq [b]$ if, and only if, $f_{[a]} \leq f_{[b]}$. This is summarized in Figure E.2.

Definition E.6 *A semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ is called* regular *if*

- *its multiplicative semigroup is regular,*
- *for all $a, b \in A$, $[a] \leq [a + b]$.*

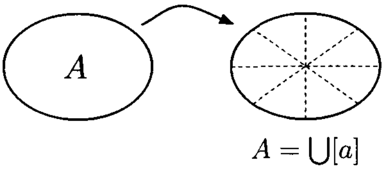


Figure E.2 A regular semigroup A decomposes directly into a union of disjoint groups using the Green relation.

Theorem E.9 *Let $\langle \Phi, D \rangle$ be a valuation algebra induced by a regular semiring. Then $\langle \Phi, D \rangle$ is regular.*

Proof: Suppose $\phi \in \Phi$ with $t \subseteq s = d(\phi)$ and $\mathbf{y} \in \Omega_t$. We define

$$\chi(\mathbf{y}) \quad = \quad (\phi^{\downarrow t}(\mathbf{y}))^{-1}.$$

Then, it follows that for any $\mathbf{x} \in \Omega_s$

$$\begin{aligned}
(\phi \otimes \phi^{\downarrow t} \otimes \chi)(\mathbf{x}) &= \phi(\mathbf{x}) \times \phi^{\downarrow t}(\mathbf{x}^{\downarrow t}) \times \chi(\mathbf{x}^{\downarrow t}) \\
&= \phi(\mathbf{x}) \times \phi^{\downarrow t}(\mathbf{x}^{\downarrow t}) \times (\phi^{\downarrow t}(\mathbf{x}^{\downarrow t}))^{-1} \\
&= \phi(\mathbf{x}) \times f_{\phi^{\downarrow t}}.
\end{aligned}$$

Here, the abbreviations f_ϕ and $f_{\phi^{\downarrow t}}$ are used for $f_{[\phi(\mathbf{x})]}$ and $f_{[\phi^{\downarrow t}(\mathbf{x}^{\downarrow t})]}$ respectively. Since the semiring is regular, we have $[\phi(\mathbf{x})] \leq [\phi^{\downarrow t}(\mathbf{x}^{\downarrow t})]$ and $f_\phi \leq f_{\phi^{\downarrow t}}$. Thus

$$\begin{aligned}
\phi(\mathbf{x}) \times f_{\phi^{\downarrow t}} &= (\phi(\mathbf{x}) \times f_\phi) \times f_{\phi^{\downarrow t}} \\
&= \phi(\mathbf{x}) \times (f_\phi \times f_{\phi^{\downarrow t}}) \\
&= \phi(\mathbf{x}) \times f_\phi \;=\; \phi(\mathbf{x}).
\end{aligned}$$

This proves the requirement of Definition D.5. ■

We remark that regular semirings are also separative. Regularity of the multiplicative semigroup implies that every element in A has an inverse and we derive from $a \times a = a \times b = b \times b$ that a and b are contained in the same group of the semiring decomposition, i.e. $[a] = [b]$. Multiplying with the inverse of a gives then $a = f_{[a]} \times b = b$. This proves that the multiplicative semigroup of a regular semiring is separative. Requirement (E.2) follows from the strengthening of positivity in Definition E.6.

Let us again consider some concrete examples of regular semirings and their induced regular valuation algebra. As a confirmation of what we found out in Instance D.8, we first show that the arithmetic semiring is regular.

■ E.13 Arithmetic Potentials and Regularity

The arithmetic semiring of non-negative real numbers is regular. It decomposes into a union of two disjoint groups $\mathbb{R}_{>0}$ and $\{0\}$, and we clearly have $[0] \leq [a]$ for all semiring elements a. Applying Theorem E.9 we conclude that its induced valuation algebra called arithmetic potentials is regular. However, we have also seen in Instance E.11 that this property changes if we define the arithmetic semiring over different sets of numbers. On this note, the semiring view affords a whole family of valuation algebra instances that are closely related to arithmetic potentials but differ in their properties.

■ E.14 Probabilistic Constraints and Regularity

The t-norm semiring with usual multiplication as t-norm is regular too as we may deduce from the foregoing example. Therefore, its induced valuation algebra called probabilistic constraints is also regular. In fact, it is the only example of the t-norms listed in Example 5.8 that leads to a valuation algebra with a non-trivial division. The two t-norm discussed in Instance E.12 do not allow division at all, and the minimum t-norm induces an idempotent valuation algebra as shown in Instance E.19 below.

■ E.15 Spohn Potentials and Regularity

The tropical semiring $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0\rangle$ is not regular because the solution $b = -a$ of the regularity equation in Definition E.5 is not contained in the semiring. But if we extend the tropical semiring to all integers, then semiring is regular. The induced regular valuations of this extended semiring are called *quasi-Spohn potentials.*

E.3 CANCELLATIVE SEMIRING VALUATION ALGEBRAS

In Appendix E.1, we started from a separative semigroup, which can be embedded into a semigroup consisting of a union of disjoint groups. Here, we discuss a special case in which the semigroup is embedded into a single group. The required property for the application of this technique is cancellativity. A semigroup A with operation $\times$ is called cancellative, if for $a, b, c \in A$,

$$a \times b \quad = \quad a \times c$$

implies $b = c$. Such a semigroup can be embedded into a group G by application of essentially the same technique as in Appendix D.1.1. We consider pairs (a, b) of elements $a, b \in A$ and define equality:

$$(a, b) = (c, d) \quad \text{if} \quad a \times d = b \times c.$$

Multiplication between pairs of semigroup elements is defined component-wise,

$$(a, b) \times (c, d) \quad = \quad (a \times c, b \times d).$$

This operation is well-defined, since $(a, b) = (a', b')$ and $(c, d) = (c', d')$ implies $(a, b) \times (c, d) = (a', b') \times (c', d')$. It is furthermore associative, commutative and the multiplicative unit e is given by the pairs (a, a) for all $a \in A$. Note that all these pairs are equal with respect to the above definition. We further have

$$(a, b) \times (b, a) \quad = \quad (a \times b, a \times b),$$

which shows that (a, b) and (b, a) are inverses. Consequently, the set G of pairs (a, b) is a group into which A is embedded by the mapping $a \mapsto (a \times a, a)$. If A itself has a unit, then $1 \mapsto (1, 1) = e$. Without loss of generality, we may therefore consider A as a subset of G. This is summarized in Figure E.3.

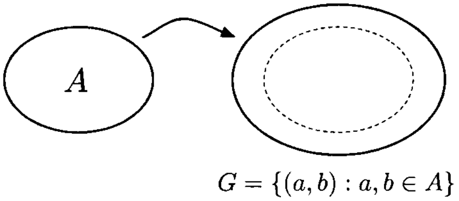


Figure E.3 A cancellative semigroup A is embedded into a group G consisting of pairs of elements from A.

Let us return to semirings and show how cancellative semirings induce valuation algebras with inverse elements.

Definition E.7 *A semiring* $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *is called* cancellative *if its multiplicative semigroup is cancellative.*

A cancellative semiring is also separative since from $a \times a = a \times b$ it follows that $a = b$. We know that cancellative semirings can be embedded into a single group consisting of pairs of semiring elements, and since we have only one group, (E.2) is trivially fulfilled. Thus, cancellative semirings represent a particularly simple case of separative semirings and consequently induce separative valuation algebras due to Appendix E.1. We also point out that no direct relationship between cancellative and regular semirings exist. Cancellative semirings are embedded into a single group whereas regular semirings decompose into a union of groups. These considerations prove the following theorem:

Theorem E.10 *Cancellative semirings induce separative valuation algebras.*

We now examine an important example of a separative valuation algebra that is induced by a cancellative semiring.

■ E.16 Weighted Constraints and Cancellativity

The tropical semiring of Example 5.4 is cancellative. Since the semiring values are non-negative and multiplication corresponds to addition, we always have that $a \times b = a \times c$ implies $b = c$. To any pair of numbers $a, b \in A$ we assign the difference $a - b$, which is not necessarily in the semiring anymore. In other words, the tropical semiring is embedded into the additive group of all integers.

E.4 IDEMPOTENT SEMIRING VALUATION ALGEBRAS

Section 4.2.1 and Appendix D.1.3 introduced idempotency as a very strong additional condition. If this property is present in a valuation algebra, then every valuation becomes the inverse of itself, which allows us to simplify inference algorithms considerably. This lead to the very simple and time-efficient idempotent architecture of Section 4.5. Thus, we next go about studying which semiring properties give rise to idempotent valuation algebras.

Theorem E.11 *c-semirings with idempotent multiplication induce idempotent valuation algebras.*

Proof: From Lemma 5.6, (SP6) we conclude that

$$\phi(\mathbf{x}) \times \phi^{\downarrow t}(\mathbf{x}^{\downarrow t}) \quad \leq \quad \phi(\mathbf{x}).$$

On the other hand, we have by Lemma 5.5, (SP4) and idempotency

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

This holds because $\phi(\mathbf{x})$ is a term of the sum. We therefore have $\phi(\mathbf{x}) \times \phi^{\downarrow t}(\mathbf{x}^{\downarrow t}) = \phi(\mathbf{x})$ which proves idempotency. ■

Idempotency of a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ directly implies the regularity of its multiplicative semigroup. Additionally, all groups $[a]$ consist of a unique element as we learned in Appendix D.1.3. Since for all $a, b \in A$ we have

$$\begin{aligned} a \times (a+b) \;&=\; (a \times a) + (a \times b) \;=\; a + (a \times b) \\ &=\; a \times (1+b) \;=\; a \times 1 \;=\; a. \end{aligned}$$

This implies $[a] \leq [a+b]$, from which we conclude that idempotent semirings are regular. Moreover, Lemma 5.5, (SP5) states that idempotent semirings are also positive which ensures the existence of neutral elements, null elements and stability in the induced valuation algebra as shown in Section 5.5.1 and 5.5.3. Altogether, this proves the following lemma:

Lemma E.8 *c-semirings with idempotent* $\times$ *induce information algebras.*

Theorem E.11 states that a c-semiring with idempotent multiplication is a sufficient condition for an idempotent semiring valuation algebra. It will next be shown that it is also necessary, if two additional conditions hold. In order to make a statement about all semiring elements starting from a semiring valuation algebra, all semiring elements must occur in the valuation algebra. In other words, Φ must be the set of all possible valuations taking values from a given semiring, whereas in other contexts it is often sufficient that Φ is only closed under combination and projection. The second condition excludes trivial lattices where the operation of projection has no effect.

Theorem E.12 *Let* $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *be a semiring,* r *a set of variables and* Φ *the set of all possible semiring valuations defined over* A *and* r. *If* $\langle \Phi, \mathcal{P}(r)\rangle$ *is idempotent, then the semiring has idempotent multiplication. Moreover, if* r *contains at least one variable* $X \in r$ *with* $|\Omega_X| > 1$, *then the semiring is also a c-semiring.*

Proof: Since Φ is the set of all possible valuations, there exists a for all $a \in A$ a valuation $\phi \in \Phi$ with $d(\phi) = \emptyset$ and $\phi(\diamond) = a$. Idempotency implies that

$$\phi(\diamond) = \phi \otimes \phi(\diamond) = \phi(\diamond) \times \phi(\diamond).$$

We therefore have $a \times a = a$ for all $a \in A$ with proves idempotency of semiring multiplication. Next, assume that there exists $X \in r$ with $\Omega_X = \{x_1, x_2, \ldots, x_n\}$. For all $a \in A$, we then have a valuation $\phi \in \Phi$ with $d(\phi) = \{X\}$ and $\phi(x_1) = \mathbf{1}$, $\phi(x_2) = a$ and $\phi(x_i) = \mathbf{0}$ for all $i > 2$. Idempotency implies that

$$\mathbf{1} = \phi(x_1) = \phi(x_1) \times \phi^{\downarrow\emptyset}(\diamond) = \phi(x_1) \times (\phi(x_1) + \phi(x_2)) = \mathbf{1} \times (\mathbf{1} + a) = \mathbf{1} + a.$$

This shows that the semiring is a c-semiring. ■

Let us again consider some examples of c-semirings with idempotent multiplication and their induced information algebras. First, we apply this analysis to the Boolean semiring to confirm what we have found out in Instance 4.2.

■ E.17 Indicator Functions and Idempotency

It has been shown in Example 5.10 that the Boolean semiring is a c-semiring. Since multiplication corresponds to minimization, this semiring is also idempotent. We therefore conclude from Lemma E.8, the induced valuation algebra of indicator functions must be an information algebra.

■ E.18 Bottleneck Constraints and Idempotency

The bottleneck semiring of Example 5.3 is clearly a c-semiring with idempotent multiplication. Its induced valuation algebra is therefore another example of an information algebra.

■ E.19 Possibility Potentials and Idempotency

We already pointed out in Example 5.10 that all t-norm semirings are c-semirings. However, only the minimum t-norm among them is idempotent which again turns its induced valuation algebra into an information algebra. This example is very similar to the Bottleneck constraints above.

A summary of the division-related properties of different semiring valuation algebras is shown in Figure E.4.

E.5 SCALABLE SEMIRING VALUATION ALGEBRAS

As a last complement on division for semiring valuation algebras, we recall from Section 4.7 that scaling or normalization may be an important semantical issue

	Instance	Separative	Cancellative	Regular	Idempotent
1.1	Indicator Functions	√	√	√	√
1.3	Arithmetic Potentials on $\mathbb{R}_{\geq 0}$	√	○	√	○
1.3	Arithmetic Potentials on $\mathbb{N}_{\geq 0}$	√	○	○	○
5.1	Weighted Constraints	√	√	○	○
5.2	Probabilistic Constraints	√	○	√	○
5.2	Lukasiewicz Constraints	○	○	○	○
5.3	Set-based Constraints	√	√	√	√
E.15	Quasi-Spohn Potentials	√	○	√	○

Figure E.4 Semiring valuation algebras and division-related properties.

for some valuation algebras. The algebraic requirement that makes scaling possible is separativity. Thus, we may conclude that separative semirings induce valuation algebra with scaling if there exists a semantical need for this operation. Among the instances discussed in this appendix, we have seen that the arithmetic semiring is regular and allows scaling in its induced valuation algebra. This leads again to the probability potentials introduced in Instance 4.4. Here are two further examples of scalable valuation algebras:

■ E.20 Normalized Weighted Constraints

Weighted constraints are induced by the cancellative tropical semiring and are therefore separative. According to equation (4.29), the scale of a weighted constraint $c \in \Phi_s$ is given by

$$c^{\downarrow}(\mathbf{x}) = c(\mathbf{x}) + \left(c^{\downarrow\emptyset}\right)^{-1}(\diamond) = c(\mathbf{x}) - \min_{\mathbf{y}\in\Omega_s} c(\mathbf{y}).$$

This operation ensures that the minimum weight is always zero.

■ E.21 Normalized Probabilistic Constraints

The product t-norm semiring induces probabilistic constraints which naturally explains the interest in scaling. This semiring is regular and therefore also fulfills the algebraic requirement. According to equation (4.29), the scale of the probabilistic potential $p \in \Phi_s$ is given by

$$p^{\downarrow}(\mathbf{x}) \;=\; p(\mathbf{x}) \cdot \left(p^{\downarrow\emptyset}\right)^{-1}(\diamond) \;=\; \frac{p(\mathbf{x})}{\max_{\mathbf{y}\in\Omega_s} p(\mathbf{y})}.$$

PROBLEM SETS AND EXERCISES

E.1 ★ Section 5.6 shows that the space complexity of the Shenoy-Shafer architecture of equation (4.9) can be improved when dealing with semiring valuations. Think about a similar improvement for set-based semiring valuation algebras.

E.2 ★ Consider mappings from singleton configuration sets to the values of a commutative semiring and prove that they form a subalgebra of the valuation algebra of set-based semiring valuations. Identify the requirements for neutral and null elements and observe that neutral elements are different in the two algebras. Compare the results with usual semiring valuations.

E.3 ★ Exercise D.4 in Chapter 4 introduced a partial order between valuations in an information algebra. We-know from Lemma E.8 that c-semirings with idempotent × induce information algebras. How is the semiring order of equation (5.4) related to the partial order between valuations in the induced information algebra?

E.4 ★ We claim in Example 5.8 and 5.10 that $\langle [0,1], \max, \times, 0, 1\rangle$ always forms a c-semiring when an arbitrary t-norm is taken for multiplication. Prove this statement. The definition of a t-norm requires that the number 1 is the unit element. Without this condition the structure it is called a *uninorm*. Prove that we still obtain a semiring, albeit not a c-semiring, when a uninorm is taken instead of a t-norm. Finally, show that we may also replace maximization for semiring addition by an arbitrary uninorm.

E.5 ★★ Exercise D.8 in Chapter 4 introduced conditionals $\phi_{s|t}$ for disjoint sets $s, t \subseteq d(\phi)$ in regular valuation algebras $\langle \Phi, D\rangle$.

a) Identify the conditionals in the valuation algebras of probabilistic constraints from Instance 5.2 and quasi-Spohn potentials of Instance E.15.

b) Prove that if $\phi \in \Phi$ and $s, t, u \subseteq d(\phi)$ are disjoint sets we have

$$\phi^{\downarrow s \cup t \cup u} \;=\; \phi_{s|u} \otimes \phi_{t|u} \otimes \phi^{\downarrow t}. \qquad \text{(E.3)}$$

c) Convince yourself that the arctic semiring is regular and therefore also the induced valuation algebra of GAI preferences from Instance 5.1. Show that equation (E.3) corresponds to *Bellman's principle of optimality* (Bellman, 1957) when it is expressed in the valuation algebra of GAI preferences.

The solution to these exercises is given in Section 5.2 of (Kohlas, 2003).

E.6 ★★ Identify the algebraic properties (i.e. division, neutral and null elements) for the valuation algebras induced by the t-norm semirings with the following t-norms:

1. *Nil-potent maximum*: For $a, b \in [0,1]$

$$T(a,b) \quad = \quad \begin{cases} \min\{a,b\} & \text{a + b > 1,} \\ 0 & \text{otherwise.} \end{cases}$$

2. *Hamacher product*: For $a, b \in [0,1]$

$$T(a,b) \quad = \quad \begin{cases} 0 & \text{a = b = 0,} \\ \frac{ab}{a+b-ab} & \text{otherwise.} \end{cases}$$

E.7 ★★★ Exercise E.4 defines a uninorm as a generalization of a t-norm where the unit element does not necessarily correspond to the number 1. If we have a uninorm with the number 0 as unit element, then it is called a *t-conorm*. We directly conclude from Exercise E.4 that we obtain a semiring when we take a t-conorm for multiplication and maximization for addition. Give a t-norm T and $a, b \in [0,1]$, we define its associated t-conorm by

$$C(a,b) \quad = \quad 1 - T(1-a, 1-b). \tag{E.4}$$

Is it possible to conclude the form of division in a valuation algebra induced by a t-conorm semiring from the form of division that is present in the valuation algebra induced by the corresponding t-norm semiring?

E.8 ★★★ Instance D.6 shows that the valuation algebra of set potentials is separative. Generalize these considerations to set-based semiring valuations and identify the algebraic requirements for a semiring to induce separative, regular and idempotent set-based semiring valuation algebras.

CHAPTER 6

VALUATION ALGEBRAS FOR PATH PROBLEMS

Together with semiring constraint systems introduced in Chapter 5, the solution of path problems in graphs surely ranks among the most popular and important application fields of semiring theory in computer science. In fact, the history of research in path problems strongly resembles that of valuation algebras and local computation in its search for genericity and generality. In the beginning, people studied specific path problems which quickly brought out a large number of seemingly different algorithms. Based on this research, it was then observed that path problems often provide a common algebraic structure. This gave birth of an abstract framework called *algebraic path problem* which unifies the formerly isolated tasks. Typical instances of this framework include the computation of shortest distances, maximum capacities or reliabilities, but also other problems that are not directly related to graphs such as partial differentiation or the determination of the language accepted by a finite automaton. This common algebraic viewpoint initiated the development of generic procedures for the solution of the algebraic path problem, similar to our strategy of defining generic local computation architectures for the solution of inference problems. Since the algebraic path problem is not limited to graph related applications, its definition is based on a matrix of semiring values instead. We will show in this chapter that depending on the semiring properties, such matrices induce valuation

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

algebras. This has several important consequences: on the one hand, we then know that matrices over special families of semirings provide further generic constructions that deliver many new valuation algebra instances. These instances are very different from the semiring valuation algebras of Chapter 5, because they are subject to a pure polynomial time and space complexity. On the other hand, we will see in Chapter 9 that inference problems over such valuation algebras model path problems that directly enables their solution by local computation. This comes along with many existing approaches that focus on solving path problems by *sparse matrix techniques*.

The first section introduces the algebraic path problem from a topological perspective and later points out an alternative description as a particular solution to a fixpoint equation, which generally is more adequate for an algebraic manipulation. Section 6.3 then discusses quasi-regular semirings that always guarantee the existence of at least one solution to this equation. Moreover, it will be shown that a particular quasi-inverse matrix can be computed from the quasi-inverses of the underlying semiring. This leads in Section 6.4 to the first generic construction related to the solution of path problems. Since quasi-regular semirings are very general and only possess little algebraic structure, it will be remarked in Section 6.5 that the same holds for their induced valuation algebras. We therefore present in Section 6.6 a special family of quasi-regular semirings called Kleene algebras that uncover a second generic construction related to path problems in Section 6.7. This approach always leads to idempotent valuation algebras and also distinguishes itself in other important aspects from quasi-regular valuation algebras, as pointed out in Section 6.8.

6.1 SOME PATH PROBLEM EXAMPLES

The algebraic path problem establishes the connection between semiring theory and path problems. To motivate a closer investigation of this combination, let us first look at some typical examples of path problems.

■ 6.1 The Connectivity Problem

The graph of Figure 6.1 represents a traffic network where nodes model junctions and arrows one-way streets. Every street is labeled with a Boolean value where 1 stands for an accessible street and 0 for a blocked street due to construction works. The *connectivity problem* now asks the question whether node T can still be reached from node S. This is computed by taking the maximum value over all possible paths leading from S to T, where the value of a path corresponds to the minimum of all its edge values. We compute for the three possible paths from S to T:

$$\begin{aligned}\min\{0,1\} &= 0\\ \min\{1,1,1\} &= 1\\ \min\{1,1,0,1\} &= 0\end{aligned}$$

and then

$$\max\{0,1,0\} \quad = \quad 1.$$

Thus, node T can still be reached from node S.

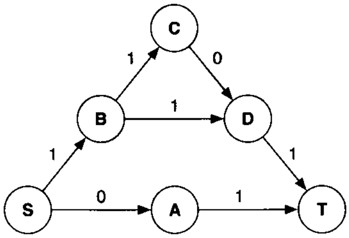


Figure 6.1 The connectivity path problem.

6.2 The Shortest and Longest Distance Problem

A similar traffic network is shown in Figure 6.2. This time, however, edges are labeled with natural numbers to express the distance between neighboring nodes. The *shortest distance problem* then asks for the minimum distance between node S and node T, which corresponds to the minimum value of all possible paths leading from S to T, where the value of a path consists of the sum of all its edge values. We compute

$$\begin{aligned} 9+4 &= 13 \\ 1+6+5 &= 12 \\ 1+2+3+5 &= 11 \end{aligned}$$

and then

$$\min\{13,12,11\} \quad = \quad 11.$$

Thus, the shortest distance from node S to node T is 11. We could also replace minimization by maximization to determine the longest or most critical path.

6.3 The Maximum Capacity Problem

Let us next interpret Figure 6.3 as a communication network where edges represent communication channels between network hosts. Every channel is labeled with its capacity. The *maximum capacity problem* requires to compute the maximum capacity of the communication path between node S and node T, where the capacity of a communication path is determined by the minimum

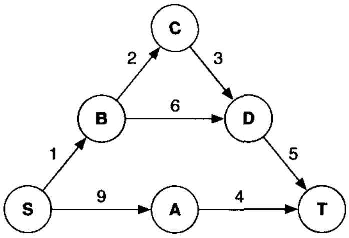


Figure 6.2 The shortest distance problem.

capacity over all channels in this path. Similar to Instance 6.1 we compute

$$\begin{aligned} \min\{3.4, 4.5\} &= 3.4 \\ \min\{3.6, 5.5, 5.1\} &= 3.6 \\ \min\{3.6, 4.2, 3.5, 5.1\} &= 3.5 \end{aligned}$$

and then

$$\max\{3.4, 3.6, 3.5\} = 3.6.$$

Thus, the maximum capacity over all channels connecting S and T is 3.6.

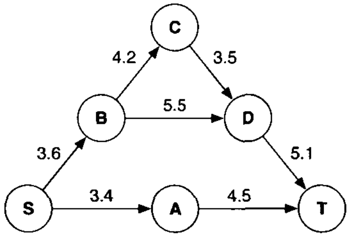


Figure 6.3 The maximum capacity problem.

6.4 The Maximum Reliability Problem

We again consider Figure 6.4 as a communication network where edges represent communication channels. Such channels may fail and are therefore labeled with a probability value to express their probability of availability. Thus, we are interested in the most reliable communication path between node S and node T, where the reliability of a path is obtained by multiplying all its channel probabilities. We compute

$$\begin{aligned} 0.4 \cdot 0.8 &= 0.32 \\ 0.9 \cdot 0.2 \cdot 0.7 &= 0.126 \\ 0.9 \cdot 0.9 \cdot 1.0 \cdot 0.7 &= 0.567 \end{aligned}$$

and then

$$\max\{0.32, 0.126, 0.567\} = 0.567.$$

Thus, the most reliable communication path has a reliability of 0.567.

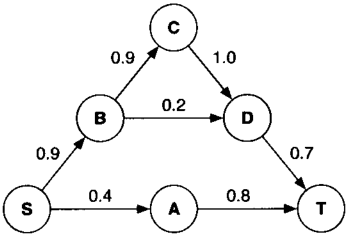


Figure 6.4 The maximum reliability problem.

6.5 The Language of an Automaton

Finally, we consider the automaton of Figure 6.5 where nodes represent states, and the edge values input words that bring the automaton into the neighbor state. Here, the task consists in determining the language that brings the automaton from state S to T. This is achieved by collecting the strings of each path from S to T, which are obtained by concatenating their edge values. We compute

$$\begin{aligned} \{a\} \cdot \{c\} &= \{ac\} \\ \{a\} \cdot \{b\} \cdot \{a\} &= \{aba\} \\ \{a\} \cdot \{c\} \cdot \{b\} \cdot \{a\} &= \{acba\} \end{aligned}$$

and then

$$\bigcup\{\{ac\}, \{aba\}, \{acba\}\} = \{ac, aba, acba\}.$$

Thus, the language that brings the automaton from state S to T is $\{ac, aba, acba\}$.

Further examples of path problems will be given at the end of this chapter.

6.2 THE ALGEBRAIC PATH PROBLEM

We next introduce a common formal background of these well-known path problems: Consider a *weighted, directed graph* (V, E, A, w) where (V, E) is a directed graph and w a weight function that assigns a value from an arbitrary semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ to each edge of the graph, $w : E \rightarrow A$. If $p = (V_0, \dots, V_n)$ denotes a path of length

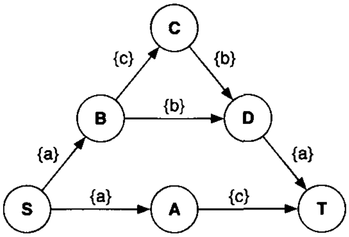


Figure 6.5 Determining the language of an automaton.

n between a source node $S = V_0$ and a target node $T = V_n$, the *path weight* $w(p)$ is defined as the product of all semiring values that are assigned to its edges,

$$w(p) \quad = \quad w(V_0, V_1) \times w(V_1, V_2) \times \cdots \times w(V_{n-1}, V_n). \tag{6.1}$$

Since we later identify graph nodes with variables, we here use capital letters to refer to graph nodes. It is important to note that we assumed an arbitrary semiring where multiplication is not necessarily commutative, which forces us to mind the order of multiplication. Also, many paths between S and T may exist. We write $P_{S,T}$ for the set of all possible paths connecting a source S with a terminal T. Solving the path problem for $S \neq T$ then consists in computing

$$\mathbf{D}(S,T) \quad = \quad \bigoplus_{p \in P_{S,T}} w(p). \tag{6.2}$$

If $S = T$ we simply define $\mathbf{D}(S,T) = \mathbf{1}$ as the weight of the empty path. This is the unit element of the semiring with respect to multiplication and ensures that path weights are not affected by cycling an arbitrary number of times within an intermediate node. If no path exists between the two nodes S and T, equation (6.2) sums over the empty set which, by definition, returns the zero element of the semiring. Alternatively, we may define $P^r_{S,T}$ as the set of all paths from node S to node T of length $r \geq 0$. We obtain

$$P_{S,T} \quad = \quad \bigcup_{r \geq 0} P^r_{S,T}$$

and consequently for $S \neq T$

$$\mathbf{D}(S,T) \quad = \quad \bigoplus_{r \geq 0} \bigoplus_{p \in P^r_{S,T}} w(p). \tag{6.3}$$

Example 6.1 *The connectivity problem of Instance 6.1 is based on the Boolean semiring of Example 5.2. Finding shortest distances according to Instance 6.2 amounts to computing in the tropical semiring of Example 5.4 where semiring multiplication corresponds to addition and semiring addition to minimization. The maximum capacity problem of Instance 6.3 is induced by the bottleneck semiring of Example 5.3, and* *the product t-norm semiring of Example 5.8 allows to compute the most reliable path in Instance 6.4. Finally, the language of the automaton in Instance 6.5 is computed by the semiring of regular languages of Example 5.7. Observe that this semiring is not commutative.*

There are several critical points regarding the equations (6.2) and (6.3): First, these formulas is entirely based on a graph related conception due to the explicit use of a path set. This may be sufficient for the examples given above, but we already pointed out in the introduction of this chapter that other applications without this interpretation exist. Second, it may well be that an infinite number of paths between two nodes exists, as it is for example the case in graphs with cycles. We then obtain an infinite sum in both equations that is not necessarily defined. These are the two major topics for the remaining part of this section. To start with, we develop a more general definition of path problems based on semiring matrices instead of path sets.

Example 6.2 *The graph in Figure 6.6 contains a cycle between the nodes* 1 *and* 6. *The path set $P_{1,6}$ therefore contains an infinite number of paths. Depending on the semiring, the sum in equation (6.2) is not defined.*

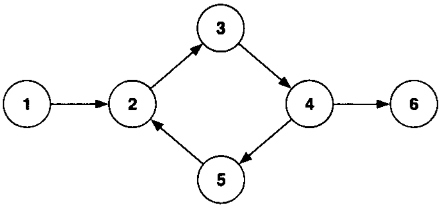


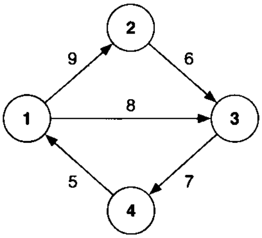
Figure 6.6 Path sets in cycled graphs may be infinite.

From now on, we again identify graph nodes with natural numbers $\{1, \ldots, n\}$ with $n = |V|$. The advantage of this convention is that we may represent a weighted graph in terms of its *adjacency matrix* $\mathbf{M} \in \mathcal{M}(A, n)$ defined as follows:

$$\mathbf{M}(X,Y) = \begin{cases} w(X,Y), & \text{if } (X,Y) \in E \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{6.4}$$

Due to the particular node numbering, the matrix entries directly identify the edge weights in the graph. It is sometimes common to set the diagonal elements of this semiring matrix to the unit element $\mathbf{1}$, if no other weight is specified in the graph. This, however, is not necessary for our purposes. Note that an adjacency matrix describes a full graph where any two nodes are linked, although some edges have weight zero. As a consequence, any sequence of nodes describes a path, but only the paths of weight different from zero describe paths in the underlying graph. Also, the definition of an adjacency matrix excludes *multigraphs* where multiple edges with identical directions but different weights may exist between two nodes. But this can be taken into consideration by assigning the sum of all weights of edges leading from node X to Y to $\mathbf{M}(X,Y)$.

Example 6.3 *Figure 6.7 shows a directed, weighted graph defined over the tropical semiring* $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0\rangle$ *and its associated adjacency matrix. Unknown distances are set to the zero element of the semiring, which here corresponds to* ∞.



$$\mathbf{M} = \begin{bmatrix} \infty & 9 & 8 & \infty \\ \infty & \infty & 6 & \infty \\ \infty & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty \end{bmatrix}$$

Figure 6.7 The adjacency matrix of a directed, weighted graph.

The solution of path problems based on adjacency matrices uses semiring matrix addition and multiplication as introduced in Section 5.1.2. Remember that matrices over semirings form themselves a semiring with the identity matrix as unit and the zero matrix as zero element. Let us next consider the *successive powers* of the adjacency matrix:

$$\mathbf{M}^2(S,T) = \sum_{K_1 \in \{1,\ldots,n\}} \mathbf{M}(S,K_1) \times \mathbf{M}(K_1,T)$$

$$\mathbf{M}^3(S,T) = \sum_{K_1,K_2 \in \{1,\ldots,n\}} \mathbf{M}(S,K_1) \times \mathbf{M}(K_1,K_2) \times \mathbf{M}(K_2,T)$$

and more generally

$$\mathbf{M}^r(S,T) = \sum_{K_1,\ldots,K_{r-1} \in \{1,\ldots,n\}} \mathbf{M}(S,K_1) \times \mathbf{M}(K_1,K_2) \times \ldots \times \mathbf{M}(K_{r-1},T).$$

We immediately observe that $\mathbf{M}^r(S,T)$ corresponds to the sum of all path weights of length $r \in \mathbb{N}$ between the selected source and target node. It therefore holds that

$$\mathbf{M}^r(S,T) = \bigoplus_{p \in P^r_{S,T}} w(p)$$

and we obtain the sum of all path weights of length at most $r \geq 0$ by

$$\mathbf{M}^{(r)} = \mathbf{I} + \mathbf{M} + \mathbf{M}^2 + \ldots + \mathbf{M}^r. \tag{6.5}$$

The identity matrix handles paths of length 0 by defining $\mathbf{I}(X,Y) = \mathbf{M}^{(0)}(X,Y) = \mathbf{1}$ if $X = Y$ and $\mathbf{I}(X,Y) = \mathbf{M}^{(0)}(X,Y) = \mathbf{0}$ otherwise. Its importance will be illustrated in Example 6.4 below. For $r > 0$, this sum of matrix powers can also be seen as the sum of the weights of all paths between S and T that visit at most $r - 1$ intermediate nodes. We are tempted to express the matrix $\mathbf{D}$ of equation (6.3) as the infinite sum

$$\mathbf{D} = \bigoplus_{r \geq 0} \mathbf{M}^r = \mathbf{I} + \mathbf{M} + \mathbf{M}^2 + \ldots \tag{6.6}$$

This corresponds to the sum of all paths between S and T visiting an arbitrary and unrestricted number of nodes. Contrary to equation (6.3) we have now derived a description of path problems that is independent of path sets and therefore not restricted to graph based applications anymore. On the other hand, we still face the problem of infinite sums of semiring values. Subsequently, we therefore introduce a topology on partially ordered semirings that allows us to define and study such formal sequences.

Example 6.4 *We reconsider the adjacency matrix from Example 6.3 and compute its successive powers for* $1 \leq r \leq 4$ *based on the tropical semiring with addition for semiring multiplication and minimization for semiring addition.*

$$\mathbf{M}^1 = \begin{bmatrix} \infty & 9 & 8 & \infty \\ \infty & \infty & 6 & \infty \\ \infty & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty \end{bmatrix}$$

$$\mathbf{M}^2 = \begin{bmatrix} \infty & 9 & 8 & \infty \\ \infty & \infty & 6 & \infty \\ \infty & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty \end{bmatrix} \times \begin{bmatrix} \infty & 9 & 8 & \infty \\ \infty & \infty & 6 & \infty \\ \infty & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty \end{bmatrix} = \begin{bmatrix} \infty & \infty & 15 & 15 \\ \infty & \infty & \infty & 13 \\ 12 & \infty & \infty & \infty \\ \infty & 14 & 13 & \infty \end{bmatrix}$$

$$\mathbf{M}^3 = \begin{bmatrix} \infty & 9 & 8 & \infty \\ \infty & \infty & 6 & \infty \\ \infty & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty \end{bmatrix} \times \begin{bmatrix} \infty & \infty & 15 & 15 \\ \infty & \infty & \infty & 13 \\ 12 & \infty & \infty & \infty \\ \infty & 14 & 13 & \infty \end{bmatrix} = \begin{bmatrix} 20 & \infty & \infty & 22 \\ 18 & \infty & \infty & \infty \\ \infty & 21 & 20 & \infty \\ \infty & \infty & 20 & 20 \end{bmatrix}$$

$$\mathbf{M}^4 = \begin{bmatrix} \infty & 9 & 8 & \infty \\ \infty & \infty & 6 & \infty \\ \infty & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty \end{bmatrix} \times \begin{bmatrix} 20 & \infty & \infty & 22 \\ 18 & \infty & \infty & \infty \\ \infty & 21 & 20 & \infty \\ \infty & \infty & 20 & 20 \end{bmatrix} = \begin{bmatrix} 27 & 29 & 28 & \infty \\ \infty & 27 & 26 & \infty \\ \infty & \infty & 27 & 27 \\ 25 & \infty & \infty & 27 \end{bmatrix}$$

$\mathbf{M}^4(1,3) = 28$ *for example means that the distance of the shortest path from node* 1 *to node* 3 *containing exactly* 4 *edges is* 28. *Using equation (6.5) we compute* $\mathbf{M}^{(4)}$ *that contains all shortest paths of at most* 4 *edges:*

$$\begin{aligned}
\mathbf{M}^{(4)} &= \mathbf{I}+\mathbf{M}+\mathbf{M}^2+\mathbf{M}^3+\mathbf{M}^4 \\
&= \begin{bmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{bmatrix} + \begin{bmatrix} \infty & 9 & 8 & \infty \\ \infty & \infty & 6 & \infty \\ \infty & \infty & \infty & 7 \\ 5 & \infty & \infty & \infty \end{bmatrix} + \begin{bmatrix} \infty & \infty & 15 & 15 \\ \infty & \infty & \infty & 13 \\ 12 & \infty & \infty & \infty \\ \infty & 14 & 13 & \infty \end{bmatrix} \\
&+ \begin{bmatrix} 20 & \infty & \infty & 22 \\ 18 & \infty & \infty & \infty \\ \infty & 21 & 20 & \infty \\ \infty & \infty & 20 & 20 \end{bmatrix} + \begin{bmatrix} 27 & 29 & 28 & \infty \\ \infty & 27 & 26 & \infty \\ \infty & \infty & 27 & 27 \\ 25 & \infty & \infty & 27 \end{bmatrix} = \begin{bmatrix} 0 & 9 & 8 & 15 \\ 18 & 0 & 6 & 13 \\ 12 & 21 & 0 & 7 \\ 5 & 14 & 13 & 0 \end{bmatrix}
\end{aligned}$$

Here, $\mathbf{M}^{(4)}(1,3) = 8$ *means that the distance of the shortest path from node* 1 *to node* 3 *containing at most* 4 *edges is* 8. *In this very special case,* $\mathbf{M}^{(4)}$ *already contains all shortest paths in the graph of Figure 6.7 which is not generally the case. Also, we observe the importance of the identity matrix to ensure that each node can reach itself with a shortest distance of* 0.

We next generalize the power sequence of semiring matrices to arbitrary semirings A and define for $r \geq 0$ and $a \in A$:

$$a^{(r)} \quad = \quad \mathbf{1} + a + a^2 + a^3 + \ldots + a^r. \tag{6.7}$$

The following recursions are direct consequences of this definition:

$$a^{(r+1)} \quad = \quad a \times a^{(r)} + \mathbf{1} \quad = \quad a^{(r)} + a^{r+1}. \tag{6.8}$$

From Lemma 5.2, (SP3) follows that the power sequence is non-decreasing,

$$a^{(r)} \quad \leq \quad a^{(r+1)}. \tag{6.9}$$

This refers to the canonical preorder of semirings introduced in equation (5.3). We also remember that if this relation is a partial order, the semiring is called a dioid. Idempotency of addition is always a sufficient condition to turn the preorder into a partial order which then is equivalently expressed by equation (5.4). Following (Bétréma, 1982) dioids can be equipped with a particular topology called *sup-topology*, where a sequence of dioid values is *convergent*, if it is non-decreasing, bounded from above and if it has a least upper bound. The *limit* of a convergent sequence then corresponds to its least upper bound. However, the simple presence of the sup-topology is not yet sufficient for our purposes. Instead, two additional properties must hold which are summarized in the definition of a *topological dioid* (Gondran & Minoux, 2008).

Definition 6.1 *A dioid endowed with the sup-topology with respect to its canonical partial order is called* topological dioid *if the following properties hold:*

1. *Every non-decreasing sequence bounded from above has a least upper bound.*

2. *Taking the limit is compatible with semiring addition and multiplication, i.e.*

$$\begin{aligned}\lim_{r\to\infty}(a^{(r)}+b^{(r)}) &= \lim_{r\to\infty}a^{(r)}+\lim_{r\to\infty}b^{(r)}\\ \lim_{r\to\infty}(a^{(r)}\times b^{(r)}) &= \lim_{r\to\infty}a^{(r)}\times\lim_{r\to\infty}b^{(r)}.\end{aligned}$$

Thus, if the semiring is a topological dioid and if the power sequence is bounded from above, it follows from Definition 6.1 that the sequence has a least upper bound. We then know that the sequence is convergent with respect to the sup-topology and that its limit corresponds to the least upper bound. We subsequently denote the limit of the power sequence by

$$a^* = \sup_{r\geq 0}\{a^{(r)}\} = \lim_{r\to\infty}a^{(r)}. \tag{6.10}$$

Example 6.5 *Bounded semirings of Definition 5.3 satisfy $a+\mathbf{1}=\mathbf{1}$ for all $a\in A$. They are idempotent since $\mathbf{1}+\mathbf{1}=\mathbf{1}$ always implies that $a+a=a$. Thus, bounded semirings are dioids, and since $a\leq\mathbf{1}$ we also have $a^r\leq\mathbf{1}$ and therefore $a^{(r)}=\mathbf{1}+a+\ldots+a^r\leq\mathbf{1}$. This shows that power sequences over bounded semirings are always non-decreasing and bounded from above. If we further have a topological dioid, then the limit of this sequence exists for each $a\in A$. Some instances of bounded semirings were given in Example 5.10.*

The following theorem links the limit of a power sequence in a topological dioid to the solution of a *linear fixpoint equation*. Semiring elements that satisfy this equation are generally referred to as *quasi-inverses*. Note that we subsequently write *ab* for $a\times b$ whenever it is contextually clear that we refer to semiring multiplication. Also, we keep our convention to refer to variables by capital letters.

Theorem 6.1 *If a^* exists in a topological dioid, it is always the least solution to the fixpoint equations*

$$X = aX+\mathbf{1} \quad \textit{and} \quad X = Xa+\mathbf{1}. \tag{6.11}$$

Let us first prove the following intermediate result:

Lemma 6.1 *Let A be an arbitrary semiring and $y\in A$ a solution to (6.11). We have*

$$y = a^{r+1}y+a^{(r)} \tag{6.12}$$

and the solution y is an upper bound for the sequence $a^{(r)}$.

Proof: We proceed by induction: For $r=0$ we have $y=ay+\mathbf{1}$. This holds because y is a solution to equation (6.11). Assume now that equation (6.12) holds for r. For $r+1$ it then follows from equation (6.8) and the induction hypothesis that

$$\begin{aligned}a^{r+2}y+a^{(r+1)} &= a^{r+2}y+aa^{(r)}+\mathbf{1}\\ &= a(a^{r+1}y+a^{(r)})+\mathbf{1} = ay+\mathbf{1} = y.\end{aligned}$$

Using Lemma 5.2, (SP3) we finally conclude that $a^{(r)} \leq y$. ■

Using this result we next prove Theorem 6.1:

Proof: Taking the limit in a topological dioid is compatible with semiring addition and multiplication. We thus conclude that the limit of $aa^{(r)}$ is aa^* and that $aa^{(r)} + \mathbf{1}$ is convergent with limit $aa^* + \mathbf{1}$. It follows from equation (6.8) that $a^* = aa^* + \mathbf{1}$ and similarly that $a^* = a^*a + \mathbf{1}$. This shows that a^* is a solution to equation (6.11). From Lemma 6.1 follows that for all solutions y we have $a^{(r)} \leq y$. Due to equation (6.10) the particular solution a^* is the least upper bound of the power sequence and therefore also the least solution to both fixpoint equations. ■

There may be several solutions to these fixpoint equations and thus several quasi-inverses for a semiring element as shown in Example 6.6 below. But if we have a topological dioid where the power sequence converges, then its limit is the least element among them. Further, we show in Lemma 6.2 below that the limit also determines the least solution to more general fixpoint equations.

Example 6.6 *Consider the tropical semiring $\langle \mathbb{R} \cup \{\infty\}, \min, +, \infty, 0\rangle$ of Example 5.4. We then obtain for equation (6.11) $x = \min\{a + x, 0\}$. For $a > 0$ this equation always has a unique solution $x = 0$. If $a < 0$ there is no solution and if $a = 0$, then every $x \leq 0$ is a solution to this fixpoint equation.*

Lemma 6.2

1. *If x is a solution to the fixpoint equation $X = aX + \mathbf{1}$ in an arbitrary semiring A, then xb is a solution to $X = aX + b$ with $b \in A$. Likewise, if x is a solution to $X = Xa + \mathbf{1}$, then bx is a solution to $X = Xa + b$.*

2. *If the limit a^* exists in a topological dioid, then a^*b is always the least solution to $X = aX + b$. Likewise, ba^* is always the least solution to $X = Xa + b$.*

Proof:

1. If x satisfies $x = ax + \mathbf{1}$ we have

$$axb + b \;=\; (ax + \mathbf{1})b \;=\; xb.$$

   Thus, xb is a solution to $X = aX + b$. The second statement follows similarly.

2. If a^* exists, it is a solution to $X = aX + \mathbf{1}$ by Theorem 6.1. By the first statement, a^*b is a solution to $X = aX + b$. In order to prove that it is the least solution, assume $y \in A$ to be another solution to $X = aX + b$. An induction proof similar to Lemma 6.1 shows that $a^{(k)}b \leq y$ for all $k \in \mathbb{N} \cup \{0\}$. The solution y is therefore an upper bound for the sequence $\{a^{(k)}b\}$ and since this sequence is non-decreasing and bounded from above, it converges towards the least upper bound a^*b of the sequence. We therefore have $a^*b \leq y$.

■

Let us return to the solution of path problems and the power sequence of semiring matrices in equation (6.6). We first recall that matrices with values from a semiring themselves form a semiring with a component-wise definition of the canonical preorder. Thus, if this relation is a partial order, then the corresponding semiring of matrices is also a dioid. Moreover, if the semiring is a topological dioid, then so is the semiring of matrices (Gondran & Minoux, 2008). We conclude from equation (6.10) that if the power sequence of semiring matrices with values from a topological dioid converges, then solving a path problem is equivalent to computing the least quasi-inverse matrix. This computational task is stated by the following definition:

Definition 6.2 *Let* $\mathbf{M} \in \mathcal{M}(A, n)$ *be a square matrix over a topological dioid* A. *If the limit exists, the* algebraic path problem *consists in computing*

$$\mathbf{M}^* \quad = \quad \lim_{r \to \infty} \mathbf{M}^{(r)}. \tag{6.13}$$

We now have a formally correct and general description of a path problem that is not restricted to graph related applications anymore. However, it is based on the limit of an infinite sequence and therefore rather inconvenient for computational purposes. An alternative way to express the algebraic path problem by means of the least solution to a linear fixpoint equation is indicated by Theorem 6.1. For that purpose, we first derive a converse result to this theorem:

Lemma 6.3 *If in a topological dioid* y *is the least solution to the fixpoint equations*

$$X \;=\; aX + \mathbf{1} \quad \textit{and} \quad X \;=\; Xa + \mathbf{1}, \tag{6.14}$$

then it corresponds to the quasi-inverse of $a \in A$ *satisfying*

$$y \quad = \quad \lim_{r \to \infty} a^{(r)}.$$

Proof: We recall from equation (6.9) that the sequence $a^{(r)}$ is non-decreasing. Further, because we have assumed the existence of a least solution, the sequence is also bounded by Lemma 6.1. Definition 6.1 then assures that the sequence has a least upper bound which corresponds to the limit of the power sequence. But then, this limit is the least solution to the fixpoint equation according to Theorem 6.1 and corresponds to y since the least element is uniquely determined. ∎

We obtain from Theorem 6.1 and Lemma 6.3 the following alternative but equivalent definition of the algebraic path problem:

Definition 6.3 *Let* $\mathbf{M} \in \mathcal{M}(A, n)$ *be a square matrix over a topological dioid* A. *The algebraic path problem consists in computing the least solution to*

$$\mathbf{X} \;=\; \mathbf{M}\mathbf{X} + \mathbf{I} \;=\; \mathbf{X}\mathbf{M} + \mathbf{I}, \tag{6.15}$$

if such a solution exists.

Here, the variable $\mathbf{X}$ refers to an unknown matrix in $\mathcal{M}(A, n)$. It is important to note that this definition is still conditioned on the existence of a solution, respectively on the convergence of the power sequence. Imagine for example that we solve a shortest distance problem on the cyclic graph of Figure 6.6 with negative values assigned to the edges of the cycle. If we then consider the path weights between the source and target node, we obtain a shorter total distance for each additional rotation in the cycle. In this case, the limit of equation (6.13) does not exist or, in other words, the fixpoint equations have no solution. This is comparable to the case $a < 0$ in Example 6.6. Investigating the requirements for the existence of quasi-inverses is an ongoing research topic. Many approaches impose restrictions on graphs which for example forbid cycles of negative path weights in case of the shortest distance problem. But this naturally only captures graph related applications. Alternatively, we may ensure the existence of quasi-inverses axiomatically which also covers other applications. Then, the semiring of Example 6.6, where no quasi-inverses exist for negative elements, would not be part of the framework anymore. However, we can easily imagine that shortest distances can still be found in graphs with negative values, as long as there are no cycles with negative path weights. In many cases we can adjoin an additional element to the semiring that acts as an artificial limit for those sequences that are divergent in the original semiring. Reconsider the shortest distance problem with edge weights from the semiring $\langle \mathbb{R} \cup \{\infty\}, \min, +, \infty, 0\rangle$. Here, it is reasonable to define $\mathbf{M}^*(S,T) = -\infty$, if the path from node S to T contains a cycle of negative weight. This amounts to computing in the extended semiring $\langle \mathbb{R} \cup \{-\infty, \infty\}, \min, +, \infty, 0\rangle$ with $-\infty + \infty = \infty$. Then, the problem of non-existence in Definition 6.3 is avoided and the algebraic path problem reduces to finding a particular solution to a semiring fixpoint equation.

6.3 QUASI-REGULAR SEMIRINGS

We now focus on semirings where each element has at least one quasi-inverse. Then, it is always possible to introduce a third semiring operation called *star operation* that delivers a particular quasi-inverse for each semiring element. Such semirings are called *closed semirings* in (Lehmann, 1976) but to avoid confusions with other semiring structures, we prefer to call them *quasi-regular semirings*.

Definition 6.4 *A* quasi-regular *semiring $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ is an algebraic structure where $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ is a semiring and $*$ an unary operation such that for $a \in A$*

$$a^* \;=\; aa^* + \mathbf{1} \;=\; a^*a + \mathbf{1}.$$

Restricting Example 6.6 to non-negative integers, we see that quasi-inverse elements are not necessarily unique in a semiring. Consequently, there may be different ways of defining the star operation which each time leads to a different quasi-regular semiring. Here, we list some possible extensions to quasi-regular semirings for the examples that marked the beginning of this chapter:

Example 6.7 *The tropical semiring $\langle \mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +, \infty, 0\rangle$ of non-negative real numbers (or integers) introduced in Example 5.4 is quasi-regular with $a^* = 0$ for all*

$a \in \mathbb{R}_{\geq 0} \cup \{\infty\}$. This can be extended to the tropical semiring of all real numbers $\langle \mathbb{R} \cup \{-\infty, \infty\}, \min, +, \infty, 0\rangle$ by $a^* = 0$ for $a \geq 0$ and $a^* = -\infty$ for $a < 0$. Similarly, the arctic semiring from Example 5.5 $\langle \mathbb{R}_{\geq 0} \cup \{-\infty, \infty\}, \max, +, -\infty, 0\rangle$ is quasi-regular with $a^* = \infty$ for $a \geq 0$ and $a^* = 0$ for $a \in \{-\infty, 0\}$. The Boolean semiring $\langle \{0,1\}, \max, \min, 0, 1\rangle$ of Example 5.2 is quasi-regular with $0^* = 1^* = 1$. This also holds for the bottleneck semiring $\langle \mathbb{R} \cup \{-\infty, \infty\}, \max, \min, -\infty, \infty\rangle$ from Example 5.3 where $a^* = \infty$ for all elements. The probabilistic or product t-norm semiring $\langle [0,1], \max, \cdot, 0, 1\rangle$ of Example 5.8 is extended to a quasi-regular semiring by defining $a^* = 1$ for all $a \in [0,1]$. Finally, the semiring of formal languages $\langle \mathcal{P}(\Sigma^*), \cup, \cdot, \emptyset, \{\epsilon\}\rangle$ presented in Example 5.7 is quasi-regular with

$$a^* = \bigcup_{i \geq 1} a^i = a \cup aa \cup aaa \cup \ldots \tag{6.16}$$

Example 6.1 showed that these semirings were used in the introductory path problems of this chapter which are thus based on quasi-regular semirings.

If we consider matrices with values from a quasi-regular semiring, then it is possible to derive quasi-inverse matrices from the star operation in the underlying semiring through the following induction:

- For $n = 1$ we define $[a]^* = [a^*]$.
- For $n > 1$ we decompose the matrix $\mathbf{M}$ into submatrices $\mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}$ such that $\mathbf{B}$ and $\mathbf{E}$ are square and define

$$\mathbf{M}^* = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}^* = \begin{bmatrix} \mathbf{B}^* + \mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{B}^*\mathbf{C}\mathbf{F}^* \\ \mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{F}^* \end{bmatrix} \tag{6.17}$$

  where $\mathbf{F} = \mathbf{E} + \mathbf{D}\mathbf{B}^*\mathbf{C}$.

It is important to note that equation (6.17) does not depend on the way how the matrix $\mathbf{M}$ is decomposed. This can easily be proved by applying equation (6.17) to a matrix with nine submatrices, decomposed in two different ways:

$$\left[\begin{array}{c|cc} \mathbf{B} & \mathbf{C} & \mathbf{D} \\ \hline \mathbf{E} & \mathbf{F} & \mathbf{G} \\ \mathbf{H} & \mathbf{J} & \mathbf{K} \end{array}\right] \quad \text{and} \quad \left[\begin{array}{cc|c} \mathbf{B} & \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} & \mathbf{G} \\ \hline \mathbf{H} & \mathbf{J} & \mathbf{K} \end{array}\right]$$

and by verifying nine identities using the semiring properties from Definition 5.1. A graph based justification of equation (6.17) will later be given in Example 6.12. The following theorem has been proposed by (Lehmann, 1976):

Theorem 6.2 *Let $\mathbf{M} \in \mathcal{M}(A, n)$ be a matrix with values from a quasi-regular semiring A. Then, the matrix $\mathbf{M}^*$ obtained from equation (6.17) satisfies*

$$\mathbf{M}^* = \mathbf{M}\mathbf{M}^* + \mathbf{I} = \mathbf{M}^*\mathbf{M} + \mathbf{I}. \tag{6.18}$$

Proof: Because the two equalities in (6.18) are symmetric we only prove the first one. Let us proceed by induction: For $n = 1$ the equality holds by Definition 6.4. The induction hypothesis is that equation (6.18) holds for matrices of size less than n. For $n > 1$ suppose the decomposition

$$\mathbf{M} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}$$

with $\mathbf{B} \in \mathcal{M}(A,k)$ for some $n > k > 0$. By equation (6.17) we have

$$\mathbf{M}^* = \begin{bmatrix} \mathbf{B}^* + \mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{B}^*\mathbf{C}\mathbf{F}^* \\ \mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{F}^* \end{bmatrix}$$

and therefore

$$\mathbf{M}\mathbf{M}^* = \begin{bmatrix} \mathbf{B}\mathbf{B}^* + \mathbf{B}\mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* + \mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{B}\mathbf{B}^*\mathbf{C}\mathbf{F}^* + \mathbf{C}\mathbf{F}^* \\ \mathbf{D}\mathbf{B}^* + \mathbf{D}\mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* + \mathbf{E}\mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{D}\mathbf{B}^*\mathbf{C}\mathbf{F}^* + \mathbf{E}\mathbf{F}^* \end{bmatrix}$$

By the induction hypothesis we have

$$\mathbf{B}^* = \mathbf{I}_k + \mathbf{B}\mathbf{B}^* \quad \text{and} \quad \mathbf{F}^* = \mathbf{I}_{n-k} + \mathbf{F}\mathbf{F}^*.$$

This together with $\mathbf{F} = \mathbf{E} + \mathbf{D}\mathbf{B}^*\mathbf{C}$ gives us the following four equalities:

$$\begin{aligned} \mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* + \mathbf{B}\mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* &= (\mathbf{I}_k + \mathbf{B}\mathbf{B}^*)\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* = \mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* \\ \mathbf{C}\mathbf{F}^* + \mathbf{B}\mathbf{B}^*\mathbf{C}\mathbf{F}^* &= (\mathbf{I}_k + \mathbf{B}\mathbf{B}^*)\mathbf{C}\mathbf{F}^* = \mathbf{B}^*\mathbf{C}\mathbf{F}^* \\ \mathbf{D}\mathbf{B}^* + \mathbf{D}\mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* + \mathbf{E}\mathbf{F}^*\mathbf{D}\mathbf{B}^* &= \mathbf{D}\mathbf{B}^* + \mathbf{F}\mathbf{F}^*\mathbf{D}\mathbf{B}^* = \\ &\quad (\mathbf{I}_{n-k} + \mathbf{F}\mathbf{F}^*)\mathbf{D}\mathbf{B}^* = \mathbf{F}^*\mathbf{D}\mathbf{B}^* \\ \mathbf{D}\mathbf{B}^*\mathbf{C}\mathbf{F}^* + \mathbf{E}\mathbf{F}^* &= \mathbf{F}\mathbf{F}^* \end{aligned}$$

We finally obtain

$$\begin{aligned} \mathbf{I}_n + \mathbf{M}\mathbf{M}^* &= \begin{bmatrix} \mathbf{I}_k + \mathbf{B}\mathbf{B}^* + \mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{B}^*\mathbf{C}\mathbf{F}^* \\ \mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{I}_{n-k} + \mathbf{F}\mathbf{F}^* \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{B}^* + \mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{B}^*\mathbf{C}\mathbf{F}^* \\ \mathbf{F}^*\mathbf{D}\mathbf{B}^* & \mathbf{F}^* \end{bmatrix} = \mathbf{M}^*. \end{aligned}$$

■

Thus, given a quasi-regular semiring, we can find a quasi-inverse for each matrix defined over this semiring. In other words, each quasi-regular semiring induces a particular quasi-regular semiring of matrices through the above construction. Algorithms to compute such quasi-inverse matrices will be presented in Chapter 9. Here, we benefit from this important result to reformulate the algebraic path problem of Definition 6.3 for matrices over quasi-regular semirings.

Definition 6.5 *For a matrix* $\mathbf{M} \in \mathcal{M}(A,n)$ *over a quasi-regular semiring* A*, the algebraic path problem consists in computing* $\mathbf{M}^*$ *defined by equation (6.17).*

It is important to note that the two Definitions 6.3 and 6.5 of the algebraic path problem are not equivalent and in fact difficult to compare. We mentioned several times that quasi-inverses do not need to be unique. Definition 6.3 presupposes a partial order and focusses on the computation of the least quasi-inverse with respect to this order. Quasi-regular semirings do not necessarily provide a partial order such that we obtain an arbitrary quasi-inverse from solving the algebraic path problem in Definition 6.5. But even if we assume a partial order in the quasi-regular semiring of Definition 6.5 and also that all sequences in Definition 6.3 converge (Aho *et al.*, 1974), we do not necessarily have equality between the two quasi-inverses (Kozen, 1990). We can therefore only observe that a solution to the algebraic path problem with respect to Definition 6.5 always satisfies the fixpoint equation of Theorem 6.1, although it is not necessarily the least element of the solution set. In Section 6.6 we will introduce a further semiring structure that extends quasi-regular semirings by idempotency and also adds a monotonicity property with respect to the star operation. In this setting, the quasi-inverse matrix obtained from equation (6.17) will again be the least solution to the fixpoint equation. We will now give an example of a quasi-inverse matrix which is obtained from the above construction:

Example 6.8 *We have seen in Example 6.7 that the tropical semiring of non-negative integers* $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0\rangle$ *is quasi-regular with* $a^* = 0$ *for all* $a \in \mathbb{N} \cup \{0, \infty\}$. *Let us assume the following matrix defined over this semiring and compute its quasi-inverse by equation (6.17). The following decomposition is chosen:*

$$\mathbf{M} \;=\; \left[\begin{array}{c|cc} \infty & 7 & 1 \\ \hline 4 & \infty & \infty \\ \infty & 2 & \infty \end{array}\right] \;=\; \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}.$$

With semiring addition as minimization, semiring multiplication as integer addition and $\infty^* = 0$ *we first compute*

$$\begin{aligned} \mathbf{F} = \mathbf{E} + \mathbf{D}\mathbf{B}^*\mathbf{C} \;&=\; \begin{bmatrix} \infty & \infty \\ 2 & \infty \end{bmatrix} + \begin{bmatrix} 4 \\ \infty \end{bmatrix} [\infty]^* \begin{bmatrix} 7 & 1 \end{bmatrix} \\ &=\; \begin{bmatrix} \infty & \infty \\ 2 & \infty \end{bmatrix} + \begin{bmatrix} 11 & 5 \\ \infty & \infty \end{bmatrix} \;=\; \begin{bmatrix} 11 & 5 \\ 2 & \infty \end{bmatrix} \end{aligned}$$

Applying equation (6.17) we obtain

$$\mathbf{F}^* \;=\; \begin{bmatrix} 11 & 5 \\ 2 & \infty \end{bmatrix}^* = \begin{bmatrix} 0 & 5 \\ 2 & 0 \end{bmatrix}.$$

Since $\mathbf{B}^* = [0]$ *we directly conclude*

$$\mathbf{B}^* + \mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* \;=\; [0]$$

and compute for the two remaining submatrices:

$$\mathbf{B}^*\mathbf{C}\mathbf{F}^* = \mathbf{C}\mathbf{F}^* = \begin{bmatrix} 7 & 1 \end{bmatrix}\begin{bmatrix} 0 & 5 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 1 \end{bmatrix}$$

$$\mathbf{F}^*\mathbf{D}\mathbf{B}^* = \mathbf{F}^*\mathbf{D} = \begin{bmatrix} 0 & 5 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} 4 \\ \infty \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}.$$

Putting these results together, we obtain the quasi-inverse matrix $\mathbf{M}^*$*:*

$$\mathbf{M}^* = \begin{bmatrix} 0 & 3 & 1 \\ 4 & 0 & 5 \\ 6 & 2 & 0 \end{bmatrix}.$$

If we interpret $\mathbf{M}$ *as a distance matrix, we immediately observe that* $\mathbf{M}^*$ *contains all shortest distances. An explanation why the above construction for quasi-inverse matrices always leads to shortest distances in case of the tropical semiring over non-negative numbers will be given in Section 6.6.*

To conclude this section, Example 6.9 alludes to an alternative way of deriving the fixpoint equation of Definition 6.5 which is not motivated by the semiring power sequence, but instead uses *Bellmann's principle of optimality* (Bellman, 1957).

Example 6.9 *Let* $\mathbf{M}$ *be the adjacency matrix of a directed, weighted graph with edge weights from the tropical semiring* $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0\rangle$ *of non-negative integers. We may then write equation (6.18) for* $X \neq Y$ *such that* $\mathbf{I}(X,Y) = \infty$ *as*

$$\mathbf{M}^*(X,Y) = \min_Z\Big(\mathbf{M}(X,Z) + \mathbf{M}^*(Z,Y)\Big).$$

Similarly, we obtain for $X = Y$

$$\mathbf{M}^*(X,Y) = \min\left(\min_Z\Big(\mathbf{M}(X,Z) + \mathbf{M}^*(Z,Y)\Big), 0\right) = 0.$$

The first equation states that the shortest path from node X *to node* Y *corresponds to the minimum sum of the direct distance to an intermediate node* Z *and the shortest path from* Z *to the terminal node* Y*. This is Bellmann's principle of optimality. The second equality states that each node can reach itself with distance zero.*

The complete matrix $\mathbf{M}^*$ of Example 6.9 contains the shortest distances between all pairs of graph nodes. It is therefore common to refer to the generalized computational task of Definition 6.3 as the *all-pairs algebraic path problem.* If we are only interested in the shortest distances between a selected source node S and all possible target nodes, it is sufficient to compute the row of the matrix $\mathbf{M}^*$ that corresponds to this source node, i.e. we compute $\mathbf{b}\mathbf{M}^*$ where $\mathbf{b}$ is a row vector with $\mathbf{b}(S) = \mathbf{1}$ and $\mathbf{b}(Y) = \mathbf{0}$ for all $Y \neq S$. It then follows from $\mathbf{M}^* = \mathbf{M}\mathbf{M}^* + \mathbf{I}$ that $\mathbf{b}\mathbf{M}^* = \mathbf{b}\mathbf{M}\mathbf{M}^* + \mathbf{b}$. This shows that $\mathbf{b}\mathbf{M}^*$ is a solution to

$$\mathbf{X} = \mathbf{X}\mathbf{M} + \mathbf{b}, \tag{6.19}$$

where $\mathbf{X}$ is a variable row vector. Hence, we refer to the task of computing $\mathbf{bM}^*$ defined by equation (6.17) as the *single-source algebraic path problem*. In a similar manner, the column vector $\mathbf{M}^*\mathbf{b}$ is a solution to

$$\mathbf{X} \quad = \quad \mathbf{MX} + \mathbf{b} \tag{6.20}$$

and refers in the above instantiation to the shortest distances between all possible source nodes and the selected target node. The task of computing equation $\mathbf{M}^*\mathbf{b}$ defined by equation (6.17) is therefore called *single-target algebraic path problem*. Clearly, instead of solving the all-pairs problem directly, we can either solve the single-source or single-target problem for each possible vector $\mathbf{b}$, i.e. for each possible source or target node. We will see in the following section that matrices over quasi-regular semirings induce a family of valuation algebras called *quasi-regular valuation algebras*. It will be shown in Chapter 9 that the single-source algebraic path problem can be solved by local computation with quasi-regular valuation algebras.

6.4 QUASI-REGULAR VALUATION ALGEBRAS

In this section, we are going to derive a generic construction that leads to a new valuation algebra for each quasi-regular semiring. These formalisms, called *quasi-regular valuation algebras*, are related to the solution of the single-target algebraic path problem or, in other words, to the computation of a single column of the quasi-inverse matrix that results from equation (6.17). Until now, we have dealt in this chapter with ordinary semiring matrices. But for the study of generic constructions and valuation algebras, we will reconsider labeled matrices as introduced in Chapter 1 for real numbers. This time, however, we assume labeled matrices with values from a quasi-regular semiring A. Thus, let r be a set of variables, $s \subseteq r$ and $\mathcal{M}(A,s)$ the set of labeled matrices $\mathbf{M} : s \times s \rightarrow A$. Inspired by (Lehmann, 1976; Radhakrishnan *et al.*, 1992; Backhouse & Carré, 1975), we observe that the pair $(\mathbf{M}, \mathbf{b})$ for $\mathbf{M} \in \mathcal{M}(A,s)$ and an s-vector $\mathbf{b} : s \rightarrow A$ sets up the fixpoint equation

$$\mathbf{X} \quad = \quad \mathbf{MX} + \mathbf{b} \tag{6.21}$$

where $\mathbf{X}$ denotes a variable column vector with domain s. This equation can be regarded as a labeled version of equation (6.20) generalized to vectors $\mathbf{b}$ of arbitrary values from a quasi-regular semiring. We further define the set

$$\Phi \quad = \quad \{(\mathbf{M}, \mathbf{b}) \mid \mathbf{M} \in \mathcal{M}(A,s) \text{ and } \mathbf{b} : s \rightarrow A \text{ for } s \subseteq r\} \tag{6.22}$$

and also write $d(\mathbf{M}, \mathbf{b}) = s$ if $\mathbf{M} \in \mathcal{M}(A,s)$.

In order to introduce a projection operator for the elements in Φ, we first decompose the above system with respect to $t \subseteq s$ and obtain

$$\begin{bmatrix} \mathbf{X}^{\downarrow s-t} \\ \mathbf{X}^{\downarrow t} \end{bmatrix} = \begin{bmatrix} \mathbf{M}^{\downarrow s-t,s-t} & \mathbf{M}^{\downarrow s-t,t} \\ \mathbf{M}^{\downarrow t,s-t} & \mathbf{M}^{\downarrow t,t} \end{bmatrix} \begin{bmatrix} \mathbf{X}^{\downarrow s-t} \\ \mathbf{X}^{\downarrow t} \end{bmatrix} + \begin{bmatrix} \mathbf{b}^{\downarrow s-t} \\ \mathbf{b}^{\downarrow t} \end{bmatrix}$$

from which we derive the two equations

$$\mathbf{X}^{\downarrow s-t} = \mathbf{M}^{\downarrow s-t,s-t}\mathbf{X}^{\downarrow s-t} + \mathbf{M}^{\downarrow s-t,t}\mathbf{X}^{\downarrow t} + \mathbf{b}^{\downarrow s-t}$$

and

$$\mathbf{X}^{\downarrow t} = \mathbf{M}^{\downarrow t,s-t}\mathbf{X}^{\downarrow s-t} + \mathbf{M}^{\downarrow t,t}\mathbf{X}^{\downarrow t} + \mathbf{b}^{\downarrow t}.$$

Lemma 6.2 points out that $(\mathbf{M}^{\downarrow s-t,s-t})^*\tilde{\mathbf{b}}^{\downarrow s-t}$ with

$$\tilde{\mathbf{b}}^{\downarrow s-t} = \mathbf{M}^{\downarrow s-t,t}\mathbf{X}^{\downarrow t} + \mathbf{b}^{\downarrow s-t}$$

provides a solution to the first equation. Note, however, that $\tilde{\mathbf{b}}^{\downarrow s-t}$ is not a semiring value, so that our discussion so far is only informal. Nevertheless, let us proceed with this expression and insert it into the second equation. After regrouping of variables we obtain

$$\begin{aligned}\mathbf{X}^{\downarrow t} &= \Big(\mathbf{M}^{\downarrow t,t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}\Big)\mathbf{X}^{\downarrow t} \\ &+ \Big(\mathbf{b}^{\downarrow t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t}\Big).\end{aligned}$$

For $(\mathbf{M}, \mathbf{b}) \in \Phi$ with $d(\mathbf{M}, \mathbf{b}) = s$ and $t \subseteq s$ this indicates that we could define

$$(\mathbf{M}, \mathbf{b})^{\downarrow t} = (\mathbf{M}_t, \mathbf{b}_t) \tag{6.23}$$

with

$$\mathbf{M}_t = \mathbf{M}^{\downarrow t,t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t} \tag{6.24}$$

and

$$\mathbf{b}_t = \mathbf{b}^{\downarrow t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t} \tag{6.25}$$

such that

$$\mathbf{X}^{\downarrow t} = \mathbf{M}_t\mathbf{X}^{\downarrow t} + \mathbf{b}_t.$$

The following theorem justifies this informal argument. It states that if a particular solution to equation (6.21) is given, then the restriction of this solution to the subdomain also solves the projected equation system. Conversely, it is always possible to reconstruct the solution to the original system from the projected solution.

Theorem 6.3 *For a quasi-regular semiring A with $\mathbf{M} \in \mathcal{M}(A, s)$, $\mathbf{b} : s \rightarrow A$ and $t \subseteq s$, the following statements hold:*

1. *If $\mathbf{x} = \mathbf{M}^*\mathbf{b}$ is a solution to $\mathbf{X} = \mathbf{M}\mathbf{X} + \mathbf{b}$ then*

$$\mathbf{x}^{\downarrow t} = (\mathbf{M}_t)^*\mathbf{b}_t \tag{6.26}$$

is a solution to $\mathbf{X}^{\downarrow t} = \mathbf{M}_t\mathbf{X}^{\downarrow t} + \mathbf{b}_t$.

2. *If* $\mathbf{x} = (\mathbf{M}_t)^*\mathbf{b}_t$ *is a solution to* $\mathbf{X}^{\downarrow t} = \mathbf{M}_t\mathbf{X}^{\downarrow t} + \mathbf{b}_t$ *and*

$$\mathbf{y} \quad = \quad (\mathbf{M}^{\downarrow s-t,s-t})^*(\mathbf{M}^{\downarrow s-t,t}\mathbf{x} + \mathbf{b}^{\downarrow s-t}) \tag{6.27}$$

then $(\mathbf{y}, \mathbf{x}) = \mathbf{M}^*\mathbf{b}$ *is a solution to* $\mathbf{X} = \mathbf{M}\mathbf{X} + \mathbf{b}$.

Proof:

1. We first consider the submatrices of $\mathbf{M}^*$ with respect to $s-t$ and t. From

$$\mathbf{M}^* \;=\; \begin{bmatrix} \mathbf{M}^{\downarrow s-t,s-t} & \mathbf{M}^{\downarrow s-t,t} \\ \mathbf{M}^{\downarrow t,s-t} & \mathbf{M}^{\downarrow t,t} \end{bmatrix}^* = \begin{bmatrix} (\mathbf{M}^*)^{\downarrow s-t,s-t} & (\mathbf{M}^*)^{\downarrow s-t,t} \\ (\mathbf{M}^*)^{\downarrow t,s-t} & (\mathbf{M}^*)^{\downarrow t,t} \end{bmatrix}$$

we derive the following identity using equation (6.17) and (6.24):

$$\mathbf{F} \;=\; \mathbf{M}^{\downarrow t,t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t} \;=\; \mathbf{M}_t$$

and then further:

$$\begin{aligned} (\mathbf{M}^*)^{\downarrow s-t,s-t} &= (\mathbf{M}^{\downarrow s-t,s-t})^* + \\ &\quad\; (\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^* \\ (\mathbf{M}^*)^{\downarrow s-t,t} &= (\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^* \\ (\mathbf{M}^*)^{\downarrow t,s-t} &= (\mathbf{M}_t)^*\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^* \\ (\mathbf{M}^*)^{\downarrow t,t} &= (\mathbf{M}_t)^* \end{aligned} \tag{6.28}$$

If $\mathbf{x} = \mathbf{M}^*\mathbf{b}$ is a solution to $\mathbf{X} = \mathbf{M}\mathbf{X} + \mathbf{b}$ we have

$$\mathbf{M}^*\mathbf{b} \quad = \quad \mathbf{M}\mathbf{M}^*\mathbf{b} + \mathbf{b}.$$

Decomposing the system with respect to $s-t$ and t gives

$$\begin{bmatrix} (\mathbf{M}^*)^{\downarrow s-t,s-t} & (\mathbf{M}^*)^{\downarrow s-t,t} \\ (\mathbf{M}^*)^{\downarrow t,s-t} & (\mathbf{M}^*)^{\downarrow t,t} \end{bmatrix} \begin{bmatrix} \mathbf{b}^{\downarrow s-t} \\ \mathbf{b}^{\downarrow t} \end{bmatrix} \;=\; \begin{bmatrix} \mathbf{b}^{\downarrow s-t} \\ \mathbf{b}^{\downarrow t} \end{bmatrix} +$$

$$\begin{bmatrix} \mathbf{M}^{\downarrow s-t,s-t} & \mathbf{M}^{\downarrow s-t,t} \\ \mathbf{M}^{\downarrow t,s-t} & \mathbf{M}^{\downarrow t,t} \end{bmatrix} \begin{bmatrix} (\mathbf{M}^*)^{\downarrow s-t,s-t} & (\mathbf{M}^*)^{\downarrow s-t,t} \\ (\mathbf{M}^*)^{\downarrow t,s-t} & (\mathbf{M}^*)^{\downarrow t,t} \end{bmatrix} \begin{bmatrix} \mathbf{b}^{\downarrow s-t} \\ \mathbf{b}^{\downarrow t} \end{bmatrix}.$$

We identify the following subsystem for t:

$$\begin{aligned} (\mathbf{M}^*)^{\downarrow t,s-t}\mathbf{b}^{\downarrow s-t} \;+\; (\mathbf{M}^*)^{\downarrow t,t}\mathbf{b}^{\downarrow t} \;&=\; \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^*)^{\downarrow s-t,s-t}\mathbf{b}^{\downarrow s-t} \\ &+\; \mathbf{M}^{\downarrow t,t}(\mathbf{M}^*)^{\downarrow t,s-t}\mathbf{b}^{\downarrow s-t} \\ &+\; \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^*)^{\downarrow s-t,t}\mathbf{b}_t \\ &+\; \mathbf{M}^{\downarrow t,t}(\mathbf{M}^*)^{\downarrow t,t}\mathbf{b}^{\downarrow t} \;+\; \mathbf{b}^{\downarrow t}. \end{aligned}$$

If we insert the above identities for the left-hand side of this equation, we obtain

$$\begin{aligned}(\mathbf{M}^*)^{\downarrow t,s-t}\mathbf{b}^{\downarrow s-t} + (\mathbf{M}^*)^{\downarrow t,t}\mathbf{b}^{\downarrow t} &= (\mathbf{M}_t)^*\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t}\\ &+ (\mathbf{M}_t)^*\mathbf{b}^{\downarrow t}\\ &= (\mathbf{M}_t)^*\left(\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t} + \mathbf{b}^{\downarrow t}\right)\\ &= (\mathbf{M}_t)^*\mathbf{b}_t.\end{aligned}$$

This follows from equation (6.25). Similarly, we obtain for the right-hand part:

$$\begin{aligned}\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^*)^{\downarrow s-t,s-t}\mathbf{b}^{\downarrow s-t} + \mathbf{M}^{\downarrow t,t}(\mathbf{M}^*)^{\downarrow t,s-t}\mathbf{b}^{\downarrow s-t} &+\\ \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^*)^{\downarrow s-t,t}\mathbf{b}^{\downarrow t} + \mathbf{M}^{\downarrow t,t}(\mathbf{M}^*)^{\downarrow t,t}\mathbf{b}^{\downarrow t} + \mathbf{b}^{\downarrow t} &=\\ \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t} &+\\ \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*b^{\downarrow s-t} &+\\ \mathbf{M}^{\downarrow t,t}(\mathbf{M}_t)^*\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t} &+\\ \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\mathbf{b}^{\downarrow t} &+\\ \mathbf{M}^{\downarrow t,t}(\mathbf{M}_t)^*\mathbf{b}^{\downarrow t} + \mathbf{b}^{\downarrow t}&.\end{aligned}$$

Reordering the terms of this expression gives:

$$\begin{aligned}\left(\mathbf{M}^{\downarrow t,t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}\right)(\mathbf{M}_t)^*&\\ \left(\mathbf{b}^{\downarrow t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t}\right) &= \mathbf{M}_t(\mathbf{M}_t)^*\mathbf{b}_t.\end{aligned}$$

The last equality follows from equations (6.24) and (6.25).

$$(\mathbf{M}_t)^*\mathbf{b}_t = \mathbf{M}_t(\mathbf{M}_t)^*\mathbf{b}_t$$

which by Lemma 6.2 proves that $(\mathbf{M}_t)^*\mathbf{b}_t$ is a solution to

$$\mathbf{X}^{\downarrow t} = \mathbf{M}_t\mathbf{X}^{\downarrow t} + \mathbf{b}_t.$$

This proves the first statement of the theorem.

2. It follows from equations (6.25) and (6.28) that

$$\begin{aligned}(\mathbf{M}_t)^*\mathbf{b}_t &= (\mathbf{M}_t)^*\left(\mathbf{b}^{\downarrow t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t}\right)\\ &= (\mathbf{M}_t)^*\mathbf{b}^{\downarrow t} + (\mathbf{M}_t)^*\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t}\\ &= (\mathbf{M}^*)^{\downarrow t,t}\mathbf{b}^{\downarrow t} + (\mathbf{M}^*)^{\downarrow t,s-t}\mathbf{b}^{\downarrow s-t}.\end{aligned}$$

Similarly, we derive

$$
\begin{aligned}
\mathbf{y} \;=\; (\mathbf{M}^{\downarrow s-t,s-t})^*\Big(\mathbf{M}^{\downarrow s-t,t}\mathbf{x} + \mathbf{b}^{\downarrow s-t}\Big) &\;=\\
(\mathbf{M}^{\downarrow s-t,s-t})^*\Big(\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\mathbf{b}_t + \mathbf{b}^{\downarrow s-t}\Big) &\;=\\
(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\mathbf{b}_t + (\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t} &\;=\\
(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\Big(\mathbf{b}^{\downarrow t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t}\Big) &\;+\\
(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{b}^{\downarrow s-t} &\;=\\
(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\mathbf{b}^{\downarrow t} &\;+\\
\Big((\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^* &\;+\\
(\mathbf{M}^{\downarrow s-t,s-t})^*\Big)\mathbf{b}^{\downarrow s-t} &\;=\\
(\mathbf{M}^*)^{\downarrow s-t,t}\mathbf{b}^{\downarrow t} + (\mathbf{M}^*)^{\downarrow s-t,s-t}\mathbf{b}^{\downarrow s-t}. &
\end{aligned}
$$

We therefore obtain:

$$
\begin{aligned}
(\mathbf{y},\mathbf{x}) &= \Big((\mathbf{M}^*)^{\downarrow s-t,t}\mathbf{b}^{\downarrow t} + (\mathbf{M}^*)^{\downarrow s-t,s-t}\mathbf{b}^{\downarrow s-t},\\
&\qquad (\mathbf{M}^*)^{\downarrow t,t}\mathbf{b}^{\downarrow t} + (\mathbf{M}^*)^{\downarrow t,s-t}\mathbf{b}^{\downarrow s-t}\Big)\\
&= \begin{bmatrix} (\mathbf{M}^*)^{\downarrow s-t,s-t} & (\mathbf{M}^*)^{\downarrow s-t,t} \\ (\mathbf{M}^*)^{\downarrow t,s-t} & (\mathbf{M}^*)^{\downarrow t,t} \end{bmatrix} \begin{bmatrix} \mathbf{b}^{\downarrow s-t} \\ \mathbf{b}^{\downarrow t} \end{bmatrix} \;=\; \mathbf{M}^*\mathbf{b}.
\end{aligned}
$$

But according to Lemma 6.2 $\mathbf{M}^*\mathbf{b}$ is always a solution to the fixpoint equation

$$\mathbf{X} \;=\; \mathbf{M}\mathbf{X} + \mathbf{b}.$$

This proves the second statement of the theorem.

■

Subsequently, we use to say that $\mathbf{x}$ is a solution to $(\mathbf{M},\mathbf{b}) \in \Phi$ if $\mathbf{x}$ is a solution to the fixpoint equation $\mathbf{X} = \mathbf{M}\mathbf{X} + \mathbf{b}$. The following lemma highlights a direct consequence of the above theorem:

Lemma 6.4 *If* $\mathbf{x} = \mathbf{M}^*\mathbf{b}$ *is a solution to* $(\mathbf{M},\mathbf{b})$ *and* $t \subseteq s = d(\mathbf{M},\mathbf{b})$ *then*

$$\mathbf{x} \;=\; \Big((\mathbf{M}_t)^*\mathbf{b}_t,\; (\mathbf{M}_{s-t})^*\mathbf{b}_{s-t}\Big).$$

Proof: If $\mathbf{M}^*\mathbf{b}$ is a solution to $(\mathbf{M},\mathbf{b})$, then $(\mathbf{M}_t)^*\mathbf{b}_t$ and $(\mathbf{M}_{s-t})^*\mathbf{b}_{s-t}$ are solutions to $(\mathbf{M}_t,\mathbf{b}_t)$ and $(\mathbf{M}_{s-t},\mathbf{b}_{s-t})$ by the first statement of Theorem 6.3. On the other hand, if $(\mathbf{M}_t)^*\mathbf{b}_t$ is a solution to $(\mathbf{M}_t,\mathbf{b}_t)$ and

$$\mathbf{y}_1 \;=\; \Big(\mathbf{M}^{\downarrow s-t,s-t}\Big)^*\Big(\mathbf{M}^{\downarrow s-t,t}(\mathbf{M}_t)^*\mathbf{b}_t + \mathbf{b}^{\downarrow s-t}\Big),$$

then $((\mathbf{M}_t)^*\mathbf{b}_t, \mathbf{y}_1) = \mathbf{M}^*\mathbf{b}$, which follows from the second statement of Theorem 6.3. Likewise, if $(\mathbf{M}_{s-t})^*\mathbf{b}_{s-t}$ is a solution to $(\mathbf{M}_{s-t}, \mathbf{b}_{s-t})$, there exists an extension $\mathbf{y}_2$ such that $(\mathbf{y}_2, (\mathbf{M}_{s-t})^*\mathbf{b}_{s-t}) = \mathbf{M}^*\mathbf{b}$. We conclude from

$$\mathbf{x} \;=\; \Big((\mathbf{M}_t)^*\mathbf{b}_t,\, \mathbf{y}_1\Big) \;=\; \Big(\mathbf{y}_2,\, (\mathbf{M}_{s-t})^*\mathbf{b}_{s-t}\Big) \;=\; \mathbf{M}^*\mathbf{b}$$

that $\mathbf{y}_1 = (\mathbf{M}_{s-t})^*\mathbf{b}_{s-t}$ and $\mathbf{y}_2 = (\mathbf{M}_t)^*\mathbf{b}_t$. ■

We next prove that the projection operator is transitive.

Lemma 6.5 *For* $(\mathbf{M}, \mathbf{b}) \in \Phi$ *with* $d(\mathbf{M}, \mathbf{b}) = s$ *and* $t \subseteq u \subseteq s$ *we have*

$$(\mathbf{M}, \mathbf{b})^{\downarrow t} \;=\; \Big((\mathbf{M}, \mathbf{b})^{\downarrow u}\Big)^{\downarrow t}. \tag{6.29}$$

Proof: Since projection is defined component-wise in equation (6.23), we can prove the statement for $\mathbf{M}$ and $\mathbf{b}$ separately. Thus, consider the following decomposition:

$$\mathbf{M} \;=\; \begin{bmatrix} \mathbf{M}^{\downarrow s-u,s-u} & \mathbf{M}^{\downarrow s-u,u-t} & \mathbf{M}^{\downarrow s-u,t} \\ \mathbf{M}^{\downarrow u-t,s-u} & \mathbf{M}^{\downarrow u-t,u-t} & \mathbf{M}^{\downarrow u-t,t} \\ \mathbf{M}^{\downarrow t,s-u} & \mathbf{M}^{\downarrow t,u-t} & \mathbf{M}^{\downarrow t,t} \end{bmatrix}.$$

Applying equation (6.24) gives

$$\mathbf{M}_u \;=\; \mathbf{M}^{\downarrow u,u} + \mathbf{M}^{\downarrow u,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u}$$

which corresponds to

$$\begin{bmatrix} \mathbf{M}^{\downarrow u-t,u-t} & \mathbf{M}^{\downarrow u-t,t} \\ \mathbf{M}^{\downarrow t,u-t} & \mathbf{M}^{\downarrow t,t} \end{bmatrix} + \begin{bmatrix} \mathbf{M}^{\downarrow u-t,s-u} \\ \mathbf{M}^{\downarrow t,s-u} \end{bmatrix} (\mathbf{M}^{\downarrow s-u,s-u})^* \begin{bmatrix} \mathbf{M}^{\downarrow s-u,u-t} & \mathbf{M}^{\downarrow s-u,t} \end{bmatrix}.$$

Executing the second projection to $t \subseteq u$ gives

$$\begin{aligned}\Big(\mathbf{M}_u\Big)_t \;=\; & \Big(\mathbf{M}^{\downarrow t,t} + \mathbf{M}^{\downarrow t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,t}\Big) + \\ & \Big(\mathbf{M}^{\downarrow t,u-t} + \mathbf{M}^{\downarrow t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u-t}\Big) \\ & \Big(\mathbf{M}^{\downarrow u-t,u-t} + \mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u-t}\Big)^* \\ & \Big(\mathbf{M}^{\downarrow u-t,t} + \mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,t}\Big).\end{aligned}$$

We define

$$\mathbf{F}^* \;=\; \Big(\mathbf{M}^{\downarrow u-t,u-t} + \mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u-t}\Big)^*$$

and obtain by rearranging the above expression

$$\begin{aligned}
\Big(\mathbf{M}_u\Big)_t &= \mathbf{M}^{\downarrow t,t} + \mathbf{M}^{\downarrow t,s-u}\Big((\mathbf{M}^{\downarrow s-u,s-u})^* \\
&+ (\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u-t}\mathbf{F}^*\mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\Big)\mathbf{M}^{\downarrow s-u,t} \\
&+ \mathbf{M}^{\downarrow t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u-t}\mathbf{F}^*\mathbf{M}^{\downarrow u-t,t} \\
&+ \mathbf{M}^{\downarrow t,u-t}\mathbf{F}^*\mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,t} \\
&+ \mathbf{M}^{\downarrow t,u-t}\mathbf{F}^*\mathbf{M}^{\downarrow u-t,t}.
\end{aligned}$$

Using equation (6.17) we further conclude that

$$\begin{aligned}
\Big(\mathbf{M}_u\Big)_t &= \mathbf{M}^{\downarrow t,t} \\
&+ \mathbf{M}^{\downarrow t,s-u}\Big((\mathbf{M}^{\downarrow s-t,s-t})^*\Big)^{\downarrow s-u,s-u}\mathbf{M}^{\downarrow s-u,t} \\
&+ \mathbf{M}^{\downarrow t,s-u}\Big((\mathbf{M}^{\downarrow s-t,s-t})^*\Big)^{\downarrow s-u,u-t}\mathbf{M}^{\downarrow u-t,t} \\
&+ \mathbf{M}^{\downarrow t,u-t}\Big((\mathbf{M}^{\downarrow s-t,s-t})^*\Big)^{\downarrow u-t,s-t}\mathbf{M}^{\downarrow s-u,t} \\
&+ \mathbf{M}^{\downarrow t,u-t}\Big((\mathbf{M}^{\downarrow s-t,s-t})^*\Big)^{\downarrow u-t,u-t}\mathbf{M}^{\downarrow u-t,t}
\end{aligned}$$

and because $s - t = (s - u) \cup (u - t)$ we finally obtain

$$\begin{aligned}
\Big(\mathbf{M}_u\Big)_t &= \mathbf{M}^{\downarrow t,t} + \begin{bmatrix} \mathbf{M}^{\downarrow t,s-u} & \mathbf{M}^{\downarrow t,u-t} \end{bmatrix} (\mathbf{M}^{\downarrow s-t,s-t})^* \begin{bmatrix} \mathbf{M}^{\downarrow s-u,t} \\ \mathbf{M}^{\downarrow u-t,t} \end{bmatrix} \\
&= \mathbf{M}^{\downarrow t,t} + \mathbf{M}^{\downarrow t,s-t}(\mathbf{M}^{\downarrow s-t,s-t})^*\mathbf{M}^{\downarrow s-t,t} \;=\; \mathbf{M}_t.
\end{aligned}$$

A similar derivation proves the statement for the vector component. ■

Complementary to the operation of projection for the elements Φ, we next introduce an operation of vacuous extension. For $(\mathbf{M}, \mathbf{b}) \in \Phi$ with $d(\mathbf{M}, \mathbf{b}) = s$ and $t \supseteq s$, this operation corresponds for both components to the operation of matrix and tuple extension introduced in Chapter 1. However, instead of assigning the number $0 \in \mathbb{R}$ to the variables in $t - s$, we take the zero element $\mathbf{0} \in A$ of the quasi-regular semiring. We therefore have

$$(\mathbf{M}, \mathbf{b})^{\uparrow t} \quad = \quad (\mathbf{M}^{\uparrow t,t}, \mathbf{b}^{\uparrow t}).$$

This operation is used for the introduction of combination: Assume $(\mathbf{M}_1, \mathbf{b}_1) \in \Phi$ and $(\mathbf{M}_2, \mathbf{b}_2) \in \Phi$ with $d(\mathbf{M}_1, \mathbf{b}_1) = s$ and $d(\mathbf{M}_2, \mathbf{b}_2) = t$. We define

$$\begin{aligned}
(\mathbf{M}_1, \mathbf{b}_1) \otimes (\mathbf{M}_2, \mathbf{b}_2) &= (\mathbf{M}_1, \mathbf{b}_1)^{\uparrow s\cup t} + (\mathbf{M}_2, \mathbf{b}_2)^{\uparrow s\cup t} \qquad (6.30) \\
&= (\mathbf{M}_1^{\uparrow s\cup t, s\cup t} + \mathbf{M}_2^{\uparrow s\cup t, s\cup t}, \ \mathbf{b}_1^{\uparrow s\cup t} + \mathbf{b}_2^{\uparrow s\cup t}).
\end{aligned}$$

The operation of combination just extends both matrices and vectors to their union domain and performs a component-wise addition. We show next that the combination axiom is satisfied by this definition.

Lemma 6.6 *For* $(\mathbf{M}_1, \mathbf{b}_1), (\mathbf{M}_2, \mathbf{b}_2) \in \Phi$ *with* $d(\mathbf{M}_1, \mathbf{b}_1) = s$, $d(\mathbf{M}_2, \mathbf{b}_2) = t$ *and* $s \subseteq z \subseteq s \cup t$ *it holds that*

$$\Big((\mathbf{M}_1, \mathbf{b}_1) \otimes (\mathbf{M}_2, \mathbf{b}_2)\Big)^{\downarrow z} \quad = \quad (\mathbf{M}_1, \mathbf{b}_1) \otimes (\mathbf{M}_2, \mathbf{b}_2)^{\downarrow z \cap t}.$$

Proof: We again consider the statement for both components separately. Consider the following decomposition with respect to z and $(s \cup t) - z = t - z$ and $z \cap t$. Here, $\mathbf{O} \in \mathcal{M}(A, s \cup t)$ denotes the zero matrix for the domain $s \cup t$.

$$\mathbf{M}_1^{\uparrow s\cup t, s\cup t} + \mathbf{M}_2^{\uparrow s\cup t, s\cup t} \quad = \quad \begin{bmatrix} \multicolumn{2}{c}{\mathbf{M}_1^{\uparrow z,z} + (\mathbf{M}_2^{\downarrow z\cap t, z\cap t})^{\uparrow z,z}} & \mathbf{O}^{\downarrow z-t,t-z} \\ & & \mathbf{M}_2^{\downarrow z\cap t,t-z} \\ \mathbf{O}^{\downarrow t-z,z-t} & \mathbf{M}_2^{\downarrow t-z,z\cap t} & \mathbf{M}_2^{\downarrow t-z,t-z} \end{bmatrix}.$$

Applying equation (6.24) gives

$$\begin{aligned} \left(\mathbf{M}_1^{\uparrow s\cup t, s\cup t} + \mathbf{M}_2^{\uparrow s\cup t, s\cup t}\right)^{\downarrow z} \quad &= \quad \mathbf{M}_1^{\uparrow z,z} + (\mathbf{M}_2^{\downarrow z\cap t, z\cap t})^{\uparrow z,z} + \begin{bmatrix} \mathbf{O}^{\downarrow z-t,t-z} \\ \mathbf{M}_2^{\downarrow z\cap t,t-z} \end{bmatrix} \\ & \quad\quad (\mathbf{M}_2^{\downarrow t-z,t-z})^* \begin{bmatrix} \mathbf{O}^{\downarrow t-z,z-t} & \mathbf{M}_2^{\downarrow t-z,z\cap t} \end{bmatrix} \\ &= \quad \mathbf{M}_1^{\uparrow z,z} + \Big((\mathbf{M}_2^{\downarrow z\cap t, z\cap t})^{\uparrow z,z} \\ &+ \quad (\mathbf{M}_2^{\downarrow z\cap t,t-z}(\mathbf{M}_2^{\downarrow t-z,t-z})^* \mathbf{M}_2^{\downarrow t-z,z\cap t})^{\uparrow z,z}\Big) \\ &= \quad \mathbf{M}_1^{\uparrow z,z} + \Big((\mathbf{M}_2^{\downarrow z\cap t, z\cap t}) \\ &+ \quad \mathbf{M}_2^{\downarrow z\cap t,t-z}(\mathbf{M}_2^{\downarrow t-z,t-z})^* \mathbf{M}_2^{\downarrow t-z,z\cap t}\Big)^{\uparrow z,z} \\ &= \quad \mathbf{M}_1^{\uparrow z,z} + (\mathbf{M}_{2\ z\cap t})^{\uparrow z,z}. \end{aligned}$$

This proves that the combination axiom holds for the matrix components and a similar derivation applies to the vector components. ∎

All these properties imply the following theorem if D denotes the lattice of finite subsets of the countable set r of variables:

Theorem 6.4 *The system* $\langle \Phi, D \rangle$ *with labeling, projection (6.23) and combination (6.30) satisfies the axioms of a valuation algebra.*

Proof: Since combination reduces to semiring addition, we conclude that Φ is a commutative semigroup under combination. Further, the axioms (A2), (A3) and (A6) follow directly from the definition of projection. Transitivity (A4) is proved in Lemma 6.5 and the combination axiom (A5) corresponds to Lemma 6.6. ∎

How quasi-regular valuation algebras are used to solve path problems will be discussed in Section 9.3.2. There, we also compare this approach with the traditional way of solving path problems and with another generic construction that will be presented in Section 6.7 of this chapter. Knowing that labeled fixpoint equations over quasi-regular semirings satisfy the valuation algebra axioms allows us to apply the local computation architectures of Chapter 3 and 4 for their processing. But some architectures require additional properties such as, for example, the presence of neutral elements or idempotency of combination. Therefore, we next look for such properties in quasi-regular valuation algebras.

6.5 PROPERTIES OF QUASI-REGULAR VALUATION ALGEBRAS

Since combination simply corresponds to matrix addition, we directly identify the neutral element for the domain $s \subseteq r$ by the pair $(\mathbf{O}, \mathbf{o})$ of null matrix $\mathbf{O} \in \mathcal{M}(A, s)$ and null vector $\mathbf{o} : s \rightarrow A$ such that $\mathbf{o}(X) = \mathbf{0}$ for all $X \in s$. It is clear that this element behaves neutrally with respect to all other valuations of domain s and also that the combination of two neutral elements again results in a neutral element. Moreover, it follows directly from equation (6.23) that neural elements project to neutral elements. Together, this proves the following lemma.

Lemma 6.7 *Quasi-regular valuation algebras have neutral elements and are stable.*

In contrast to neutral elements, quasi-regular valuation algebras do not provide null elements, although the supplementary semiring property $a + \mathbf{1} = \mathbf{1}$ for all $a \in A$ would in fact be sufficient to obtain an absorbing element for each domain. Such semirings were called bounded or simple semirings in Section 5.2. The corresponding absorbing elements also project to absorbing elements as a consequence of the sum in the definition of projection. But this sum also makes it possible that non-absorbing elements may project to absorbing elements which contradicts the nullity axiom of Section 3.4. Second, we also point out that quasi-regular valuation algebras are not idempotent, not even when semiring addition is idempotent due to the quasi-inverse in the definition of projection. These considerations show that quasi-regular valuation algebras do not provide a lot of interesting properties and are therefore rather poor in structure. First and foremost, it is the property of idempotency that would be desirable, since it enables the simple local computation architecture of Section 4.5 which is especially suited for valuation algebras with polynomial time and space complexity. But we have just seen that imposing additional semiring properties such as idempotent addition is not sufficient to obtain an idempotent valuation algebra. In Section 6.7, we therefore introduce another family of valuation algebras for the solution of path problems which, besides tackling the all-pairs algebraic path problem directly, are always idempotent. This is achieved by moving the computational effort from the projection to the combination operation. However, the prize we pay is that more algebraic structure is needed in the underlying semiring as it is the case in a *Kleene algebra* presented in the following section. We will therefore dispose of two different families of valuation algebras for the solution of path problems. How path problems are solved with local computation, and how the two approaches differ from each other will be discussed in Chapter 9.

6.6 KLEENE ALGEBRAS

This section introduces a special family of quasi-regular semirings called Kleene algebras (Conway, 1971; Kozen, 1990) which have many interesting properties. First, Kleene algebras are idempotent semirings and therefore partially ordered. Based on this order, they further satisfy a monotonicity law with respect to the star operation to guarantee that it always returns the least solution to the fixpoint equation (6.11). This provides the yet missing interpretation of quasi-inverse elements as a solution of path problems. Another important consequence of the monotonicity property is that the star operation in a Kleene algebra satisfies the axioms of a *closure operator* (Davey & Priestley, 1990). This important insight enables to derive a new generic construction based on matrix closures which always produces idempotent valuation algebras. However, let us first give a formal introduction to Kleene algebras and derive their most important properties:

Definition 6.6 *A tuple* $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ *is called Kleene algebra if:*

- $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *is an idempotent semiring;*
- $\mathbf{1} + aa^* \leq a^*$ *for* $a \in A$; *(K1)*
- $\mathbf{1} + a^*a \leq a^*$ *for* $a \in A$; *(K2)*
- $ax \leq x$ *implies that* $a^*x \leq x$ *for* $a, x \in A$; *(K3)*
- $xa \leq x$ *implies that* $xa^* \leq x$ *for* $a, x \in A$. *(K4)*

Here, the relation $\leq$ refers to the canonical partial order of an idempotent semiring defined in equation (5.4). The following properties are immediate consequences of the above axioms:

Lemma 6.8 *In a Kleene algebra* $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ *we have:*

1. $\mathbf{1} \leq a^*$ *for all* $a \in A$; *(KP1)*
2. $a \leq a^*$ *for all* $a \in A$; *(KP2)*
3. $\mathbf{0}^* = \mathbf{1} = \mathbf{1}^*$; *(KP3)*
4. $a^*a^* = a^*$ *for all* $a \in A$; *(KP4)*
5. $(a^*)^* = a^*$ *for all* $a \in A$; *(KP5)*
6. $a \leq b$ *implies that* $a^* \leq b^*$ *for all* $a, b \in A$. *(KP6)*

Proof:

1. From (SP3) of Lemma 5.2 and Axiom (K1) we conclude $\mathbf{1} \leq \mathbf{1} + aa^* \leq a^*$.
2. By (SP2), (SP3) and Axiom (K1), $\mathbf{1} \leq a^*$ implies $a \leq aa^* \leq aa^* + \mathbf{1} \leq a^*$.
3. $\mathbf{1} \leq \mathbf{0}^*$ follows from (KP1) and $\mathbf{0}^* \leq \mathbf{1}$ from Axiom (K3) taking $a = \mathbf{0}$ and $x = \mathbf{1}$. So, $\mathbf{0}^* = \mathbf{1}$. Further, we conclude from (KP2), $\mathbf{1} \leq \mathbf{1}^*$ and $\mathbf{1}^* \leq \mathbf{1}$ follows again from Axiom (K3) taking $a = x = \mathbf{1}$, hence $\mathbf{1}^* = \mathbf{1}$.
4. Using (SP3) and (K1) we derive $aa^* \leq \mathbf{1} + aa^* \leq a^*$. From Axiom (K3) it follows that $aa^* \leq a^*$ implies $a^*a^* \leq a^*$. Conversely, we have $\mathbf{1} \leq a^*$ by (KP1) and thus $a^* \leq a^*a^*$ by (SP2).
5. From (KP2) follows that $a^* \leq (a^*)^*$. Further, we derive from (KP4) and Axiom (K3) that $a^*a^* \leq a^*$ implies $(a^*)^*a^* \leq a^*$. We finally conclude from (KP1) and (SP2) that $\mathbf{1} \leq a^*$ implies $(a^*)^* \leq (a^*)^*a^* \leq a^*$. Hence, it follows that $(a^*)^* = a^*$.
6. Let $a \leq b$ and thus $a \leq b^*$ due to (KP2). Using (SP2) and (KP4) we then have $ab^* \leq b^*b^* = b^*$ which implies $a^*b^* \leq b^*$ by (K3). Finally, we conclude from (KP1) and (SP2) that $a^* = a^*\mathbf{1} \leq a^*b^* \leq b^*$.

■

We will next see that the star operation in a Kleene algebra is uniquely determined.

Lemma 6.9 *For each element $a \in A$ in a Kleene algebra $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$, there is a unique element $a^* \in A$ that satisfies (K1) to (K4).*

Proof: Let $r \in A$ be another element for a that satisfies (K1) to (K4). From (K1) and (SP3) we conclude that $ar \leq r$ which in turn implies $a^*r \leq r$ by Axiom (K3). Since $\mathbf{1} \leq r$ we obtain using (SP2) that $a^* \leq a^*r \leq r$ and therefore $a^* \leq r$. On the other hand, we also have $aa^* \leq a^*$ from (K1) and (SP3) which implies that $ra^* \leq a^*$ by (K3). Since $\mathbf{1} \leq a^*$ we conclude using (SP2) that $r \leq ra^* \leq a^*$ and therefore $r \leq a^*$. Together, the two arguments prove $r = a^*$. ■

Lemma 6.10 *In a Kleene algebra it always holds that*

$$(a+b)^* \quad = \quad a^*(ba^*)^*.$$

Proof: From (SP3) and (KP6) we derive

$$\begin{aligned} a^*(ba^*)^* &\leq (a+b)^*((a+b)(a+b)^*)^* \\ &\leq (a+b)^*((a+b)^*)^* \;=\; (a+b)^*(a+b)^* \;=\; (a+b)^*. \end{aligned}$$

The second inequality follows from (K1) and (SP3), and the two equalities from (KP5) and (KP4). On the other hand, (K1) and (SP3) imply that $aa^* \leq a^*$, hence $aa^*(ba^*)^* \leq a^*(ba^*)^*$ by (SP2). $aa^* \leq a^*$ also implies $ba^*(ba^*)^* \leq (ba^*)^*$ which together with (KP1) gives $ba^*(ba^*)^* \leq a^*(ba^*)^*$. Adding the two results gives:

$$aa^*(ba^*)^* + ba^*(ba^*)^* \quad \leq \quad a^*(ba^*)^* + a^*(ba^*)^*$$

thus by distributivity and idempotency

$$(a+b)a^*(ba^*)^* \quad \leq \quad a^*(ba^*)^*$$

which due to (K3) implies

$$(a+b)^*a^*(ba^*)^* \quad \leq \quad a^*(ba^*)^*.$$

Finally, we conclude from $\mathbf{1} \leq a^*(ba^*)^*$ that

$$(a+b)^* \;\leq\; (a+b)^*a^*(ba^*)^* \;\leq\; a^*(ba^*)^*.$$

The statement of Lemma 6.10 follows now by antisymmetry. ■

Lemma 6.11 *In a Kleene algebra it always holds that*

$$(a+b)^* \;=\; (a^*+b)^* \;=\; (a+b^*)^* \;=\; (a^*+b^*)^*.$$

Proof: Using Lemma 6.10 and Property (KP5) we derive

$$(a+b)^* \;=\; a^*(ba^*)^* \;=\; a^{**}(ba^{**})^* \;=\; (a^*+b)^*.$$

The other equalities follow directly from the commutativity of addition. ■

The following lemma states what we have already mentioned in the introduction of this section. Namely, that Kleene algebras are quasi-regular semirings.

Lemma 6.12 *In a Kleene algebra we have for all* $a \in A$

$$a^* \;=\; aa^*+\mathbf{1} \;=\; a^*a+\mathbf{1}.$$

Proof: Using (KP1), (SP1) and (SP2) we have $\mathbf{1}+aa^* \leq a^*$ implies that $a(\mathbf{1}+aa^*) \leq aa^*$ and then $\mathbf{1}+a(\mathbf{1}+aa^*) \leq (\mathbf{1}+aa^*)$. We conclude from (SP3) that $a(\mathbf{1}+aa^*) \leq (\mathbf{1}+aa^*)$. Applying (K3) we obtain $a^*(\mathbf{1}+aa^*) \leq (\mathbf{1}+aa^*)$ and finally derive from $\mathbf{1} \leq \mathbf{1}+aa^*$ that $a^* \leq a^*(\mathbf{1}+aa^*) \leq (\mathbf{1}+aa^*)$. It follows that $a^* \leq \mathbf{1}+aa^*$ which together with (K1) implies equality. The second statement is proved in a similar manner. ■

Thus, the star operation always provides a solution to the fixpoint equation (6.11) which allows us to consider Kleene algebras as special cases of quasi-regular semirings. We again point out that multiple solutions to this equation may exist in a Kleene algebra, but due to Lemma 6.9, a^* is the only solution among them that satisfies the properties (K3) and (K4). As shown in the following lemma, this implies that a^* is the least element of the solution set.

Lemma 6.13 *The element a^* in a Kleene algebra is the least solution to*

$$X \;=\; aX+\mathbf{1} \quad \text{and} \quad X \;=\; Xa+\mathbf{1}.$$

Proof: We know from Lemma 6.12 that a^* is a solution to the two fixpoint equations. Let $r \in A$ be another solution such that $r = ar+\mathbf{1}$. From (SP3) we conclude $ar \leq r$ which implies $a^*r \leq r$ by (K3). Since $\mathbf{1} \leq r$ we obtain $a^* \leq a^*r \leq r$ by (SP2). ■

As a consequence of idempotency, all Kleene algebras are dioids. If we further assume that a particular Kleene algebra is a topological dioid according to Definition 6.1 where the power sequence of equation (6.7) converges, then its limit is equal to the result of the star operation. The converse direction of this statement is not true: (Kozen, 1990) gives a very artificial example of a Kleene algebra whose star operation does not correspond to the limit of the corresponding power sequence. However, besides being quasi-regular, all semirings of Example 6.7 are idempotent, and it can be shown that they all are topological dioids. Further, Example 6.10 identifies them as Kleene algebras under the particular definition of the star operation. This finally explains why the algebraic path problem of Definition 6.3 applied to these semirings results in minimum distances, maximum capacities, maximum reliabilities, etc.

Example 6.10 *All quasi-regular semirings of Example 6.7 are idempotent and their star operation satisfies (K1) and (K2). We therefore only focus on the two axiom (K3) and (K4). Let us look at the tropical semiring in more detail: Taking non-negative numbers* $\langle \mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +, *, \infty, 0\rangle$ *with* $a^* = 0$ *property (K3) is always satisfied since* $a^* + x \leq x$ *trivially holds. If we take all real numbers* $\langle \mathbb{R} \cup \{-\infty, \infty\}, \min, +, *, \infty, 0\rangle$ *with* $a^* = 0$ *for* $a \geq 0$ *and* $a^* = -\infty$ *for* $a < 0$ *the property does still hold: For* $a < 0$ *we always have* $a^* + x = -\infty + x \leq x$*, for* $a = 0$ *the statement holds as above and for* $a \geq 0$*, the assumption* $a + x \leq x$ *cannot be satisfied. These tropical semirings are therefore Kleene algebras and we can show in the same manner that all other semirings from Example 6.7 are Kleene algebras too.*

Example 6.11 *A important example of a quasi-regular semiring that is not a Kleene algebra is given by the arithmetic semiring* $\langle \mathbb{N} \cup \{0, \infty\}, +, \cdot, *, 0, 1\rangle$ *of non-negative integers. To make sure that* 0 *is indeed the zero element of this semiring, we define* $a \times \infty = \infty$ *for all* $a > 0$ *and* $0 \times \infty = 0$*. Then, the star operation is defined as* $a^* = \infty$ *for all* $a > 0$ *and* $0^* = 1$*. It can easily be shown that this satisfies the definition of a quasi-regular semiring. Alternatively, we may take the arithmetic semiring* $\langle \mathbb{R} \cup \{\infty\}, +, \cdot, *, 0, 1\rangle$ *of real numbers with the definition*

$$a^* = \frac{1}{1-a}$$

for $a \neq 1$ *and* $a^* = \infty$ *for* $a = 1$*. This again fulfills the requirements for a quasi-regular semiring and also explains why* a^* *is called a quasi-inverse. We observe that both semirings are not idempotent and can therefore not be Kleene algebras.*

6.6.1 Matrices over Kleene Algebras

Let us now consider matrices with values from a Kleene algebra. We already observed in Section 5.1.2 that matrices over idempotent semirings again form an idempotent semiring. Further, all Kleene algebras are quasi-regular, which implies by Theorem 6.2 that the result of the construction (6.17) applied to matrices with values from a Kleene algebra satisfies axiom (K1) and (K2). It has further been shown in (Conway, 1971) that they also respect the two monotonicity properties (K3) and (K4) which altogether proves the following theorem:

Theorem 6.5 *Let $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ be a Kleene algebra. Then, the semiring of $\mathcal{M}(A, n)$ with the star operation of equation (6.17) also forms a Kleene algebra.*

Due to the many algebraic properties of Kleene algebras, there are several definitions of the star operation for matrices which are all equivalent to equation (6.17). Here, we give one such alternative definition proposed by (Kozen, 1994) that adopts the nice graphical interpretation of Example 6.12 below.

- For $n = 1$ we define $[a]^* = [a^*]$.
- For $n > 1$ we decompose the matrix $\mathbf{M}$ into submatrices $\mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}$ such that $\mathbf{B}$ and $\mathbf{E}$ are square. We define

$$\mathbf{M}^* = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}^* = \begin{bmatrix} \mathbf{F}^* & \mathbf{F}^*\mathbf{C}\mathbf{E}^* \\ \mathbf{E}^*\mathbf{D}\mathbf{F}^* & \mathbf{E}^* + \mathbf{E}^*\mathbf{D}\mathbf{F}^*\mathbf{C}\mathbf{E}^* \end{bmatrix} \tag{6.31}$$

  where $\mathbf{F} = \mathbf{B} + \mathbf{C}\mathbf{E}^*\mathbf{D}$.

Lemma 6.14 *In case of a Kleene algebra, the two definitions of $\mathbf{M}^*$ in the equations (6.17) and (6.31) are identical.*

We propose the proof of this lemma as Exercise F.4 to the reader.

Example 6.12 *Let us first write equation (6.31) for the case $n = 2$:*

$$\mathbf{M}^* = \begin{bmatrix} b & c \\ d & e \end{bmatrix}^* = \begin{bmatrix} f^* & f^*ce^* \\ e^*df^* & e^* + e^*df^*ce^* \end{bmatrix} \tag{6.32}$$

with $f = b + ce^*d$. Then, consider the automaton of Figure 6.8. $\mathbf{M}^*(1,1) = f^*$ represents all possible strings that bring the automaton from state 1 back into state 1, i.e. we may either go directly to 1 using b, or change to state 2 with symbol c, then repeat symbol e an arbitrary number of times, which is expressed by e^*, and finally return to state 1 with symbol d. This whole procedure can also be repeated indefinitely which altogether gives $(b + ce^*d)^* = f^* = \mathbf{M}^*(1,1)$. The other components of $\mathbf{M}^*$ are interpreted in a similar manner.

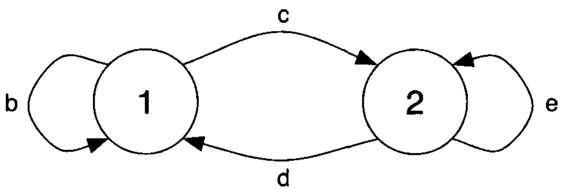


Figure 6.8 An interpretation of equation (6.32).

The following property of matrices over Kleene algebras will later be useful:

Lemma 6.15 *Assume*

$$\mathbf{M}^* = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}.$$

Then, it holds that $\mathbf{B} = \mathbf{B} + \mathbf{C}\mathbf{E}^*\mathbf{D}$ *or equivalently that* $\mathbf{C}\mathbf{E}^*\mathbf{D} \leq \mathbf{B}$.

Proof: From (KP5) follows that $\mathbf{M}^* = \mathbf{M}^{**}$ and therefore $\mathbf{B} = (\mathbf{B} + \mathbf{C}\mathbf{E}^*\mathbf{D})^*$ by equation (6.31). Using (SP3) and (KP2) we then have $\mathbf{B} \leq \mathbf{B} + \mathbf{C}\mathbf{E}^*\mathbf{D} \leq (\mathbf{B} + \mathbf{C}\mathbf{E}^*\mathbf{D})^* = \mathbf{B}$ and therefore $\mathbf{B} = \mathbf{B} + \mathbf{C}\mathbf{E}^*\mathbf{D}$. ■

The proof of this lemma is based on a particular property of Kleene algebras that henceforth becomes important. Namely, that the star operation satisfies the three axioms of a *closure operator* (Davey & Priestley, 1990). It is therefore common to refer to a^* as the *closure* of $a \in A$. In terms of Kleene algebras, the properties of a closure operator are:

1. $a \leq a^*$,
2. $a \leq b$ implies that $a^* \leq b^*$,
3. $a^* = a^{**}$.

They follow directly from (KP2), (KP5) and (KP6). Subsequently, we are going to derive a new generic construction where each valuation corresponds to the closure of a matrix with values from a Kleene algebra. This generates a whole family of new valuation algebras called *Kleene valuation algebras*. The principal idea behind this construction is motivated from graph related applications as for example from the shortest distance problem. Here, it is obvious that the adjacency matrices of multiple graphs may share the same matrix closure, i.e. different graphs defined over the same set of vertices may have the same shortest distances. Let us explore this observation in more detail by reconsidering square, labeled matrices. Thus, let r be a set of variables, $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ a Kleene algebra, $\mathbf{M}_1 \in \mathcal{M}(A, s)$ and $\mathbf{M}_2 \in \mathcal{M}(A, t)$ with $s, t \subseteq r$. It is then easy to show that the following relation is an equivalence relation between labeled matrices:

$$\mathbf{M}_1 \equiv \mathbf{M}_2 \quad \text{if, and only if,} \quad d(\mathbf{M}_1) = d(\mathbf{M}_2) \text{ and } \mathbf{M}_1^* = \mathbf{M}_2^*. \tag{6.33}$$

This equivalence relation decomposes the set of labeled matrices into disjoint equivalence classes of matrices with equal domains and closures. Moreover, since each closure belongs itself to a different equivalence class, they can be considered as representatives of their associated classes. Thus, defining closures as valuations allows us to compute with a unique representation of graph related knowledge or information.

Let us next introduce some operations to manipulate labeled matrices: For a Kleene algebra $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ the *direct sum* of $\mathbf{M}_1 \in \mathcal{M}(A, s)$ and $\mathbf{M}_2 \in \mathcal{M}(A, t)$ with

$s \cap t = \emptyset$ and $X, Y \in s \cup t$ is defined as

$$(\mathbf{M}_1 \oplus \mathbf{M}_2)(X,Y) = \begin{cases} \mathbf{M}_1(X,Y) & \text{if } X,Y \in s, \\ \mathbf{M}_2(X,Y) & \text{if } X,Y \in t, \\ \mathbf{0} & \text{otherwise.} \end{cases} \tag{6.34}$$

This operation satisfies the distributive law with respect to the closure operation:

Lemma 6.16 *It holds that*

$$(\mathbf{M}_1 \oplus \mathbf{M}_2)^* = \mathbf{M}_1^* \oplus \mathbf{M}_2^*.$$

Proof: Let $\mathbf{O}_1 \in \mathcal{M}(A, s \times t)$ and $\mathbf{O}_2 \in \mathcal{M}(A, t \times s)$ be two zero matrices for the domains $s \times t$ and $t \times s$. Applying equation (6.31) gives

$$(\mathbf{M}_1 \oplus \mathbf{M}_2)^* = \begin{bmatrix} \mathbf{M}_1 & \mathbf{O}_1 \\ \mathbf{O}_2 & \mathbf{M}_2 \end{bmatrix}^* = \begin{bmatrix} \mathbf{M}_1^* & \mathbf{O}_1 \\ \mathbf{O}_2 & \mathbf{M}_2^* \end{bmatrix} = \mathbf{M}_1^* \oplus \mathbf{M}_2^*.$$

■

Next, we reconsider the usual projection operator for labeled matrices, but since we only deal with square matrices, we introduce the following shorthand notation: For $\mathbf{M} \in \mathcal{M}(A,s)$ and $t \subseteq s$ we define

$$\mathbf{M}^{\downarrow t} = \mathbf{M}^{\downarrow t,t}. \tag{6.35}$$

However, in the context of Kleene valuation algebras, we need to redefine the operation of vacuous extension for labeled matrices. Instead of assigning the zero element of the semiring to all the new components, we assign the unit element to the diagonal elements and the zero element to all other components. Using the direct sum of matrices, this operation can be defined as follows: For $\mathbf{M} \in \mathcal{M}(A,s)$ and $s \subseteq t$

$$\mathbf{M}^{\uparrow t} = \mathbf{M} \oplus \mathbf{I}. \tag{6.36}$$

where $\mathbf{I} \in \mathcal{M}(A, t-s)$ is the unit matrix for the domain $t-s$. The following lemma states that the application of the closure operation and vacuous extension are interchangeable.

Lemma 6.17 *For* $\mathbf{M} \in \mathcal{M}(A,s)$ *and* $s \subseteq t$ *we have*

$$\left(\mathbf{M}^{\uparrow t}\right)^* = \left(\mathbf{M}^*\right)^{\uparrow t}.$$

Proof: Due to Lemma 6.16 and (KP3), which implies $\mathbf{I} = \mathbf{I}^*$, we have

$$\left(\mathbf{M}^{\uparrow t}\right)^* = \left(\mathbf{M} \oplus \mathbf{I}\right)^* = \left(\mathbf{M}^* \oplus \mathbf{I}^*\right) = \left(\mathbf{M}^* \oplus \mathbf{I}\right) = \left(\mathbf{M}^*\right)^{\uparrow t}.$$

■

6.7 KLEENE VALUATION ALGEBRAS

We are going to show in this section that closures of labeled matrices over a Kleene algebra $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ satisfy the axioms of a valuation algebra. For a countable set of variables r and its lattice of finite subsets D we first define the set of matrix closures as

$$\Phi \quad = \quad \{\mathbf{M}^* \mid \mathbf{M} \in \mathcal{M}(A,s) \text{ and } s \in D\}. \tag{6.37}$$

Under the instantiation of closure matrices as shortest distances in a graph, the operation of projection simply corresponds to dropping distances. On the other hand, vacuous extension corresponds to adding new graph nodes which cannot be accessed from other nodes. It is therefore clear that both operations do not affect other distances. This observation is generalized by the following two lemmas.

Lemma 6.18 *Φ is closed under projection.*

Proof: We show that for $\mathbf{M}^* \in \Phi$ and $s \subseteq d(\mathbf{M}^*) = t$ it holds that

$$(\mathbf{M}^*)^{\downarrow s} \quad = \quad \left((\mathbf{M}^*)^{\downarrow s}\right)^*.$$

First, observe that it is always possible to decompose $\mathbf{M}^*$ such that

$$\mathbf{M}^* \quad = \quad \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}$$

where $\mathbf{B} \in \mathcal{M}(A, t-s)$ and $\mathbf{E} \in \mathcal{M}(A,s)$. We then have $(\mathbf{M}^*)^{\downarrow s} = \mathbf{E}$ and therefore

$$\left((\mathbf{M}^*)^{\downarrow s}\right)^* \quad = \quad \mathbf{E}^*.$$

We conclude from (KP2) that $\mathbf{E} \leq \mathbf{E}^*$. But using (KP5) and (6.31) we also obtain

$$\mathbf{E} \;=\; (\mathbf{M}^*)^{\downarrow s} \;=\; (\mathbf{M}^{**})^{\downarrow s} \;=\; \mathbf{E}^* + \mathbf{E}^*\mathbf{D}\mathbf{F}^*\mathbf{C}\mathbf{E}^*$$

which implies $\mathbf{E}^* \leq \mathbf{E}$ according to (SP3) of Lemma 5.2. This proves that $\mathbf{E} = \mathbf{E}^*$. ■

Lemma 6.19 *Φ is closed under vacuous extension.*

Proof: This follows from Lemma 6.17 and (KP5):

$$\left((\mathbf{M}^*)^{\uparrow t}\right)^* \;=\; \left((\mathbf{M}^*)^*\right)^{\uparrow t} \;=\; \left(\mathbf{M}^*\right)^{\uparrow t}.$$

■

We next introduce a very intuitive combination rule for elements in Φ. Imagine, for example, that we have two closure matrices which express the shortest distances between two possibly overlapping regions of a large graph. Then, the shortest distance matrix for the unified region can be found by vacuously extending the two matrices to their union domain, taking the component-wise minimum which corresponds to semiring addition and computing the new shortest distances. Thus, for $\mathbf{M}_1^*, \mathbf{M}_2^* \in \Phi$ with $d(\mathbf{M}_1^*) = s$ and $d(\mathbf{M}_2^*) = t$ we define

$$\mathbf{M}_1^* \otimes \mathbf{M}_2^* \quad = \quad \left((\mathbf{M}_1^*)^{\uparrow s \cup t} + (\mathbf{M}_2^*)^{\uparrow s \cup t}\right)^*. \tag{6.38}$$

We directly conclude from this definition that Φ is also closed under combination. Moreover, Φ becomes a commutative semigroup as shown by the following lemma.

Lemma 6.20 *Combination in Φ is commutative and associative.*

Proof: Commutativity of combination follows directly from the commutativity of addition in a semiring. To prove associativity, assume $\mathbf{M}_1^*, \mathbf{M}_2^*, \mathbf{M}_3^* \in \Phi$ with $d(\mathbf{M}_1^*) = s$, $d(\mathbf{M}_2^*) = t$ and $d(\mathbf{M}_3^*) = u$. Using Lemma 6.17 we obtain:

$$\begin{aligned}
(\mathbf{M}_1^* \otimes \mathbf{M}_2^*) \otimes \mathbf{M}_3^* &= \left(\left[\left((\mathbf{M}_1^*)^{\uparrow s \cup t} + (\mathbf{M}_2^*)^{\uparrow s \cup t}\right)^*\right]^{\uparrow s \cup t \cup u} + (\mathbf{M}_3^*)^{\uparrow s \cup t \cup u}\right)^* \\
&= \left(\left[\left((\mathbf{M}_1^*)^{\uparrow s \cup t} + (\mathbf{M}_2^*)^{\uparrow s \cup t}\right)^{\uparrow s \cup t \cup u}\right]^* + (\mathbf{M}_3^*)^{\uparrow s \cup t \cup u}\right)^* \\
&= \left(\left[\left((\mathbf{M}_1^*)^{\uparrow s \cup t \cup u} + (\mathbf{M}_2^*)^{\uparrow s \cup t \cup u}\right)\right]^* + (\mathbf{M}_3^*)^{\uparrow s \cup t \cup u}\right)^* \\
&= \left((\mathbf{M}_1^*)^{\uparrow s \cup t \cup u} + (\mathbf{M}_2^*)^{\uparrow s \cup t \cup u} + (\mathbf{M}_3^*)^{\uparrow s \cup t \cup u}\right)^*.
\end{aligned}$$

The last equality follows from Lemma 6.11 and the associativity of addition. Exactly the same expression can be derived in a similar way for $\mathbf{M}_1^* \otimes (\mathbf{M}_2^* \otimes \mathbf{M}_3^*)$ which proves associativity. ■

Next, we verify the combination axiom:

Lemma 6.21 *If $\mathbf{M}_1^*, \mathbf{M}_2^* \in \Phi$, $d(\mathbf{M}_1^*) = s$, $d(\mathbf{M}_2^*) = t$ and $s \subseteq z \subseteq s \cup t$ we have*

$$(\mathbf{M}_1^* \otimes \mathbf{M}_2^*)^{\downarrow z} \quad = \quad \mathbf{M}_1^* \otimes (\mathbf{M}_2^*)^{\downarrow z \cap t}. \tag{6.39}$$

Proof: From the definition of vacuous extension, it follows directly that vacuous extension can be performed step-wise. We therefore obtain

$$(\mathbf{M}_1^*)^{\uparrow s \cup t} = \left((\mathbf{M}_1^*)^{\uparrow z}\right)^{\uparrow s \cup t} = (\mathbf{M}_1^*)^{\uparrow z} \oplus \mathbf{I},$$

where $\mathbf{I}$ is the unit matrix with domain $t - z$. On the other hand, we also observe that since $(s \cup t) - z = t - z$

$$(\mathbf{M}_2^*)^{\uparrow s \cup t} \quad = \quad \begin{bmatrix} \left((\mathbf{M}_2^*)^{\downarrow z \cap t}\right)^{\uparrow z} & \left((\mathbf{M}_2^*)^{\uparrow s \cup t}\right)^{\downarrow z, t-z} \\ \left((\mathbf{M}_2^*)^{\uparrow s \cup t}\right)^{\downarrow t-z, z} & (\mathbf{M}_2^*)^{\downarrow t-z} \end{bmatrix}$$

Since Φ is closed under vacuous extension, we may apply Lemma 6.15 and obtain

$$\begin{aligned}\left((\mathbf{M}_2^*)^{\downarrow z\cap t}\right)^{\uparrow z} &= \left((\mathbf{M}_2^*)^{\downarrow z\cap t}\right)^{\uparrow z} + \\ &\quad \left((\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^{\downarrow z,t-z}(\mathbf{M}_2^*)^{\downarrow t-z}\left((\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^{\downarrow t-z,z}.\end{aligned}$$

We next compute

$$\begin{aligned}\mathbf{M}_1^*\otimes\mathbf{M}_2^* &= \left((\mathbf{M}_1^*)^{\uparrow s\cup t}+(\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^* \\ &= \begin{bmatrix}(\mathbf{M}_1^*)^{\uparrow z}+\left((\mathbf{M}_2^*)^{\downarrow z\cap t}\right)^{\uparrow z} & \left((\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^{\downarrow z,t-z} \\ \left((\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^{\downarrow t-z,z} & \mathbf{I}+(\mathbf{M}_2^*)^{\downarrow t-z}\end{bmatrix}^* \\ &= \begin{bmatrix}(\mathbf{M}_1^*)^{\uparrow z}+\left((\mathbf{M}_2^*)^{\downarrow z\cap t}\right)^{\uparrow z} & \left((\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^{\downarrow z,t-z} \\ \left((\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^{\downarrow t-z,z} & (\mathbf{M}_2^*)^{\downarrow t-z}\end{bmatrix}^*.\end{aligned}$$

This follows from (KP1) and because Φ is closed under projection. We next determine the closure of this matrix using (6.31) and project the result to the domain z:

$$\begin{aligned}(\mathbf{M}_1^*\otimes\mathbf{M}_2^*)^{\downarrow z} &= \Bigg[(\mathbf{M}_1^*)^{\uparrow z}+\left((\mathbf{M}_2^*)^{\downarrow z\cap t}\right)^{\uparrow z} \\ &\quad + \left((\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^{\downarrow z,t-z}(\mathbf{M}_2^*)^{\downarrow t-z}\left((\mathbf{M}_2^*)^{\uparrow s\cup t}\right)^{\downarrow t-z,z}\Bigg]^* \\ &= \left[(\mathbf{M}_1^*)^{\uparrow z}+\left((\mathbf{M}_2^*)^{\downarrow z\cap t}\right)^{\uparrow z}\right]^* = \mathbf{M}_1^*\otimes(\mathbf{M}_2^*)^{\downarrow z\cap t}.\end{aligned}$$

■

All these properties imply the following theorem if D denotes the lattice of finite subsets of the countable set r of variables:

Theorem 6.6 *The system $\langle\Phi, D\rangle$ with labeling, projection (6.35) and combination (6.38) as defined above satisfies the axioms of a valuation algebra.*

Proof: Axioms (A2) to (A4) and (A6) follow directly from the above definitions. Axiom (A1) is proved in Lemma 6.20 and Axiom (A5) in Lemma 6.21. ■

Closures of labeled matrices with values from a Kleene algebra therefore provide another example of a generic construction that leads to as many new valuation algebra instances as Kleene algebras exist. How exactly Kleene valuation algebras are used to solve path problems will be discussed in Section 9.3.3 where we also compare this approach with the solution of factorized path problems using the quasi-regular valuation algebras from Section 6.4. Remember, one reason that motivated the introduction of Kleene valuation algebras was the promise that they always produce idempotent valuation algebras. This and other properties will next be studied.

6.8 PROPERTIES OF KLEENE VALUATION ALGEBRAS

Kleene algebras have many interesting properties as we have seen in Section 6.6. It is therefore not astonishing that Kleene valuation algebras are also very rich in structure. In fact, the following results show that Kleene valuation algebras are idempotent and always provide neutral elements.

Lemma 6.22 *Kleene valuation algebras have neutral elements and are stable.*

Proof: We first show that the neutral element for the domain $s \in D$ is given by the identity matrix $\mathbf{I} \in \mathcal{M}(A, s)$. Due to Property (KP3) we have $\mathbf{I}^* = \mathbf{I}$ and therefore $\mathbf{I} \in \Phi$. Further, we obtain for $\mathbf{M}^* \in \Phi$ with $d(\mathbf{M}^*) = s$

$$\mathbf{M}^* \otimes \mathbf{I} \;=\; (\mathbf{M}^* + \mathbf{I})^* \;=\; (\mathbf{M}^*)^* \;=\; \mathbf{M}^*,$$

due to (KP1) and (KP5). It is furthermore clear that projecting a neutral element always results in a neutral element for the subdomain which proves stability. ∎

Lemma 6.23 *Kleene valuation algebras are idempotent.*

Proof: For $\mathbf{M}^* \in \Phi$ with $s \subseteq t = d(\mathbf{M}^*)$ we have

$$\mathbf{M}^* \otimes (\mathbf{M}^*)^{\downarrow s} \;=\; \left[\mathbf{M}^* + ((\mathbf{M}^*)^{\downarrow s})^{\uparrow t}\right]^* \;=\; \mathbf{M}^{**} \;=\; \mathbf{M}^*.$$

This follows from the idempotency of addition, (KP1) and (KP5). ∎

However, Kleene valuation algebras generally do not have null elements. Even if we assume an element for each domain that behaves absorbingly with respect to combination, the nullity axiom of Section 3.4 will not be satisfied. Since projection only drops matrix rows and columns, there may always exist other valuations that project to null elements which contradicts the nullity axiom.

6.9 FURTHER PATH PROBLEMS

This chapter presented two different families of formalisms to model path problems that both satisfy the valuation algebra axioms. Namely, these are quasi-regular valuation algebras and Kleene valuation algebras. Also, it was shown that Kleene valuation algebras provide more algebraic structure as for example the property of idempotency that is not fulfilled in quasi-regular valuation algebras. Looking at the proof of Lemma 6.23, we immediately see that idempotency of combination is a consequence of idempotent addition and the monotonicity laws in a Kleene algebra. Both requirements are not present in a quasi-regular semiring. On the other hand, the absence of these properties makes quasi-regular semirings more general than Kleene algebras, and the same holds for their induced valuation algebras. Since all path problems shown at the beginning of this chapter are based on Kleene algebras, we end with a few more examples based on only quasi-regular semirings. These applications can therefore not be solved in the formalism of Kleene valuation algebras. The examples are based on the arithmetic semirings of non-negative integers and of real numbers from Example 6.11 that are both quasi-regular but not Kleene algebras. The first instance is a typical graph-related application, whereas the subsequent problems are not directly related to graphs anymore. This keeps the promise articulated in the introduction that the algebraic path problem also covers such rather unexpected applications.

6.6 The Path Counting Problem

Figure 6.9 shows a network similar to the connectivity problem of Instance 6.1. This time however, we ask for the number of paths that connect the source node S with the target node T. We therefore compute the sum of the weights of all possible paths leading from S to T, where the weight of a path corresponds to the product of its edge weights. We compute for the three possible paths:

$$\begin{aligned} 0 \cdot 1 &= 0 \\ 1 \cdot 1 \cdot 1 &= 1 \\ 1 \cdot 1 \cdot 1 \cdot 1 &= 1 \end{aligned}$$

and then

$$0 + 1 + 1 = 2.$$

Clearly, this description corresponds to equation (6.6) based on the arithmetic semiring of non-negative integers.

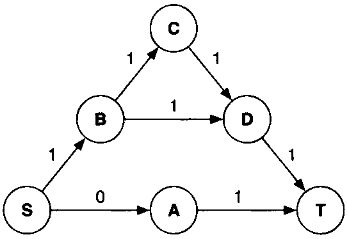


Figure 6.9 The path counting problem.

6.7 Markov Chains

A *time-homogeneous, first order Markov chain* is specified by a countable sequence of random variables $X_1, X_2, \ldots$ taking values from a finite *state* *space* Ω and satisfying the property:

$$P(X_{n+1} = x | X_1 = x_1, \ldots, X_n = x_n) \quad = \quad P(X_{n+1} = x | X_n = x_n)$$

for all $n \in \mathbb{N}$ and $x, x_i \in \Omega$. Thus, the probability to arrive in state x at some time depends only on the last state but not on previous states. Such a Markov chain is determined by its *transition matrix* $\mathbf{M} : \Omega \times \Omega \rightarrow [0, 1]$ satisfying

$$\sum_{Y \in \Omega} \mathbf{M}(X, Y) \quad = \quad 1$$

for all $X \in \Omega$, where the value $\mathbf{M}(X, Y)$ specifies the probability to arrive in state Y from the previous state X. A Markov chain can always be represented by a weighted, directed graph where the nodes correspond to the states and the edge weights to the transition probabilities. Let us for example assume a Markov chain with state space $\Omega = \{1, 2, 3\}$ and transition matrix

$$\mathbf{M} \quad = \quad \begin{bmatrix} 0 & 0.6 & 0.4 \\ 0 & 0.1 & 0.9 \\ 0 & 1 & 0 \end{bmatrix}.$$

Its graph is shown in Figure 6.10. For two selected states S and T, the value of the matrix power $\mathbf{M}^n(S, T)$ corresponds to the weight of the path between S and T that contains exactly n edges. If the computations are executed in the arithmetic semiring of real numbers $\langle \mathbb{R} \cup \{\infty\}, +, \cdot, *, 0, 1 \rangle$, then this path weight corresponds to the probability of reaching state T in n steps from node S. Accordingly, the values of the quasi-inverse matrix $\mathbf{M}^*(S, T)$ refer to the sum of all probabilities of reaching state T from state S in an arbitrary number of steps. It is clear that this will not be a probability anymore. If we for example obtain $\mathbf{M}^*(S, T) = \infty$, we say that the state T has *finite hitting time* from state S. Moreover, if a state has finite hitting time from itself, then it is called *recurrent*. Such states return to themselves infinitely often with probability 1. On the other hand, if we obtain a value different from ∞, then the state is called *transient* and leads back to itself only a finite number of times. Identifying recurrent and transient states in Markov chains is an important task that can thus be seen as an instance of the algebraic path problem. For the above matrix $\mathbf{M}$ we obtain by equation (6.17):

$$\mathbf{M}^* \quad = \quad \begin{bmatrix} 1 & 0 & 0 \\ \infty & \infty & \infty \\ \infty & \infty & \infty \end{bmatrix}.$$

The diagonal values tell us that state 1 is transient and all other states are recurrent. Alternatively, we may repeat these computations with the probabilistic semiring $\langle [0, 1], \max, \cdot, *, 0, 1 \rangle$ where $a^* = 1$ for all $a \in [0, 1]$. This is a Kleene algebra as shown in Example 6.10 and $\mathbf{M}^*(S, T)$ corresponds to the value of the most likely path from state S to T. We obtain for the above example:

$$\mathbf{M}^* = \begin{bmatrix} 1 & 0.6 & 0.54 \\ 0 & 1 & 0.9 \\ 0 & 1 & 1 \end{bmatrix}.$$

The reader may convince himself that $0.6 \cdot 0.9 = 0.54$ is indeed the highest probability among all sequences of transitions that lead from state 1 to state 3. We refer to (Norris, 1998; Meyn & Tweedie, 1993; Kemeny *et al.*, 1960) for a systematic discussion of Markov chains.

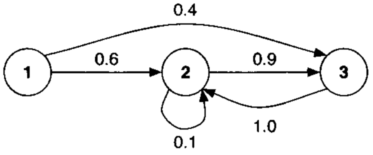


Figure 6.10 A graphical representation of a Markov chain.

6.8 Numeric Partial Differentiation

The numerical computation of the *partial derivations* of a function can be reduced to an algebraic path problem as shown in (Rote, 1990), from where we also borrow this example. Assume the following function over three variables:

$$f(Z_1, Z_2, Z_3) = \left(\frac{Z_2Z_3 + \sqrt{(Z_2Z_3)^2 + 4Z_1^2Z_2^2}}{2Z_1^2} \right)^2$$

This function is step-wise computed by the following program:

$$\begin{aligned} Y_1 &:= Z_2Z_3 + \sqrt{(Z_2Z_3)^2 + 4Z_1^2Z_2^2} \\ Y_2 &:= (Y_1 \,/\, (2Z_1^2))^2 \end{aligned}$$

This is a rather simple example but we could as well imagine some complicated program involving loops and conditional statements. However, we next decompose this program into elementary operations:

$$\begin{array}{lcl@{\qquad}lcl@{\qquad}rcl} A & := & Z_2 \cdot Z_3 & E & := & C \cdot D & Y_1 & := & A + I \\ B & := & A^2 & G & := & 4 \cdot E & J & := & 2 \cdot C \\ C & := & Z_1^2 & H & := & B + G & K & := & Y/J \\ D & := & Z_2^2 & I & := & \sqrt{H} & Y_2 & := & K^2 \end{array}$$

This decomposition allows us to construct the *computational graph* of Figure 6.11 by the following procedure: First, a node for is created for each variable

Z_1 to Z_3. Then, we add a node for each elementary operation and connect it with a directed edge to its arguments. If the expression contains a constant, then we first add an additional node for this value.

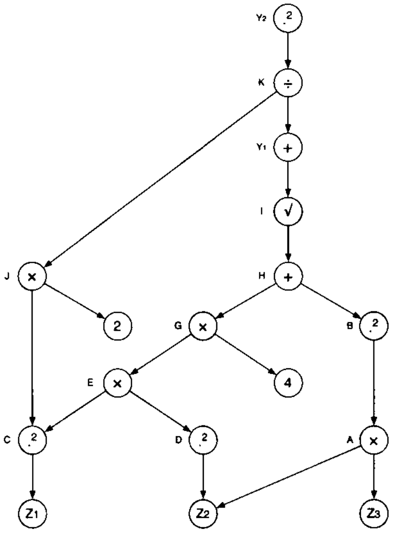


Figure 6.11 The computational graph of a function.

Now, let us look at some node w with two outgoing edges leading to u and v. This subtree represents a composition function $w(u,v)$, whose partial derivatives $\partial w/\partial Z_i$ can be determined by the *chain rule*:

$$\frac{\partial w}{\partial Z_i} \quad = \quad \frac{\partial w}{\partial u}\cdot\frac{\partial u}{\partial Z_i}+\frac{\partial w}{\partial v}\cdot\frac{\partial v}{\partial Z_i}. \tag{6.40}$$

Note that $\partial w/\partial u$ and $\partial w/\partial v$ can easily be calculated, since only elementary operations are involved. Let us for example take $w = u/v$. We then have $\partial w/\partial u = 1/v$ and $\partial w/\partial v = -u/v^2 = -w/v$. It is therefore possible to assign the weight $\mathbf{M}(w,v) = \partial w/\partial v$ to each edge (w,v) of the computational graph. We then obtain for the above application of the chain rule:

$$X_{w,Z_i} \quad = \quad \mathbf{M}(w,u)X_{u,Z_i} + \mathbf{M}(w,v)X_{v,Z_i}$$

which is a recursive expression in the unknowns

$$X_{u,Z_i} = \partial u/\partial Z_i \quad \text{and} \quad X_{v,Z_i} = \partial v/\partial Z_i.$$

For the root node r we have $X_{r,Z_i} = \partial f/\partial Z_i$. By developing this expression recursively, we obtain the derivative of f with respect to Z_i by the sum of the weights of all paths leading from f to Z_i, whereas the weight of a path corresponds to the product of its edge weights. So far, we considered the edge weights as symbolic expressions. But if we assign values from the arithmetic semiring of real numbers from Example 6.11 to the variables Z_1 to Z_3, then all edge weights also correspond to values in this semiring. Then, the above scheme for calculating the partial derivations of f amounts to the solution of a path problem in the quasi-regular semiring of real numbers. Moreover, computing the *Jacobi matrix* $(\partial f_j / \partial Z_i)$ then corresponds to a submatrix of the quasi-inverse matrix that is obtained from solving the all-pairs problem on this setting. We refer to (Rote, 1990) for a more comprehensive analysis of this approach and for related references.

6.9 Matrix Multiplication

Consider a quasi-regular semiring $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ and two square matrices $\mathbf{M}_1, \mathbf{M}_2 : \{1, \ldots, n\} \times \{1, \ldots, n\} \rightarrow A$ defined over the same index set. The following idea of reducing the task of multiplying $\mathbf{M}_1$ and $\mathbf{M}_2$ to the computation of a quasi-inverse matrix was proposed by (Aho *et al.*, 1974). We first construct a new matrix $\mathbf{M} : \{1, \ldots, 3n\} \times \{1, \ldots, 3n\} \rightarrow A$ defined as:

$$\mathbf{M} = \begin{bmatrix} \mathbf{O} & \mathbf{M}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{M}_2 \\ \mathbf{O} & \mathbf{O} & \mathbf{O} \end{bmatrix}.$$

We assume the following decomposition of $\mathbf{M}$ and compute the quasi-inverse $\mathbf{M}^*$ by equation (6.17):

$$\mathbf{M}^* = \left[\begin{array}{cc|c} \mathbf{O} & \mathbf{M}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{M}_2 \\ \hline \mathbf{O} & \mathbf{O} & \mathbf{O} \end{array}\right]^* = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}^*.$$

Since $\mathbf{D}$ and $\mathbf{E}$ are zero matrices, we directly obtain $\mathbf{F} = \mathbf{E} + \mathbf{D}\mathbf{B}^*\mathbf{C} = \mathbf{O}$ and thus $\mathbf{F}^* = \mathbf{I}$ as a consequence of equation (6.18). We also compute

$$\mathbf{B}^* = \begin{bmatrix} \mathbf{O} & \mathbf{M}_1 \\ \mathbf{O} & \mathbf{O} \end{bmatrix}^* = \begin{bmatrix} \mathbf{I} & \mathbf{M}_1 \\ \mathbf{O} & \mathbf{I} \end{bmatrix}$$

and finally obtain for the three remaining submatrices:

$$\begin{aligned} \mathbf{B}^*\mathbf{C}\mathbf{F}^* &= \mathbf{B}^*\mathbf{C} = \begin{bmatrix} \mathbf{I} & \mathbf{M}_1 \\ \mathbf{O} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{O} \\ \mathbf{M}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{M}_1\mathbf{M}_2 \\ \mathbf{M}_2 \end{bmatrix} \\ \mathbf{F}^*\mathbf{D}\mathbf{B}^* &= \mathbf{D}\mathbf{B}^* = \begin{bmatrix} \mathbf{O} & \mathbf{O} \end{bmatrix} \end{aligned}$$

and $\mathbf{B}^* + \mathbf{B}^*\mathbf{C}\mathbf{F}^*\mathbf{D}\mathbf{B}^* = \mathbf{B}^*$. This determines the quasi-inverse matrix as

$$\mathbf{M}^* = \begin{bmatrix} \mathbf{I} & \mathbf{M}_1 & \mathbf{M}_1\mathbf{M}_2 \\ \mathbf{O} & \mathbf{I} & \mathbf{M}_2 \\ \mathbf{O} & \mathbf{O} & \mathbf{I} \end{bmatrix}.$$

Two square matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ with equal index sets and values from a quasi-regular semiring can thus be multiplied by computing the quasi-inverse matrix of $\mathbf{M}$ and extracting the corresponding submatrix.

6.10 CONCLUSION

This chapter started with an introduction to the algebraic path problem which requires to solve a particular fixpoint equation of matrices taking values from a semiring. In the general case, such equations do not necessarily have a solution, but if they exist, solutions are called quasi-inverses. In order to avoid the problem of non-existence, we limited ourselves to so-called quasi-regular semirings where at least one quasi-inverse exists for each semiring element. Based on the construction of (Lehmann, 1976) it is then possible to compute a particular quasi-inverse of a matrix from the quasi-inverses of the underlying semiring. This leads to a first generic construction called quasi-regular valuation algebra where valuations correspond to labeled pairs of matrices and vectors over a quasi-regular semiring. Such valuation algebras can be used for the solution of the single-target algebraic path problem which will be discussed in Section 9.3.2. A second approach focussing directly on the solution of the all-pairs algebraic path problem is based on a special family of quasi-regular semirings called Kleene algebras. Under this setting, we always obtain the least quasi-inverse for each semiring element and the corresponding operation further satisfies the axioms of a closure operator. These two properties lead to another family of valuation algebras where valuations are closures of labeled matrices over a Kleene algebra. Projection corresponds to simple matrix restriction and combination to the computation of the closure after taking the sum of the two factor matrices. In contrast to quasi-regular valuation algebras, the second approach therefore moves the computational effort from the projection operation to the combination which manifests itself in the fact that Kleene valuation algebras are always idempotent. This combination essentially consists in the computation of a quasi-inverse matrix which, according to (Lehmann, 1976), is possible in polynomial time. Algorithms for this task will be presented in Chapter 9 where we also describe the solution of factorized path problems using quasi-regular and Kleene valuation algebras. Further, we observe that both approaches only store matrices and vectors which also implies that space complexity is polynomial. We therefore have two generic constructions that both induce valuation algebras with a pure polynomial behaviour.

PROBLEM SETS AND EXERCISES

F.1 ★ Instance 6.5 describes the task of determining the language that brings an automaton from state S to state T as a path problem over the semiring of formal languages $\langle \mathcal{P}(\Sigma^*), \cup, \cdot, \emptyset, \{\epsilon\}\rangle$ from Example 5.7. Alternatively, this can also be interpreted as the problem of listing all paths between node S and node T in a graph, provided that the edge weights express the identifier of the corresponding connection.

However, the set of all possible paths between two nodes is often infinite. Thus, we are rather interested in listing only *simple paths* that contain every edge at most once. Identify the corresponding subsemiring of the semiring of formal languages and determine whether it is still a Kleene algebra.

F.2 ★ Testing whether a graph is bipartite: An undirected graph is *bipartite*, if it contains no cycle with an odd number of edges. Since cycles are special paths, we can formulate this as a path problem. Consider the set of symbols $A = \{\emptyset, E, O, EO\}$. The weight of a path is either E, if it contains an even number of edges, or O, if it contains an odd number of edges. Accordingly, the weight of a set of paths is either $\emptyset$, E, O or EO, depending on whether the set is empty, contains only even paths, only odd paths or both types of paths. Continue the development of this path problem by specifying how the edge weights are initialized and how the semiring operations are defined. Prove that the semiring is quasi-regular and test if it is also a Kleene algebra. The solution can be found in Section 6.7.1 of (Rote, 1990).

F.3 ★ Identifying cut nodes of a graph: A *cut node* in a connected graph is a node whose removal causes the graph to become disconnected. Formulate the problem of finding cut nodes as a path problem and analyze the underlying semiring. The solution can be found in Section 6.7.2 of (Rote, 1990).

F.4 ★ Prove the following property of Kleene algebras: For $a, b \in A$ we have

$$(ab)^*a \quad = \quad a(ba)^*. \tag{6.41}$$

The solution to this exercise is given in Corollary 5 of (Kozen, 1994). Then, prove Lemma 6.14 by applying this identity and the properties of Kleene algebras listed in Section 6.6. The solution to this exercise is given in Chapter 3 of (Conway, 1971).

F.5 ★ We have seen in Section 3.1 that in case of valuation algebras defined over variable systems, the operation of projection can be replaced by variable elimination. We may then use the simpler versions of the valuation algebra axioms given in Lemma 3.1. Develop the quasi-regular and Kleene valuation algebra from Section 6.4 with variable elimination instead of projection.

F.6 ★★ It is sometimes necessary to solve many path problems with different Kleene algebras over the same graph. Instead of repeating the computations each time from scratch, we propose a two-stage *compilation process*: We first determine some formal expression for the paths between S and T which can later be evaluated for different Kleene algebras without recomputing the paths. To define this formal language, we assume a symbol $w_{X \rightarrow Y}$, if an edge between node X and Y exists. In addition, we identify the two semiring operations $+$ and $\times$ with the symbols $\oplus$ and $\odot$. Then, the formal expression for the path between node S and T in Figure 6.1 is

$$\begin{aligned} w_{S\to T} \quad = \quad & w_{S\to A} \odot w_{A\to T} \otimes \\ & w_{S\to B} \odot w_{B\to D} \odot w_{D\to T} \otimes \\ & w_{S\to B} \odot w_{B\to C} \odot w_{C\to D} w_{D\to T}. \end{aligned}$$

This string can now be evaluated by replacing the symbols with concrete edge values and operations from a Kleene algebra.

a) Specify this formal language completely such that it forms a Kleene algebra. The solution to this exercise can be found in Section 3.2.2 of (Jonczy, 2009). There, it is also shown how such path expressions can be represented and evaluated efficiently, and that many further queries can be computed that go far beyond the computation of path weights.

b) Instance 6.8 considers the numeric computation of partial differentiations as a path problem. Apply the compilation technique to this particular path problem for the aim of symbolic differentiation.

CHAPTER 7

LANGUAGE AND INFORMATION

Information often concerns the values of variables or the truth of propositions satisfying equations or logical formulae. It is in many cases stated through linguistic constructs with a determined interpretation. This general situation is a source of many valuation algebras that all have the interesting property of idempotent combination. Moreover, they always provide neutral and null elements and therefore adopt the structure of an information algebra according to Section 4.2.1. In such cases, the computations are usually not carried out on the level of the information or valuations, but rather in the corresponding linguistic system using syntactic procedures. In this chapter, we illustrate the general concept of representing information by formal languages with a determined interpretation. We first give two important examples of such formalisms and then derive a more general concept that serves as a generic construction to produce the two previous examples and many other instances.

In the first section, we show how propositional logic serves to state the truth about propositions or logical variables. The interpretation of propositional formulae represents the information expressed by the formulae. We shall show that such pieces of information form a valuation algebra and more particularly an information algebra. The second example presented in Section 7.2 concerns systems of linear equations

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

whose solutions provide information about the values of a set of variables. Their solution spaces are again shown to form an information algebra, and the exploitation of this algebraic structure leads to *sparse matrix techniques* for the solution of sparse linear systems, see Chapter 9. In this chapter, we only focus on the algebraic properties of linear equation systems. Although we consider fields instead of semirings, this nevertheless is closely related to the path problems of the foregoing chapter that also require the solution of particular equation systems. We therefore move the discussion of the computational aspects of all these applications to Chapter 9. Finally, the concluding section of this chapter presents the abstract generic framework generalizing both linguistic systems. There, we also allude to some further important instances that emerge from this new generic construction.

7.1 PROPOSITIONAL LOGIC

Propositional logic or *sentential calculus* is concerned with the truth of elementary propositions or with the values of Boolean variables. With regard to information, the question addressed by this formalism is which propositions or variables are true and which are false. Such truth values are usually expressed by $'1'$ for *true* and $'0'$ for *false*, but the available information regarding these values may also be expressed in the propositional language. In most cases, these expressions are much simpler and shorter than the explicit enumeration of all possible truth values for the variables under consideration. Subsequently, we define this formal language and equip it with an interpretation to show how *propositional information* (i.e. the information about the truth values of Boolean variables) is expressed. We then point out in the following section that a dual pair of valuation algebras arises from the language and its interpretation.

7.1.1 Language and Semantics

The language of propositional logic $\mathcal{L}_p$ is constructed over a countable set of *propositional symbols* or *propositional variables* $p = \{P_1, P_2, \ldots\}$. It consists of *well-formed formulae (wff)* defined inductively as follows:

1. Each element $P \in p$ as well as $\top$ and $\perp$ are wffs (called *atomic formulae*).

2. If f is a wff, then $\neg f$ is a wff.

3. If f and g are wffs, then $f \wedge g$ is a wff.

4. If f is a wff and $P \in p$ then $(\exists P)f$ is a wff.

All wffs are generated from atomic formulae by finitely many applications of the rules 2, 3 and 4. The construction 4 is usually not part of the propositional language, but it suits our purposes. The symbol $\exists$ is called *existential quantifier*. We conclude from these considerations that any wff has finite length and contains only a finite number of propositional symbols. For simplification, it is also common to extend this language by adding the following constructions:

1. $f \vee g \quad := \quad \neg(\neg f \wedge \neg g);$
2. $f \rightarrow g \quad := \quad \neg f \vee g;$
3. $f \leftrightarrow g \quad := \quad (f \rightarrow g) \wedge (g \rightarrow f).$

The intended meaning of these constructions is given by the interpretation of the formulae defined below. But before doing this, let us remark that we often consider propositional languages over subsets $q \subseteq p$. In particular, we often limit ourselves to finite subsets q of propositional symbols. So, $\mathcal{L}_q$ is defined as above by replacing p in Rule 1 by q. The subsets of p form a lattice under inclusion with intersection as meet and union as join, see Example A.2 in the appendix of Chapter 1. This lattice structure is also reflected in the family of languages $\mathcal{L}_q$ as follows:

1. If $p_1 \subseteq p_2$ we also have $\mathcal{L}_{p_1} \subseteq \mathcal{L}_{p_2}$;
2. $\mathcal{L}_{p_1 \cap p_2} \;=\; \mathcal{L}_{p_1} \cap \mathcal{L}_{p_2} \;=\; \mathcal{L}_{p_1} \wedge \mathcal{L}_{p_2}$;
3. $\mathcal{L}_{p_1 \cup p_2} \;=\; \mathcal{L}_{p_1} \vee \mathcal{L}_{p_2} \;=\; \bigcap\{\mathcal{L}_q : \mathcal{L}_{p_1} \cup \mathcal{L}_{p_2} \subseteq \mathcal{L}_q\}$.

In particular, we have $\mathcal{L}_\emptyset = \{\bot, \top\}$ for the language without propositional symbols.

We now give meaning to this language by *interpreting* its wffs. This is achieved by mappings $\mathbf{v} : p \rightarrow \{0,1\}$ which are called *valuations*. Here, valuations do not correspond to elements of a valuation algebra but to the traditional name of an assignment of truth values to propositions which is equivalent to the notion of tuple, configuration or vector of Chapter 1. However, we accept this double sense for a short while and always explicitly refer to the valuation algebra when the context changes. Valuations are extended to propositional formulae $\hat{\mathbf{v}} : \mathcal{L}_p \rightarrow \{0,1\}$ by:

1. $\hat{\mathbf{v}}(P_i) = \mathbf{v}(P_i)$ for all $P_i \in p$;
2. $\hat{\mathbf{v}}(\top) = 1$ and $\hat{\mathbf{v}}(\bot) = 0$;
3. $\hat{\mathbf{v}}(\neg f) = 1$ if $\hat{\mathbf{v}}(f) = 0$ and $\hat{\mathbf{v}}(\neg f) = 0$ otherwise;
4. $\hat{\mathbf{v}}(f \wedge g) = 1$ if $\hat{\mathbf{v}}(f) = 1$ and $\hat{\mathbf{v}}(g) = 1$, $\hat{\mathbf{v}}(f \wedge g) = 0$ otherwise;
5. $\hat{\mathbf{v}}((\exists P)f) = \hat{\mathbf{v}}(f[P/\top] \vee f[P/\bot])$.

Here, $f[P/\top]$ denotes the formula obtained from f by replacing all occurrences of the proposition $P \in p$ by $\top$. Similarly, $f[P/\bot]$ refers to the formula obtained from f by replacing all occurrences of P by $\bot$.

If we read $'1'$ as true and $'0'$ as false, we then see that the connector $\neg$ represents the *negation* of a propositional formula, i.e. the truth value $\hat{v}(\neg f)$ is the opposite of $\hat{v}(f)$. The connector $\wedge$ expresses the *logical and* or *logical conjunction*: $f \wedge g$ is true if, and only if, both formulae f and g are true. The existential quantifier $\exists P$ means that the quantified formula $(\exists P)f$ evaluates to true, if f evaluates to true when P is either replaced by $\top$ or $\bot$ in f. In other words, f is true if it is true with P either interpreted as true or false. The evaluation of the formulae of the extended language follow from these definitions:

1. $\hat{\mathbf{v}}(f \vee g) = 1$ if $\hat{\mathbf{v}}(f) = 1$ or $\hat{\mathbf{v}}(g) = 1$, $\hat{\mathbf{v}}(f \wedge g) = 0$ otherwise;
2. $\hat{\mathbf{v}}(f \rightarrow g) = 1$ if either $\hat{\mathbf{v}}(f) = 0$ or $\hat{\mathbf{v}}(g) = 1$, $\hat{\mathbf{v}}(f \rightarrow g) = 0$ otherwise;
3. $\hat{\mathbf{v}}(f \leftrightarrow g) = 1$, if $\hat{\mathbf{v}}(f) = \hat{\mathbf{v}}(g)$, $\hat{\mathbf{v}}(f \leftrightarrow g) = 0$ otherwise.

$f \vee g$ is interpreted as the *logical or* or the *logical disjunction*. The expression $f \rightarrow g$ is called *implication* and means that if f is true, then so is g. Finally, $f \leftrightarrow g$ expresses the *equivalence* between f and g, i.e. f and g are either both true or both false.

Example 7.1 *We describe the full adder circuit of Figure 2.7 from Instance 2.4 in the language of propositional logic over* $p = \{In_1, In_2, In_3, Out_1, Out_2\}$. *The XOR gates in Figure 2.7 output* 1 *if, and only if, both inputs have different values. This can be modeled by the propositional formula* $f \veebar g := (f \vee \neg g) \wedge (\neg f \vee g)$. *We then have* $\hat{\mathbf{v}}(f \veebar g) = 1$ *if* $\hat{\mathbf{v}}(f) \neq \hat{\mathbf{v}}(g)$. *Otherwise, we have* $\hat{\mathbf{v}}(f \veebar g) = 0$. *Using this construction, the full adder circuit is completely described by the following formulae:*

$$Out_1 \leftrightarrow (In_1 \veebar In_2) \veebar In_3 \tag{7.1}$$

and

$$Out_2 \leftrightarrow ((In_1 \veebar In_2) \wedge In_3) \vee (In_1 \wedge In_2). \tag{7.2}$$

A valuation $\mathbf{v}$ *satisfies* a propositional formula f if $\hat{\mathbf{v}}(f) = 1$. Then, $\mathbf{v}$ is called a *model* of f, and we write $\mathbf{v} \models f$. The set of all valuations satisfying f is denoted by

$$\hat{r}(f) \quad = \quad \{\mathbf{v} : \mathbf{v} \models f\}.$$

More generally, $\hat{r}(S)$ denotes the set of valuations which satisfy *all* formulae $f \in S$,

$$\hat{r}(S) \quad = \quad \{\mathbf{v} : \mathbf{v} \models f, \forall f \in S\}.$$

Conversely, $\check{r}(\mathbf{v})$ denotes the set of all wffs which are satisfied by a valuation $\mathbf{v}$,

$$\check{r}(\mathbf{v}) \quad = \quad \{f : \mathbf{v} \models f\}.$$

Thus, $\mathbf{v}$ is a model of each element of $\check{r}(\mathbf{v})$. This notation is again extended to sets of valuations: If M is a set of valuations, then $\check{r}(M)$ is the set of all sentences satisfied by all valuations in M,

$$\check{r}(M) \quad = \quad \{f : \mathbf{v} \models f, \forall \mathbf{v} \in M\}.$$

Let $\mathcal{M}_p$ denote the set of all valuations $\mathbf{v} : p \rightarrow \{0, 1\}$. A tuple $\langle \mathcal{L}_p, \mathcal{M}_p, \models \rangle$ consisting of a language $\mathcal{L}_p$, a set of models $\mathcal{M}_p$ and a satisfying relation $\models \subseteq \mathcal{L}_p \times \mathcal{M}_p$ is called a *context*. This concept will be generalized in Section 7.3.1.

A formula f is called *satisfiable*, if there is at least one valuation $\mathbf{v}$ that satisfies f, i.e. if $\hat{r}(f) \neq \emptyset$. Checking the satisfiability of a formula is the fundamental problem of propositional logic (see Instance 2.4). Two formulae f and g are called (logically) *equivalent* if they have the same models, i.e. if $\hat{r}(f) = \hat{r}(g)$. Note that $(\exists P)f$ is always equivalent to a formula where the proposition $P \in p$ does not occur anymore. So, existential quantification serves to eliminate variables from a formula. Similarly, two sets of formulae S_1 and S_2 are equivalent, if $\hat{r}(S_1) = \hat{r}(S_2)$. We denote by $(\exists P_{i_1} \ldots P_{i_n})S$ the set of formulae $(\exists P_{i_1}) \ldots (\exists P_{i_n})f$ for all $f \in S$. It is equivalent to a set of formulae without the variables P_{i_1} to P_{i_n}.

Example 7.2 *The valuation* $\mathbf{v}(In_1) = \mathbf{v}(Out_1) = 1$ *and* $\mathbf{v}(In_2) = \mathbf{v}(In_3) = 0$ *satisfies the propositional formula (7.1) and is therefore called a model of this formula. On the other hand, the assignment* $\mathbf{v}(Out_1) = 1$ *and* $\mathbf{v}(In_1) = \mathbf{v}(In_2) = \mathbf{v}(In_3) = 0$ *does not satisfy the formula. If S consists of the two formulae (7.1) and (7.2), the set of valuations $\hat{r}(S)$ satisfying both formulae is given in Table 2.8 of Instance 2.4.*

In the next section, we describe how these concepts are used to capture *propositional information* and how two dual information algebras are related to propositional languages and models.

7.1.2 Propositional Information

The available information concerning propositional variables is usually expressed by sets of propositional formulae. More precisely, a set of propositional formulae S determines the information $\hat{r}(S) \subseteq \mathcal{M}$, and the formulae S say that the unknown model, sometimes called a *possible world*, is a member of $\hat{r}(S)$. With respect to the above adder example, the equations (7.1) and (7.2) specify the possible configurations (Table 2.8) of the circuit variables, if the components of the adder work correctly.

In order to move towards a valuation or even an information algebra, we consider the lattice D of *finite subsets* of the countable set of propositional symbols p. Any subsets $q \subseteq p$ represents a question, namely the question about the truth values of the propositions $P_i \in q$. The projection $\mathbf{v}^{\downarrow q}$ of a valuation $\mathbf{v} : p \rightarrow \{0, 1\}$ is called an *interpretation* of the language $\mathcal{L}_q$. Let $\mathcal{M}_q$ be the set of all possible interpretations of $\mathcal{L}_q$. This is a finite set of $2^{|q|}$ elements. As mentioned before, the elements of $\mathcal{M}_q$ can also be considered as Boolean q-tuples or configurations $\mathbf{m} : q \rightarrow \{0, 1\}$. If f is a wff from $\mathcal{L}_q$, it contains only propositional symbols from q. For $\mathbf{m} \in \mathcal{M}_q$ we write $\mathbf{m} \models_q f$ if $\mathbf{m} = v^{\downarrow q}$ and $\mathbf{v} \models f$. In other words, $\mathbf{m}$ is a model of f with respect to the variables in q. The values of $\mathbf{v}$ on variables outside q clearly do not influence the relation $\models_q$. We thus have for all $q \in D$ a context $c_q = \langle \mathcal{L}_q, \mathcal{M}_q, \models_q \rangle$ and we next extend the notions of $\hat{r}(f)$ and $\check{r}(\mathbf{v})$ to interpretations or models in $\mathcal{M}_q$ and formulae in $\mathcal{L}_q$. If $S \subseteq \mathcal{L}_q$ is a set of formulae and $M \subseteq \mathcal{M}_q$ a set of models, we have

$$\hat{r}_q(S) \quad = \quad \{\mathbf{m} \in \mathcal{M}_q : \mathbf{m} \models_q f, \forall f \in S\}$$

and

$$\check{r}_q(M) \quad = \quad \{f \in \mathcal{L}_q : \mathbf{m} \models f, \forall \mathbf{m} \in M\}.$$

$\hat{r}_q(S)$ is called the *propositional information* of S with respect to q, and $\check{r}_q(M)$ is the *theory* of M in q. The *information sets*, i.e. the subsets of $\mathcal{M}_q$ for all $q \in D$, form an information algebra. In fact, this formalism corresponds to the relational algebra of Instance 1.2 where all relations $M \subseteq \mathcal{M}_q$ are Boolean. Given an information set M, we define its label by $d(M) = q$ if $M \subseteq \mathcal{M}_q$. Combination is defined by natural join. If M_1 is labeled with q and M_2 with u we have

$$M_1 \otimes M_2 \;=\; M_1 \bowtie M_2 \;=\; \{m \in \mathcal{M}_{q \cup u} : \mathbf{m}^{\downarrow q} \in M_1 \text{ and } \mathbf{m}^{\downarrow u} \in M_2\}. \quad (7.3)$$

Finally, projection of an information set M to some domain $q \subseteq d(M)$ is defined by

$$M^{\downarrow q} \;=\; \{\mathbf{m}^{\downarrow q} : \mathbf{m} \in M\}. \quad (7.4)$$

The neutral element for the domain $q \subseteq p$ is $\mathcal{M}_q$, and the null element corresponds to the empty subsets of $\mathcal{M}_q$.

Example 7.3 *Let S_1 and S_2 denote the two singleton sets containing the adder formula (7.1) and (7.2) respectively. We identify their model sets $M_1 = \hat{r}_q(S_1)$ and $M_2 = \hat{r}_q(S_2)$ which are information sets with domain $d(M_1) = \{In_1, In_2, In_3, Out_1\}$ and $d(M_2) = \{In_1, In_2, In_3, Out_2\}$:*

$M_1 =$

In_1	In_2	In_3	Out_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$M_2 =$

In_1	In_2	In_3	Out_2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The combination $M_1 \otimes M_2$ again corresponds to the information in Table 2.8.

To this algebra of information sets, we may associate a corresponding algebra of sentences. For that purpose, we show how the above rules for combination and projection can be expressed by sentences. Consider an information $M_1 = \hat{r}_q(S_1)$ described by a set S_1 of wffs in $\mathcal{L}_q$ and an information $M_2 = \hat{r}_u(S_2)$ described by $S_2 \subseteq \mathcal{L}_u$. We observe that S_1 and S_2 may also be seen as formulae of the language $S_1 \cup S_2 \subseteq \mathcal{L}_{q \cup u}$. Further, let

$$M^{\uparrow q \cup u} \;=\; \{\mathbf{m} \in \mathcal{M}_{q \cup u} : \mathbf{m}^{\downarrow q} \in M\}$$

be the *cylindric* or *vacuous extension* of $M \subseteq \mathcal{M}_q$ to $q \cup u$. Then, if $M = \hat{r}_q(S)$, we also have $M^{\uparrow q \cup u} = \hat{r}_{q \cup u}(S)$ and therefore

$$M_1 \otimes M_2 \;=\; M_1^{\uparrow q \cup u} \cap M_2^{\uparrow q \cup u} = \hat{r}_{q \cup u}(S_1) \cap \hat{r}_{q \cup u}(S_2) \;=\; \hat{r}_{q \cup u}(S_1 \cup S_2).$$

The last identity follows immediately from the definition of $\hat{r}(S)$. Thus, combining pieces of information means to take the union of their defining sentences. Further, if $M \subseteq \mathcal{M}_u$ is a set of models with $M = \hat{r}_u(S)$ for some set of sentences $S \subseteq \mathcal{L}_u$, it holds for $q \subseteq u$ that

$$M^{\downarrow q} = \{\mathbf{m}^{\downarrow q} : \mathbf{m} \in \hat{r}_u(S)\} = \hat{r}_q((\exists P_{i_1} \dots P_{i_n})S),$$

where $u - q = \{P_{i_1} \dots P_{i_n}\}$. This shows that projection or extraction of information corresponds to the existential quantification of the sentences describing the original information. In this way, an algebra of formulae is associated to the algebra of models. In fact, to each model set $M = \hat{r}_q(S)$ of $\mathcal{M}_q$ its theory $\check{r}_q(M) = \check{r}_q(\hat{r}_q(S))$ is associated. The operator $C_q(S) = \check{r}_q(\hat{r}_q(S))$ satisfies the axioms of a *closure operator* (Davey & Priestley, 1990). This will be proved in the general context in Lemma 7.2 of Section 7.3 below. Subsets $S \subseteq \mathcal{L}_q$ with $S = C_q(S)$ are therefore called *closed*. Clearly, $M = \hat{r}_q(S)$ implies that $M = \hat{r}_q(C_q(S))$ and therefore $C_q(S) = C_q(C_q(S))$. Thus, theories $C_q(S)$ are always closed sets, and the algebra of formulae is in fact an algebra of closed sets of formulae. Combination between closed sets $S_1 \subseteq \mathcal{L}_q$ and $S_2 \subseteq \mathcal{L}_U$ is defined by

$$S_1 \otimes S_2 = C_{q \cup u}(S_1 \cup S_2), \tag{7.5}$$

and projection of a closed set $S \subseteq \mathcal{L}_u$ to some subset $q \subseteq u$ by

$$S^{\downarrow q} = C_q((\exists P_{i_1} \dots P_{i_n})S) \tag{7.6}$$

where $u - q = \{P_{i_1} \dots P_{i_n}\}$. The neutral elements in this algebra are the *tautologies* $C_q(\emptyset)$, whereas the null elements equal the whole language $\mathcal{L}_q$. Both are clearly closed sets. This algebra of closed sets is closely related to the well-known *Lindenbaum algebra* of propositional logic (Davey & Priestley, 1990) which is a Boolean algebra that covers combination but not projection. Our algebra is a reduct of the Lindenbaum algebra as a Boolean algebra, but extended by the operation of projection.

To conclude this section, we formalize the relations between contexts c_q for different $q \in D$. If $u \subseteq q$, then any wff $s \in \mathcal{L}_u$ is also a wff in $\mathcal{L}_q$. Formally, we define this embedding $f_{u,q} : \mathcal{L}_u \rightarrow \mathcal{L}_q$ simply by the identity mapping $f_{u,q}(s) = s$. On the other hand, models $\mathbf{m} \in \mathcal{M}_q$ can be projected to $\mathcal{M}_u$. We therefore define the projection mapping $g_{q,u} : \mathcal{M}_q \rightarrow \mathcal{M}_u$ by $g_{q,u}(\mathbf{m}) = \mathbf{m}^{\downarrow u}$. Then, the pair of *contravariant mappings* $f_{u,q}$ and $g_{q,u}$ clearly satisfies the following property:

$$\mathbf{m} \models_q f_{u,q}(s) \iff g_{q,u}(\mathbf{m}) \models_u s.$$

Such a pair of contravariant mappings is called an *infomorphism* between the contexts c_q and c_u (Barwise & Seligman, 1997). They are considered in a more general and abstract setting in Section 7.3 below.

7.1.3 Some Computational Aspects

The formalism of propositional logic induces a dual pair of valuation algebras that arises from the language and its interpretation. In the first case, valuations are closed sets of formulae and their rules of combination and projection are given in equations (7.3) and (7.4). In the second case, valuations are sets of models and their operations of combination and projection are given in equations (7.5) and (7.6). The proof that the valuation algebra axioms are satisfied in both systems will be given in a more general context in Section 7.3.2 below. Hence, inference problems with knowledgebases from propositional logic can be computed by the local computation architectures of Chapters 3 and 4. We now give two important examples of such inference problems.

■ 7.1 Satisfiability in Propositional Logic

Let $\{\phi_1, \ldots, \phi_n\} \subseteq \Phi$ be a set of propositional information pieces, either represented in the valuation algebra of closed sets of formulae or in the valuation algebra of model sets. In any case, this set is interpreted conjunctively, i.e. the objective function $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ is satisfiable if, and only if, each factor ϕ_i with $i = 1, \ldots, n$ is satisfiable. A satisfiablity test therefore reduces to a single-query inference problem with the empty set as query:

$$\phi^{\downarrow\emptyset} \quad = \quad (\phi_1 \otimes \ldots \otimes \phi_n)^{\downarrow\emptyset}.$$

Assume that we work on the language level. If we obtain the tautology $\phi^{\downarrow\emptyset} = \{\top\}$ then the knowledgebase is satisfiable. Otherwise, if $\phi^{\downarrow\emptyset} = \{\bot\}$ then the knowledgebase is *contradictory*. If we work on model level, then $\phi^{\downarrow\emptyset} = \{\diamond\}$ indicates a satisfiable knowledgebase and $\phi^{\downarrow\emptyset} = \emptyset$ a contradictory knowledgebase.

■ 7.2 Theorem Proving in Propositional Logic

Let $\{\phi_1, \ldots, \phi_n\} \subseteq \Phi$ be a set of propositional information pieces and $\phi_h \in \Phi$ a *hypothesis*. Testing whether the hypothesis is true under the given knowledgebase is equivalent to verifying whether $\{\phi_1, \ldots, \phi_n\} \cup \{\phi_{\neg h}\}$ is contradictory. This induces an inference problem according to the foregoing instance. Here, $\phi_{\neg h}$ denotes the negation of the hypothesis. Depending on the knowledgebase, this is either expressed in the valuation algebra on language or on model level.

If some propositional information over the variables $q \subseteq p$ is expressed by a set of formulae $S \subseteq \mathcal{L}_q$, then the corresponding valuation on language level is the closed set of formulae $C_q(S)$. Closed sets of a set of formulae correspond to their theory and are infinite constructs. For practical applications, it is therefore essential to work with some finite representation of $C_q(S)$, most appropriately with some *normal form* of S. It is then important that the valuation algebra operations can be expressed with respect to the chosen normal form. This requires that executing a combination or projections again leads to a corresponding normal form. There are many different normal forms that could be used (Darwiche & Marquis, 2001). As an example, we use the *conjunctive normal form* .

A *positive literal* $P \in p$ is a propositional variable and its negation $\neg P$ is called a *negative literal*. A *clause* φ is a disjunction of either positive or negative literals l_i for $i = 1, \ldots, m$, i.e. $\varphi = l_1 \vee \ldots \vee l_m$. Such clauses are called *proper* if every propositional variable appears at most once. A formula is in conjunctive normal form (CNF) if it is written as a conjunction of proper clauses, i.e. $f = \varphi_1 \wedge \ldots \wedge \varphi_n$ where φ_i are proper clauses for $i = 1, \ldots, n$. It is known that every formula can be transformed into an equivalent CNF (Chang & Lee, 1973) and because sets of formulae are interpreted conjunctively, we may also transform sets of formulae into a CNF. It is common to represent CNFs as *clause sets* $\Sigma = \{\varphi_1, \ldots, \varphi_n\}$, which henceforth are used as normal form. Following (Kohlas *et al.*, 1999) we define the combination of two clause sets Σ_1 and Σ_2 by

$$\Sigma_1 \otimes \Sigma_2 \quad = \quad \mu(\Sigma_1 \cup \Sigma_2). \tag{7.7}$$

Here, μ denotes the *subsumption operator* that eliminates non-minimal clauses. This means that if all literals of some clause are contained in another clause, then the second is a logical consequence of the first and can be removed. We observe that if $C_q(\Sigma_1) = C_q(S_1)$ and $C_u(\Sigma_2) = C_u(S_2)$, the result of this combination is again a clause set whose closure satisfies

$$C_{q \cup u}(\mu(\Sigma_1 \cup \Sigma_2)) \quad = \quad C_{q \cup u}(S_1 \cup S_2). \tag{7.8}$$

For the definition of projection it is more suitable to switch to variable elimination. We first remark that every clause set Σ, obtained from a set of formulae $S \subseteq \mathcal{L}_u$, can be decomposed into disjoint subsets with respect to a proposition $X \in u$:

$$\begin{aligned}
\Sigma_X &= \{\varphi \in \Sigma : \varphi \text{ contains } X \text{ as positive literal}\}, \\
\Sigma_{\overline{X}} &= \{\varphi \in \Sigma : \varphi \text{ contains } X \text{ as negative literal}\}, \\
\Sigma_{\dot{X}} &= \{\varphi \in \Sigma : \varphi \text{ does not contain } X \text{ at all}\}.
\end{aligned}$$

This is possible because Σ contains only proper clauses. Then, the elimination of a variable $X \in u$ is defined by

$$\Sigma^{-X} \quad = \quad \mu(\Sigma_{\dot{X}} \cup R_X(\Sigma)), \tag{7.9}$$

where

$$R_X(\Sigma) \quad = \quad \{\vartheta_1 \vee \vartheta_2 \ : \ X \vee \vartheta_1 \in \Sigma_X \text{ and } \neg X \vee \vartheta_2 \in \Sigma_{\overline{X}}\} \tag{7.10}$$

is the set of *resolvants* between clauses in Σ_X and $\Sigma_{\overline{X}}$. We again observe that the result of this operation is a clause set. Moreover, equation (7.9) corresponds to existential quantification with respect to the proposition $X \in u$, such that we have

$$C_{u-\{X\}}(\Sigma^{-X}) \quad = \quad C_{u-\{X\}}((\exists X)S), \tag{7.11}$$

see (Kohlas *et al.*, 1999). Clause sets are closed under combination (7.7) and variable elimination (7.9). Because further the equations (7.8) and (7.11) hold, we conclude that clause sets form a valuation algebra. A direct proof of this statement can also be found in (Haenni *et al.*, 2000). The following example illustrates the computation with propositional clause sets.

Example 7.4 *Consider the propositional variables $p = \{U, V, W, X, Y, Z\}$ and two clause sets Σ_1 and Σ_2 defined as*

$$\begin{aligned}\Sigma_1 &= \{X \vee Y, X \vee \neg Y \vee Z, Z \vee \neg W\}\\ \Sigma_2 &= \{U \vee \neg W \vee Z, \neg X \vee W, X \vee V\}.\end{aligned}$$

Combining Σ_1 and Σ_2 gives

$$\begin{aligned}\Sigma_1 \otimes \Sigma_2 &= \mu\{X \vee Y, X \vee \neg Y \vee Z, Z \vee \neg W, U \vee \neg W \vee Z, \neg X \vee W, X \vee V\}\\ &= \{X \vee Y, X \vee \neg Y \vee Z, Z \vee \neg W, \neg X \vee W, X \vee V\}.\end{aligned}$$

Observe that the variable $U \in p$ is contained in Σ_2 but disappears in the combination. This seems to contradict the labeling axiom of valuation algebras. However, it does not because $p - \{U\} \subset p$ implies that $\mathcal{L}_{p-\{U\}} \subset \mathcal{L}_p$. This allows us to consider $\Sigma_1 \otimes \Sigma_2$ as a valuation with domain $d(\Sigma_1 \otimes \Sigma_2) = p$. To eliminate variable $X \in p$ we partition the clause set as follows:

$$\begin{aligned}(\Sigma_1 \otimes \Sigma_2)_X &= \{X \vee Y, X \vee \neg Y \vee Z, X \vee V\},\\ (\Sigma_1 \otimes \Sigma_2)_{\overline{X}} &= \{\neg X \vee W\},\\ (\Sigma_1 \otimes \Sigma_2)_{\dot{X}} &= \{Z \vee \neg W\}.\end{aligned}$$

We then obtain

$$\begin{aligned}(\Sigma_1 \otimes \Sigma_2)^{-X} &= \mu(\{Z \vee \neg W\} \cup \{Y \vee W, \neg Y \vee Z \vee W, V \vee W\})\\ &= \{Z \vee \neg W, Y \vee W, \neg Y \vee Z \vee W, V \vee W\}.\end{aligned}$$

For practical purposes, the use of the CNF normal form is often unsuitable since clause sets may be very large. We therefore prefer a more compact representation as for example *prime implicates*. A proper clause φ is called *implicate* of a sentence $\gamma \in \mathcal{L}$, if $\gamma \models \varphi$. An implicate φ of γ is then a *prime implicate* if no proper subclause of φ is also an implicate of γ. In other words, prime implicates of some sentence γ are the logically strongest consequences of γ. The set of all prime implicates of γ defines a conjunctive normal form denoted by $\Phi(\gamma)$. It is therefore possible to apply the above rules of combination and projection to sets of prime implicates, if we additionally change the μ operator in such a way that the result is again a set of prime implicates. We refer to (Haenni *et al.*, 2000) for a discussion of computing with prime implicates and other normal forms of propositional logic.

7.2 LINEAR EQUATIONS

The second formalism we are going to consider are systems of linear equations over sets of variables. We examine the solution spaces of such systems and introduce operations between solution spaces that are also mirrored by corresponding operations on the systems of linear equations. This program needs some notational conventions and we also remind some basic elements of linear algebra.

7.2.1 Equations and Solution Spaces

We consider an index set r of a set of variables which, in the interest of simplicity, is assumed to be finite, although the following discussion could be extended to a countable set without much difficulty. For a subset $s \subseteq r$ and $m \in \mathbb{N}$ we define a *linear system* of m equations

$$\sum_{j \in s} a_{i,j} X_j \quad = \quad b_i \tag{7.12}$$

for $i = 1, \ldots, m$. The number of equations m can be smaller, equal or larger than the number of variables $|s|$. Note that such equations are purely syntactic constructs. They obtain a meaning, if we specify the coefficients $a_{i,j}$ and b_i to be real numbers. Then, we are looking for real values for the variables X_j that satisfy the above equations, when the arithmetic sum and multiplication are used on the left-hand side. These assignments again correspond to real s-vectors $\mathbf{x} : s \rightarrow \mathbb{R}$ such that

$$\sum_{j \in s} a_{i,j} \mathbf{x}_j \quad = \quad b_i$$

for $i = 1, \ldots, m$. Any s-vector that satisfies this relation is called a *solution* to the linear system. There may be no solution, exactly one solution or infinitely many solutions. We consider the set of solutions of the system (7.12) as the information expressed with respect to the variables involved, and refer to it as the *solution space* of system (7.12).

In order to discuss solution spaces of linear systems in general, we need to remind some basic facts of linear algebra. The s-vectors form a *linear space* where addition is defined component-wise, i.e. $\mathbf{x} + \mathbf{y}$ has components $\mathbf{x}_i + \mathbf{y}_i$ for $i \in s$, and scalar multiplication $c \cdot \mathbf{x}$ of $\mathbf{x}$ with a real value c has components $c \cdot \mathbf{x}_i$. The linear space of s-vectors is denoted by $\mathbb{R}^s$ and its *dimension* equals $|s|$. Similarly, for $m \in \mathbb{N}$, we define m-vectors by mappings $\mathbf{b} : \{1, \ldots, m\} \rightarrow \mathbb{R}$ which also form a linear space with component-wise addition and scalar multiplication. This linear space of dimension m is denoted by $\mathbb{R}^m$. Finally, the $m \times s$ matrix $\mathbf{A}$ of the linear system (7.12) is a mapping $\{1, \ldots, m\} \times s \rightarrow \mathbb{R}$ with components $a_{i,j}$ for $i \in \{1, \ldots, m\}$ and $j \in s$. It defines a *linear mapping* $\mathbf{A} : \mathbb{R}^s \rightarrow \mathbb{R}^m$ determined by the matrix-vector product $\mathbf{x} \mapsto \mathbf{A}\mathbf{x} \in \mathbb{R}^m$ for $\mathbf{x} \in \mathbb{R}^s$. The mapping is linear since $\mathbf{A}(\mathbf{x} + \mathbf{y}) = \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y}$ and $\mathbf{A}(c \cdot \mathbf{x}) = c \cdot \mathbf{A}\mathbf{x}$. Its *range* is the set of all vectors in $\mathbb{R}^m$ which are images of s-vectors in $\mathbb{R}^s$,

$$\mathcal{R}(\mathbf{A}) \quad = \quad \{\mathbf{z} \in \mathbb{R}^m : \exists \mathbf{x} \in \mathbb{R}^s \text{ such that } \mathbf{A}\mathbf{x} = \mathbf{z}\}.$$

The range $\mathcal{R}(\mathbf{A})$ is a linear subspace of $\mathbb{R}^m$ and therefore has a determined dimension $dim(\mathcal{R}(\mathbf{A}))$ called the *rank* of $\mathbf{A}$. We subsequently write $rank(\mathbf{A})$ for the rank of the linear mapping $\mathbf{A}$ which corresponds to the maximal number of linearly independent rows or columns of $\mathbf{A}$. The *null space* of the mapping $\mathbf{A}$ is the set of s-vectors that map to the zero vector $\mathbf{0} \in \mathbb{R}^m$,

$$\mathcal{N}(A) = \{\mathbf{x} \in \mathbb{R}^s : \mathbf{Ax} = \mathbf{0}\}.$$

This is again a linear space, a subspace of $\mathbb{R}^s$ with a determined dimension $dim(\mathcal{N}(\mathbf{A}))$. A well-known theorem of linear algebra states that

$$|s| = rank(\mathbf{A}) + dim(\mathcal{N}(\mathbf{A})). \tag{7.13}$$

Thus, a linear mapping is associated to every system of linear equations determined by the matrix of the system. The solution space of the linear system can be described in terms of this mapping: If $\mathbf{y}$ is a solution of (7.12), i.e. $\mathbf{Ay} = \mathbf{b}$, then clearly $\mathbf{y} + \mathbf{x}_0$ is a solution of (7.12) for any $\mathbf{x}_0 \in \mathcal{N}(A)$, and any solution can be written in this form. The solution space of (7.12) is therefore

$$\mathcal{S}(\mathbf{A}, \mathbf{b}) = \{\mathbf{x} \in \mathbb{R}^s : \mathbf{x} = \mathbf{y} + \mathbf{x}_0,\ \mathbf{x}_0 \in \mathcal{N}(\mathbf{A})\} = \mathbf{y} + \mathcal{N}(\mathbf{A}).$$

The solutions to (7.12) are thus determined by a particular solution $\mathbf{y}$ to the linear system and the set of solutions to the corresponding *homogeneous system* $\mathbf{AX} = \mathbf{0}$. A set $\mathbf{y} + \mathcal{L}$, where $\mathbf{y}$ is an s-vector and $\mathcal{L}$ is a linear subspace of $\mathbb{R}^s$, is called an *affine space* in $\mathbb{R}^s$. Its dimension equals the dimension of $\mathcal{L}$. So, the solution spaces of linear equations over variables in s are affine spaces in $\mathbb{R}^s$. The following example illustrates these concepts by a linearly dependent, under-determined system:

Example 7.5 *For a set of variables $\{X_1, X_2, X_3\}$ consider the linear system:*

$$\begin{array}{rcrcrcr} X_1 & - & 2X_2 & + & 2X_3 & = & -1 \\ 3X_1 & + & 5X_2 & - & 3X_3 & = & 8 \\ 4X_1 & + & 3X_2 & - & X_3 & = & 7 \end{array}$$

We have $|s| = m = 3$ and the associated matrix determining the linear mapping is:

$$\mathbf{A} = \begin{bmatrix} 1 & -2 & 2 \\ 3 & 5 & -3 \\ 4 & 3 & -1 \end{bmatrix}$$

This equation system is linearly dependent because the third equation corresponds to the sum of the two others. We have $rank(\mathbf{A}) = dim(\mathcal{R}(\mathbf{A})) = 2$ and according to (7.13) $dim(\mathcal{N}(\mathbf{A})) = 1$. Transforming $\mathbf{A}$ into a lower triangular matrix by subtracting the triple of the first row from the second row, and by subtracting the sum of the first two rows from the third, gives

$$\mathbf{A} = \begin{bmatrix} 1 & -2 & 2 \\ 0 & 11 & -9 \\ 0 & 0 & 0 \end{bmatrix}$$

Thus, we may describe the null space of the linear mapping represented by A as

$$\mathcal{N}(\mathbf{A}) = \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_1 = -\frac{4}{11}x_3,\ x_2 = \frac{9}{11}x_3\}.$$

A particular solution to the original system is: $X_1 = 1$, $X_2 = 1$ and $X_3 = 0$. We can therefore describe the solution space by

$$\mathcal{S}(\mathbf{A}, \mathbf{b}) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \mathcal{N}(\mathbf{A}).$$

There are four possible cases to be distinguished at this stage:

1. $rank(\mathbf{A}) = m < |s|$;
2. $rank(\mathbf{A}) < m < |s|$;
3. $rank(\mathbf{A}) = |s| \leq m$;
4. $rank(\mathbf{A}) < |s| \leq m$.

In the first case, the system (7.12) has a solution for any m-vector $\mathbf{b}$, since $\mathcal{R}(\mathbf{A}) = \mathbb{R}^m$, and it follows from (7.13) that $dim(\mathcal{N}(\mathbf{A})) = |s| - m > 0$. Consequently, the solution space has dimension $dim(\mathcal{N}(\mathbf{A}))$. In the second case, the system (7.12) has solutions only if $\mathbf{b} \in \mathcal{R}(\mathbf{A})$. Then, the solution space is an affine space of dimension $dim(\mathcal{N}(\mathbf{A})) > 0$. In the third case, we see that $dim(\mathcal{N}(\mathbf{A})) = 0$ according to (7.13). So, if the system (7.12) has a solution, it is *unique*. The solution space is a point, and it has a solution if $b \in \mathcal{R}(\mathbf{A})$. If $|s| = m$, then this is automatically the case. Finally, in the fourth case, we again have $dim(\mathcal{N}(\mathbf{A})) > 0$ and solutions exist, if and only if, $\mathbf{b} \in \mathcal{R}(\mathbf{A})$. The solution space again has dimension $dim(\mathcal{N}(\mathbf{A}))$. Example 7.5 presents a system in this case. So, solution spaces of systems (7.12) are either empty or affine spaces in $\mathbb{R}^s$. It is convenient to consider the empty set also as an affine space in $\mathbb{R}^s$. Then, solution spaces are always affine spaces without exception. Conversely, we remark that any affine space $\mathbf{y} + \mathcal{L}$ in $\mathbb{R}^s$ is the solution space of some linear system over variables in s. This is another well-known result from linear algebra, and it clarifies the relation between affine spaces in $\mathbb{R}^s$ and systems of linear equations over a set s of variables.

We now look at systems of linear equations and solutions in a way similar to the propositional logic case in the previous section. Consider syntactic constructions like

$$e := \sum_{i \in s} a_i X_i = b,$$

with a_i and b being real numbers. Such single equations are the sentences or formulae of a language $\mathcal{L}_s$ defined over the variables X_i with indices from a set $s \subseteq r$. If an s-vector $\mathbf{x}$ is a solution of such an equation e, we write $\mathbf{x} \models_s e$ which defines a binary relation $\models_s$ in $\mathcal{L}_s \times \mathbb{R}^s$. We now consider the structure $\langle \mathcal{L}_s, \mathbb{R}^s, \models_s \rangle$ and reformulate the above discussion about systems of linear equations in terms of this structure: For an equation $e \in \mathcal{L}_s$ let $\hat{r}(e)$ be the set of all solutions to this equation,

$$\hat{r}(e) = \{\mathbf{x} \in \mathbb{R}^s : \mathbf{x} \models_s e\}.$$

If $E \subseteq \mathcal{L}_s$ is a set of equations, then $\hat{r}(E)$ denotes the set of all solutions to the equations in E,

$$\hat{r}(E) = \{\mathbf{x} \in \mathbb{R}^s : \mathbf{x} \models_s e, \forall e \in E\}.$$

If E is a finite set, then $\hat{r}(E)$ is an affine space. Conversely, we may look for all equations having a given s-vector as solution,

$$\check{r}(\mathbf{x}) = \{e \in \mathcal{L}^s : \mathbf{x} \models_s e\}.$$

Similarly, for a set $M \subseteq \mathbb{R}^s$ of s-vectors,

$$\check{r}(M) = \{e \in \mathcal{L}^s : \mathbf{x} \models_s e, \forall \mathbf{x} \in M\}$$

is the set of equations for which all elements of M are solutions. We observe that

$$E_1 \subseteq E_2 \Rightarrow \hat{r}(E_1) \supseteq \hat{r}(E_2) \quad \text{and} \quad M_1 \subseteq M_2 \Rightarrow \check{r}(M_1) \supseteq \check{r}(M_2).$$

These concepts allow us to derive and express some well-known facts from linear algebra: first, two systems E and E' of linear equations are called *equivalent*, if they have the same solution space, i.e. $E \equiv_s E'$ if, and only if, $\hat{r}(E) = \hat{r}(E')$. Then, $E' \subseteq E$ is called a *minimal system*, if $E \equiv_s E'$ and there is no subset $E'' \subset E'$ such that $E \equiv_s E''$. For any set of equations E, and thus also for any affine space $\mathbf{y} + \mathcal{L}$, there clearly exist such minimal systems E'. They are characterized by the property that their matrix $\mathbf{A}'$ has full rank m, where $m = |E'| \leq |s|$ is the number of equations in E'. In fact, if $rank(\mathbf{A}) < m \leq |s|$, then the equations are linearly dependent, and we may eliminate $m - rank(\mathbf{A})$ of them to get an equivalent system with matrix $\mathbf{A}'$. We then have $rank(\mathbf{A}') = m = |s| - dim(\mathcal{L})$ if $|s| \leq m$. We may likewise eliminate $m - r$ equations if the system has rank r. Further, if M is a subset of an affine space $\mathbf{y} + \mathcal{L}$ then $\check{r}(M) \supseteq \check{r}(\mathbf{y} + \mathcal{L})$, hence $\hat{r}(\check{r}(M)) \subseteq \hat{r}(\mathbf{y} + \mathcal{L})$. If equality holds, then M is said to span the affine space $\mathbf{y} + \mathcal{L}$. In any case, $\hat{r}(\check{r}(M))$ is itself an affine space spanned by M. It is also clear that $\mathbf{y} + \mathcal{L}$ spans itself, so that $\hat{r}(\check{r}(\mathbf{y} + \mathcal{L})) = \mathbf{y} + \mathcal{L}$ must hold. For any affine space $\mathbf{y} + \mathcal{L}$ there are minimal sets M which span the affine space, and the cardinality of these minimal sets equals the dimension of the affine space.

Structures $\langle \mathcal{L}_s, \mathbb{R}^s, \models_s \rangle$ of systems of linear equations over sets of variables $s \subseteq r$ may of course be considered for any non-empty subset $s \subseteq r$. The relations between these structures for different subsets will next be considered. Assume $t \subseteq s$. Then, any equation $e \in \mathcal{L}_t$ with

$$e := \sum_{i \in t} a_i X_i = b$$

may also be seen as an equation in $\mathcal{L}_s$ via the embedding

$$f_{t,s}(e) \quad := \quad \sum_{i \in s} a_i X_i \;=\; b,$$

where $a_i = 0$ if $i \in s - t$. Also, s-vectors $\mathbf{x}$ may be projected to the subspace $\mathbb{R}^t$. This projection is denoted by $g_{s.t}$ and defined component-wise for $i \in t$ by

$$g_{t,s}(\mathbf{x})(i) \quad = \quad \mathbf{x}_i.$$

We observe that if the s-vector $\mathbf{x}$ is a solution of the equation $f_{t,s}(e)$, then the t-vector $g_{s,t}(\mathbf{x})$ is a solution of the equation e and vice-versa. It therefore holds that

$$\mathbf{x} \models_s f_{t,s}(e) \quad \Longleftrightarrow \quad g_{s,t}(\mathbf{x}) \models_t e.$$

Subsequently, we again write $\mathbf{x}^{\downarrow t}$ for the projection of an s-vector to $t \subseteq s$. Hence, the contravariant pair of mappings $f_{t,s} : \mathcal{L}_s \rightarrow t$ and $g_{s,t} : \mathbb{R}^s \rightarrow \mathbb{R}^t$ have the same property as the corresponding mappings introduced for propositional logic in Section 7.1. They form again an *informorphism*. This hints at some common structure behind the two systems of propositional logic and linear equations which will be discussed in Section 7.3. Here, we continue by showing that affine spaces form an information algebra. This yields a foundation for discussing local computation techniques for the solution of linear systems in Chapter 9.

7.2.2 Algebra of Affine Spaces

Let r be a finite index set and D its lattice of subsets. We denote by Φ_s the family of affine spaces in the linear space $\mathbb{R}^s$ where for $s = \emptyset$, $\Phi_\emptyset = \{\emptyset\}$ is the only affine space. We further define

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

and introduce within this family of affine spaces the operations of labeling, combination and projection. Note that the affine spaces in $\mathbb{R}^s$ represented by $\mathbf{y} + \mathcal{L}'$ und $\mathbf{y}' + \mathcal{L}$ are identical, if $\mathcal{L} = \mathcal{L}'$ and $\mathbf{y} - \mathbf{y}' \in \mathcal{L}$.

The labeling operation is simply defined by $d(\phi) = s$ if $\phi = \mathbf{y} + \mathcal{L}$ is an affine space in $\mathbb{R}^s$, i.e. $\mathcal{L}$ is a linear subspace of $\mathbb{R}^s$. For the combination of two affine spaces in $\mathbb{R}^s$ and $\mathbb{R}^t$, we first extend them to $\mathbb{R}^{s \cup t}$ and compute their intersection. So, let us define the extension of an affine space: If ϕ is an affine space with $d(\phi) = s$ and $s \subseteq t$, then

$$\phi^{\uparrow t} \quad = \quad \{\mathbf{x} \in \mathbb{R}^t : \mathbf{x}^{\downarrow s} \in \phi\}.$$

We show that $\phi^{\uparrow t}$ is again an affine space: From $\phi = \mathbf{y} + \mathcal{L}$ and $\mathbf{x}^{\downarrow s} \in \phi$ we conclude that $\mathbf{x}^{\downarrow s} = \mathbf{y} + \mathbf{x}_0$ for some $\mathbf{x}_0 \in \mathcal{L}$. This implies that $\phi^{\uparrow t} = \mathbf{y}^{\uparrow t} + \mathcal{L}^{\uparrow t}$ with

$$\mathbf{y}^{\uparrow t} \quad = \quad \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix}$$

where 0 denotes the zero vector for the domain $t - s$ and

$$\mathcal{L}^{\uparrow t} = \{\mathbf{x}_0 \in \mathbb{R}^t : \mathbf{x}_0^{\downarrow s} \in \mathcal{L}\}.$$

It is easy to verify that $\mathcal{L}^{\uparrow t}$ is a linear space since $(\mathbf{x}_0 + \tilde{\mathbf{x}}_0)^{\downarrow s} = \mathbf{x}_0^{\downarrow s} + \tilde{\mathbf{x}}_0^{\downarrow s}$ and $(c \cdot \mathbf{x}_0)^{\downarrow s} = c \cdot \mathbf{x}_0^{\downarrow s}$. Therefore, $\phi^{\uparrow t}$ is an affine space in $\mathbb{R}^t$ which finally allows us to define the combination between an affine space $\phi \in \Phi_s$ and an affine space $\psi \in \Phi_t$:

$$\phi \otimes \psi = \phi^{\uparrow s \cup t} \cap \psi^{\uparrow s \cup t}.$$

This is again identical to the operation of natural join in the relational algebra:

$$\phi \otimes \psi = \phi \bowtie \psi = \{\mathbf{x} \in \mathbb{R}^{s \cup t} : \mathbf{x}^{\downarrow s} \in \phi \text{ and } \mathbf{x}^{\downarrow t} \in \psi\}.$$

It remains to verify that $\phi \otimes \psi$ is still an affine space which follows immediately from the following argument: Let ϕ be represented by a system of linear equation E over variables in s, and ψ by a system E' over variables in t. Then, the solution space of the union $E \cup E'$ of these two systems is clearly $\phi \bowtie \psi = \phi \otimes \psi$, hence an affine space.

For the definition of projection let $\phi = \mathbf{y} + \mathcal{L}$ be an affine space in $\mathbb{R}^s$ and $t \subseteq s$. The projection $\phi^{\downarrow t}$ is defined as in the relational algebra by

$$\phi^{\downarrow t} = \{\mathbf{x}^{\downarrow t} : \mathbf{x} \in \phi\}.$$

Since $\mathbf{x} \in \phi$ implies that $\mathbf{x} = \mathbf{y} + \mathbf{x}_0$ for some $\mathbf{x}_0 \in \mathcal{L}$, we obtain $\mathbf{x}^{\downarrow t} = \mathbf{y}^{\downarrow t} + \mathbf{x}_0^{\downarrow t}$ and therefore

$$\phi^{\downarrow t} = \mathbf{y}^{\downarrow t} + \mathcal{L}^{\downarrow t}.$$

Clearly, $\mathcal{L}^{\downarrow t}$ is a linear subspace of $\mathbb{R}^t$ and therefore $\phi^{\downarrow t}$ an affine space in $\mathbb{R}^t$.

Altogether, the algebraic structure $\langle \Phi, D \rangle$ with the operations of labeling, combination and projection is well-defined. Combination has $\mathbb{R}^s$ as a neutral element in domain s and the empty set as null element. Thus, we see that affine spaces form a subalgebra of the relational algebra of s-vectors and therefore inherit the properties of an information algebra. In other words, affine spaces provide another instance of an information algebra and are therefore amenable to the application of local computation. However, we generally cannot compute with affine spaces directly, since they are infinite sets. Instead, we revert to finite representations of them through systems of linear equations. We refer to Chapter 9 for such computational aspects. Finally we note that the above discussion also applies to linear equations over arbitrary fields. This identifies another generic construction that produces a new information algebra for each field. Particularly important among them are Galois fields that have many important applications in coding theory.

7.3 INFORMATION IN CONTEXT

In the previous two sections, we described the two rather different formalisms of propositional logic and systems of linear equations. Different as they are, we have nevertheless worked out some common features. Essentially, information as set of models or set of vectors is represented or described by sentences of some formal language, and sets of such sentences determine the information described by them. In this section, we examine the abstract structure behind both formalisms that leads to a further generic construction for obtaining information algebras.

7.3.1 Contexts: Models and Language

We introduce an abstract concept called *context* that represents the connection between language and information. Here, language refers to a very simple form of a *formal language* such as the language of propositional logic or linear equations. For our purposes, it is sufficient to represent a language $\mathcal{L}$ by the set of its *sentences* without regard to the syntactic structure of these sentences. In general terms, information concerns a *question* of interest which formally is represented by the set of its *possible answers* denoted by $\mathcal{M}$. In the example of propositional logic $\mathcal{M}$ is the set of interpretations of propositions. In the case of linear equations over variables of an index set r, $\mathcal{M}$ is the set of r-vectors. We refer to the elements of $\mathcal{M}$ as *models* and further assume a binary relation $\models \subseteq \mathcal{L} \times \mathcal{M}$. For a pair $(s, \mathbf{m}) \in \mathcal{L} \times \mathcal{M}$ we write $\mathbf{m} \models s$ if $(s, \mathbf{m}) \in \models$. In propositional logic this means that model $\mathbf{m}$ satisfies formula s. For linear equations it means that vector $\mathbf{m}$ is a solution to equation s.

A triple $\langle \mathcal{L}, \mathcal{M}, \models \rangle$ is called a *context* and serves to express information by sentences: If $s \in \mathcal{L}$ is a sentence, then

$$\hat{r}(s) \quad = \quad \{\mathbf{m} \in \mathcal{M} : \mathbf{m} \models s\}$$

is the set of models satisfying s, which is thought of being the information described by stating s. If $S \subseteq \mathcal{L}$,

$$\hat{r}(S) \quad = \quad \{\mathbf{m} \in \mathcal{M} : \mathbf{m} \models s, \forall s \in S\}$$

is the set of models satisfying all sentences of S or the information expressed by S. Similarly, if $\mathbf{m} \in \mathcal{M}$ is a model, then

$$\check{r}(\mathbf{m}) \quad = \quad \{s \in \mathcal{L} : \mathbf{m} \models s\}$$

is the set of sentences satisfied by $\mathbf{m}$. And if $M \subseteq \mathcal{M}$ is a set of models, then

$$\check{r}(M) \quad = \quad \{s \in \mathcal{L} : \mathbf{m} \models s, \forall \mathbf{m} \in M\}$$

is the set of sentences satisfied by all elements of M. It can be considered as the theory of M, i.e. the set of all sentences which express M.

It is important to note the formal duality between the concepts derived from $\mathcal{L}$ and $\mathcal{M}$. This is emphasized by the following lemma which collects some elementary properties of these mappings between the power sets of $\mathcal{M}$ and $\mathcal{L}$:

Lemma 7.1 *Let $\hat{r}$ and $\check{r}$ be the above operators for a context $\langle \mathcal{L}, \mathcal{M}, \models \rangle$. Then if $S, S_j \subseteq \mathcal{L}$ and $M, M_j \subseteq \mathcal{M}$, the following dual pairs of properties hold:*

$$\begin{array}{lrl}
1. & S \subseteq \check{r}(\hat{r}(S)) & M \subseteq \hat{r}(\check{r}(M)) \\
2. & S_1 \subseteq S_2 \Rightarrow \hat{r}(S_1) \supseteq \hat{r}(S_2) & M_1 \subseteq M_2 \Rightarrow \check{r}(M_1) \supseteq \check{r}(M_2) \\
3. & \hat{r}(S) = \hat{r}(\check{r}(\hat{r}(S))) & \check{r}(M) = \check{r}(\hat{r}(\check{r}(M))) \\
4. & \hat{r}(\bigcup_j S_j) = \bigcap_j \hat{r}(S_j) & \check{r}(\bigcup_j M_j) = \bigcap_j \check{r}(M_j).
\end{array}$$

Proof: We prove the properties for the language part. The corresponding results for the model part follow by duality.

1. Let $s \in S$. Then, $\mathbf{m} \in \hat{r}(S)$ means that $\mathbf{m} \models s$ for all $s \in S$. Consequently, s belongs to $\check{r}(\mathbf{m})$ for all $\mathbf{m} \in \hat{r}(S)$, which means that $s \in \check{r}(\hat{r}(S))$.

2. If $\mathbf{m} \in \hat{r}(S_2)$ we have $\mathbf{m} \models s$ for all $s \in S_2$, and therefore $\mathbf{m} \models s$ for all $s \in S_1 \subseteq S_2$. This implies that $\mathbf{m} \in \hat{r}(S_1)$.

3. From Property 1 and 2 it follows that $\hat{r}(S) \supseteq \hat{r}(\check{r}(\hat{r}(S)))$. But on the other hand, we conclude from the model part of Property 1 that $\hat{r}(S) \subseteq \hat{r}(\check{r}(\hat{r}(S)))$.

4. It follows from $S_j \subseteq \bigcup_j S_j$ and Property 2 that $\hat{r}(S_j) \supseteq \hat{r}(\bigcup_j S_j)$, hence $\bigcap_j \hat{r}(S_j) \supseteq \hat{r}(\bigcup_j S_j)$. Conversely, $\mathbf{m} \in \bigcap_j \hat{r}(S_j)$ means that $\mathbf{m} \models s$ for all $s \in S_j$ and all j. Therefore $\mathbf{m} \in \hat{r}(\bigcup_j S_j)$.

■

We next introduce the operators

$$C_{\models}(S) = \check{r}(\hat{r}(S)) \quad \text{and} \quad C^{\models}(M) = \hat{r}(\check{r}(M)).$$

We remark that $s \in C_{\models}(S)$ means that $\mathbf{m} \models s$ for all $\mathbf{m}$ which are models of S. In logic, such a relation is called a *logical consequence*, and it is said that s is a logical consequence of S, written as $S \models s$. So, $C_{\models}(S)$ is the set of all logical consequences of S. We therefore call $C_{\models}$ a *consequence operator*. Similar relations hold for the operator $C^{\models}$ on the model side. The following dual properties of consequence operators follow immediately from Lemma 7.1:

Lemma 7.2 *For the consequence operators $C_{\models}$ and $C^{\models}$ in $\langle \mathcal{L}, \mathcal{M}, \models \rangle$, and $S \subseteq \mathcal{L}$ and $M \subseteq \mathcal{M}$ the following properties hold:*

$$\begin{array}{lrl}
1. & S \subseteq C_{\models}(S) & M \subseteq M^{\models}(M) \\
2. & S_1 \subseteq S_2 \Rightarrow C_{\models}(S_1) \subseteq C_{\models}(S_2) & M_1 \subseteq M_2 \Rightarrow C^{\models}(M_1) \subseteq C^{\models}(M_2) \\
3. & C_{\models}(C_{\models}(S)) = C_{\models}(S) & C^{\models}(C^{\models}(M)) = C^{\models}(M).
\end{array}$$

Operators satisfying these three conditions are called *closure operators* and play an important role in the foundations of logic and in topology (Wojcicki, 1988; Norman & Pollard, 1996). We already came across closure operators in the context of Kleene valuation algebras in Chapter 6. Sets $S \subseteq \mathcal{L}$ and $M \subseteq \mathcal{M}$ such that $C_{\models}(S) = S$

and $C^{\models}(M) = M$ are called *closed*. Closed sets of sentences and models can be understood in the following sense: if a set S of sentences is stated, then $\hat{r}(S)$ is by Property 3 of Lemma 7.1 a closed set of models representing the *information* expressed by S. We therefore call a closed sets of models an *information set* or shortly, an *information*. In particular, $\hat{r}(S)$ is the information expressed by S. The closure $C_{\models}(S)$ of S is the set of all logical consequences of S called its *theory*. Similarly, if M is an arbitrary set of models, $\check{r}(M)$ is a closed set of sentences called the theory of M. Any set M of models determines an information set $C^{\models}(M)$. Note that $\hat{r}$ is a map $\hat{r} : \mathcal{P}(\mathcal{L}) \rightarrow \mathcal{P}(\mathcal{M})$ and $\check{r}$ is a contravariant map $\check{r} : \mathcal{P}(\mathcal{M}) \rightarrow \mathcal{P}(\mathcal{L})$ to $\hat{r}$. This pair of contravariant mappings satisfies the property that

$$M \subseteq \hat{r}(S) \quad \Longleftrightarrow \quad S \subseteq \check{r}(M). \tag{7.14}$$

This follows from Property 1 and 2 of Lemma 7.1. A contravariant pair of mappings satisfying (7.14) is an instance of a *Galois connection*. We mention that such structures are also used in *formal concept analysis* (Davey & Priestley, 1990).

It is time to illustrate and to link these abstract concepts with the examples of the previous two sections, as well as with some additional examples. In Section 7.3.2 it will then be shown that these formalisms satisfy the axioms of an information algebra under some additional assumptions. They are therefore new instances of our generic framework.

■ 7.3 Propositional Logic

Section 7.1 introduced the language and interpretation of propositional logic. Models are valuations $\mathbf{v}$ of propositional symbols and serve to interpret propositional formulae. More precisely, $\mathbf{v} \models s$ holds if the formula s evaluates to true under the valuation $\mathbf{v}$, i.e. if $\hat{\mathbf{v}}(s) = 1$. If we restrict ourselves to finite sets q of propositional symbols, then all sets of models are closed. The closure operators C_q correspond to the consequence operators $C_{\models}$ in the abstract setting above.

■ 7.4 Linear Equation Systems

Another example is provided by systems of linear equations discussed in Section 7.2. The language is formed by linear equations e over variables from an index set s with coefficients in a field, for example in the field of real or rational numbers. Models $\mathbf{x}$ are s-tuples with values in the field. The relation $\mathbf{x} \models e$ means that the s-tuple $\mathbf{x}$ is a solution to the equation e. In case of real numbers, the closed sets of models are exactly the affine spaces in $\mathbb{R}^s$. Note that an affine space of dimension k is spanned by k linearly independent vectors. The affine space spanned by a set of vectors M is the closure of M. The theories $C_{\models}(S)$ are formed by the totality of equations sharing the same affine space as solutions with the set S.

7.5 Linear Inequality Systems

Instead of linear equations, we may also consider *linear inequalities*

$$e \quad := \quad \sum_{i \in s} a_i X_i \ \leq a_0$$

over variables with indices from a set s and real-valued coefficients a_i and a_0. Such formulae are the elements of the language, and models are again s-vectors $\mathbf{x}$. The relation $\mathbf{x} \models e$ holds if $\mathbf{x}$ satisfies the inequality e, i.e. if

$$\sum_{i \in s} a_i \mathbf{x}_i \quad \leq \quad a_0.$$

Closed sets of models are *convex polyhedra* in the vector space $\mathbb{R}^s$ and, as we will see below, form an information valuation algebra. Systems of linear inequalities are especially used in *linear programming* (Chvátal, 1983; Winston & Venkataramanan, 2002).

7.6 Predicate Logic

A further example in the domain of logic is provided by *predicate logic*. The vocabulary of predicate logic consists of a countable set of variables $X_1, X_2, \ldots$ and a countable set of predicate symbols $P_1, P_2, \ldots$ and further includes the logical constants $\top$ and $\bot$ and the connectors $\wedge, \neg, \exists$. Each predicate symbol P_i has a definite *rank* ρ_i, and a predicate with rank ρ is referred to as a ρ-place predicate. Formulae of predicate logic are built according to the following rules:

1. $P_i X_{i_1} \ldots X_{i_\rho}$, where ρ is the rank of P_i, $\bot$ and $\top$ are (atomic) formulae.

2. If f is a formula, then $\neg f$ and $(\exists X_i) f$ are formulae.

3. If f and g are formulae, then $f \wedge g$ is a formula.

The predicate language $\mathcal{L}$ consists of all formulae which are obtained by applying these rules a finite number of times. We consider also the predicate language $\mathcal{L}_s$ where only variables from a subset s of variables are allowed. Often, s is restricted to be a finite set.

In order to define an interpretation for formulae of predicate logic, we choose a *relational structure* $\mathcal{R} = (U, R_1, R_2, \ldots)$ where U is a nonempty set called *universe*, and the R_i are relations among elements of U with arity ρ_i equal to the rank of predicate P_i. In other words, R_i are subsets of U^{ρ_i}. A *valuation* is a mapping $\mathbf{v} : \{1, 2, \ldots\} \rightarrow U$ which assigns to each variable X_i a value $\mathbf{v}(i) \in U$. We write U^ω for the set of all possible valuations. Given a valuation $\mathbf{v}$ and an index i, we define the set of all possible valuations that agree with $\mathbf{v}$ on all values except $\mathbf{v}(i)$,

$$\mathbf{v}^{\Rightarrow i} \quad = \quad \{\mathbf{u} \in U^\omega : \mathbf{u}(j) = \mathbf{v}(j) \text{ for } j \neq i\}.$$

Valuations $\mathbf{v}$ are used to assign a *truth value* $\hat{\mathbf{v}}(f) \in \{0,1\}$ to formulae $f \in \mathcal{L}$. This assignment is defined inductively as follows:

1. $\hat{\mathbf{v}}(\top) = 1$ and $\hat{\mathbf{v}}(\bot) = 0$;

2. $\hat{\mathbf{v}}(P_i X_{i_1} \ldots X_{i_\rho}) = 1$, if $(\mathbf{v}(i_1), \ldots, \mathbf{v}(i_\rho)) \in R_i$ and $\hat{\mathbf{v}}(P_i X_{i_1} \ldots X_{i_\rho}) = 0$ otherwise;

3. $\hat{\mathbf{v}}(\neg f) = 1$ if $\hat{\mathbf{v}}(f) = 0$ and $\hat{\mathbf{v}}(\neg f) = 0$ otherwise;

4. $\hat{\mathbf{v}}((\exists X_i)f) = 1$ if there is a valuation $\mathbf{u} \in \mathbf{v}^{\Rightarrow i}$ such that $\hat{\mathbf{u}}(f) = 1$ and $\hat{\mathbf{v}}((\exists X_i)f) = 0$ otherwise;

5. $\hat{\mathbf{v}}(f \wedge g) = 1$ if $\hat{\mathbf{v}}(f) = \hat{\mathbf{v}}(g) = 1$ and $\hat{\mathbf{v}}(f \wedge g) = 0$ otherwise.

A valuation $\mathbf{v}$ is called a model of a formula f in the structure $\mathcal{R}$ if $\hat{\mathbf{v}}(f) = 1$. We then write $\mathbf{v} \models f$. This defines the relation $\models$ between formulae of $\mathcal{L}$ and valuations in U^ω. Further, the relation can be restricted to languages $\mathcal{L}_s$ and s-tuples $\mathbf{m}$ in U^s which are simply projections of valuations $\mathbf{v}$ to subsets s. In this case, we write $\mathbf{m} \models_s f$. Information sets $\hat{r}_s(S)$ then are subsets of U^s, i.e. relations in s. They form the relational algebra over subsets s which exhibits the close and well-known relation of predicate logic to relational algebra and hence to relational databases. We refer to (Langel, 2010) for more details about predicate logic and information.

Besides propositional and predicate logic, many other logic systems possess this context structure. We refer to (Wilson & Mengin, 2001) for further instances from the field of logic.

Two sets of sentences S_1 and S_2 are called *equivalent*, if they have the same models, $\hat{r}(S_1) = \hat{r}(S_2)$. This means that S_1 and S_2 express the same information and it is equivalent to saying that they have the same theories $C_\models(S_1) = C_\models(S_2)$. Any set S of sentences is in particular equivalent to its theory $C_\models(S)$. In the same way, two sets of models M_1 and M_2 are equivalent, if they have the same theory $\check{r}(M_1) = \check{r}(M_2)$. They span the same information (see for example Instance 7.4).

In a first step towards a valuation algebra, we may already discuss a preliminary concept of combination of information. In fact, suppose two sources which both send a piece of information by stating sets S_1 and S_2 of sentences. They express the two information sets $M_1 = \hat{r}(S_1)$ and $M_2 = \hat{r}(S_2)$. On the level of sentences, combining these two pieces of information means clearly putting the two sets of sentences together as $S_1 \cup S_2$. Think for example of two systems of linear equations. On the model level, the combined information is then given by the models of $S_1 \cup S_2$. Hence, we may tentatively define the following combination operation between two information sets:

$$M_1 \otimes M_2 \quad = \quad \hat{r}(S_1 \cup S_2).$$

Using Property 4 of Lemma 7.1 we may also write this operation as:

$$M_1 \otimes M_2 \;=\; \hat{r}(S_1) \cap \hat{r}(S_2) \;=\; M_1 \cap M_2.$$

Combining two information sets simply corresponds to set intersection. It thus follows that the intersection of two closed sets is again a closed set. In fact, since Property 4 of Lemma 7.1 holds for any family of sets S_j, we conclude that the intersection of any family of closed sets is again a closed set. From this, it follows by a standard results of lattice theory that closed sets form a *complete lattice* (see Definition A.5) (Davey & Priestley, 1990). This is a fundamental result of context analysis. We are pursuing however a different direction. In the following section we shall construct an information algebra out of context systems.

7.3.2 Tuple Model Structures

In all examples we have seen so far, we dealt with subsets s of variables, and the models were valuations of these variables in some sets, for example real numbers as in linear systems of equations or Boolean values as in propositional logic. For every subset of variables we have a context c_s linking the sentences or formulae with the corresponding values, for example as solutions to sets of linear equations or interpretations satisfying propositional formulae. These contexts are linked together by projection of models and embeddings of sentences. In this section, we subsume this situation into an abstract frame which induces a generic construction producing these examples as particular instances. In a first step, we abstract the structure of s-vectors and interpretations (Boolean vectors) into a general abstract frame called *tuple system*. In fact s-vectors and s-interpretations for subsets s of a set r of variables are instances of a tuple system defined as follows:

Definition 7.1 *A* tuple system *over the lattice D of subsets $s \subseteq r$ is a set T together with two operations $d : T \to D$ and $\downarrow$: $T \times D \to T$ defined for $x \subseteq d(f)$ which satisfies the following axioms: For $f, g \in T$ and $x, y \in D$,*

1. *If $x \subseteq d(f)$ then $d(f^{\downarrow x}) = x$.*

2. *If $x \subseteq y \subseteq d(f)$ then $(f^{\downarrow y})^{\downarrow x} = f^{\downarrow x}$.*

3. *If $d(f) = x$ then $f^{\downarrow x} = f$.*

4. *For $d(f) = x$, $d(g) = y$ and $f^{\downarrow x \cap y} = g^{\downarrow x \cap y}$ there exists $h \in T$ such that $d(h) = x \cup y$, $h^{\downarrow x} = f$ and $h^{\downarrow y} = g$.*

5. *For $d(f) = x$ and $x \subseteq y$ there exists $g \in T$ such that $d(g) = y$ and $g^{\downarrow x} = f$.*

The operation $\downarrow$ of course corresponds to the *projection* of s-vectors, and the label d indicates to which group of variables a vector or tuple belongs. Vectors and ordinary tuples as occurring in relational databases are evidently instances of abstract tuples according to the definition above.

Example 7.6 *The relational algebra of Instance 1.2 can be generalized to abstract tuple systems. Then, a relation R over $s \in D$ is a set of tuples $f \in T$ which all have the domain s. This also defines the label of R as $d(R) = s$. Combination of two relations R_1 and R_2 with domains s and t respectively is defined by natural join,*

$$R_1 \bowtie R_2 \quad = \quad \{f \in T : d(f) = s \cup t \textit{ and } f^{\downarrow s} \in R_1 \textit{ and } f^{\downarrow t} \in R_2\},$$

and projection of relation R with domain s to $t \subseteq s$ is defined as

$$R^{\downarrow t} \quad = \quad \{f^{\downarrow t} : f \in R\}.$$

Verifying that this generalized relational algebra *also satisfies the valuation algebra axioms is similar to the ordinary relational algebra.*

The algebra of models in Section 7.1 corresponds to such a generalized relational algebra over Boolean tuples. The algebra of affine spaces in Section 7.2 is a subalgebra of the relational algebra over s-vectors, and the same holds for the algebra of convex polyhedra in Instance 7.5. Here follows a more unusual example.

Example 7.7 *Let $\langle\Phi, D\rangle$ be an idempotent valuation algebra, i.e. for $\phi \in \Phi$ and $x \subseteq d(\phi)$ it holds that*

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

We further assume a neutral element e_x for each domain $x \in D$, and that the valuation algebra is stable, i.e. that $e_x^{\downarrow y} = e_y$ for $y \subseteq x$. Then Φ is a tuple system: The properties 1, 2 *and* 3 *of a tuple system simply correspond to the Axioms (A3), (A4) and (A6) of the valuation algebra. To verify Property* 4, *assume $d(\phi) = x$, $d(\psi) = y$ and $\phi^{\downarrow x \cap y} = \psi^{\downarrow x \cap y}$. Defining $\chi = \phi \otimes \psi$, we obtain from the combination axiom (A5) and idempotency that*

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

In a similar way, we derive $\chi^{\downarrow y} = \psi$. For Property 5, *let $d(\phi) = x$ and $\chi = \phi \otimes e_y$. It follows that $d(\chi) = y$ and by the combination axiom and the property assumed for neutral elements that*

$$\chi^{\downarrow x} \;=\; (\phi \otimes e_y)^{\downarrow x} \;=\; \phi \otimes e_y^{\downarrow x} \;=\; \phi \otimes e_x \;=\; \phi.$$

This shows that each idempotent valuation algebra with neutral elements adopts itself the structure of a tuple system.

As in the above examples, we now link tuples systems with formal languages to form a family of contexts. For that purpose, remember that the contexts $c_x = \langle \mathcal{L}_x, \mathcal{M}_x, \models_x\rangle$ for all $x \subseteq r$ in the examples are linked together by embedding and projection mappings. Thus, let $\mathcal{M}$ be a tuple system over the subsets of a set r. We define the set of all tuples with domain x by

$$\mathcal{M}_x \quad = \quad \{m \in \mathcal{M} : d(m) = x\}.$$

Further, we assume the existence of language $\mathcal{L}_x$ associated with $\mathcal{M}_x$ to express information in $\mathcal{M}_x$. More precisely, we suppose a context $c_x = \langle \mathcal{L}_x, \mathcal{M}_x, \models_x \rangle$ for each subset $x \subseteq r$, and for $y \subseteq x$ an embedding $f_{y,x} : \mathcal{L}_y \to \mathcal{L}_x$ such that it forms together with the projection $g_{x,y} : \mathcal{M}_x \to \mathcal{M}_y$ defined by $g_{x,y}(m) = m^{\downarrow y}$ an *infomorphism*. We thus have

$$m \models_x f_{y,x}(s) \quad \Longleftrightarrow \quad g_{x,y}(m) \models_y s.$$

For a set of sentences S of the language $\mathcal{L}_x$ we have the information set $M = \hat{r}_x(S)$ in $\mathcal{M}_x$ which is a $\models_x$-closed set $M = C^{\models_x}(M) = \hat{r}_x(\check{r}_x(M))$. Let Φ_x be the set of all $\models_x$-closed sets in $\mathcal{M}_x$ and

$$\Phi = \bigcup_{x \subseteq r} \Phi_x.$$

We examine whether Φ adopts the structure of an information algebra relative to the lattice of subsets of r. In fact, since the elements of Φ are generalized relations, it is sufficient to show that it is a subalgebra of the generalized relational algebra associated with the tuple system $\mathcal{M}$ (see Example 7.6), and this requires that Φ is closed under join and projection. Unfortunately, it follows from the formal duality between languages and models that Φ is not necessarily closed under projection. This can, for example, be seen by the example of propositional logic: the embedding $f_{y,x}(S)$ of a theory in the context c_y is not yet closed in the context c_x. Dually, we may generally not expect that the projection $g_{x,y}(M)$ of a closed set in the context c_x is still closed in c_y. However, this is the case in many examples and in particular in all examples we have seen so far. Therefore we tacitly *assume* or require that $g_{x,y}(M)$ is $\models_y$-closed when M is $\models_x$-closed.

The situation is different for combination: Let $M_1 \in \Phi_x$, $M_2 \in \Phi_y$ and recall that

$$M_1 \bowtie M_2 = M_1^{\uparrow x \cup y} \cap M_2^{\uparrow x \cup y}, \tag{7.15}$$

where $M^{\uparrow y}$ for $M \subseteq \mathcal{M}_x$ and $x \subseteq y$ is defined as:

$$M^{\uparrow y} = \{m \in \mathcal{M}_y : m^{\downarrow x} \in M\}.$$

We claim that $m^{\uparrow y}$ is $\models_y$-closed if M is $\models_x$-closed. In fact, assume that $M = \hat{r}_x(S)$ for some set of sentences $S \subseteq \mathcal{L}_x$. We then have the following equivalences due to the infomorphism property:

$$\begin{aligned} m \in \mathcal{M}^{\uparrow y} \quad & \Leftrightarrow \quad m^{\downarrow x} \in M = \hat{r}_x(S) \\ & \Leftrightarrow \quad g_{y,x}(m) \models_x s, \forall s \in S \\ & \Leftrightarrow \quad m \models_y f_{x,y}(s), \forall s \in S \\ & \Leftrightarrow \quad m \in \hat{r}_y(f_{x,y}(S)). \end{aligned}$$

This shows that $\mathcal{M}^{\uparrow y} = \hat{r}_y(f_{x,y}(S))$ which is a $\models_y$-closed set. Therefore, both $M_1^{\uparrow x \cup y}$ and $M_2^{\uparrow x \cup y}$ in equation (7.15) are $\models_{x \cup y}$-closed sets, and since the intersection of closed sets is still closed, we have $M_1 \bowtie M_2 \in \Phi$.

To sum it up, Φ is a subset of the generalized relational algebra from which we already know that it satisfies the information algebra axioms. It is furthermore closed under combination and projection which turns the structure $\langle \Phi, D \rangle$ into a subalgebra of the generalized relational algebra. Therefore, $\langle \Phi, D \rangle$ also forms an information algebra. This holds for all families of contexts where the corresponding operations of embedding and projection are linked by an infomorphism, given the additional condition that the projection of closed model sets always leads to a closed model set. The algebra contains neutral elements $\mathcal{M}_x$ for all contexts x, corresponding to the tautologies $C_{\models_x}(\emptyset)$ in context x. This generic construction called *context valuation algebra* produces among other formalisms the information algebras of propositional and predicate logic, and of linear equation and inequality systems. This is shown by the following examples:

Example 7.8 *In the case of propositional logic (see Section 7.1 and Instance 7.3) the tuple system $\mathcal{M}$ consists of all Boolean s-vectors $\mathbf{m} : s \rightarrow \{0, 1\}$, and Φ is equal to the relational algebra over this tuple system, if r is finite. This is because any subset of $\mathcal{M}_x$ is $\models_x$-closed. The same situation is found in the predicate logic of Instance 7.6 where the models also form a relational algebra.*

Example 7.9 *In the context of linear equations (see Section 7.2 and Instance 7.4) the tuple system $\mathcal{M}$ consists of all s-vectors $\mathbf{m} : s \rightarrow \mathbb{R}$. Here, Φ is the subalgebra of affine spaces of the relational algebra of s-vectors. As we have seen, affine spaces project to affine spaces again.*

Example 7.10 *Similar as in systems of linear equations, the tuple system $\mathcal{M}$ of linear inequalities (see Instance 7.5) consists of all s-vectors $\mathbf{m} : s \rightarrow \mathbb{R}$. But here, Φ is the subalgebra of convex polyhedra in the linear space $\mathbb{R}^s$. Again, convex polyhedra always project to convex polyhedra.*

7.4 CONCLUSION

We started into this chapter by considering the two formalisms of propositional logic and linear equations. Both examples provide a formal language to describe their information sets. In the first case, propositional formulae describe sets of truth value assignments to propositional symbols under which the corresponding formula evaluates to true, and in the second case, linear equations describe their associated sets of solutions. This interaction of sentences and model sets uncovers two information algebras for each formalism. On the model level, both formalism correspond to (a subalgebra of) the relational algebra of Instance 1.2 and therefore adopt the structure of a information algebra. These properties mirror the context structure. Closed sets of sentences therefore also form an information algebra. Although our considerations for linear equations were limited to real numbers, we noted its applicability for arbitrary fields. This identified a first generic construction which actually produces two new information algebras for each field. However, the final section of this chapter showed that propositional logic and linear equations over arbitrary fields are instances of a far more general generic construction called context valuation algebra. Based on so-called tuple systems, this further induces the information algebra behind linear inequalities, predicate logic, and many other formalisms where this duality between language and models exists.

PROBLEM SETS AND EXERCISES

G.1 ★ Express *consequence finding* in propositional logic as an inference problem. This solution to this exercise can be found in (Inoue, 1992).

G.2 ★ Exercise D.4 in Chapter 4 introduced a partial order between the elements of an idempotent valuation algebra. Apply this order to propositional logic, linear equation systems and to contexts in general and explore its semantics.

G.3 ★★ In this chapter, we discussed propositional and predicate logic as instances of context valuation algebras. Other non-classical logics are studied in (Wilson & Mengin, 2001). Identify the context structure in these logics.

G.4 ★★★ Context valuations are infinite constructs but with a finite representation. For propositional logic such representations could be so-called normal forms. Exemplarily, we studied the conjunctive normal form and the normal form of prime implicates in Section 7.1.3. But there are many other normal forms listed in (Darwiche & Marquis, 2002; Wachter & Haenni, 2006) that could be used as well. Analyse further normal forms and provide suitable definitions for combination and projection.

PART III

APPLICATIONS

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas
Copyright © 2011 John Wiley & Sons, Inc.

CHAPTER 8

DYNAMIC PROGRAMMING

Many valuation algebras have an associated notion of a *solution*, *best* or *preferred value* with respect to some criteria. Most typical are solutions to linear equations or inequalities. In logic, solutions may correspond to the truth value assignments under which a sentence evaluates to true. Likewise, the solutions of a constraint system are the variable assignments that satisfy the constraints and in optimization problems, solutions lead to either maximum or minimum values. Given a set r or variables, we again write Ω_X for the frame of each variable $X \in r$ and Ω_s for the set of all possible tuples or configurations of a subset $s \subseteq r$, see Section 1.2. If $\langle \Phi, D \rangle$ denotes a valuation algebra defined over the subset lattice $D = \mathcal{P}(r)$, a solution for some valuation $\phi \in \Phi$ with domain $d(\phi) = s$ always corresponds to some element $\mathbf{x} \in \Omega_s$. We know from Instance 1.2 that such tuple systems form a relational algebra. It is therefore reasonable to consider the solutions of ϕ as a specific relation $c_\phi \subseteq \Omega_s$. In this chapter, we focus on the computation of the solutions for valuations ϕ that are given as factorizations $\phi = \phi_1 \otimes \ldots \otimes \phi_n$. This must again be done without the explicit computation of ϕ. Moreover, it is a general observation that determining a solution to an equation system becomes easier the less variables are involved. In standard Gaussian elimination for regular systems, we therefore eliminate one variable after the other until a single equation with only one variable remains. The solution to this simple equation

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

can then be determined and the complete solution to the total system is obtained from a backward substitution process. The more general procedure presented in this chapter follows exactly this course of action. We execute an arbitrary local computation architecture to obtain the marginal $\phi^{\downarrow t}$ for some $t \subseteq d(\phi)$. This corresponds to the elimination of the variables in $s - t$. We then observe that every solution to this marginal is also a partial solution to ϕ with respect to the variables in t. In a second step, we extend this partial solution to a complete solution for the valuation ϕ using only the intermediate results obtained from the local computation process. This ensures that solution construction adopts the complexity of a local computation scheme.

The first section of this chapter gives a more rigorous definition of solutions in the context of valuation algebras and derives some basic properties. We then present different algorithms for the construction of single solutions or complete solution sets in Section 8.2. These methods are again generic and can be applied to all valuation algebras with a suitable notion of solution. A particular important field of application is constraint reasoning or the solution of optimization problems. It will be shown in Section 8.3.2 that these problems emerge from a particular subclass of the family of semiring valuation algebras from Chapter 5. In fact, executing the fusion or bucket elimination algorithm for the computation of a marginal of an optimization problem and applying the generic solution construction procedure presented in this chapter coincides with a well-known programming paradigm called *dynamic programming* (Bertele & Brioschi, 1972). This close relationship between valuation algebras and dynamic programming was established by (Shenoy, 1996), who further proved that the axioms for discrete dynamic programming given in (Mitten, 1964) entail those of the valuation algebra. Hence, the axiomatic system of a valuation algebra that enables local computation also constitutes the mathematical foundation to dynamic programming. Essentially the same idea for the solution of constraint systems based on bucket elimination was also found by (Dechter, 1999). This chapter can therefore be seen as a generalization of both approaches from constraint systems to arbitrary valuation algebras with solutions. Further applications of solution construction to symmetric, positive definite equation systems and quasi-regular semiring fixpoint equation systems will be discussed in Chapter 9.

8.1 SOLUTIONS AND SOLUTION EXTENSIONS

Let r be a set of variables and $\langle \Phi, D \rangle$ a valuation algebra defined over the subset lattice $D = \mathcal{P}(r)$. Intuitively, solutions in valuation algebras are defined by the fundamental property that a partial solution $\mathbf{x} \in \Omega_t$ to ϕ with respect to some variables in $t \subseteq s = d(\phi)$ can always be extended to a solution $(\mathbf{x}, \mathbf{y}) \in \Omega_s$ to ϕ. Moreover, this property of extensibility must hold for all configurations, i.e. if $\mathbf{x} \in \Omega_t$, it must be possible to find an extension $\mathbf{y} \in \Omega_{s-t}$ such that $(\mathbf{x}, \mathbf{y}) \in \Omega_s$ leads to the "preferred" value of ϕ among all configurations $\mathbf{z} \in \Omega_s$ with $\mathbf{z}^{\downarrow t} = \mathbf{x}$. It is required that this configuration extension $\mathbf{y}$ can either be computed directly by extending $\mathbf{x}$ to the domain s, or step-wise by first extending $\mathbf{x}$ to u and then to s for $t \subseteq u \subseteq s$. This is the basic idea of the following definition:

Definition 8.1 *Let $\phi \in \Phi$ with $t \subseteq s = d(\phi)$ and $\mathbf{x} \in \Omega_t$. A set*

$$W_\phi^t(\mathbf{x}) \quad \subseteq \quad \Omega_{s-t}$$

is called configuration extension set *of ϕ from t to s, given* $\mathbf{x}$, *if the following property holds for all $u \in D$ such that $t \subseteq u \subseteq s$:*

$$W_\phi^t(\mathbf{x}) \quad = \quad \left\{\mathbf{z} \in \Omega_{s-t} : \mathbf{z}^{\downarrow u-t} \in W_{\phi^{\downarrow u}}^t(\mathbf{x}) \textit{ and } \mathbf{z}^{\downarrow s-u} \in W_\phi^u(\mathbf{x}, \mathbf{z}^{\downarrow u-t})\right\}.$$

Intuitively, the set $W_\phi^t(\mathbf{x})$ is assumed to contain all extensions of the configuration $\mathbf{x} \in \Omega_t$ to the domain of ϕ and is characterized by the fundamental property that configuration extensions can be computed step-wise. Instead of searching an extension of a configuration $\mathbf{x} \in \Omega_t$ to some preferred value in ϕ, we can first extend it to $\phi^{\downarrow u}$ and then to ϕ. Observe that configuration extension sets may also be empty. Further, we will see in Section 8.4 that we may define solution extension sets in different ways for the same valuation algebra. They are therefore not unique. In the introduction we presented the conception of solutions as the set of configurations that lead to the preferred value in ϕ. This can now be expressed as the configuration extension set of the empty configuration to ϕ.

Definition 8.2 *A* solution set c_ϕ *of a valuation $\phi \in \Phi$ is defined as*

$$c_\phi \quad = \quad W_\phi^\emptyset(\diamond).$$

An important relationship between solution sets and configuration extension sets follows directly by applying Definition 8.1 to $t = \emptyset$ and $\mathbf{x} = \diamond$:

$$\begin{aligned} c_\phi \quad = \quad W_\phi^\emptyset(\diamond) \quad &= \quad \left\{\mathbf{z} \in \Omega_s : \mathbf{z}^{\downarrow u} \in W_{\phi^{\downarrow u}}^\emptyset(\diamond) \text{ and } \mathbf{z}^{\downarrow s-u} \in W_\phi^u(\mathbf{z}^{\downarrow u})\right\} \\ &= \quad \left\{\mathbf{z} \in \Omega_s : \mathbf{z}^{\downarrow u} \in c_{\phi^{\downarrow u}} \text{ and } \mathbf{z}^{\downarrow s-u} \in W_\phi^u(\mathbf{z}^{\downarrow u})\right\}. \qquad (8.1) \end{aligned}$$

This shows how solution sets are computed step-wise. The following lemma states an important property of solution sets, saying that every solution to a projection of ϕ is also a projection of some solution to ϕ and vice versa.

Lemma 8.1 *For $\phi \in \Phi$ and $t \subseteq d(\phi)$ it holds that*

$$c_{\phi^{\downarrow t}} \quad = \quad c_\phi^{\downarrow t}.$$

Proof: Let $d(\phi) = s$. It then follows from equation (8.1) that

$$\begin{aligned} c_\phi^{\downarrow t} \quad &= \quad \left[W_\phi^\emptyset(\diamond)\right]^{\downarrow t} \\ &= \quad \left\{\mathbf{z} \in \Omega_s : \mathbf{z}^{\downarrow t} \in c_{\phi^{\downarrow t}} \text{ and } \mathbf{z}^{\downarrow s-t} \in W_\phi^t(\mathbf{z}^{\downarrow t})\right\}^{\downarrow t} \quad = \quad c_{\phi^{\downarrow t}}. \end{aligned}$$

■

Subsequently, we refer to a projection $\mathbf{x}^{\downarrow t}$ of a solution $\mathbf{x} \in c_\phi$ as a *partial solution* of ϕ with respect to $t \subseteq d(\phi)$. If s and t are two arbitrary subsets of the domain of a valuation $\phi \in \Phi$, the partial solutions of ϕ with respect to $s \cup t$ may be obtained by extending the partial solutions of ϕ with respect to s from $s \cap t$ to t. This is the statement of the following theorem:

Theorem 8.1 *For $s, t \subseteq d(\phi)$ we have*

$$c_\phi^{\downarrow s \cup t} \quad = \quad \left\{\mathbf{z} \in \Omega_{s \cup t} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \textit{ and } \mathbf{z}^{\downarrow t-s} \in W_{\phi^{\downarrow t}}^{s \cap t}(\mathbf{z}^{\downarrow s \cap t})\right\}.$$

Proof: We first show that if $\mathbf{z} \in c_\phi^{\downarrow s}$ then

$$W_{\phi^{\downarrow s \cup t}}^{s}(\mathbf{z}) \quad \subseteq \quad W_{\phi^{\downarrow t}}^{s \cap t}(\mathbf{z}^{\downarrow s \cap t}). \tag{8.2}$$

Applying equation (8.1) and Lemma 8.1 gives

$$c_\phi^{\downarrow s \cup t} \quad = \quad \left\{\mathbf{z} \in \Omega_{s \cup t} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and } \mathbf{z}^{\downarrow t-s} \in W_{\phi^{\downarrow s \cup t}}^{s}(\mathbf{z}^{\downarrow s})\right\} \tag{8.3}$$

and

$$c_\phi^{\downarrow t} \quad = \quad \left\{\mathbf{z} \in \Omega_t : \mathbf{z}^{\downarrow s \cap t} \in c_\phi^{\downarrow s \cap t} \text{ and } \mathbf{z}^{\downarrow t-s} \in W_{\phi^{\downarrow t}}^{s \cap t}(\mathbf{z}^{\downarrow s \cap t})\right\}.$$

If $\mathbf{z} \in c_\phi^{\downarrow s}$ and $\mathbf{y} \in W_{\phi^{\downarrow s \cup t}}^{s}(\mathbf{z})$ we have $(\mathbf{y}, \mathbf{z}) \in c_\phi^{\downarrow s \cup t}$ and therefore $(\mathbf{y}, \mathbf{z}^{\downarrow s \cap t}) \in c_\phi^{\downarrow t}$. Since $\mathbf{z} \in c_\phi^{\downarrow s}$ implies that $\mathbf{z}^{\downarrow s \cap t} \in c_\phi^{\downarrow s \cap t}$ we conclude that $\mathbf{y} \in W_{\phi^{\downarrow t}}^{s \cap t}(\mathbf{z}^{\downarrow s \cap t})$. This proves (8.2). Next, we obtain by applying two times equation (8.1) and Lemma 8.1:

$$\begin{aligned} c_\phi^{\downarrow s \cup t} \quad = \quad & \Big\{\mathbf{z} \in \Omega_{s \cup t} : \mathbf{z}^{\downarrow s \cap t} \in c_\phi^{\downarrow s \cap t} \text{ and } \mathbf{z}^{\downarrow t-s} \in W_{\phi^{\downarrow t}}^{s \cap t}(\mathbf{z}^{\downarrow s \cap t}) \text{ and} \\ & \mathbf{z}^{\downarrow s-t} \in W_{\phi^{\downarrow s \cup t}}^{t}(\mathbf{z}^{\downarrow t})\Big\}. \end{aligned} \tag{8.4}$$

Similarly, we also have

$$\begin{aligned} c_\phi^{\downarrow s \cup t} \quad = \quad & \Big\{\mathbf{z} \in \Omega_{s \cup t} : \mathbf{z}^{\downarrow s \cap t} \in c_\phi^{\downarrow s \cap t} \text{ and } \mathbf{z}^{\downarrow s-t} \in W_{\phi^{\downarrow s}}^{s \cap t}(\mathbf{z}^{\downarrow s \cap t}) \text{ and} \\ & \mathbf{z}^{\downarrow t-s} \in W_{\phi^{\downarrow s \cup t}}^{s}(\mathbf{z}^{\downarrow s})\Big\}. \end{aligned}$$

It follows from these two expressions that

$$\begin{aligned} c_\phi^{\downarrow s \cup t} \quad = \quad & \Big\{\mathbf{z} \in \Omega_{s \cup t} : \mathbf{z}^{\downarrow s \cap t} \in c_\phi^{\downarrow s \cap t} \text{ and } \mathbf{z}^{\downarrow s-t} \in W_{\phi^{\downarrow s}}^{s \cap t}(\mathbf{z}^{\downarrow s \cap t}) \text{ and} \\ & \mathbf{z}^{\downarrow t-s} \in W_{\phi^{\downarrow s \cup t}}^{s}(\mathbf{z}^{\downarrow s}) \text{ and } \mathbf{z}^{\downarrow s-t} \in W_{\phi^{\downarrow s \cup t}}^{t}(\mathbf{z}^{\downarrow t})\Big\}. \end{aligned}$$

Since $\mathbf{z} \in c_\phi^{\downarrow s \cup t}$ implies that $\mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s}$ we can add this as a further condition to the last expression. But $\mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s}$ in turn implies that $\mathbf{z}^{\downarrow s \cap t} \in c_\phi^{\downarrow s \cap t}$. Hence, the added condition $\mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s}$ subsumes $\mathbf{z}^{\downarrow s\cap t} \in c_\phi^{\downarrow s\cap t}$ and we obtain

$$c_\phi^{\downarrow s\cup t} = \Big\{\mathbf{z} \in \Omega_{s\cup t} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and } \mathbf{z}^{\downarrow s-t} \in W_{\phi^{\downarrow s}}^{s\cap t}(\mathbf{z}^{\downarrow s\cap t}) \text{ and } \mathbf{z}^{\downarrow t-s} \in W^{s}_{\phi^{\downarrow s\cup t}}(\mathbf{z}^{\downarrow s}) \text{ and } \mathbf{z}^{\downarrow s-t} \in W^{t}_{\phi^{\downarrow s\cup t}}(\mathbf{z}^{\downarrow t})\Big\}.$$

It then follows from property (8.2) that

$$c_\phi^{\downarrow s\cup t} = \Big\{\mathbf{z} \in \Omega_{s\cup t} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and } \mathbf{z}^{\downarrow t-s} \in W^{s}_{\phi^{\downarrow s\cup t}}(\mathbf{z}^{\downarrow s}) \text{ and } \mathbf{z}^{\downarrow s-t} \in W^{t}_{\phi^{\downarrow s\cup t}}(\mathbf{z}^{\downarrow t})\Big\}. \tag{8.5}$$

Finally, we conclude by comparing (8.4) with (8.5) that if $\mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s}$ that

$$W^{s}_{\phi^{\downarrow s\cup t}}(\mathbf{z}^{\downarrow s}) = W^{s\cap t}_{\phi^{\downarrow s}}(\mathbf{z}^{\downarrow s\cap t})$$

must hold. Replacing the expression in (8.3) finally gives:

$$c_\phi^{\downarrow s\cup t} = \Big\{\mathbf{z} \in \Omega_{s\cup t} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and } \mathbf{z}^{\downarrow t-s} \in W^{s\cap t}_{\phi^{\downarrow t}}(\mathbf{z}^{\downarrow s\cap t})\Big\}.$$

■

8.2 COMPUTING SOLUTIONS

We now focus on the efficient computation of solutions for some valuation $\phi \in \Phi$ that is given as a factorization $\phi = \phi_1 \otimes \ldots \otimes \phi_n$. It was pointed out in many places that every reasonable approach for this task must dispense with the explicit computation of ϕ. Since local computation is a suitable way to avoid building the objective function, it is sensible to embark on a similar strategy. This section presents a class of methods that assemble solutions for ϕ using partial solution extension sets with respect to the previously computed marginals in a join tree. This mirrors the fundamental assumption that computing solutions becomes easier, if less variables are involved. It is important to note that we do not say in this section how solutions or configuration extension sets are computed for concrete valuation algebras, but only how they are assembled from smaller sets. Concrete case studies for different valuation algebras will later be given in Section 8.3.2 and Chapter 9.

8.2.1 Computing all Solutions with Distribute

The most obvious approach to identify all solution configurations of ϕ follows from the application of local computation for the solution of multi-query inference problems. Depending on the architecture, each node $i \in V$ in the covering join tree (V, E, λ, D) can compute or already contains the marginal of ϕ relative to its node label $\lambda(i)$ after the complete message-passing. We remind that join tree nodes are numbered in such a way that if $j \in V$ is a node on the path from node i to the root node $r = |V|$, then $j > i$. From the computed marginals, we can build the set of all solutions by the following procedure: due to Lemma 8.1 and Definition 8.2, we obtain for the root node:

$$c_\phi^{\downarrow\lambda(r)} \quad = \quad W^{\emptyset}_{\phi^{\downarrow\lambda(r)}}(\diamond). \tag{8.6}$$

Since $\lambda(r)$ is generally much smaller than $d(\phi)$, we here assume that $c_\phi^{\downarrow\lambda(r)}$ can be computed efficiently. Then, the following lemma shows how this partial solution set can be extended to the complete solution set c_ϕ.

Lemma 8.2 *For $i = r-1, \ldots, 1$ and $s = \lambda(r) \cup \ldots \cup \lambda(i+1)$ we have*

$$c_\phi^{\downarrow s\cup\lambda(i)} = \Big\{\mathbf{z} \in \Omega_{s\cup\lambda(i)} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \textit{ and } \mathbf{z}^{\downarrow\lambda(i)-s} \in W^{\lambda(i)\cap\lambda(ch(i))}_{\phi^{\downarrow\lambda(i)}}\big(\mathbf{z}^{\downarrow\lambda(i)\cap\lambda(ch(i))}\big)\Big\}.$$

Proof: It follows from Theorem 8.1 that

$$c_\phi^{\downarrow s\cup\lambda(i)} = \Big\{\mathbf{z} \in \Omega_{s\cup\lambda(i)} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and } \mathbf{z}^{\downarrow\lambda(i)-s} \in W^{s\cap\lambda(i)}_{\phi^{\downarrow t}}\big(\mathbf{z}^{\downarrow s\cap\lambda(i)}\big)\Big\}.$$

From the running intersection property we conclude

$$s \cap \lambda(i) = \Big(\lambda(r) \cup \ldots \cup \lambda(i+1)\Big) \cap \lambda(i) = \lambda(i) \cap \lambda(ch(i)).$$

Then, the statement of the lemma follows directly. ■

To sum it up, we compute c_ϕ by the following procedure.

1. Execute a multi-query local computation procedure on $\{\phi_1, \ldots, \phi_n\}$.
2. Identify $c_\phi^{\downarrow\lambda(r)}$ in the root node.
3. For $i = r-1, \ldots, 1$
   (a) compute $W^{\lambda(i)\cap\lambda(ch(i))}_{\phi^{\downarrow\lambda(i)}}$ in node $i \in V$;
   (b) build $c_\phi^{\downarrow\lambda(r)\cup\ldots\cup\lambda(i)}$ by application of Lemma 8.2.
4. Return $c_\phi = c_\phi^{\downarrow\lambda(r)\cup\ldots\cup\lambda(1)}$.

The pseudo-code of this procedure is given in Algorithm 8.1. It is important to note that the domains of the configuration extension sets computed in step 3 are always bounded by the domain $\lambda(i)$ of the corresponding join tree node. Hence, although the construction of c_ϕ naturally depends on the actual number of solutions, the computations of the configuration extension sets adopt the complexity of a local computation scheme. Moreover, the successive buildup of c_ϕ can be seen as a top-down message-passing in the join tree. The messages

$$\mu_{ch(i)\rightarrow i} \quad = \quad c_\phi^{\downarrow\lambda(i)\cap\lambda(ch(i))} \tag{8.7}$$

are tuple sets and thus elements of the relational algebra. As an alternative to building up c_ϕ directly with Theorem 8.1, we may content ourselves with computing the marginal of c_ϕ relative to the node label $\lambda(i)$ in each node $i \in V$. This could be seen as a pre-compilation of the solution set c_ϕ and requires to compute

$$\begin{aligned} c_\phi^{\downarrow\lambda(i)} &= \Big\{\mathbf{z} \in \Omega_{\lambda(i)} : \mathbf{z}^{\downarrow\lambda(i)\cap\lambda(ch(i))} \in c_\phi^{\downarrow\lambda(i)\cap\lambda(ch(i))} \text{ and} \\ &\qquad \mathbf{z}^{\downarrow\lambda(i)-\lambda(ch(i))} \in W_{\phi^{\downarrow\lambda(i)}}^{\lambda(i)\cap\lambda(ch(i))}\big(\mathbf{z}^{\downarrow\lambda(i)\cap\lambda(ch(i))}\big)\Big\} \\ &= \Big\{\mathbf{z} \in \Omega_{\lambda(i)} : \mathbf{z}^{\downarrow\lambda(i)\cap\lambda(ch(i))} \in \mu_{i\rightarrow ch(i)} \text{ and} \\ &\qquad \mathbf{z}^{\downarrow\lambda(i)-\lambda(ch(i))} \in W_{\phi^{\downarrow\lambda(i)}}^{\lambda(i)\cap\lambda(ch(i))}\big(\mathbf{z}^{\downarrow\lambda(i)\cap\lambda(ch(i))}\big)\Big\}. \end{aligned}$$

in each node. Observe that this formula is closely related to the operation of natural join, which points out that solution construction corresponds to local computation in the relational algebra. The detailed elaboration of this perspective is proposed as Exercise H.1 to the reader. However, an important disadvantage of this algorithm is that it requires a complete run of a multi-query local computation architecture to obtain the marginals of the objective function relative to all node labels used in Lemma 8.2. We shall therefore study in the following subsection, under what restrictions one can omit the execution of the outward phase and compute the configuration extensions only from the result of a single-query local computation architecture.

Algorithm 8.1 Computing all Solutions I

```
input:  $\phi^{\downarrow\lambda(i)}$ for $1 \le i \le r$  output:  $c_\phi$

begin
  $c$ := $W^{\emptyset}_{\phi^{\downarrow\lambda(r)}}(\diamond)$;
  for $i = r - 1 \ldots 1$ do
    $s$ := $\lambda(r) \cup \ldots \cup \lambda(i+1)$;
    $c$ := $\{\mathbf{z} : \mathbf{z}^{\downarrow s} \in c \wedge \mathbf{z}^{\downarrow\lambda(i)-s} \in W_{\phi^{\downarrow\lambda(i)}}^{\lambda(i)\cap\lambda(ch(i))}(\mathbf{z}^{\downarrow\lambda(i)\cap\lambda(ch(i))})\}$;
  end;
  return $c$;
end
```

8.2.2 Computing some Solutions without Distribute

We now start from a complete run of the generalized collect algorithm from Section 3.10 or any other single-query local computation procedure. At the end of the message-passing, the root node r contains the marginal $\phi^{\downarrow\lambda(r)}$. This ensures that the initialization step of Algorithm 8.1 can still be performed, even if we omit the execution of the outward phase. More challenging is the loop statement that constructs the solution set c_ϕ using Lemma 8.2. Here, assume the existence of configuration extension sets with respect to $\phi^{\downarrow\lambda(i)}$ for $i = 1, \ldots, r-1$, but these marginals are not available at the end of a single-query local computation procedure. We therefore need some way to build solution sets from configuration extension sets with respect to the node contents

$$\psi_i^{(r)} \;=\; \psi_i \otimes \bigotimes_{j \in pa(i)} \mu_{j \rightarrow i}$$

at the end of the collect algorithm. For simplicity, we subsequently write

$$\omega_i \;=\; d(\psi_i^{(r)}) \;=\; d(\psi_i) \cup \bigcup_{j \in pa(i)} d(\mu_{j \rightarrow i}) \;\subseteq\; \lambda(i).$$

As a first important result, we show that the node domain at the end of the collect phase already contain all required information:

Lemma 8.3 *For* $\mathbf{x} \in c_\phi^{\downarrow \lambda(i) \cap \lambda(ch(i))}$ *it holds that*

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

Proof: It follows from Lemma 4.2 that $\lambda(i) - \lambda(ch(i)) = \omega_i - \lambda(ch(i))$. Since

$$c_\phi^{\downarrow \omega_i} \;=\; \left[c_\phi^{\downarrow \lambda(i)}\right]^{\downarrow \omega_i}$$

we have

$$\begin{aligned}
c_\phi^{\downarrow \omega_i} \;=\; & \Big\{\mathbf{z} \in \Omega_{\omega_i} : \mathbf{z}^{\downarrow \omega_i \cap \lambda(ch(i))} \in c_\phi^{\downarrow \omega_i \cap \lambda(ch(i))} \text{ and} \\
& \qquad \mathbf{z}^{\downarrow \omega_i - \lambda(ch(i))} \in W_{\phi^{\downarrow \omega_i}}^{\omega_i \cap \lambda(ch(i))}(\mathbf{z}^{\downarrow \omega_i \cap \lambda(ch(i))})\Big\} \\
=\; & \Big\{\mathbf{z} \in \Omega_{\lambda(i)} : \mathbf{z}^{\downarrow \lambda(i) \cap \lambda(ch(i))} \in c_\phi^{\downarrow \lambda(i) \cap \lambda(ch(i))} \text{ and} \\
& \qquad \mathbf{z}^{\downarrow \lambda(i) - \lambda(ch(i))} \in W_{\phi^{\downarrow \lambda(i)}}^{\lambda(i) \cap \lambda(ch(i))}(\mathbf{z}^{\downarrow \lambda(i) \cap \lambda(ch(i))})\Big\}^{\downarrow \omega_i} \\
=\; & \Big\{\mathbf{z} \in \Omega_{\lambda(i)} : \mathbf{z}^{\downarrow \lambda(i) \cap \lambda(ch(i))} \in c_\phi^{\downarrow \lambda(i) \cap \lambda(ch(i))} \text{ and} \\
& \qquad \mathbf{z}^{\downarrow \omega_i - \lambda(ch(i))} \in W_{\phi^{\downarrow \lambda(i)}}^{\lambda(i) \cap \lambda(ch(i))}(\mathbf{z}^{\downarrow \lambda(i) \cap \lambda(ch(i))})\Big\}^{\downarrow \omega_i} \\
=\; & \Big\{\mathbf{z}^{\downarrow \omega_i} : \mathbf{z} \in \Omega_{\lambda(i)} \text{ and } \mathbf{z}^{\downarrow \lambda(i) \cap \lambda(ch(i))} \in c_\phi^{\downarrow \lambda(i) \cap \lambda(ch(i))} \text{ and} \\
& \qquad \mathbf{z}^{\downarrow \omega_i - \lambda(ch(i))} \in W_{\phi^{\downarrow \lambda(i)}}^{\lambda(i) \cap \lambda(ch(i))}(\mathbf{z}^{\downarrow \lambda(i) \cap \lambda(ch(i))})\Big\}. \qquad (8.8)
\end{aligned}$$

Now, assume $\mathbf{x} \in c_\phi^{\downarrow \lambda(i) \cap \lambda(ch(i))}$ and $\mathbf{y} \in W_{\phi^{\downarrow \lambda(i)}}^{\lambda(i) \cap \lambda(ch(i))}(\mathbf{x})$. Then $(\mathbf{x}, \mathbf{y}) \in c_\phi^{\downarrow \lambda(i)}$ which implies $(\mathbf{x}^{\downarrow \omega_i \cap \lambda(ch(i))}, \mathbf{y}) \in c_\phi^{\downarrow \omega_i}$ and hence $\mathbf{y} \in W_{\phi^{\downarrow \omega_i}}^{\omega_i \cap \lambda(ch(i))}(\mathbf{x}^{\downarrow \omega_i \cap \lambda(ch(i))})$. On the other hand, assume that $\mathbf{z} = (\mathbf{x}, \mathbf{y}) \in \Omega_{\lambda(i)}$ with $\mathbf{x} \in c_\phi^{\downarrow \lambda(i) \cap \lambda(ch(i))}$ and $\mathbf{y} \in W_{\phi^{\downarrow \omega_i}}^{\omega_i \cap \lambda(ch(i))}(\mathbf{x}^{\downarrow \omega_i \cap \lambda(ch(i))})$. Since $\mathbf{x}^{\downarrow \omega_i \cap \lambda(ch(i))} \in c_\phi^{\downarrow \omega_i \cap \lambda(ch(i))}$ we have $\mathbf{z}^{\downarrow \omega_i} = (\mathbf{x}^{\downarrow \omega_i \cap \lambda(ch(i))}, \mathbf{y}) \in c_\phi^{\downarrow \omega_i}$. It then follows from equation (8.8) that $\mathbf{y} \in W_{\phi^{\downarrow \lambda(i)}}^{\lambda(i) \cap \lambda(ch(i))}(\mathbf{x})$ must hold. Hence, the equality claimed holds. ■

Applying this result, we can replace the node labels $\lambda(i)$ in Lemma 8.2 by the corresponding node domains ω_i just after collect.

We mentioned in the foregoing section that the process of solution construction can be interpreted as local computation in the relational algebra of solution sets. It is clear that for $\phi \in \Phi$ with $d(\phi) = s$ we have $d(c_\phi) = s$. Note that the last expression refers to the labeling operator in the relational algebra. In addition, Lemma 8.1 shows how the operation of projection in the valuation algebra $\langle \Phi, D \rangle$ is related to projection in the relational algebra of solution sets. The following lemma presupposes a similar property for combination and shows that if this property holds, the marginals of ϕ in Lemma 8.2 can be replaced by the node contents at the end of a single-query local computation procedure.

Lemma 8.4 *If the configuration extension sets in a valuation algebra $\langle \Phi, D \rangle$ satisfy the property that for all $\psi_1, \psi_2 \in \Phi$ with $d(\psi_1) = s$, $d(\psi_2) = t$, $s \subseteq u \subseteq s \cup t$ and $\mathbf{x} \in \Omega_u$ we have*

$$W_{\psi_2}^{u\cap t}(\mathbf{x}^{\downarrow u \cap t}) \quad \subseteq \quad W^{u}_{\psi_1 \otimes \psi_2}(\mathbf{x}), \tag{8.9}$$

then it holds for $i = r-1, \ldots, 1$ and $s = \lambda(r) \cup \ldots \cup \lambda(i+1)$ that

$$c_\phi^{\downarrow\lambda(r)\cup\ldots\cup\lambda(i)} \quad \supseteq \quad \Big\{\mathbf{z} \in \Omega_{s\cup\lambda(i)} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \textit{ and } \\ \mathbf{z}^{\downarrow\lambda(i)-s} \in W^{\omega_i\cap\lambda(ch(i))}_{\psi_i^{(r)}}(\mathbf{z}^{\downarrow\omega_i\cap\lambda(ch(i))})\Big\} \tag{8.10}$$

Proof: From the distribute phase, transitivity and the combination axiom follows:

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

We next apply property (8.9) with $s = u = \omega_i \cap \lambda(ch(i))$ and $t = \omega_i$ and obtain for all $\mathbf{x} \in \Omega_u$

$$\begin{aligned} W^{\omega_i\cap\lambda(ch(i))}_{\phi^{\downarrow\omega_i}}(\mathbf{x}^{\downarrow\omega_i\cap\lambda(ch(i))}) \quad &= \quad W^{\omega_i\cap\lambda(ch(i))}_{\mu_{ch(i)\rightarrow i}^{\downarrow\omega_i\cap\lambda(ch(i))}\otimes\psi_i^{(r)}}(\mathbf{x}^{\downarrow\omega_i\cap\lambda(ch(i))}) \\ &\supseteq \quad W^{\omega_i\cap\lambda(ch(i))}_{\psi_i^{(r)}}(\mathbf{x}^{\downarrow\omega_i\cap\lambda(ch(i))}). \end{aligned} \tag{8.11}$$

Applying Lemma 8.3, it then follows for the statement of Lemma 8.2:

$$\begin{aligned} c_\phi^{\downarrow s\cup\lambda(i)} \quad &= \quad \Big\{\mathbf{z} \in \Omega_{s\cup\lambda(i)} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and} \\ &\qquad\qquad \mathbf{z}^{\downarrow\lambda(i)-s} \in W^{\lambda(i)\cap\lambda(ch(i))}_{\phi^{\downarrow\lambda(i)}}(\mathbf{z}^{\downarrow\lambda(i)\cap\lambda(ch(i))})\Big\} \\ &= \quad \Big\{\mathbf{z} \in \Omega_{s\cup\lambda(i)} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and} \\ &\qquad\qquad \mathbf{z}^{\downarrow\lambda(i)-s} \in W^{\omega_i\cap\lambda(ch(i))}_{\phi^{\downarrow\omega_i}}(\mathbf{z}^{\downarrow\omega_i\cap\lambda(ch(i))})\Big\} \\ &\supseteq \quad \Big\{\mathbf{z} \in \Omega_{s\cup\lambda(i)} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and} \\ &\qquad\qquad \mathbf{z}^{\downarrow\lambda(i)-s} \in W^{\omega_i\cap\lambda(ch(i))}_{\psi_i^{(r)}}(\mathbf{z}^{\downarrow\omega_i\cap\lambda(ch(i))})\Big\}. \end{aligned} \tag{8.12}$$

■

Theorem 8.2 *If property (8.9) holds and if configuration extension sets are always non-empty, then at least one solution of $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ can be found, using the node contents at the end of any single-query local computation architecture.*

This follows from Lemma 8.4 applying the following procedure:

1. Execute a single-query local computation procedure on $\{\phi_1, \ldots, \phi_n\}$.

2. Compute $c_\phi^{\downarrow\lambda(r)}$ in the root node.

3. For $i = r-1, \ldots, 1$

   (a) compute $W_{\psi_i^{(r)}}^{\omega_i \cap \lambda(ch(i))}$ in node $i \in V$;

   (b) build a subset of $c_\phi^{\downarrow\lambda(r)\cup\ldots\cup\lambda(i)}$ by application of equation (8.10).

4. Return the subset of $c_\phi = c_\phi^{\downarrow\lambda(r)\cup\ldots\cup\lambda(1)}$.

Hence, if configuration extension sets are always non-empty and if property (8.9) is satisfied, we obtain at least one solution from the above procedure that is entirely based on the node contents at the end of a single-query local computation procedure. The two conditions imposed on the valuation algebra and on the solution extension sets therefore reflect the prize we pay to avoid the distribute phase. Here is the corresponding algorithm for the procedure above:

Algorithm 8.2 Computing some Solutions

```
input:  $\psi_i^{(r)}$ for $1 \leq i \leq r$   output:  $c \subseteq c_\phi$

begin
  $c$ := $W^{\emptyset}_{\psi_r^{(r)}}(\diamond)$;
  for $i = r-1 \ldots 1$ do
    $s$ := $\lambda(r) \cup \ldots \cup \lambda(i+1)$;
    $c$ := $\{\mathbf{z} : \mathbf{z}^{\downarrow s} \in c \wedge \mathbf{z}^{\downarrow\lambda(i)-s} \in W_{\psi_i^{(r)}}^{\omega_i\cap\lambda(ch(i))}(\mathbf{z}^{\downarrow\omega_i\cap\lambda(ch(i))})\}$;
  end;
  return $c$;
end
```

In particular, if both conditions are satisfied, we can always find one solution by the following non-deterministic algorithm:

Algorithm 8.3 Computing one Solution

```
input: $\psi_i^{(r)}$ for $1 \leq i \leq r$   output: $\mathbf{x} \in c_\phi$

begin
  choose $\mathbf{x} \in W^{\emptyset}_{\psi_r^{(r)}}(\diamond)$;
  for $i = r-1 \ldots 1$ do
    choose $\mathbf{y} \in W^{\omega_i \cap \lambda(ch(i))}_{\psi_i^{(r)}}(\mathbf{x}^{\downarrow \omega_i \cap \lambda(ch(i))})$;
    $\mathbf{x}$ := $(\mathbf{x}, \mathbf{y})$;
  end;
  return $\mathbf{x}$;
end
```

8.2.3 Computing all Solutions without Distribute

We have seen so far that all solutions can be found, if we agree on a complete run of the outward propagation. Without this computational effort, at least a subset of solutions can be identified, if condition (8.9) is satisfied. This section will show that if we impose a more limiting condition, all solutions can be found even with the second method. In fact, it is sufficient to ensure equality in Lemma 8.4:

Theorem 8.3 *If equality holds in equation (8.9), then all solutions can be computed based on the node contents at the end of a single-query local computation architecture.*

Proof: Since equality holds in property (8.9), we also have equality in the equations (8.11) and (8.12). This gives us the following reformulation of Lemma 8.2: For $i = r-1, \ldots, 1$ and $s = \lambda(r) \cup \ldots \cup \lambda(i+1)$ we have

$$c_\phi^{\downarrow s \cup \lambda(i)} = \Big\{ \mathbf{z} \in \Omega_{s \cup \lambda(i)} : \mathbf{z}^{\downarrow s} \in c_\phi^{\downarrow s} \text{ and } \mathbf{z}^{\downarrow \lambda(i) - s} \in W^{\omega_i \cap \lambda(ch(i))}_{\psi_i^{(r)}}\big(\mathbf{z}^{\downarrow \omega_i \cap \lambda(ch(i))}\big) \Big\} \tag{8.13}$$

■

We can then compute c_ϕ by the following procedure:

1. Execute a single-query local computation procedure on $\{\phi_1, \ldots, \phi_n\}$.
2. Identify $c_\phi^{\downarrow \lambda(r)}$ in the root node.
3. For $i = r-1, \ldots, 1$
   (a) compute $W^{\omega_i \cap \lambda(ch(i))}_{\psi_i^{(r)}}$ in node $i \in V$;
   (b) build $c_\phi^{\downarrow \lambda(r) \cup \ldots \cup \lambda(i)}$ by application of equation (8.13).
4. Return $c_\phi = c_\phi^{\downarrow \lambda(r) \cup \ldots \cup \lambda(1)}$.

This is expressed in the following algorithm:

Algorithm 8.4 Computing all Solutions II

```
input:   $\phi^{\downarrow\lambda(i)}$ for $1 \leq i \leq r$   output:   $c_\phi$

begin
  $c$ := $W^{\emptyset}_{\phi^{\downarrow\lambda(r)}}(\diamond)$;
  for $i = r-1 \dots 1$ do
    $s$ := $\lambda(r) \cup \dots \cup \lambda(i+1)$;
    $c$ := $\{\mathbf{z} : \mathbf{z}^{\downarrow s} \in c \wedge \mathbf{z}^{\downarrow\lambda(i)-s} \in W^{\omega_i \cap \lambda(ch(i))}_{\psi_i^{(r)}}(\mathbf{x}^{\downarrow\omega_i \cap \lambda(ch(i))})\}$;
  end;
  return $c$;
end
```

This completes our general study on local computation based solution construction in valuation algebras. In the following section, we provide a detailed analysis of a particularly important field of application. This is constraint reasoning or more generally, the solution of optimization problems. In addition, this section will also show that important examples exist, where only strict inclusion holds in (8.9). An application to solving linear systems of equations will be presented in Chapter 9.

8.3 OPTIMIZATION AND CONSTRAINT PROBLEMS

Chapter 5 introduced the extensive family of semiring valuation algebras that emerge from a simple mapping from configurations to the values of a commutative semiring. Particularly important among these valuation algebras are *constraint formalisms*, which are obtained from the subclass of c-semirings in Definition 5.3 (Bistarelli *et al.*, 1997; Bistarelli *et al.*, 1999). C-semirings are characterized by an absorbing unit element, which further implies the idempotency of semiring addition. Moreover, it follows from Lemma 5.3 that all idempotent semirings are partially ordered. We will see in this section that inference problems over valuation algebras induced by idempotent semirings represent optimization problems with respect to the partial semiring order. If in addition this order is total, we may ask for one or more configurations that lead to the optimum value. This is a very typical field of application for the theory of solution construction developed in the first part of this chapter. In particular, since c-semirings are idempotent too, it covers the solution of constraint systems as a special case.

8.3.1 Totally Ordered Semirings

In the general case, a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ only provides the canonical preorder introduced in equation (5.3). This is a very fundamental remark since it leads to a split of algebra into semirings with additive inverses (i.e. the monoid $(A, +)$ forms a group) and semirings with a canonical partial order called dioids (Gondran & Minoux, 2008). Take for example the arithmetic semiring of integers $\langle \mathbb{Z}, +, \cdot, 0, 1 \rangle$

and observe that $a \leq b$ and $b \leq a$ does not imply that $a = b$. Antisymmetry therefore contradicts the existence of additive inverses or, in other words, the group structure of $(A, +)$. As mentioned above, idempotency of addition is a sufficient condition to turn the semiring preorder into a partial order. If this relation is furthermore a total order as it is in many of the examples in Section 5.1, we obtain a class of semirings that are sometimes called *addition-is-max semirings* (Wilson, 2005). This name comes from the following lemma.

Lemma 8.5 *In an idempotent semiring we have* $a + b = \max\{a, b\}$ *if, and only if, its canonical order is total.*

Proof: If the canonical order is total, we either have $a \leq b$ or $b \leq a$ for all $a, b \in A$. Assume that $a \leq b$. Then, $a + b = b$ and therefore $a + b = \max\{a, b\}$. The converse claim holds trivially. ∎

Alternatively, we may also see this lemma as a strengthening of Property (SP4) from Lemma 5.5, and according to (SP1) and (SP2) in Lemma 5.2 the total order behaves monotonic with respect to both semiring operations. For later applications however, we often require a more restrictive version of monotonicity whose definition is based on the following notation:

$$a < b \quad \Leftrightarrow \quad a \leq b \text{ and } a \neq b. \tag{8.14}$$

Definition 8.3 *An idempotent semiring* $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ *is called* strict monotonic, *if for* $a, b, c \in A$ *and* $c \neq \mathbf{0}$, $a < b$ *implies that* $a \times c < b \times c$.

Let us see which of the semirings from Section 5.1 are strict monotonic.

Example 8.1 *The Boolean semiring* $\langle \{0, 1\}, \max, \min, 0, 1 \rangle$ *of Example 5.2 is strict monotonic, since* $0 < 1$ *implies that* $\min(0, 1) < \min(1, 1)$. *Its generalization on the other hand, the Bottleneck semiring* $\langle \mathbb{R} \cup \{-\infty, +\infty\}, \max, \min, -\infty, +\infty \rangle$ *of Example 5.3, is not strict monotonic since we cannot conclude from* $a < b$ *that* $\min(a, c) < \min(b, c)$. *A bit more challenging is the tropical semiring* $\langle \mathbb{N} \cup \{0, +\infty\}, \min, +, \infty, 0 \rangle$ *from Example 5.4. Here, the canonical semiring order corresponds in fact to the inverse order of natural numbers. Nevertheless, this semiring is strict monotonic,* $\min(a, b) = b$ *and* $a \neq b$ *implies that* $\min(a + c, b + c) = b + c$ *and* $a + c \neq b + c$, *if* $c \neq \infty$. *Also, the arctic semiring* $\langle \mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0 \rangle$ *is strict monotonic which follows from the same argument replacing minimization by maximization. Further, we directly see that strict monotonicity cannot hold in the truncation semiring of Example 5.6, but it does so in the semiring of formal languages from Example 5.7. Here, the strict order from equation (8.3) corresponds to strict set inclusion which is maintained under concatenation. Finally, in the triangular norm semirings of Example 5.8, strict monotonicity has to be verified for each choice of the triangular norm separately since they are only non-decreasing in their arguments.*

An important consequence of a total order flows out of Definition 8.3:

Lemma 8.6 *In a totally ordered, strict monotonic, idempotent semiring we have for* $a \neq \mathbf{0}$ *that* $a \times b = a \times c$ *if, and only if,* $b = c$.

Proof: Clearly, $b = c$ implies that $a \times b = a \times c$. On the other hand, assume $a \times b = a \times c$. Since the semiring is totally ordered, either $b < c$, $c < b$ or $b = c$ must hold. Suppose $b < c$. Then, because $\times$ behaves strict monotonic, we have $a \times b < a \times c$ which contradicts the assumption. A similar contradiction is obtained if we assume $c < b$ which proves $b = c$. ■

So far, we called an idempotent semiring totally ordered, if its canonical order is total. In fact, the following lemma shows that if other total and monotonic orders are present in an idempotent semiring, they are always equivalent to either the canonical order or the inverse canonical order. The exact case can be found out by comparing the zero and unit element of the semiring. Thus, we may simply speak about totally ordered, idempotent semirings without specifying the exact order relation. On the other hand, we can limit our considerations to the total canonical order and the results then also apply to other total orders.

Lemma 8.7 *Let* $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ *be an idempotent semiring with a total canonical order* $\leq_1$ *and an arbitrary total, monotonic order* $\leq_2$ *defined over* A. *We then have for all* $a, b \in A$*:*

1. $\mathbf{0} \leq_2 \mathbf{1} \;\Rightarrow\; [a \leq_2 b \;\Leftrightarrow\; a \leq_1 b]$.

2. $\mathbf{1} \leq_2 \mathbf{0} \;\Rightarrow\; [a \leq_2 b \;\Leftrightarrow\; b \leq_1 a]$.

Proof: We first remark that $\mathbf{0} \leq_2 \mathbf{1}$ implies $\mathbf{0} = \mathbf{0} \times b \leq_2 \mathbf{1} \times b = b$ and therefore $a = a + \mathbf{0} \leq_2 a + b$ by monotonicity of the assumed total order. Hence, $a, b \leq_2 a + b$ for all $a, b \in A$. Similarly, we derive from $\mathbf{1} \leq_2 \mathbf{0}$ that $a + b \leq_2 a, b$.

1. Assume that $a \leq_2 b$ and by monotonicity and idempotency $a + b \leq_2 b + b = b$. Since $\mathbf{0} \leq_2 \mathbf{1}$ we have $b \leq_2 a + b$ and therefore $b \leq_2 a + b \leq_2 b$. Thus, we conclude by antisymmetry that $a + b = b$, i.e. $a \leq_1 b$. On the other hand, if $a \leq_1 b$, we obtain from $\mathbf{0} \leq_2 \mathbf{1}$ that $a \leq_2 a + b = b$ and therefore $a \leq_2 b$.

2. Assume that $a \leq_2 b$ and by idempotency and monotonicity $a = a + a \leq_2 a + b$. Since $\mathbf{1} \leq_2 \mathbf{0}$ we have $a + b \leq_2 a$ and therefore $a \leq_2 a + b \leq_2 a$. Thus, we conclude by antisymmetry that $a + b = a$, i.e. $b \leq_1 a$. On the other hand, if $b \leq_1 a$, we obtain from $\mathbf{1} \leq_2 \mathbf{0}$ that $a = a + b \leq_2 b$ and therefore $a \leq_2 b$. ■

Note that for semirings where $\mathbf{0} = \mathbf{1}$, both consequences of Lemma 8.7 are trivially satisfied because these semirings consist of only one element.

Example 8.2 *Consider the tropical semiring* $\langle \mathbb{N} \cup \{0, +\infty\}, \min, +, \infty, 0\rangle$ *which is idempotent and provides the natural order between non-negative integers. This order is monotonic with respect to* $+$ *and* min. *Since* $\mathbf{0} = \infty$ *and* $\mathbf{1} = 0$ *we have* $\mathbf{1} \leq \mathbf{0}$. *We therefore conclude from Lemma 8.7 that the natural order of the tropical semiring corresponds to the inverse canonical order.*

8.3.2 Optimization Problems

We learned in Section 2.3 that inference problems capture the computational interest in valuation algebras, and their efficient solution was the aim of the development of local computation algorithms. Next, it will be shown that inference problems acquire a very particular meaning when dealing with valuation algebras that are induced by semirings with idempotent addition. Due to Lemma 5.5, Property (SP4) we then have for an inference problem with query $t \subseteq d(\phi) = s$ and $\mathbf{x} \in \Omega_t$,

$$\phi^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y}\in\Omega_{s-t}} \phi(\mathbf{x},\mathbf{y}) \;=\; \sup\{\phi(\mathbf{x},\mathbf{y}),\; \mathbf{y} \in \Omega_{s-t}\}, \tag{8.15}$$

if $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ denotes the objective function. This corresponds to the computation of the lowest upper bound of all semiring values that are assigned to those configurations of ϕ that project to $\mathbf{x}$. In particular, we obtain for the empty query

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

which amounts to the computation of the lowest upper bound of all values of ϕ. If we furthermore assume that the underlying semiring is totally ordered, we obtain according to Lemma 8.5

$$\phi^{\downarrow t}(\mathbf{x}) \;=\; \sum_{\mathbf{y}\in\Omega_{s-t}} \phi(\mathbf{x},\mathbf{y}) \;=\; \max\{\phi(\mathbf{x},\mathbf{y}),\; \mathbf{y} \in \Omega_{s-t}\}, \tag{8.16}$$

and

$$\phi^{\downarrow\emptyset}(\diamond) \;=\; \max\{\phi(\mathbf{x}),\; \mathbf{x} \in \Omega_s\}. \tag{8.17}$$

In other words, inference problems on valuation algebras induced by a totally ordered, idempotent semiring amount to the computation of maximum (or minimum) values. They therefore also called *optimization problems* or, if they are induced by c-semirings, *constraint problems*. Let us survey some typical examples:

■ 8.1 Classical Optimization

The most classical optimization problems are obtained from the tropical or arctic semirings of Example 5.4, where × corresponds to the usual addition and $+$ to either minimization or maximization. Let us consider the tropical semiring $\langle \mathbb{R} \cup \{\infty\}, \min, +, \infty, 0\rangle$ and a knowledgebase $\{\phi_1, \ldots, \phi_n\}$ of induced valuations. We then obtain for the inference problem with empty query

$$\begin{aligned}\phi^{\downarrow\emptyset}(\diamond) &= (\phi_1 \otimes \ldots \otimes \phi_n)^{\downarrow\emptyset} \\ &= \min\left\{\phi_1(\mathbf{x}^{\downarrow d(\phi_1)}) + \ldots + \phi_n(\mathbf{x}^{\downarrow d(\phi_n)}),\; \mathbf{x} \in \Omega_s\right\}\end{aligned}$$

where $s = d(\phi)$. Recall from Example 8.2 that the natural order of a tropical semiring corresponds to its inverse canonical order. This explains the difference to equation (8.17). Solving optimization problems induced by tropical or arctic semirings is the typical task of reasoning with weighted constraints. A concrete application from coding will be presented in Instance 8.5 below.

8.2 Satisfiability

Satisfiability with crisp constraints corresponds to the task of solving an optimization problem induced by the Boolean semiring $\langle\{0,1\},\max,\min,0,1\rangle$. Given a knowledgebase of valuations induced by the Boolean semiring, the crisp constraint $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ 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 $\phi^{\downarrow\emptyset}(\diamond) = 1$. Otherwise, if $\phi^{\downarrow\emptyset}(\diamond) = 0 = z_\emptyset(\diamond)$, the set of constraints is contradictory due to the nullity axiom of Section 3.4. This also includes the satisfiability problem of propositional logic from Section 7.1. If $\{f_1,\ldots,f_n\}$ denotes a set of formulae, we may express their possible interpretations by a set of crisp constraints or indicator functions $\{\phi_1,\ldots,\phi_n\}$. The objective function ϕ then reflects the possible interpretations of the conjunction $f_1 \wedge \ldots \wedge f_n$, and R_ϕ corresponds to its set of models. Thus, if $|R_\phi| > 0$, the conjunction is satisfiable.

8.3 Maximum Satisfiability

Maximum satisfiability (Garey & Johnson, 1990) can be regarded as an approximation task for the satisfiability problem. For a set $\{\phi_1,\ldots,\phi_n\}$ of crisp constraints or formulae, we do not require that all elements are satisfied as in the foregoing instance, but we content ourselves with satisfying a maximum number of elements. Thus, if ϕ_1 to ϕ_n still map configurations to Boolean values, we may compute

$$\phi^{\downarrow\emptyset}(\diamond) \quad = \quad \max\left\{\phi_1(\mathbf{x}^{\downarrow d(\phi_1)}) + \ldots + \phi_n(\mathbf{x}^{\downarrow d(\phi_n)}),\ \mathbf{x} \in \Omega_s\right\}.$$

This identifies the maximum number of constraints or formulae that can be satisfied by any configuration. The semiring used in this application is the arctic semiring $\langle\mathbb{Z} \cup \{-\infty,\infty\},\max,+,-\infty,0\rangle$ of Example 5.5.

8.4 Most & Least Probable Values

We have seen in Section 5.4 that the arithmetic semiring induces the valuation algebra of probability potentials. However, instead of a probability value, we are sometimes interested in identifying either a minimum or maximum value of all probabilities in the objective function ϕ. The maximum can for example be determined using the product t-norm semiring. Here, addition is maximization and $\phi^{\downarrow\emptyset}(\diamond)$ corresponds to the probability of the *most probable configuration* or *explanation* of a given situation. This is especially important in diagnostic problems. Similarly, if we replace maximization by minimization, the marginal identifies the value of the *least probable configuration* or *explanation*.

When dealing with optimization or constraint problems, we are in general less interested in the optimum value of the objective function, but more interested in the actual configurations that adopt this value. Following (Shenoy, 1996) we refer to such configurations as *solution configurations*. The most and least probable configurations of Instance 8.4 are typical examples of such solution configurations.

Definition 8.4 *Let $\langle \Phi, D \rangle$ be a valuation algebra induced by a totally ordered, idempotent semiring. For $\phi \in \Phi$ with $d(\phi) = s$, we call $\mathbf{x} \in \Omega_s$ a* solution configuration, *if $\phi(\mathbf{x}) = \phi^{\downarrow\emptyset}(\diamond)$.*

We shall see in Section 8.4 that solution configurations give rise to solution extension sets according to Definition 8.1. Let us consider some typical applications, where knowing the solution configuration is more important than computing their assigned semiring value. These examples all require to find a single solution configuration for a valuation $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ with $d(\phi) = s$. Depending on whether the canonical semiring order corresponds to minimization or maximization, we write

$$\arg\max_{\mathbf{x} \in \Omega_s} \phi \quad \text{or} \quad \arg\min_{\mathbf{x} \in \Omega_s} \phi,$$

for the task of finding an arbitrary solution configuration.

8.5 Bayesian and Maximum Likelihood Decoding

Consider an unreliable, memoryless communication channel to transmit symbols out of a finite coding alphabet $\mathcal{A}$. If we assign the random variables X and Y to the input and output of the channel, then the latter is fully specified by its transmission probabilities $p(Y = y_i | X = x_j)$ for $y_i, x_j \in \mathcal{A}$. This is the probability that input symbol x_j is changed into output symbol y_i by sending it through the channel. Figure 8.1 illustrates such a channel for a binary coding alphabet $\mathcal{A} = \{0, 1\}$. Instead of single symbols, we now transmit code words that consist of $n \in \mathbb{N}$ consecutive symbols. Thus, an unknown input word $\mathbf{x} = (x_1, \ldots, x_n)$ is transmitted over the channel and on the receiver side, an output word $\mathbf{y} = (y_1, \ldots, y_n)$ is observed. The decoding process asks to deduce the input from the received output and this can for example be done by choosing the input word $\mathbf{x} = (x_1, \ldots, x_n)$ that leads most probably to the observed output $\mathbf{y}$. In other words, we choose $\mathbf{x}$ such that $p(\mathbf{y}|\mathbf{x})$ is maximum. Since $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}, \mathbf{x})/p(\mathbf{y})$ and maximization only concerns the input code words, it is sufficient to compute

$$\arg\max_{\mathbf{x}} \left(\prod_{i=1}^{n} p(Y_i = y_i | X_i = x_i) \cdot p(X_1 = x_1, \ldots, X_n = x_n) \right).$$

Here, $p(\mathbf{x}) = p(X_1 = x_1, \ldots, X_n = x_n)$ is the prior distribution of the input word. The semiring valuation algebra used in this example is induced by the t-norm semiring $\langle [0,1], \max, \cdot, 0, 1 \rangle$ of Example 5.8.

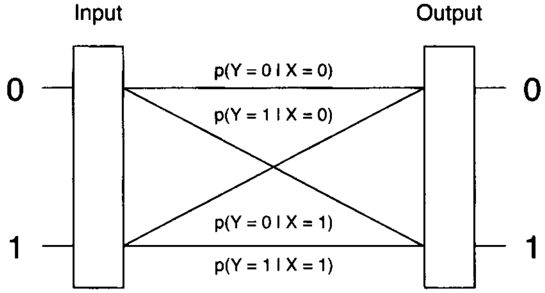


Figure 8.1 A binary, unreliable, memoryless communication channel.

The above decoding approach is generally called *Bayes decoding* since it is based on a prior distribution $p(\mathbf{x})$ of the input code words. Alternatively, this can be simplified by assuming a uniform prior distribution that can be ignored in the maximization process. We then obtain

$$\arg\max_{\mathbf{x}} \left(\prod_{i=1}^{n} p(Y_i = y_i | X_i = x_i) \right)$$

called *maximum likelihood decoding*. It is convenient for computational purposes to replace the maximization of probabilities by the minimization of their negated logarithms. We obtain for the case of maximum likelihood decoding

$$\arg\max_{\mathbf{x}} \left(\prod_{i=1}^{n} p(Y_i = y_i | X_i = x_i) \right) = \tag{8.18}$$

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

This is an optimization problem induced by the tropical semiring.

■ 8.6 Linear Decoding

Linear codes go a step further and take into consideration that not all arrangements of binary input symbols may be valid code words. Such systems therefore provide a so-called *(low density) parity check matrix H* which assures $H \cdot \mathbf{x}^T = \mathbf{0}$ if, and only if, $\mathbf{x}$ is a valid code word. For illustration purposes, assume the following example matrix (Wiberg, 1996):

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Then, $\mathbf{x} = (x_1, \ldots, x_7)$ is a valid code word if

$$(x_1 + x_2 + x_4, x_3 + x_4 + x_6, x_4 + x_5 + x_7) \quad = \quad (0, 0, 0).$$

These additional constraints can be inserted into the maximum likelihood decoding scheme using the following auxiliary functions:

$$\chi_1(x_1, x_2, x_4) \;=\; \begin{cases} 0, & \text{if } x_1 + x_2 + x_4 = 0 \\ \infty, & \text{otherwise,} \end{cases}$$

$$\chi_2(x_3, x_4, x_6) \;=\; \begin{cases} 0, & \text{if } x_3 + x_4 + x_6 = 0 \\ \infty, & \text{otherwise,} \end{cases}$$

$$\chi_3(x_4, x_5, x_7) \;=\; \begin{cases} 0, & \text{if } x_4 + x_5 + x_7 = 0 \\ \infty, & \text{otherwise.} \end{cases}$$

We then obtain for equation (8.18) and $n = 7$,

$$\arg\min_{\mathbf{x}}\Big(\sum_{i=1}^{7} \; - \log p(y_i|x_i) + \chi_1(x_1, x_2, x_4) + \\ \chi_2(x_3, x_4, x_6) + \chi_3(x_4, x_5, x_7)\Big).$$

It remains an optimization problem with knowledgebase factors induced by the tropical semiring. Furthermore, applying the fusion algorithm from Section 3.2.1 to this setting yields the *Gallager-Tanner-Wiberg algorithm* (Gallager, 1963) as it was observed by (Aji & McEliece, 2000). A survey of these and related decoding schemes is given in (MacKay, 2003) where the author presents the corresponding algorithms as message-passing schemes. This suggests that even more sophisticated and state of the art decoding systems such as *convolutional* or *turbo codes* are subsumed by optimization problems over semiring valuation algebras.

More examples can be found in constraint literature, e.g. (Apt, 2003).

8.4 COMPUTING SOLUTIONS OF OPTIMIZATION PROBLEMS

Given a factorized valuation $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ with $d(\phi) = s$, induced by a totally ordered, idempotent semiring, we obtain the optimum value $\phi^{\downarrow\emptyset}(\diamond)$ by applying an arbitrary single-query local computation architecture from Chapter 3. But we have seen in the foregoing section that in such cases, one is often interested in finding one or more configurations $\mathbf{x} \in \Omega_s$ that adopt the optimum value, i.e. with $\phi(\mathbf{x}) = \phi^{\downarrow\emptyset}(\diamond)$.

Such configurations were called solution configuration in Definition 8.4, and it follows directly from equation (8.16) that at least one solution configuration always exists. Moreover, given an arbitrary configuration $\mathbf{y} \in \Omega_t$ for $t \subseteq s$, it is always possible to find an extension $\mathbf{x} \in \Omega_{s-t}$ such that $\phi^{\downarrow t}(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{y})$. Subsequently, we refer to $\mathbf{y}$ as a configuration extension of $\mathbf{x}$ with respect to ϕ. The set of all configuration extensions of $\mathbf{x}$ with respect to ϕ is then given by

$$W_\phi^t(\mathbf{x}) \quad = \quad \left\{\mathbf{y} \in \Omega_{s-t} : \phi(\mathbf{x}, \mathbf{y}) = \phi^{\downarrow t}(\mathbf{x})\right\}. \tag{8.19}$$

Theorem 8.4 *Solution extension sets in optimization problems satisfy the property of Definition 8.1, i.e. for all $\phi \in \Phi$ with $t \subseteq u \subseteq s = d(\phi)$ and $\mathbf{x} \in \Omega_t$ we have*

$$W_\phi^t(\mathbf{x}) \quad = \quad \left\{\mathbf{z} \in \Omega_{s-t} : \mathbf{z}^{\downarrow u-t} \in W_{\phi^{\downarrow u}}^t(\mathbf{x}) \text{ and } \mathbf{z}^{\downarrow s-u} \in W_\phi^u(\mathbf{x}, \mathbf{z}^{\downarrow u-t})\right\}.$$

Proof: It follows from equation (8.19) that

$$\begin{aligned}
\left\{\mathbf{z} \in \Omega_{s-t} : \mathbf{z}^{\downarrow u-t} \in W_{\phi^{\downarrow u}}^t(\mathbf{x}) \text{ and } \mathbf{z}^{\downarrow s-u} \in W_\phi^u(\mathbf{x}, \mathbf{z}^{\downarrow u-t})\right\} &= \\
\left\{\mathbf{z} \in \Omega_{s-t} : \phi^{\downarrow t}(\mathbf{x}) = \phi^{\downarrow u}(\mathbf{x}, \mathbf{z}^{\downarrow u-t}) \text{ and } \phi^{\downarrow u}(\mathbf{x}, \mathbf{z}^{\downarrow u-t}) = \phi(\mathbf{x}, \mathbf{z})\right\} &= \\
\left\{\mathbf{z} \in \Omega_{s-t} : \phi^{\downarrow t}(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{z})\right\} &= W_\phi^t(\mathbf{x}).
\end{aligned}$$

■

Configuration extension sets in optimization problems therefore instantiate the general definition of configuration extension sets given in Section 8.1. So, we may also specialize the general definition of solution sets given in Definition 8.2 to optimization problems. We obtain

$$c_\phi \; = \; W_\phi^\emptyset(\diamond) \; = \; \left\{\mathbf{y} \in \Omega_s : \phi(\mathbf{x}) = \phi^{\downarrow \emptyset}(\diamond)\right\}. \tag{8.20}$$

Observe that this indeed corresponds to the notion of solutions in constraint or optimization problems given in Definition 8.4. Moreover, we also see that several possibilities to define configuration extension sets may exist in a valuation algebra. Instead of the configurations that map to the optimum value, we could also consider all other configurations that do not satisfy this property as solutions and define the configuration extension sets accordingly. In terms of constraint reasoning, we then search for counter-models. However, we continue with the above definition and show in the following example that computing solution extensions or solution sets in optimization problems essentially amounts to a table lookup.

Example 8.3 *Consider a set $\{A, B, C\}$ of variables with finite frames $\Omega_A = \{a, \overline{a}\}$, $\Omega_B = \{b, \overline{b}\}$ and $\Omega_C = \{c, \overline{c}\}$, and a valuation ϕ induced by the tropical semiring $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, -\infty, 0\rangle$. We perform a step-wise projection of ϕ until all variables are eliminated:*

$\phi =$

A	B	C	
a	b	c	2
a	b	$\bar{c}$	4
a	$\bar{b}$	c	9
a	$\bar{b}$	$\bar{c}$	4
$\bar{a}$	b	c	5
$\bar{a}$	b	$\bar{c}$	8
$\bar{a}$	$\bar{b}$	c	2
$\bar{a}$	$\bar{b}$	$\bar{c}$	5

$\phi^{\downarrow\{A,C\}} =$

A	C	
a	c	2
a	$\bar{c}$	4
$\bar{a}$	c	2
$\bar{a}$	$\bar{c}$	5

$\phi^{\downarrow\{C\}} =$

C	
c	2
$\bar{c}$	4

We finally obtain $\phi^{\downarrow\emptyset}(\diamond) = 2$. By consulting the tabular representation of ϕ, we determine the solution configuration set

$$c_\phi \quad = \quad \{(a,b,c),(\bar{a},\bar{b},c)\}.$$

For the partial solution $(a) \in c_\phi^{\downarrow\{A\}}$ we find the solution extension set

$$W_\phi^{\{A\}}(a) \quad = \quad \{(b,c)\}$$

again by consulting the above table, and for $(c) \in c_\phi^{\downarrow\{C\}}$ we find

$$W_\phi^{\{C\}}(c) \quad = \quad \{(a,b),(\bar{a},\bar{b})\}.$$

Since configuration extensions in optimization problems satisfy Definition 8.1, it follows from Section 8.2.1 that the complete solution set c_ϕ of a factorized valuation $\phi = \phi_1 \otimes \ldots \otimes \phi_n$ can be determined from the results of a previous run of a multi-query local computation architecture. The corresponding procedure is given in Algorithm 8.1. However, many practical applications of constraint reasoning just require to find a single solution, and this should, if possible, be done without the additional effort of a downward propagation in the local computation scheme. Indeed, we have seen in Section 8.2.2 that this is possible if configuration extension sets are always non-empty and condition 8.9 of Lemma 8.4 holds. The first requirement directly follows from equation (8.19), i.e. configuration extension sets and therefore also solution sets in optimization problems are always non-empty. The second requirement is guaranteed by the following lemma:

Lemma 8.8 *In a valuation algebra induced by a totally ordered, idempotent semiring we have for all $\psi_1, \psi_2 \in \Phi$ with $d(\psi_1) = s$, $d(\psi_2) = t$, $s \subseteq u \subseteq s \cup t$ and $\mathbf{x} \in \Omega_u$*

$$W_{\psi_2}^{u \cap t}(\mathbf{x}^{\downarrow u \cap t}) \quad \subseteq \quad W_{\psi_1 \otimes \psi_2}^{u}(\mathbf{x}).$$

Proof: Assume $\mathbf{x} \in \Omega_u$ and $\mathbf{y} \in W_{\psi_2}^{u \cap t}(\mathbf{x}^{\downarrow t \cap u})$. By equation (8.19) we have

$$\psi_2(\mathbf{x}^{\downarrow u \cap t}, \mathbf{y}) \quad = \quad \psi_2^{\downarrow u \cap t}(\mathbf{x}^{\downarrow u \cap t}),$$

hence also

$$\psi_1(\mathbf{x}^{\downarrow s}) \otimes \psi_2(\mathbf{x}^{\downarrow u \cap t}, \mathbf{y}) \quad = \quad \psi_1(\mathbf{x}^{\downarrow s}) \otimes \psi_2^{\downarrow u \cap t}(\mathbf{x}^{\downarrow u \cap t}),$$

By application of the definition of combination in semiring valuation algebras and the combination axiom we obtain:

$$\begin{aligned} (\psi_1 \otimes \psi_2)(\mathbf{x}, \mathbf{y}) &= \psi_1(\mathbf{x}^{\downarrow s}) \otimes \psi_2(\mathbf{x}^{\downarrow u \cap t}, \mathbf{y}) \\ &= \psi_1(\mathbf{x}^{\downarrow s}) \otimes \psi_2^{\downarrow u \cap t}(\mathbf{x}^{\downarrow u \cap t}) \\ &= \left(\psi_1 \otimes \psi_2^{\downarrow u \cap t}\right)(\mathbf{x}) \;=\; (\psi_1 \otimes \psi_2)^{\downarrow u}(\mathbf{x}). \end{aligned}$$

We conclude from $(\psi_1 \otimes \psi_2)(\mathbf{x}, \mathbf{y}) = (\psi_1 \otimes \psi_2)^{\downarrow u}(\mathbf{x})$ that $\mathbf{y} \in W^u_{\psi_1 \otimes \psi_2}(\mathbf{x})$. ■

Both conditions of Theorem 8.2 are therefore satisfied in optimization problems, which allows us directly to apply Algorithm 8.2 or the computation of a non-empty subset of solutions, or Algorithm 8.3 for the identification of a single solution. The following example illustrates a complete run of Algorithm 8.3.

Example 8.4 *We consider binary variables A, B, C with frames $\Omega_A = \{a, \overline{a}\}$ to $\Omega_C = \{c, \overline{c}\}$. Let ψ_1, ψ_2 and ψ_3 be three join tree factors defined over the bottleneck semiring $\langle \mathbb{R} \cup \{-\infty, \infty\}, \max, \min, -\infty, \infty\rangle$ with domains $d(\psi_1) = \{A, B\}$, $d(\psi_2) = \{B, C\}$, $\psi_3 = \{B\}$ and the following values:*

$\psi_1 =$

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

$\psi_2 =$

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

$\psi_3 =$

B	
b	1
$\overline{b}$	6

The join tree in Figure 8.2 corresponds to this factorization and numbering.

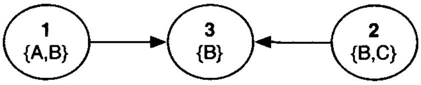


Figure 8.2 The join tree that belongs to the node factors of Example 8.4.

Let us first compute $\phi = \psi_1 \otimes \psi_2 \otimes \psi_3$ directly to verify later results:

$\phi =$

A	B	C	
a	b	c	1
a	b	$\bar{c}$	1
a	$\bar{b}$	c	3
a	$\bar{b}$	$\bar{c}$	3
$\bar{a}$	b	c	1
$\bar{a}$	b	$\bar{c}$	1
$\bar{a}$	$\bar{b}$	c	2
$\bar{a}$	$\bar{b}$	$\bar{c}$	2

We observe that $c_\phi = \{(a, \bar{b}, c), (a, \bar{b}, \bar{c})\}$ are the configurations with the maximum value. This will now be compared it with the result of Algorithm 8.2 that first requires to execute a complete run of the collect algorithm:

$\mu_{1\to 3} =$

B	
b	3
$\bar{b}$	4

$\psi_3^{(2)} =$

B	
b	1
$\bar{b}$	4

$\mu_{2\to 3} =$

B	
b	5
$\bar{b}$	3

$\psi_3^{(3)} =$

B	
b	1
$\bar{b}$	3

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

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

We can only choose $\mathbf{x}^{\downarrow\{B\}} = (\bar{b})$ and proceed for $i = 2$:

$$W^{\{B\}}_{\psi_2^{(3)}}(\bar{b}) = \{(c), (\bar{c})\}.$$

We choose (c) and obtain $\mathbf{x}^{\downarrow\{B,C\}} = (\bar{b}, c)$. Finally, for $i = 1$ we compute:

$$W^{\{B\}}_{\psi_1^{(3)}}(\bar{b}) = \{(a)\}$$

and obtain the solution configuration $\mathbf{x} = (a, \bar{b}, c) \in c_\phi$ with $\phi(a, \bar{b}, c) = 3$.

If we repeat the computations in this example with Algorithm 8.2, we find the complete solution set $c_\phi = \{(a, \bar{b}, c), (a, \bar{b}, \bar{c})\}$. But this is generally not the case. The following example shows that sometimes strict inclusion holds in Lemma 8.8, which makes the computation of all solutions using Algorithm 8.2 impossible. Moreover, we cannot even know without computing ϕ explicitly, whether all solution configurations have been found or not.

Example 8.5 *We take the bottleneck semiring $\langle \mathbb{R} \cup \{-\infty, \infty\}, \max, \min, -\infty, \infty\rangle$ of Example 5.3 and assume two semiring valuations ϕ and ψ with domains $d(\phi) = \{A\}$ and $d(\psi) = \{A, B\}$. The variable frames are $\Omega_A = \{a, \bar{a}\}$ and $\Omega_B = \{b, \bar{b}\}$.*

$\phi =$

A	
a	1
$\bar{a}$	1

$\psi =$

A	B	
a	b	6
a	$\bar{b}$	7
$\bar{a}$	b	8
$\bar{a}$	$\bar{b}$	9

$\phi \otimes \psi =$

A	B	
a	b	1
a	$\bar{b}$	1
$\bar{a}$	b	1
$\bar{a}$	$\bar{b}$	1

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

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

A	
a	7
$\bar{a}$	9

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

A	
a	1
$\bar{a}$	1

Finally, we remark that

$$W_{\psi}^{u \cap t}(a) \;=\; W_{\psi}^{u \cap t}(\bar{a}) \;=\; \{(\bar{b})\} \;\subset\; W_{\phi\otimes\psi}^{u}(a) \;=\; W_{\phi\otimes\psi}^{u}(\bar{a}) \;=\; \{(b), (\bar{b})\}.$$

All solution construction algorithms in Section 8.1 start from the marginal of ϕ with respect to the root node label $\lambda(r)$, which is obtained from the previous local computation process. From this marginal, we may further compute

$$\phi^{\downarrow\emptyset}(\diamond) \;=\; \left(\phi^{\downarrow\lambda(r)}\right)^{\downarrow\emptyset}(\diamond)$$

by one additional projection. The following lemma states that if this value corresponds to the zero element in the semiring, then all configurations are solutions of ϕ. If this is the case, we do not need to run the solution construction process.

Lemma 8.9 *If $d(\phi) = s$, then $\phi^{\downarrow\emptyset}(\diamond) = \mathbf{0}$ implies that $c_\phi = \Omega_s$.*

Proof: Property (SP3) in Lemma 5.2 implies that $\mathbf{0} \leq \phi(\mathbf{x})$ for all $\mathbf{x} \in \Omega_s$. Hence, if the maximum value is equal to zero, then all configurations are solutions. ■

Constraints with zero as maximum element are sometimes called *contradictory*. Hence, it follows from the definition of solution extension sets in constraint systems that all configurations are solutions to a contradictory constraint. If, at the end of the collect phase, we obtain $\phi^{\downarrow\emptyset}(\diamond) = \mathbf{0}$ in the root node, we know from the above lemma that all configurations are solutions. It is therefore not necessary to build the solution configuration set and we can stop the algorithm. Subsequently, we exclude this special case in the root node, before starting the solution construction procedure. Then, equality in Lemma 8.8 can be guaranteed, if the semiring is strict monotonic according to Definition 8.3.

Lemma 8.10 *In a valuation algebra induced by a totally ordered, idempotent and strict monotonic semiring we have for all $\psi_1, \psi_2 \in \Phi$ with $d(\psi_1) = s$, $d(\psi_2) = t$, $s \subseteq u \subseteq s \cup t$ and $\mathbf{x} \in \Omega_u$*

$$W_{\psi_2}^{u\cap t}(\mathbf{x}^{\downarrow u \cap t}) \;=\; W_{\psi_1\otimes\psi_2}^{u}(\mathbf{x}).$$

Proof: It remains to prove that

$$W_{\psi_2}^{u\cap t}(\mathbf{x}^{\downarrow u \cap t}) \;\supseteq\; W_{\psi_1\otimes\psi_2}^{u}(\mathbf{x}).$$

Assume $\mathbf{x} \in \Omega_u$ with $\psi_1(\mathbf{x}^{\downarrow s}) \neq \mathbf{0}$ and $\mathbf{y} \in W_{\psi_1\otimes\psi_2}^{u}(\mathbf{x})$. By definition,

$$\begin{aligned}(\psi_1 \otimes \psi_2)^{\downarrow u}(\mathbf{x}) &= (\psi_1 \otimes \psi_2)(\mathbf{x}, \mathbf{y}) \\ &= \psi_1(\mathbf{x}^{\downarrow s}) \times \psi_2(\mathbf{x}^{\downarrow u\cap t}, \mathbf{y}).\end{aligned}$$

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

$$(\psi_1 \otimes \psi_2)^{\downarrow u}(\mathbf{x}) \;=\; (\psi_1 \otimes \psi_2^{\downarrow u\cap t})(\mathbf{x}) \;=\; \psi_1(\mathbf{x}^{\downarrow s}) \times \psi_2^{\downarrow u\cap t}(\mathbf{x}^{\downarrow u\cap t}).$$

Therefore,

$$\psi_1(\mathbf{x}^{\downarrow s}) \times \psi_2(\mathbf{x}^{\downarrow u\cap t}, \mathbf{y}) \;=\; \psi_1(\mathbf{x}^{\downarrow s}) \times \psi_2^{\downarrow u\cap t}(\mathbf{x}^{\downarrow u\cap t}).$$

We conclude from Lemma 8.6 that

$$\psi_2(\mathbf{x}^{\downarrow u\cap t}, \mathbf{y}) \;=\; \psi_2^{\downarrow u\cap t}(\mathbf{x}^{\downarrow u\cap t})$$

and consequently that $\mathbf{y} \in W_{\psi_2}^{u\cap t}(\mathbf{x}^{\downarrow u\cap t})$. ■

The additional property of strict monotonicity therefore allows us to apply Algorithm 8.4 for the identification of all solution configurations based on the results of a single-query local computation procedure only. But the exclusion of the zero element as optimum value in the root node is crucial. We therefore execute an arbitrary single-query local computation procedure on the factorization $\phi = \phi_1 \otimes \cdots \otimes \phi_n$ and determine the optimum value $\phi^{\downarrow\emptyset}(\diamond)$ in the rot node. If $\phi^{\downarrow\emptyset}(\diamond) = \mathbf{0}$, we know from Lemma 8.9 that all configurations of ϕ are solutions. Hence, there is no need to start the solution construction process. On the other hand, $\phi^{\downarrow\emptyset}(\diamond) \neq \mathbf{0}$ implies that $\phi^{\downarrow\lambda(i)}(\mathbf{x}) \neq \mathbf{0}$ for all $\mathbf{x} \in c_\phi^{\downarrow\lambda(i)}$ and $\lambda(i) \subseteq d(\phi)$. Then,

$$\begin{aligned}\mathbf{0} \;\neq\; \phi^{\downarrow\emptyset}(\diamond) \;=\; \phi^{\downarrow\lambda(i)}(\mathbf{x}) \;&=\; \left(\psi_i^{(r)} \otimes \mu_{ch(i)\rightarrow i}\right)(\mathbf{x}) \\ &=\; \psi_i^{(r)}(\mathbf{x}^{\downarrow\omega_i}) \times \mu_{ch(i)\rightarrow i}(\mathbf{x}^{\downarrow\lambda(i)\cap\lambda(ch(i))}).\end{aligned}$$

This shows that $\phi^{\downarrow\emptyset}(\diamond) \neq \mathbf{0}$ and $\mathbf{x} \in c_\phi^{\downarrow\lambda(i)}$ implies

$$\psi_i^{(r)}(\mathbf{x}^{\downarrow\omega_i}) \neq \mathbf{0} \quad \text{and} \quad \mu_{ch(i)\rightarrow i}(\mathbf{x}^{\downarrow\lambda(i)\cap\lambda(ch(i))}) \neq \mathbf{0}.$$

The requirement of strict monotonicity is therefore always satisfied when Lemma 8.10 is applied in the proof of Theorem 8.3, i.e. in each step of the solution construction algorithm. To sum it up, it is always possible to compute a single solution to an optimization problem based on the results of a single-query local computation scheme. If all solutions are required, we either need the property of strict monotonicity in the underlying valuation algebra, or we must execute a complete run of a multi-query local computation architecture.

8.5 CONCLUSION

Many important valuation algebras have some notion of a solution, characterized by the property that every solution to a projected valuation is also a partial solution to the original valuation. This property allows us to extend partial solutions step-wise to a complete solution. The first part of this chapter presents three approaches to compute solution sets of a factorized valuation. First, we presuppose a completed run of a multi-query local computation procedure, determine the solution set with respect to the root node label and proceed downwards the join tree to buildup the complete solution set. It was shown that these computations take place in the relational algebra of solution sets, such that the solution construction algorithm shapes up as a local computation scheme. This first approach works for all valuation algebras with a suitable notion of solutions, but it has the drawback to depend on a multi-query local computation scheme. Alternatively, if solution sets are non-empty and if they satisfy an additional condition with respect to combination, then it is possible to find a non-empty subset of solutions based on the join tree node contents at the end of a single-query local computation scheme. Finally, if a stronger condition with respect to combination of solution sets holds, then it is possible to find all solutions based on a single-query architecture. In the second part of the chapter we focus on the important application field of constraint reasoning or solving optimization problems. It was shown that such problems arise from totally ordered, idempotent semirings and they always satisfy the weaker condition for solution construction. The stronger condition, that makes the computation of all solutions based on a single-query architecture possible, holds when the semiring is strict monotonic. Further applications of solution construction for solving systems of linear equations will be studied in Chapter 9.

PROBLEM SETS AND EXERCISES

H.1 ★ Describe the solution construction process from Section 8.2.1 in terms of local computation in the relational algebra of solution sets. The messages are given in Equation 8.7. It therefore remains to express Theorem 8.1 in terms of natural join.

H.2 ★ Explore the t-norm semirings of Example 5.8 and Exercise E.6 in Chapter 5 for strict monotonicity.

H.3 ★ Instance 8.5 shows that Bayesian and maximum likelihood decoding establish optimization problems. This application only considers one channel. Identify the optimization problems that occur when multiple channels are connected in series as shown in Figure 8.3 and in parallel as shown in Figure 8.4.

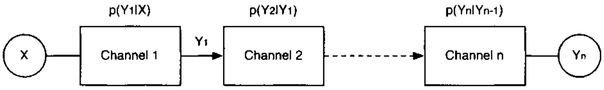


Figure 8.3 A serial connection of unreliable, memoryless channels.

H.4 ★★ Context valuation algebras from Chapter 7 are based on the duality between the sentences of a language and their models. All instances of this family satisfy

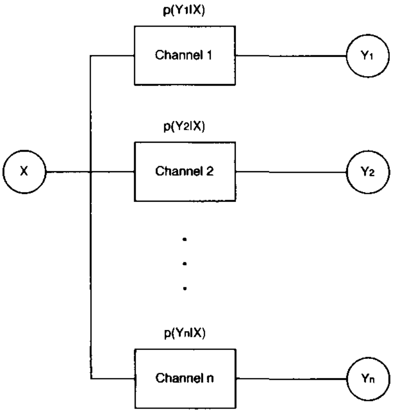


Figure 8.4 A parallel connection of unreliable, memoryless channels.

idempotency and can therefore be processed by the idempotent architecture of Section 4.5. In the introduction to this chapter, we mentioned several examples of context valuations as typical formalisms with some notion of solutions, i.e. solutions in linear equations, inequalities or models in different logics. Show that the set of models associated with a context valuation always satisfies the requirements for a solution set. Moreover, show that the interpretation of solution construction as local computation in the relational algebra of solution sets given in Section 8.2.1 and recessed in Exercise H.1 then complies with the usual distribute phase in the idempotent architecture. Finally, investigate property (8.9) for context valuation algebras.

CHAPTER 9

SPARSE MATRIX TECHNIQUES

We learned in Section 7.2 that the solution spaces of linear equation systems form a valuation algebra and even an information algebra. This hints at an application of local computation for solving systems of linear equations. However, in contrast to many other inference formalisms studied in this book, the valuations here are infinite structures and therefore not directly suitable for computational purposes. On the other hand, the equations themselves are finite constructs and provide a linguistic description of the solution spaces. It is thus reasonable for linear systems to perform the computations not on the valuation algebra directly but on the associated language level. This is the topic of the present chapter. Whereas solving systems of linear equations is an old and classical subject of numerical mathematics, looking at it from the viewpoint of local computation is new and leads to simple and clear insights especially for computing with *sparse matrices*. It is often the case that the matrix of a system of linear equations contains many zeros. Such matrices are called sparse, and this sparsity can be exploited to economize memory and computing time. However, it is well-known that zero entries may get lost during the solution process if care is not taken. Matrix entries that change from a zero to a non-zero value are called *fill-ins* and they destroy the advantages of sparsity. Therefore, much effort has been spent on developing method that maintain sparsity. We claim that local computation

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

offers a very simple and easy to understand method for controlling fill-ins and thus for maintaining sparsity. This is not only true for ordinary, real-valued linear equations, or linear equations over a field, but also for linear equations over semirings. We have seen in Chapter 6 that path problems induce semiring fixpoint equation systems, where the sparsity often comes from an underlying graph. Independently, local computation maintains the sparsity in all these applications.

An interesting aspect that arises from studying equation systems in the context of valuation algebras is the necessity to distinguish between *factorizations* and *decompositions*. The generic local computation methods from Chapter 3 and 4 solve inference problems that consist of a knowledgebase of valuations which factor the objective function. Where do these valuations come from? A key note behind the theory of valuation algebras is that information exists in pieces and comes from different sources, which indicates that factorizations occur naturally. This is for example the case in the semiring valuation systems of Chapter 5, where factorizations are often the only mean to express the objective function, due to the exponential complexity behind these formalisms. In contrast, linear systems are polynomial, and it is often more realistic that a total linear system exists, from which the knowledgebase must be artificially fabricated. We then speak about a decomposition rather than a factorization. We will continue the discussion of factorizations and decompositions throughout this chapter and show that both perspectives are equally important when dealing with linear systems.

In the beginning two sections of this chapter, we treat ordinary systems of linear equations by examining arbitrary systems in Section 9.1 and the important case of systems with symmetric, positive definite matrices in Section 9.2. As an application, the method of *least squares* is shown to fit into the local computation framework. Formally, this is very similar to the problems that we are going to treat in Chapter 10; only the semantics of the two systems are different. This uncovers a first application of the valuation algebra from Instance 1.6 in Chapter 1. The remaining sections are devoted to the solution of linear fixpoint equation systems over semirings. Section 9.3 focuses on the local computation based solution of arbitrary fixpoint equation systems with values from quasi-regular semirings. The application of this theory to path problems over Kleene algebras is compiled in Section 9.3.3.

9.1 SYSTEMS OF LINEAR EQUATIONS

We first give a short review of Gaussian variable elimination.

9.1.1 Gaussian Variable Elimination

Consider a system of linear equations with real-valued coefficients,

$$\begin{array}{ccccccccc} a_{1,1}X_1 & + & a_{1,2}X_2 & + & \cdots & + & a_{1,n}X_n & = & b_1, \\ a_{2,1}X_1 & + & a_{2,2}X_2 & + & \cdots & + & a_{2,n}X_n & = & b_2, \\ & & & & & & & \vdots & \\ a_{m,1}X_1 & + & a_{m,2}X_2 & + & \cdots & + & a_{m,n}X_n & = & b_m. \end{array} \tag{9.1}$$

We make no assumptions about the number m of equations or the number n of unknowns and neither about the rank of the matrix of the system. So this system of linear equations may have no solution, exactly one solution or infinitely many solutions. In any case, the solutions form an affine space as explained in Section 7.2. The computational task is to decide whether the system has a solution or not. If it has exactly one solution, then this solution must be determined. If it has infinitely many solutions, then the solution space must be determined. The usual way to solve systems of linear equations is by *Gaussian variable elimination.* Assuming that the element $a_{1,1}$ is different from zero, an elimination step based on $a_{1,1}$ proceeds by first solving the first equation for X_1 in terms of the remaining variables,

$$X_1 \quad = \quad \frac{b_1}{a_{1,1}} - \frac{a_{1,2}}{a_{1,1}}X_2 - \cdots - \frac{a_{1,n}}{a_{1,1}}X_n. \tag{9.2}$$

Then, the variable X_1 is replaced by the right-hand expression in all other equations. After rearranging terms this results in the new system of linear equations

$$\begin{array}{ccccccc} (a_{2,2} - \frac{a_{2,1}a_{1,2}}{a_{1,1}})X_2 & + & \cdots & + & (a_{2,n} - \frac{a_{2,1}a_{1,n}}{a_{1,1}})X_n & = & b_2 - \frac{a_{2,1}b_1}{a_{1,1}}, \\ & & & & & \vdots & \\ (a_{m,2} - \frac{a_{m,1}a_{1,2}}{a_{1,1}})X_2 & + & \cdots & + & (a_{m,n} - \frac{a_{m,1}a_{1,n}}{a_{1,1}})X_n & = & b_m - \frac{a_{m,1}b_1}{a_{1,1}}. \end{array}$$

This is called a *pivoting step* with $a_{1,1}$ as *pivot element.* Variable X_1 is called the *pivot variable* and the first equation, where the pivot element is selected from, is called the *pivot equation.* It eliminates variable X_1 and reduces the system of equations to a new system with one less variable and one less equation. The new system is equivalent to the old one in the following sense:

1. If $(x_1, x_2, \ldots, x_n)$ is a solution to the original system, then $(x_2, \ldots, x_n)$ is a solution of the new system.

2. If $(x_2, \ldots, x_n)$ is a solution of the new system, then $(x_1, x_2, \ldots, x_n)$ with

$$x_1 \quad = \quad \frac{b_1}{a_{1,1}} - \frac{a_{1,2}}{a_{1,1}}x_2 - \cdots - \frac{a_{1,n}}{a_{1,1}}x_n$$

   is a solution to the original system.

This procedure is called Gaussian variable elimination and can be repeated on the new system. In order to execute a pivot step on a variable X_i there must be at least one equation with a non-zero coefficient of X_i. If there is no such equation, X_i can be eliminated from the system without further actions. In fact, in this case X_i has zero coefficients in all equations which is tantamount to say that X_i does not appear in the system. So, Gaussian elimination permits to reduce the system stepwise by eliminating one variable after the other until only one variable remains. Several cases may arise during the elimination of variables: assume that the variables X_1 to X_p have been successfully eliminated. Then, the following three cases may arise:

1. All coefficients on the left-hand side of the new system vanish, but on the right-hand side there is at least one element different from zero. In this case the system has no solution.

2. All coefficients on both sides of an equation vanish. Then, this equation can be eliminated. If no equations remain after elimination, then the variables X_{p+1} to X_n can take arbitrary values.

3. There remains a non-vanishing system of linear equations with at least one equation less than the system before pivoting on X_p. Then, a next pivot step can be executed.

The first case identifies contradictions in the system. For this, we do not necessarily need to continue the pivoting process until all coefficients on the left-hand side of the system vanish. Instead, we may stop the process when the first equation arises, where all coefficient on the left-hand side vanish, but the right-hand side value is different from zero. These three cases solve the linear system in the following sense:

1. If case 1 occurs, the system has no solution.

2. In case 2 a certain number of variables $X_1, \ldots, X_p$ can be expressed linearly by the remaining variables $X_{p+1}, \ldots, X_n$. This determines the solution space.

3. All variables up to and including X_{n-1} can be eliminated. Then X_n gets a fixed value in the last system, and a unique solution exists for the system. It can be found by *backward substitution* of the values of $X_n, X_{n-1}, \ldots$

Note that which of the above cases arises does not depend on the order in which variables are eliminated.

The equations in the system after the first pivot step clearly show that original zero coefficients of certain variables may easily become non-zero in a pivot step. This is called a *fill-in*. The number of fill-ins depends on the choice of the pivot element or, in other words, there may be many fill-ins if the pivot element is not carefully chosen. Selection of the pivot element is partially a question of selecting elimination sequences of variables, but also of selecting the pivot equation. In the next section, we show how local computation allows to fix an elimination sequence such that fill-ins can be controlled and bounded. It should, however, be emphasized that limiting fill-ins is not the only consideration in selecting pivot elements. Numerical stability and the control of numerical errors are other important aspects that may be in contradiction to the minimization of fill-ins. The reader should keep this aspect in mind, although we here only focus on the limitation of fill-ins.

9.1.2 Fill-ins and Local Computation

To discuss a concrete scenario we assume that in our system of linear equations (9.1) we have $a_{j,i} = 0$ for all $j = 1,\ldots,r$ and $i = h,\ldots,n$ and also for $j = r+1,\ldots,m$ and $i = 1,\ldots,k$ for some $r < m$ and $k < h < n$. This corresponds to a decomposition of the system into a first subsystem of r equations, where only the variables X_1 to X_h occur with non-zero coefficients and a second subsystem of $m-r$ equations where only the variables X_{k+1} to X_n occur. The corresponding matrix decomposition is illustrated in Figure 9.1. We denote by **A** the $r \times h$ matrix of the first subsystem for the variables X_1 to X_h and by **B** the $(m-r) \times (n-k)$ matrix of the second subsystem for the variables X_{k+1} to X_n. It is already clear that if we select the pivot elements for the variables X_1 to X_k in the first subsystem, then pivoting does not change the second subsystem, and in the first subsystem only the coefficients of the variables X_1 to X_h change. So the zero-elements outside the two subsystems are maintained. This is how fill-ins are controlled by a decomposition.

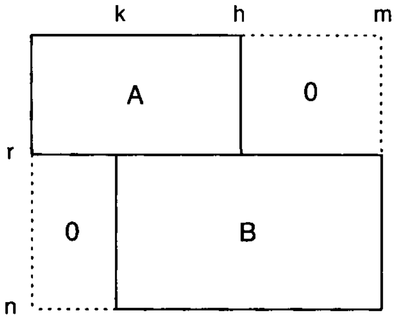


Figure 9.1 A first decomposition of a system of linear equations into two subsystems and the associated zero-pattern of the coefficients.

If we want to eliminate the variables X_1 to X_k, we have to distinguish two cases:

1. First, assume that $r < k$, i.e. there are less equations than variables to be eliminated. Then, we may eliminate at most r variables. Let us eliminate the variables in the order of their numbering $X_1,\ldots,X_r$. Then, it either turns out that the first subsystem has no solution, which implies that the whole system has no solution, or $X_1,\ldots,X_r$ are at the end of the elimination process linearly expressed by the remaining variables $X_{r+1},\ldots,X_h$ of the subsystem. We have for $i = 1,\ldots,r$,

$$X_i \quad = \quad c_i + c_{i,r+1}X_{r+1} + \cdots + c_{i,k}X_k + c_{i,k+1}X_{k+1} + \cdots c_{i,h}X_h.$$

Here, the first variables X_{r+1} to X_k can freely be chosen because they do not appear in the remaining second part of the system, whereas the last variables X_{k+1} to X_h are determined by the second part of the system which can be solved independently of the first part.

2. In the second case, if $k \leq r$, the elimination of the variables $X_1, X_2, \ldots$ may already show that the system has no solution. Otherwise, the variables X_1 to X_k are eliminated from at least k equations of the first subsystem and there remain at most $r - k$ equations containing only the variables X_{k+1} to X_h. This system is added to the second subsystem which still contains only the variables X_{k+1} to X_n. We thus obtain a new linear system for the variables X_{k+1} to X_n. The eliminated variables $X_1, \ldots, X_k$ are linearly expressed by the variables $X_{k+1}, \ldots, X_h$. Once the second subsystem is solved for $X_{k+1}, \ldots, X_n$ we can backward substitute the solution into the expressions for $X_1, \ldots, X_k$.

This describes a simple local computation scheme: the decomposition of the system can be represented by the two-node join tree of Figure 9.2. The variables of the first subsystem are covered by the label of the left-hand node and the variables of the second subsystem by the label of the right-hand node. The message sent from the left to the right node is obtained from eliminating all variables outside the intersection of the node labels. These are the variables X_1 to X_k. The message is either empty or consists of the remaining system of the first subsystem after eliminating the variables X_1 to X_k. If the elimination of the variables in the first subsystem already shows that the total system has no solution, then no message needs to be sent. Otherwise, the arriving message is simply added to the subsystem of the receiving node. Then, the variable elimination is continued in the new system until either a solution is found in the sense of the previous subsection, or it is seen that no solution exists. In the first case, assume that the variables X_{k+1} to X_l are expressed by the remaining variables $X_{l+1}, \ldots, X_n$. Note that since we did not make any assumption about the rank of the matrix, we cannot be sure that all this holds for all variables X_{k+1} to X_h, but only for some $l \leq h$. Then, the expression of the variables X_{k+1} to X_l may be backward substituted into the expressions of the variables X_1 to X_r if $r < k$, or to X_k obtained in the process of variable elimination in the first subsystem.

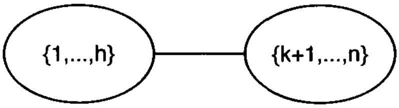


Figure 9.2 The join tree corresponding to the decomposition of Figure 9.1.

Let us point out more explicitly how the above scheme is connected to the local computation framework. We know from Section 7.2 that solution spaces of linear systems form a valuation algebra. Sets of equations provide a formal description of solution spaces and can therefore be considered as a representation of the latter. The domain of a set of equations consists of all variables that have at least one non-zero coefficient in this system. In the above description, we combined sets of equations by simple union. The original system therefore corresponds to the objective function and the knowledgebase factors are subsets of equations, whose union builds up the total system. This corresponds to the decomposition view presented in the introduction of this chapter. When producing such a decomposition from an existing system, we should keep the factor domains as small as possible. Otherwise, we will need large join tree nodes to cover these factors, which results in a bad complexity. In the case at hand, we can provide a factorization with minimum granularity where each factor contains exactly one equation. Then, the above procedure can be generalized as follows: we assume a covering join tree (V, E, λ, D) for the knowledgebase where each node $i \in V$ contains a subsystem of linear equations whose variables are covered by the node label $\lambda(i)$. This join tree decomposition reflects a certain *pattern of zeros* in the original total system. Note also that these systems may be empty on certain nodes, which mirrors the assignment of identity elements in Section 3.6. If we then execute the collect algorithm, the messages correspond to the description above. At the end of the message-passing, the root node contains the total system from which all variables outside the root label have been eliminated. We determine the remaining variables from this system and, if it exists, build the total solution by backward substitution. The depicted variable elimination process clearly maintains the pattern of zeros represented by the join tree decomposition. Consequently, join tree decompositions control fill-ins in general sparse systems of linear equations. This process will be formulated more precisely in Section 9.1.3 for the important case of regular systems that always provide a unique solution. There, we also point out that backward substitution in fact corresponds to the construction of solution configurations according to Section 8.2.1, see also Exercise H.1.

To identify the complexity of this approach, we consider a system that gives the most of work, i.e. a regular system of n variables and n equations. Eliminating one variable from this system takes a time complexity of $\mathcal{O}(n^2)$ and eliminating all variables is thus possible in $\mathcal{O}(n^3)$. Backward substitution takes linear time for one variable and is repeated n times which result in $\mathcal{O}(n^2)$. Altogether, the time complexity of Gaussian elimination without taking care of fill-ins is $\mathcal{O}(n^3)$. If local computation is applied, each join tree node contains a system of at most $\omega^* + 1$ variables, where ω^* denotes the treewidth of the inference problem derived from the total system. The effort of eliminating one variable is $\mathcal{O}((\omega^* + 1)^2)$, and a complete run of the collect algorithm that eliminates all variables takes

$$\mathcal{O}\Big(n \cdot (\omega^* + 1)^2\Big). \tag{9.3}$$

Backward substitution is performed n times with the equations that result from the variable elimination process and which contain at most ω^* variables. We therefore obtain a time complexity of $\mathcal{O}(n\omega^*)$ for the complete backward substitution process. Because each node stores a matrix whose domain is bounded by the treewidth, we obtain the same bound as (9.3) for the space complexity. If the treewidth is small with respect to n, then big savings may be expected.

9.1.3 Regular Systems

In this section we examine more closely and more precisely the local computation scheme and the related structure of fill-ins for regular systems $\mathbf{AX} = \mathbf{b}$, where $\mathbf{A}$ is a *regular* $n \times n$ matrix. This assumption means that there is a unique solution to this system. In particular, we look at the fusion algorithm of Section 3.2.1 for computing the solution to this system. We choose an arbitrary elimination sequence for the n variables. By renumbering the variables and applying corresponding permutations of the columns of the matrix $\mathbf{A}$, we may without loss of generality assume that the elimination sequence is $(X_1, X_2, \ldots, X_n)$. Further, by a well-known theorem of linear numerical analysis, it is also always possible to permute the rows of the matrix $\mathbf{A}$ such that in the sequence of the elimination of the variables $X_1, \ldots, X_n$ all diagonal elements are different form zero (Schwarz, 1997). Again, without loss of generality, we may assume that the rows of $\mathbf{A}$ are already in the required order. This means that the following process of variable elimination works fine.

In the first step variable X_1 is eliminated. Since by assumption $a_{1,1} \neq 0$ we may solve the first equation for X_1

$$X_1 = c_1 - \sum_{i=2}^{n} r_{1,i} X_i,$$

where

$$c_1 = \frac{b_1}{a_{1,1}} \quad \text{and} \quad r_{1,i} = \frac{a_{1,i}}{a_{1,1}} \tag{9.4}$$

for $i = 2, \ldots, n$. Using this linear expression, we replace the variable X_1 in the remaining equations to obtain

$$\sum_{i=2}^{n} (a_{j,i} - l_{j,1} a_{1,i}) X_i = b_j - l_{j,1} b_1$$

for $j = 2, \ldots, n$ with

$$l_{j,1} = \frac{a_{j,1}}{a_{1,1}}. \tag{9.5}$$

Let us define

$$a_{j,i}^{(1)} = a_{j,i} - l_{j,1} a_{1,i} \quad \text{and} \quad b_j^{(1)} = b_j - l_{j,1} b_1$$

for $j, i = 2, \ldots, n$. Then, the new system can be written as

$$\sum_{i=2}^{n} a_{j,i}^{(1)} X_i = b_j^{(1)}.$$

By assumption $a_{2,2}^{(1)}$ is different from zero, variable X_2 can be eliminated by solving the first equation of the new system and so forth. In the k-th step of this elimination process, $k = 1, \ldots, n$, we similarly have

$$X_k \quad = \quad c_k - \sum_{i=k+1}^{n} r_{k,i} X_i, \tag{9.6}$$

where

$$c_k \;=\; \frac{b_k^{(k-1)}}{a_{k,k}^{(k-1)}} \quad \text{and} \quad r_{k,i} \;=\; \frac{a_{k,i}^{(k-1)}}{a_{k,k}^{(k-1)}} \tag{9.7}$$

for $i = k+1, \ldots, n$. Further, with

$$a_{j,i}^{(k)} \;=\; a_{j,i}^{(k-1)} - l_{j,k} a_{k,i}^{(k-1)} \quad \text{and} \quad b_j^{(k)} \;=\; b_j^{(k-1)} - l_{j,k} b_k^{(k-1)} \tag{9.8}$$

for $j, i = k+1, \ldots, n$, where

$$l_{j,k} \quad = \quad \frac{a_{j,k}^{(k-1)}}{a_{k,k}^{(k-1)}} \tag{9.9}$$

we get the system

$$\sum_{i=k+1}^{n} a_{j,i}^{(k)} X_i \quad = \quad b_j^{(k)}.$$

In this process we define $a_{j,i}^{(0)} = a_{j,i}$. For $k = n-1$ it remains a simple system

$$a_{n,n}^{(n-1)} X_n \quad = \quad b_n^{(n-1)}$$

with one unknown that can easily be solved. By backward substitution for $k = n-2, \ldots, 1$, using equation (9.6), we obtain the solution of the regular system.

In order to study the control of fill-ins by local computation, we consider the join tree induced by the above execution of the fusion algorithm as described in Section 3.2.2. To each eliminated variable X_i corresponds a node $i \in V$ in the join tree (V, E, λ, D) with a certain label $\lambda(i) \in D = \mathcal{P}(\{1, \ldots, n\})$. Let us now interpret the fusion algorithm as a message-passing scheme in this join tree according to Section 3.5. To start with, consider node 1 where the variable X_1 is eliminated. Let

$$\epsilon(1) \quad = \quad \{j : a_{j,1} \neq 0\}$$

define the index set of all equations which contain the variable X_1. Only these equations are changed when variable X_1 is eliminated. Further, let

$$\lambda(1) \quad = \quad \{i : a_{j,i} \neq 0 \text{ for some } j \in \epsilon(1)\}$$

be the set of variables that occur in the subset of equations with indices in $\epsilon(1)$. Note that the first equation is in $\epsilon(1)$, since by assumption $a_{1,1} \neq 0$. Hence, also the variable X_1 is in the label $\lambda(1)$ of node 1. When we now look at the elimination process of variable X_1 according to the equations (9.4) and (9.5) above, then we remark that $r_{1,i} = 0$ for $i \notin \lambda(1)$ and $l_{j,1} = 0$ for $j \notin \epsilon(1)$ and also

$$a_{j,i}^{(1)} = a_{j,i}^{(0)} \text{ and } b_j^{(1)} = b_j^{(0)} \text{ for } j \notin \epsilon(1).$$

This shows that zeros in the first row and the first column of $\mathbf{A}$ are maintained when eliminating variable X_1. Further, it exhibits the locality of the process by showing that only equations which contain variable X_1 will change. The message sent to the child node $ch(1)$ of node 1 in the join tree is determined by the coefficients

$$a_{j,i}^{(1)} = a_{j,i}^{(0)} - l_{j,1}a_{1,i}^{(0)} \quad \text{and} \quad b_j^{(1)} = b_j^{(0)} - l_{j,1}b_1^{(0)}$$

for $j \in \epsilon(1) - \{1\}$ and $i \in \lambda(1) - \{1\}$. These coefficients specify the new system of equations after elimination of variable X_1. This process is repeated for variables X_k and nodes k for $k = 1, \ldots, n-1$. For node k of the join tree, where the variable X_k is eliminated, we define as above

$$\epsilon(k) = \{j : a_{j,k}^{(k-1)} \neq 0\}$$

for the equations containing variable X_k after $k-1$ elimination steps and

$$\lambda(k) = \{i : a_{j,i}^{(k-1)} \neq 0 \text{ for some } j \in \epsilon(k)\}$$

for the variables contained in these equations. Again, by assumption, $a_{k,k}^{(k-1)} \neq 0$. Hence, the k-th equation belongs to $\epsilon(k)$ and the variable X_k belongs to $\lambda(k)$. So, we may eliminate variable X_k from the k-th equation. Similar to the first step, we remark that $r_{k,i} = 0$ for $i \notin \lambda(k)$ and $l_{j,k} = 0$ for $j \notin \epsilon(k)$ and also

$$a_{j,i}^{(k)} = a_{j,i}^{(k-1)} \text{ and } b_j^{(k)} = b_j^{(k-1)} \text{ for } j \notin \epsilon(k) \text{ and } i = k, \ldots, n.$$

The message sent from node k to its child $ch(k)$ is given by the coefficients

$$a_{j,i}^{(k)} = a_{j,i}^{(k-1)} - l_{j,k}a_{k,i}^{(k-1)} \quad \text{and} \quad b_j^{(k)} = b_j^{(k-1)} - l_{j,k}b_k^{(k-1)}$$

for $j \in \epsilon(k) - \{k\}$ and $i \in \lambda(k) - \{k\}$. The process stops at node n. This discussion permits to clarify how the zero-pattern of the matrix $\mathbf{A}$ is controlled and maintained by this fusion algorithm.

Lemma 9.1 *The k-th equation of the system $\mathbf{AX} = \mathbf{b}$ is covered by some node $i \leq k$ of the join tree (V, E, λ, D). More precisely, there exists an index $i \leq k$ such that*

$$\{h : a_{k,h} \neq 0\} \subseteq \lambda(i).$$

Proof: Let i be the least index of the variables occurring in the k-the equation. Since the k-th equation contains no variables with indices $h < i$, we conclude that

$k \notin \epsilon(h)$ for $h \leq i$ and hence $a_{k,l}^{(i)} = a_{k,l}$ for $l = 1, \ldots, n$ and $a_{k,i}^{(i)} = a_{k,i} \neq 0$. So, the k-th equation belongs to $\epsilon(i)$ and is therefore covered by $\lambda(i)$. ■

This result allows us to assign each equation k to some node $i \leq k$ of the join tree, and we have $a_{k,h} = 0$ for $h \notin \lambda(i)$. This zero-pattern is maintained through the variable elimination process due to the remarks of the analysis above, i.e. $a_{k,h}^{(l)} = 0$ for $h \notin \lambda(i)$ and $l \leq i$.

To complete the description of the fusion algorithm in the case of a regular system of linear equations, we summarize it as a message-passing scheme: The system of equations assigned to the node k of the join tree is denoted by $\tilde{\psi}_k$ for $k = 1, \ldots, n$. Some of these systems may be empty. A system $\tilde{\psi}_k$ determines the affine space ψ_k of its solutions. If the system is empty, the affine space consists of all possible vectors. In other words, ψ_k is then the neutral valuation in the information algebra of affine spaces. The problem of finding the solution to the system $\mathbf{AX} = \mathbf{b}$ can then be written for $k = 1, \ldots, n$ as the inference problem

$$\phi^{\downarrow\{X_k\}} = (\psi_1 \otimes \cdots \otimes \psi_n)^{\downarrow\{X_k\}}.$$

Given the uniqueness of the solution, each such projection is just a number.

Now, the fusion algorithm is described in terms of the information elements ψ_k as follows: at step 1, node 1 sends the message

$$\mu_{1\rightarrow ch(1)} = \psi_1^{-X_1}$$

to its child node $ch(1)$. This is done by transforming the systems of equations $\tilde{\psi}_1$ according to the above procedure, i.e. by solving the first equation with respect to X_1 and replacing the variable X_1 in the remaining equations of $\tilde{\psi}_1$. The resulting system is denoted by $\tilde{\mu}_{1\rightarrow ch(1)}$ and represents the message $\mu_{1\rightarrow ch(1)}$. The message is combined to the content of node $ch(1)$ to get

$$\psi_{ch(1)}^{(2)} = \psi_{ch(1)} \otimes \mu_{1\rightarrow ch(1)}.$$

This new system is simply represented by

$$\tilde{\psi}_{ch(1)}^{(2)} = \tilde{\psi}_{ch(1)} \cup \tilde{\mu}_{1\rightarrow ch(1)}.$$

More generally, let $\psi_i^{(k)}$ be the information element on node i before step k of the fusion algorithm. We therefore have $\psi_i^{(1)} = \psi_i$. The associated system of equations is denoted by $\tilde{\psi}_i^{(k)}$. At step k, node k sends the message

$$\mu_{k\rightarrow ch(k)} = \left(\psi_k^{(k)}\right)^{-X_k}$$

to node $ch(k)$ by solving the first equation of the system $\tilde{\psi}_k^{(k)}$ with respect to X_k and replacing the variable X_k in the remaining equations, which gives the system

$\tilde{\mu}_{k\rightarrow ch(k)}$ representing the message. This message is added to the system in the child node $ch(k)$ to yield

$$\tilde{\psi}_{ch(k)}^{(k+1)} \quad = \quad \tilde{\psi}_{ch(k)}^{(k)} \cup \tilde{\mu}_{k\rightarrow ch(k)}.$$

All other nodes remain unchanged, i.e. $\tilde{\psi}_i^{(k+1)} = \tilde{\psi}_i^{(k)}$ for $i \neq ch(k)$. This process is repeated for $k = 1, \ldots, n-1$. At the end, the root node n contains the valuation $\phi^{\downarrow\{X_n\}}$ which is represented by the system

$$a_{n,n}^{(n-1)} X_n \quad = \quad b_n^{(n-1)}. \tag{9.10}$$

This is the end of the fusion algorithm interpreted as message-passing scheme, or equivalently, of the collect algorithm as described in Section 3.8, executed on the particular join tree that is induced by the fusion algorithm.

Since we are dealing with an idempotent valuation algebra, we can use the corresponding architecture from Section 4.5 for the distribute phase to compute the complete solution $\phi^{\downarrow\{X_i\}}$ for all $i = 1, \ldots, n$. We first transform equation (9.10) on the node n into the equivalent form

$$X_n \quad = \quad \frac{b_n^{(n-1)}}{a_{n.n}^{(n-1)}}, \tag{9.11}$$

from which we determine the solution value

$$x_n \quad = \quad \frac{b_n^{(n-1)}}{a_{n,n}^{(n-1)}}.$$

This in fact defines the single-point affine space $\phi^{\downarrow\{X_n\}} = \{x_n\}$. Observe now that if $n-1$ is a neighbor to node n such that $ch(n-1) = n$, it follows by the construction of the join tree from the fusion algorithm that $\lambda(n-1) \cap \lambda(n) = \lambda(n-1) - \{X_{n-1}\} = \{X_n\}$. So, the message sent from node n to node $n-1$ in the distribute phase is

$$\phi^{\downarrow\lambda(n-1)\cap\lambda(n)} \quad = \quad \phi^{\downarrow\{X_n\}}.$$

This means that equation (9.11) is added to the system $\tilde{\psi}_{n-1}^{(n-1)}$ containing the equation

$$X_{n-1} \quad = \quad c_{n-1} - r_{n-1,n} X_n, \tag{9.12}$$

besides possibly a second system that represents the message from node $n-1$ to n in the collect phase. This second system is redundant with the equation (9.11) and can be eliminated. Finally, the solution x_n from equation (9.11) can be introduced into (9.12) to yield the solution value for X_{n-1} determining the affine space $\phi^{\downarrow\{X_{n-1}\}}$. In the general step of the distribute phase, node k receives a message

$$\phi^{\downarrow\lambda(k)\cap\lambda(ch(k))} \quad = \quad \phi^{\downarrow\lambda(k)-\{k\}}$$

from its child node $ch(k)$. By the induction assumption this message is a single-point affine space given by the solution values x_i for $i \in \lambda(k) - \{k\}$. The system of equations on node k contains the equation

$$X_k \quad = \quad c_k - \sum_{i \in \lambda(k)-\{k\}} r_{k,i} X_i.$$

The solution values contained in the message are introduced into this equation to get the solution value for X_k,

$$x_k \quad = \quad c_k - \sum_{i \in \lambda(k)-\{k\}} r_{k,i} x_i.$$

So, by induction, distribute corresponds to the construction of the solution values of the variables X_k on the nodes k of the join tree by backward substitution. This is in fact a consequence of the theory of solution construction developed in Chapter 8 applied to context valuation algebras. In Exercise H.4 we proposed to define solution sets in context valuation algebras by their associated set of models. The computations in the relational algebra of solution sets performed by the solution construction algorithms of Section 8.2 therefore correspond to computations in the context valuation algebra itself. Hence, the above execution of the idempotent distribute phase coincides with the application of the generic solution construction procedure from Section 8.2.3. We continue this discussion in Section 9.2 and Section 9.3.2, where other valuation algebra are studied that do not belong to the class of context valuation algebras. There, the solution construction procedures will be applied explicitly.

This completes the picture of the solution process as a message-passing procedure on the join tree induced by the fusion algorithm. In particular, the discussion shows that locality as represented by the join tree is also maintained during the distribute phase. It is important to understand that a similar process can be executed on an arbitrary join tree, where $\lambda(k) \cap \lambda(ch(k)) = \lambda(k) - \{X_k\}$ does not necessarily hold. Here, we have chosen the very particular join tree of the fusion algorithm for illustration purposes, i.e. exactly one variable is eliminated from the equation system between two neighboring join tree nodes. It follows a numerical example:

Example 9.1 *Consider a regular* 4×4 *matrix* $\mathbf{A}$ *and a corresponding vector* $\mathbf{b}$,

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 1 \end{bmatrix} \quad \textit{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 6 \end{bmatrix}.$$

We solve the linear system $\mathbf{AX} = \mathbf{b}$ *by elimination of the variables* X_1, X_2 *and* X_3 *in this order. Eliminating* X_1 *yields the following new matrix* $\mathbf{A}^{(1)}$ *and the new right-hand side vector* $\mathbf{b}^{(1)}$*:*

$$\mathbf{A}^{(1)} = \begin{bmatrix} -1 & 1 & 0 \\ 2 & 1 & 1 \\ 0 & 2 & 1 \end{bmatrix} \quad \textit{and} \quad \mathbf{b}^{(1)} = \begin{bmatrix} -1 \\ 4 \\ 6 \end{bmatrix}.$$

We remember also the non-zero r- and l-elements $r_{1,2} = 1$ and $l_{2,1} = 2$. Next, we eliminate X_2 and obtain

$$\mathbf{A}^{(2)} = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix} \quad \textit{and} \quad \mathbf{b}^{(2)} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}.$$

In addition we have $r_{2,3} = -1$ and $l_{3,2} = -2$. It remains to eliminate X_3 giving

$$\mathbf{A}^{(3)} = \begin{bmatrix} 1/3 \end{bmatrix} \quad \textit{and} \quad \mathbf{b}^{(2)} = \begin{bmatrix} 14/3 \end{bmatrix}$$

with $l_{4,3} = \frac{2}{3}$ and $r_{3,4} = \frac{1}{3}$. We can now solve the equation

$$\frac{1}{3}X_4 = \frac{14}{3}$$

for X_4 and obtain the solution value $x_4 = 14$.

Figure 9.3 shows the join tree induced by the fusion algorithm described above. We may assign the first two equations to the left-most node and the last two equations to the second node from the left. Eliminating the variable X_1 generates the equation

$$-X_2 + X_3 = -1$$

as the message sent to the next node. There, this equation is added to the two equations already hold by this node. This yields the linear system defined by the matrix $\mathbf{A}^{(1)}$ and the vector $\mathbf{b}^{(1)}$. Next, eliminating the variable X_2 generates the system described by the matrix $\mathbf{A}^{(2)}$ and the vector $\mathbf{b}^{(2)}$ as the message to be sent from the second to the third node. Since the system contained in the third node is empty, the node content becomes equal to the received message. Finally, eliminating the variable X_3 generates the equation $(1/3)X_4 = 14/3$ as the message sent to the fourth node. Again, the content of node 4 is an empty system such that the received message directly becomes the new node content.

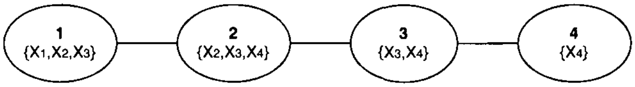


Figure 9.3 The join tree induced by the variable elimination in Example 9.1.

Now, the distribute phase starts by solving the equation of node 4, which gives us the solution value $x_4 = 14$ as remarked above. Node 4 sends this solution value back to node 3, where it can substitute the variable X_4. Using the first equation of the system defined by $\mathbf{A}^{(2)}$ and $\mathbf{b}^{(2)}$ on this node, we can solve for X_3 to obtain $x_3 = 2/3 - (1/3)x_4 = -4$. This is the value sent back to node 2, where it can be used to determine the solution value of X_2 using the first equation of the system on this node, $x_2 = 1 + x_3 = -3$. Finally, this values is sent back to node 1, where we get from the first equation $x_1 = 2 - x_2 = 5$. This constitutes the totality of the unique *solution of the system* $\mathbf{AX} = \mathbf{b}$. *Note that in the whole process, at each step only the variables belonging to the label of the current join tree node are involved. This shows the locality of the solution process.*

In the next section we study a variant of this process, which has advantages if the system of equations $\mathbf{AX} = \mathbf{b}$ must be solved for different vectors $\mathbf{b}$. By memorizing certain intermediate factors derived from the matrix $\mathbf{A}$ in the first run of the above algorithm, we obtain a factorization of $\mathbf{A}$ into an upper and a lower triangular matrix. If later another equation system with the same matrix must be solved, only the computations with respect to the new vector must be repeated. This considerably simplifies the computations of the involved local computation messages.

9.1.4 LDR-Decomposition

Let us review the process of Gaussian variable elimination described in Section 9.1.3. From the recursive definition in equation (9.8) we derive

$$\begin{aligned} a_{k,i}^{(k-1)} &= a_{k,i} - l_{k,1}a_{1,i}^{(0)} - l_{k,2}a_{2,i}^{(1)} - \cdots - l_{k,k-1}a_{k-1,i}^{(k-2)} \\ &= a_{k,i} - l_{k,1}v_{1,i} - l_{k,2}v_{2,i} - \cdots - l_{k,k-1}v_{k-1,i} \end{aligned}$$

for $1 \leq k \leq i$, if we define $v_{k,i} = a_{k,i}^{(k-1)}$. This implies that

$$a_{k,i} = \sum_{j=1}^{k-1} l_{k,j}v_{j,i} + v_{k,i}.$$

If we set $l_{k,k} = 1$, we may also write

$$a_{k,i} = \sum_{j=1}^{k} l_{k,j}v_{j,i}.$$

This can also be expressed by the matrix product

$$\mathbf{A} = \mathbf{LV},$$

where $\mathbf{L}$ is a lower triangular und $\mathbf{R}$ an upper triangular matrix,

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{2,1} & 1 & 0 & \dots & 0 \\ l_{3,1} & l_{3,2} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & 1 \end{bmatrix}$$

and

$$\mathbf{V} = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & \dots & v_{1,n} \\ 0 & v_{2,2} & v_{2,3} & \dots & v_{2,n} \\ 0 & 0 & v_{3,3} & \dots & v_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & v_{n,n} \end{bmatrix}.$$

This representation is called $\mathbf{LV}$-*decomposition* of the matrix $\mathbf{A}$. We remark that the $v_{k,i}$ used here are closely related to the elements $r_{k,i}$ in equation (9.7). We have

$$r_{k,i} \quad = \quad \frac{a_{k,i}^{(k-1)}}{a_{k,k}^{(k-1)}} \quad = \quad \frac{v_{k,i}}{v_{k,k}}.$$

Therefore, if we define the diagonal matrix $\mathbf{D}$ with elements $d_{k,k} = 1/v_{k,k}$ on the diagonal and zeros otherwise, and further the upper triangular matrix

$$\mathbf{R} \quad = \quad \begin{bmatrix} 1 & r_{1,2} & r_{1,3} & \dots & r_{1,n} \\ 0 & 1 & r_{2,3} & \dots & r_{2,n} \\ 0 & 0 & 1 & \dots & r_{3,n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix},$$

then we have the representation

$$\mathbf{A} \quad = \quad \mathbf{LDR}.$$

This is called $\mathbf{LDR}$-*decomposition* of the matrix $\mathbf{A}$.

We may neglect the right-hand side of the system $\mathbf{AX} = \mathbf{b}$ in the fusion algorithm described in the previous Section 9.1.3 and only compute the elements $l_{k,i}$, $r_{k,i}$ and $a_{k,k}^{(k-1)}$ that depend on the matrix $\mathbf{A}$. If we relate these elements to the join tree induced by the fusion algorithm, we remind that $r_{j,i} = 0$ for $i \notin \lambda(j)$ and $l_{k,j} = 0$ for $j \notin \epsilon(j)$. Thus, the two matrices $\mathbf{L}$ and $\mathbf{R}$ maintain the zero-pattern described by the join tree. The computation of the $\mathbf{LDR}$-decomposition in the fusion algorithm can be interpreted as a *compilation* of the matrix $\mathbf{A}$, which can then be used to solve $\mathbf{AX} = \mathbf{b}$ for different vectors $\mathbf{b}$. This will be shown next.

Given some vector $\mathbf{b}$ and the $\mathbf{LV}$- or $\mathbf{LDR}$-decomposition of the matrix $\mathbf{A}$, we solve the system $\mathbf{AX} = \mathbf{b}$ by first solving $\mathbf{LY} = \mathbf{b}$, yielding the solution $\mathbf{y}$, and then $\mathbf{VX} = \mathbf{DRX} = \mathbf{y}$. This solution process is simple, since both matrices $\mathbf{L}$ and $\mathbf{V}$ are triangular: We first solve the system $\mathbf{LY} = \mathbf{b}$ by executing the collect algorithm on the join tree. In fact, starting with node 1 we directly obtain the solution value $y_1 = b_1$ from the first equation $Y_1 = b_1$. We introduce this value into the equations $j \in \epsilon(1) - \{1\}$ and obtain the new system

$$\sum_{i \in \lambda(1)} l_{j,i} X_i \quad = \quad b_j - l_{j,1} b_1.$$

Defining $b_j^{(1)} = b_j - l_{j,1} b_1$, the message sent to the child node $ch(1)$ is the set

$$\{b_j^{(1)} : j \in \epsilon(1) - \{1\}\}.$$

This defines the new lower triangular system after elimination of variable Y_1. In the k-th step, the equation for Y_k on the node k is $Y_k = b_k^{(k-1)}$. The solution value $y_k = b_k^{(k-1)}$ is introduced into the equations $j \in \epsilon(k) - \{k\}$ to obtain

$$\sum_{i \in \lambda(k)} l_{j,i} Y_i \quad = \quad b_j^{(k-1)} - l_{j,k} b_k^{(k-1)}.$$

So, the message sent to the child node $ch(k)$ is the set

$$\{b_j^{(k)} : j \in \epsilon(k) - \{k\}\}$$

with $b_j^{(k)} = b_j^{(k-1)} - l_{j,k} b_k^{(k-1)}$. In this way, the solution y_k for $k = 1, \ldots, n$ is obtained. It is important to remark that in the general case, the collect algorithm only computes the solution for the variable Y_n in the root node, and the values for the remaining variables are found by backward substitution in the distribute phase. In the case at hand, we are dealing with triangular systems and therefore obtain all solutions $\mathbf{y}$ of the system $\mathbf{LY} = \mathbf{b}$ already from the collect algorithm.

In a second step, we execute the distribute algorithm to solve the system $\mathbf{VX} = \mathbf{DRX} = \mathbf{y}$, respectively $\mathbf{RX} = \mathbf{D}^{-1}\mathbf{y}$. The root node n contains the equation $X_n = y_n/d_{n,n}$ from which we determine the solution value $x_n = y_n/d_{n,n}$. This value is sent to node $n-1$ and replaces there the variable X_n in the equations from $\mathbf{RX} = \mathbf{D}^{-1}\mathbf{y}$ that are contained in the node $n-1$. More precisely, we compute for

$$x_j \quad = \quad \frac{y_j}{d_{j,j}} - r_{j,n} x_n$$

for $j \in \lambda(n-1)$, according to equation (9.12). In the general case, node k receives a message from node $ch(k)$, which consist of

$$x_j \quad = \quad \frac{y_j}{d_{j,j}} - r_{j,ch(k)} x_{ch(k)}.$$

for $j \in \lambda(k)$. Altogether, this permits to solve the triangular system of equations on the node k for $k = n, \ldots, 1$.

We again point out that the solution of $\mathbf{AX} = \mathbf{b}$ for an arbitrary, regular matrix $\mathbf{A}$ required in the previous section a complete run of the collect and distribute algorithm. But if we dispose of an $\mathbf{LDR}$-decomposition of $\mathbf{A}$, we may solve the system $\mathbf{LY} = \mathbf{b}$ with a single run of the collect algorithm, because the involved matrix $\mathbf{L}$ is lower triangular. Likewise, we directly obtain the solution to the system $\mathbf{DR} = \mathbf{y}$ by a single run of the distribute algorithm, because the matrix $\mathbf{DR}$ is upper triangular. This symmetry between the application of collect and distribute is worth noting.

We apply this compilation approach to Example 9.1 above:

Example 9.2 *In Example 9.1 we determined the elements of the matrices* $\mathbf{L}$ *and* $\mathbf{R}$,

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & 2/3 & 1 \end{bmatrix} \quad \textit{and} \quad \mathbf{R} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 1/3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We remark how the zero-pattern controlled by the join tree of Figure 9.3 is still reflected, both in the matrix $\mathbf{L}$ *as well as in the matrix* $\mathbf{R}$. *With the diagonal matrix below we obtain the* $\mathbf{LDR}$*-decomposition* $\mathbf{A} = \mathbf{LDR}$:

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1/3 \end{bmatrix}.$$

In the first step, we solve the system $\mathbf{LY} = \mathbf{b}$, *where* $\mathbf{b}$ *is given in Example 9.3. We again assign the first two equations of this system to the first join tree node and the last two equations to the second node to control fill-ins. In the first node, we obtain* $y_1 = 2$ *and introduce this value into the second equation, whose right-hand side becomes* $b_2^{(1)} = b_2 - l_{2,1}y_1 = -1$. *This value is sent to node* 2, *where now the solution value* $y_2 = b_2^{(1)} = -1$ *is obtained. Also, node* 2 *computes the message* $b_3^{(2)} = b_3 - l_{3,2}y_2 = 2$ *and sends it to node* 3. *Node* 3 *computes* $y_3 = 2$ *and sends the message* $b_4^{(3)} = b_4 - l_{4,3}y_3 = 14/3$ *to node* 4, *where we finally obtain* $y_4 = 14/3$.

The solution of $\mathbf{RX} = \mathbf{D}^{-1}\mathbf{y}$ *by the distribute algorithm, using the values* $\mathbf{y}$ *computed in the collect phase, is initiated by the root node* 4. *We obtain on node* 4 *the value* $x_4 = y_4/d_{4,4} = 14$. *This value is sent to node* 3, *where we obtain* $x_3 = y_3/d_{3,3} - r_{3,4}x_4 = -4$ *and send this message to node* 2. *Continuing this process, we obtain* $x_2 = y_2/d_{2,2} - r_{2,3}x_3 = -3$ *in node* 2 *and* $x_1 = y_1/d_{1,1} - r_{1,2}x_2 = 5$ *in node* 1. *This terminates the distribute phase. We note that on all nodes* j *only variables from the node label* $\lambda(j)$ *are involved.*

This ends our first discussion of regular systems. An important special case of regular systems is provided by symmetric, positive definite systems. They allow to refine the present approach as shown in the following section.

9.2 SYMMETRIC, POSITIVE DEFINITE MATRICES

The method of $\mathbf{LDR}$-decomposition presented in the previous section becomes especially interesting in the case of linear systems with *symmetric, positive definite matrices*. Such systems arise frequently in electrical network analysis, analysis of structural systems and hydraulic problems. Further, they arise in the classical method of *least squares*, which will be discussed in Section 9.2.5 as an application of the results developed here. In all these cases, the exploitation of sparsity is important.

However, it turns out that in the case of systems of linear equations with symmetric, positive definite matrices, the underlying algebra of affine solution spaces is of less interest and can be replaced by another valuation algebra. We first introduce this algebra, which interestingly is closely related to the valuation algebra of Gaussian potentials introduced in Instance 1.6. The reason for this is elucidated in Section 9.2.5, where systems of linear equations with symmetric, positive definite matrices are related to statistical problems.

9.2.1 Valuation Algebra of Symmetric Systems

To put the discussion into the general framework of valuation algebras, we consider variables $X_1, \ldots, X_n$ together with the associated index set $r = \{1, \ldots, n\}$. Vectors of variables $\mathbf{X}$, as well as matrices $\mathbf{A}$ and vectors $\mathbf{b}$ then refer to certain subsets $s \subseteq r$. For instance, a system $\mathbf{AX} = \mathbf{b}$ is said to be an s-system, if $\mathbf{X}$ is the variable vector whose components X_i have indices in $s \subseteq r$, $\mathbf{A}$ is a symmetric, positive definite $s \times s$ matrix and $\mathbf{b}$ is an s-vector. Such systems are fully determined by the pair $(\mathbf{A}, \mathbf{b})$ where $\mathbf{A}$ is an $s \times s$ matrix, $\mathbf{b}$ an s-vector for some $s \subseteq r$. These are the elements of the valuation algebra to be identified. Let Φ_s be the set of all pairs $(\mathbf{A}, \mathbf{b})$ relative to some subset $s \subseteq r$ and define

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

By convention, we define for the empty set $\Phi_\emptyset = \{(\diamond, \diamond)\}$. The *label* of a pair $(\mathbf{A}, \mathbf{b})$ is defined as $d(\mathbf{A}, \mathbf{b}) = s$, if $(\mathbf{A}, \mathbf{b}) \in \Phi_s$.

In order to define the operations of combination and projection for elements in Φ, we first consider variable elimination in the system $\mathbf{AX} = \mathbf{b}$. Suppose $(\mathbf{A}, \mathbf{b}) \in \Phi_s$, i.e. the system is over variables in s, and assume $t \subseteq s$. We want to eliminate the variables in the index set $s - t$. For this purpose, we decompose $\mathbf{A}$ as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^{\downarrow s-t,s-t} & \mathbf{A}^{\downarrow s-t,t} \\ \mathbf{A}^{\downarrow t,s-t} & \mathbf{A}^{\downarrow t,t} \end{bmatrix}.$$

The system $\mathbf{AX} = \mathbf{b}$ can then be written as

$$\begin{aligned} \mathbf{A}^{\downarrow s-t,s-t}\mathbf{X}^{\downarrow s-t} &+ \mathbf{A}^{\downarrow s-t,t}\mathbf{X}^{\downarrow t} &= \mathbf{b}^{\downarrow s-t}, \\ \mathbf{A}^{\downarrow t,s-t}\mathbf{X}^{\downarrow s-t} &+ \mathbf{A}^{\downarrow t,t}\mathbf{X}^{\downarrow t} &= \mathbf{b}^{\downarrow t}. \end{aligned}$$

Solving the first part for $\mathbf{X}^{\downarrow s-t}$ and substituting this in the second part leads to the reduced system for $\mathbf{X}^{\downarrow t}$

$$\left(\mathbf{A}^{\downarrow t,t} - \mathbf{A}^{\downarrow t,s-t}(\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{A}^{\downarrow s-t,t}\right)\mathbf{X}^{\downarrow t} = \mathbf{b}^{\downarrow t} - \mathbf{A}^{\downarrow t,s-t}(\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{b}^{\downarrow s-t}.$$

Here we use the fact that any diagonal submatrix of a symmetric, positive definite matrix is regular. We define

$$\begin{aligned} \mathbf{A}^{\Downarrow t} &= \mathbf{A}^{\downarrow t,t} - \mathbf{A}^{\downarrow t,s-t}(\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{A}^{\downarrow s-t,t} \\ \mathbf{b}^{\Downarrow t} &= \mathbf{b}^{\downarrow t} - \mathbf{A}^{\downarrow t,s-t}(\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{b}^{\downarrow s-t} \end{aligned} \tag{9.13}$$

and remark that $\mathbf{A}^{\Downarrow t}$ is still symmetric, positive definite.

Theorem 9.1 *If* $\mathbf{x}$ *is the unique solution of* $\mathbf{AX} = \mathbf{b}$*, then* $\mathbf{x}^{\downarrow t}$ *is the unique solution to* $\mathbf{A}^{\Downarrow t}\mathbf{X}^{\downarrow t} = \mathbf{b}^{\Downarrow t}$*. Conversely, if* $\mathbf{y}$ *is the unique solution of* $\mathbf{A}^{\Downarrow t}\mathbf{Y} = \mathbf{b}^{\Downarrow t}$ *and*

$$\mathbf{x} \quad = \quad (\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{b}^{\downarrow s-t} - (\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{A}^{\downarrow s-t,t}\mathbf{y},$$

then $(\mathbf{x}, \mathbf{y})$ *is the unique solution of* $\mathbf{AX} = \mathbf{b}$.

Proof: The proof is straightforward, using the definitions of $\mathbf{A}^{\Downarrow t}$ and $\mathbf{b}^{\Downarrow t}$. Since $\mathbf{x}$ is a solution of $\mathbf{AX} = \mathbf{b}$, it holds that

$$\begin{aligned}
\mathbf{A}^{\downarrow s-t,s-t}\mathbf{x}^{\downarrow s-t} \quad &+ \quad \mathbf{A}^{\downarrow s-t,t}\mathbf{x}^{\downarrow t} \quad = \quad \mathbf{b}^{\downarrow s-t},\\
\mathbf{A}^{\downarrow t,s-t}\mathbf{x}^{\downarrow s-t} \quad &+ \quad \mathbf{A}^{\downarrow t,t}\mathbf{x}^{\downarrow t} \quad = \quad \mathbf{b}^{\downarrow t},
\end{aligned}$$

hence

$$\left(\mathbf{A}^{\downarrow t,t} - \mathbf{A}^{\downarrow t,s-t}(\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{A}^{\downarrow s-t,t}\right)\mathbf{x}^{\downarrow t} = \mathbf{b}^{\downarrow t} - \mathbf{A}^{\downarrow t,s-t}(\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{b}^{\downarrow s-t}.$$

This proves the first part of the theorem. The second part follows from the first subsystem above, if $\mathbf{x}^{\downarrow s-t}$ is replaced by $\mathbf{x}$ and $\mathbf{x}^{\downarrow t}$ by $\mathbf{y}$ and if it is multiplied on both sides by $(\mathbf{A}^{\downarrow s-t,s-t})^{-1}$. ■

This theorem justifies to define the operation of projection for elements in Φ by

$$(\mathbf{A}, \mathbf{b})^{\downarrow t} \quad = \quad (\mathbf{A}^{\Downarrow t}, \mathbf{b}^{\Downarrow t}), \tag{9.14}$$

for $t \subseteq d(\mathbf{A}, \mathbf{b})$. It is well-known in matrix algebra that $\mathbf{A}^{\Downarrow t}$ may also be written as

$$\mathbf{A}^{\Downarrow t} \quad = \quad \left((\mathbf{A}^{-1})^{\downarrow t,t}\right)^{-1}.$$

A proof can for example be found in (Harville, 1997).

Next, consider two systems $\mathbf{A}_1\mathbf{X}_1 = \mathbf{b}_1$ and $\mathbf{A}_2\mathbf{X}_2 = \mathbf{b}_2$. What is a sensible way to combine these two systems into a new system? Clearly, taking simply the union of both systems as we did so far with systems of linear equations makes no sense, because the matrix of the combined system will no more be symmetric. Moreover, the combined system will most of the time have no solution anymore. We propose alternatively to add the matrices and the right-hand vectors component-wise. More precisely, we define for $d(\mathbf{A}_1, \mathbf{b}_1) = s$ and $d(\mathbf{A}_2, \mathbf{b}_2) = t$,

$$(\mathbf{A}_1, \mathbf{b}_1) \otimes (\mathbf{A}_2, \mathbf{b}_2) \quad = \quad (\mathbf{A}_1^{\uparrow s \cup t} + \mathbf{A}_2^{\uparrow s \cup t}, \mathbf{b}_1^{\uparrow s \cup t} + \mathbf{b}_2^{\uparrow s \cup t}). \tag{9.15}$$

At this point, the proposed definition is rather ad hoc. It will be justified by the success of local computation and even more strongly by a semantical interpretation given in the following Section 9.2.5. For the time being, the question is whether (Φ, D) together with the operations of labeling, projection and combination satisfies the axioms of a valuation algebra. Instead of verifying the axioms directly, we consider the mapping $(\mathbf{A},\mathbf{b}) \mapsto (\mathbf{A},\mathbf{u})$, where $\mathbf{u} = \mathbf{A}^{-1}\mathbf{b}$. Let Ψ be the set of all pairs $(\mathbf{A},\mathbf{u})$, where $\mathbf{A}$ is a symmetric, positive definite $s \times s$ matrix and $\mathbf{u}$ an s-vector. This mapping is a surjection between Φ and Ψ. We next look at how combination and projection in Φ is mapped to Ψ. Let $(\mathbf{A}_1,\mathbf{b}_1)$ and $(\mathbf{A}_2,\mathbf{b}_2)$ belong to Φ with labels s and t, respectively. Then, by the definition of combination above, $(\mathbf{A}_1,\mathbf{b}_1) \otimes (\mathbf{A}_2,\mathbf{b}_2) = (\mathbf{A},\mathbf{b})$ with

$$\mathbf{A} \quad = \quad \mathbf{A}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t}.$$

Hence, $(\mathbf{A}_1,\mathbf{b}_1) \otimes (\mathbf{A}_2,\mathbf{b}_2)$ maps to $(\mathbf{A},\mathbf{u})$ with

$$\begin{aligned}\mathbf{u} &= \mathbf{A}^{-1}\mathbf{b} = \mathbf{A}^{-1}\Big(\mathbf{b}_1^{\uparrow s\cup t} + \mathbf{b}_2^{\uparrow s\cup t}\Big)\\ &= \mathbf{A}^{-1}\Big((\mathbf{A}_1\mathbf{u}_1)^{\uparrow s\cup t} + (\mathbf{A}_2\mathbf{u}_2)^{\uparrow s\cup t}\Big) = \mathbf{A}^{-1}\Big(\mathbf{A}_1^{\uparrow s\cup t}\mathbf{u}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t}\mathbf{u}_2^{\uparrow s\cup t}\Big),\end{aligned}$$

where $\mathbf{u}_1 = \mathbf{A}_1^{-1}\mathbf{b}_1$ and $\mathbf{u}_1 = \mathbf{A}_2^{-1}\mathbf{b}_2$. Consequently, a combination of two elements in Φ is mapped as follows to an element in Ψ

$$(\mathbf{A}_1,\mathbf{b}_1) \otimes (\mathbf{A}_2,\mathbf{b}_2) \quad \mapsto \quad (\mathbf{A},\mathbf{u})$$

with

$$\mathbf{A} \quad = \quad \mathbf{A}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t}$$

and

$$\mathbf{u} \quad = \quad \mathbf{A}^{-1}\Big(\mathbf{A}_1^{\uparrow s\cup t}\mathbf{u}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t}\mathbf{u}_2^{\uparrow s\cup t}\Big).$$

Turning to projection, let $(\mathbf{A},\mathbf{b})$ be an element in Φ with domain s, which is mapped to $(\mathbf{A},\mathbf{u})$ with $\mathbf{u} = \mathbf{A}^{-1}\mathbf{b}$. If $t \subseteq s$, then $(\mathbf{A},\mathbf{b})^{\downarrow t}$ represents the system $\mathbf{A}^{\Downarrow t}\mathbf{X}^{\downarrow t} = \mathbf{b}^{\Downarrow t}$ with the unique solution $\mathbf{u}^{\downarrow t}$ as a consequence of Theorem 9.1. This means that $(\mathbf{A},\mathbf{b})^{\downarrow t}$ maps to

$$\Big(\mathbf{A}^{\Downarrow t}, \left(\mathbf{A}^{\Downarrow t}\right)^{-1}\mathbf{b}^{\Downarrow t}\Big) \quad = \quad (\mathbf{A}^{\Downarrow t}, \mathbf{u}^{\downarrow t})$$

where

$$\mathbf{A}^{\Downarrow t} \quad = \quad \Big((\mathbf{A}^{-1})^{\downarrow t,t}\Big)^{-1}.$$

We thus have the following projection rule in Ψ:

$$(\mathbf{A},\mathbf{u})^{\downarrow t} \quad = \quad \Big(\left((\mathbf{A}^{-1})^{\downarrow t,t}\right)^{-1}, \mathbf{u}^{\downarrow t}\Big).$$

We observe that combination and projection in Ψ correspond exactly to the operations of Gaussian potentials introduced in Instance 1.6 and we therefore know that $\langle \Psi, D\rangle$ forms a valuation algebra. Moreover, due to the introduced mapping, $\langle \Phi, D\rangle$ and

$\langle \Psi, D\rangle$ are isomorphic which implies that $\langle \Phi, D\rangle$ is a valuation algebra. We proved the following theorem:

Theorem 9.2 *The algebra of linear systems with symmetric, positive definite matrices is isomorphic to the valuation algebra of Gaussian potentials.*

Let us have a closer look at the combination of $(\mathbf{A}_1, \mathbf{b}_1)$ and $(\mathbf{A}_2, \mathbf{b}_2)$ with $d(\mathbf{A}_1, \mathbf{b}_1) = s$ and $d(\mathbf{A}_2, \mathbf{b}_2) = t$. If $\mathbf{X}$ denotes an $(s \cup t)$ variable vector, these valuations represent the systems $\mathbf{A}_1\mathbf{X}^{\downarrow s} = \mathbf{b}_1$ and $\mathbf{A}_2\mathbf{X}^{\downarrow t} = \mathbf{b}_2$ with symmetric, positive definite matrices. We consider the system $\mathbf{AX} = \mathbf{b}$ which is represented by $(\mathbf{A}_1, \mathbf{b}_1) \otimes (\mathbf{A}_2, \mathbf{b}_2)$ such that

$$\mathbf{A} \;=\; \mathbf{A}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t} \quad \text{and} \quad \mathbf{b} \;=\; \mathbf{b}_1^{\uparrow s\cup t} + \mathbf{b}_2^{\uparrow s\cup t}.$$

In order to compute the solution to the system $\mathbf{AX} = \mathbf{b}$, we proceed in two steps by first projecting to t, i.e. eliminating the variables $\mathbf{X}^{\downarrow s-t}$, and then solving the system $\mathbf{A}^{\Downarrow t}\mathbf{X}^{\downarrow t} = \mathbf{b}^{\Downarrow t}$. This is justified by Theorem 9.1. We decompose the matrices and vectors according to the sets $s - t$, $s \cap t$ and $t - s$ and write the system as

$$\begin{bmatrix} \mathbf{A}_1^{\downarrow s-t,s-t} & \mathbf{A}_1^{\downarrow s-t,s\cap t} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t,s-t} & \mathbf{A}_1^{\downarrow s\cap t,s\cap t} + \mathbf{A}_2^{\downarrow s\cap t,s\cap t} & \mathbf{A}_2^{\downarrow s\cap t,t-s} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s,s\cap t} & \mathbf{A}_2^{\downarrow t-s,t-s} \end{bmatrix} \begin{bmatrix} \mathbf{X}^{\downarrow s-t} \\ \mathbf{X}^{\downarrow s\cap t} \\ \mathbf{X}^{\downarrow t-s} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{b}_1^{\downarrow s-t} \\ \mathbf{b}_1^{\downarrow s\cap t} + \mathbf{b}_2^{\downarrow s\cap t} \\ \mathbf{b}_2^{\downarrow t-s} \end{bmatrix}.$$

Eliminating the variables $\mathbf{X}^{\downarrow s-t}$ means to solve the first subsystem for the variables $\mathbf{X}^{\downarrow s-t}$ and to substitute the solution into the other equations, which gives the system

$$\begin{bmatrix} \mathbf{A}_1^{\Downarrow s\cap t,s\cap t} + \mathbf{A}_2^{\downarrow s\cap t,s\cap t} & \mathbf{A}_2^{\downarrow s\cap t,t-s} \\ \mathbf{A}_2^{\downarrow t-s,s\cap t} & \mathbf{A}_2^{\downarrow t,t-s} \end{bmatrix} \begin{bmatrix} \mathbf{X}^{\downarrow s\cap t} \\ \mathbf{X}^{\downarrow t-s} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^{\Downarrow s\cap t} + \mathbf{b}_2^{\downarrow s\cap t} \\ \mathbf{b}_2^{\downarrow t-s} \end{bmatrix}.$$

Note that this system is represented by either

$$((\mathbf{A}_1, \mathbf{b}_1) \otimes (\mathbf{A}_2, \mathbf{b}_2))^{\downarrow t} \quad \text{or} \quad (\mathbf{A}_1, \mathbf{b}_1)^{\downarrow s\cap t} \otimes (\mathbf{A}_2, \mathbf{b}_2). \tag{9.16}$$

This illustrates the combination axiom and shows how local computation can be applied to symmetric, positive definite systems. If the second representation is used, then the variables $\mathbf{X}^{\downarrow s-t}$ in the first system are eliminated and the result is combined to the second system. It would be reasonable to apply the local computation methods of Section 9.1.2 and 9.1.3 for the variable elimination process. This however is not possible because idempotency does not hold anymore in this algebra. Symmetric, positive definite systems only form a valuation algebra but not an information algebra. For local computation, this means that we must apply the Shenoy-Shafer architecture. However, Theorem 9.1 offers an alternative possibility based on solution construction which comes near to idempotency. This is exploited in the following section, and variable elimination and **LDR**-decomposition with symmetric systems is examined in Section 9.2.3.

9.2.2 Solving Symmetric Systems

We now return to local computation as a means to control fill-ins. Note that zero-patterns in a symmetric matrix are symmetric too. Consequently, symmetric decompositions of the matrix $\mathbf{A}$ as shown in Figure 9.1 are not possible. This implies that the underlying valuation algebra of affine spaces introduced in Section 7.2 and used so far is no longer of interest. In other words, combination as intersection or join of affine spaces does not really reflect the natural decomposition of symmetric systems. Instead, this is achieved by adding symmetric, positive definite matrices as showed in the foregoing section. The corresponding algebra will now be used to control fill-ins.

Consider a join tree (V, E, λ, D) with the labeling function $\lambda : V \rightarrow D = \mathcal{P}(r)$ for $r = \{1, \ldots, n\}$. We number the nodes from 1 to $m = |V|$ and assume that any node $i \in V$ in this join tree covers a pair $\phi_i = (\mathbf{A}_i, \mathbf{b}_i)$ with $\omega_i = d(\phi_i)$. They consist of a symmetric, positive definite $\omega_i \times \omega_i$ matrix $\mathbf{A}_i$ and an ω_i-vector $\mathbf{b}_i$. We further assume that $\omega_i \subseteq \lambda(i)$ and $\omega_1 \cup \ldots \cup \omega_m = r$. In other words, ϕ_i are elements of the valuation algebra introduced in Section 9.2.1 and (V, E, λ, D) is a covering join tree for the factorization

$$\phi \quad = \quad (\mathbf{A}, \mathbf{b}) \quad = \quad \phi_1 \otimes \cdots \otimes \phi_m \tag{9.17}$$

according to Definition 3.8. As usual, the nodes of the join tree are numbered such that if node j is on the path form node i to node m, then $i < j$. We will see in Section 9.2.4 how such decompositions can be produced, and it will be shown in 9.2.5 that factorizations of symmetric, positive definite systems may also occur naturally in certain applications. The objective function ϕ represents the symmetric, positive definite system $\mathbf{AX} = \mathbf{b}$, where

$$\mathbf{A} \quad = \quad \mathbf{A}_1^{\uparrow r} + \ldots + \mathbf{A}_m^{\uparrow r}$$

and

$$\mathbf{b} \quad = \quad \mathbf{b}_1^{\uparrow r} + \ldots + \mathbf{b}_m^{\uparrow r}.$$

The join tree exhibits the sparsity of the matrix $\mathbf{A}$. We have $a_{j,h} = 0$, if j and h do not belong both to some ω_i. Its zero-pattern is denoted by the set

$$Z \quad = \quad \{(j,h) : \text{ there is no } i = 1, \ldots, m, \text{ such that } j, h \in \omega_i\}. \tag{9.18}$$

Given a factorized system $\phi = (\mathbf{A}, \mathbf{b}) = \phi_1 \otimes \ldots \otimes \phi_m$ and some covering join tree, we next focus on the solution process using local computation. As mentioned above, this algebra is fundamentally different from the algebra of affine spaces and does not directly yield solution sets. Instead, we aim at first executing a local computation scheme and later build the solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ using a generic solution construction algorithm from Chapter 8. Hence, we start by introducing a suitable notion of configuration extension sets for symmetric, positive define systems, motivated by Theorem 9.1: For a t-vector of real numbers $\mathbf{x}$, $\phi = (\mathbf{A}, \mathbf{b})$ and $t \subseteq d(\phi)$

we define

$$W^t_\phi(\mathbf{x}) \quad = \quad \left\{(\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{b}^{\downarrow s-t} - (\mathbf{A}^{\downarrow s-t,s-t})^{-1}\mathbf{A}^{\downarrow s-t,t}\mathbf{x}\right\}. \qquad (9.19)$$

For later reference we prove the following property:

Lemma 9.2 *Assume* $\phi_1 = (\mathbf{A}, \mathbf{b}_1)$ *and* $\phi_2 = (\mathbf{A}, \mathbf{b}_2)$ *with* $d(\phi_1) = d(\phi_2) = s$ *and* $t \subseteq u \subseteq s$. *If*

$$\mathbf{b}_1^{\downarrow s-t} \quad = \quad \mathbf{b}_2^{\downarrow s-t}$$

then it holds for all t-vectors $\mathbf{x}$ *that*

$$W^t_{\phi_1^{\downarrow u}}(\mathbf{x}) \quad = \quad W^t_{\phi_2^{\downarrow u}}(\mathbf{x}).$$

Proof: By equation (9.19) and $\phi_i^{\downarrow u} = (\mathbf{A}^{\Downarrow u}, \mathbf{b}_i^{\Downarrow u})$ we have for $i = 1, 2$

$$\begin{aligned} W^t_{\phi_i^{\Downarrow u}}(\mathbf{x}) \quad &= \quad \Big\{\big((\mathbf{A}^{\Downarrow u})^{\downarrow u-t,u-t}\big)^{-1}(\mathbf{b}_i^{\Downarrow u})^{\downarrow u-t} - \\ &\qquad \big((\mathbf{A}^{\Downarrow u})^{\downarrow u-t,u-t}\big)^{-1}(\mathbf{A}^{\Downarrow u})^{\downarrow u-t,t}\mathbf{x}\Big\}. \end{aligned}$$

The statement of the lemma then follows directly from:

$$\begin{aligned} \left(\mathbf{b}_1^{\Downarrow u}\right)^{\downarrow u-t} \quad &= \quad \left(\mathbf{b}_1^{\downarrow u} - \mathbf{A}^{\downarrow u,s-u}(\mathbf{A}^{\downarrow s-u,s-u})^{-1}\mathbf{b}_1^{\downarrow s-u}\right)^{\downarrow u-t} \\ &= \quad \left(\mathbf{b}_1^{\downarrow u}\right)^{\downarrow u-t} - \left(\mathbf{A}^{\downarrow u,s-u}(\mathbf{A}^{\downarrow s-u,s-u})^{-1}\mathbf{b}_1^{\downarrow s-u}\right)^{\downarrow u-t} \\ &= \quad \left(\mathbf{b}_2^{\downarrow u}\right)^{\downarrow u-t} - \left(\mathbf{A}^{\downarrow u,s-u}(\mathbf{A}^{\downarrow s-u,s-u})^{-1}\mathbf{b}_2^{\downarrow s-u}\right)^{\downarrow u-t} \\ &= \quad \left(\mathbf{b}_2^{\Downarrow u}\right)^{\downarrow u-t}. \end{aligned}$$

■

Based on this lemma, we show that the definition of equation (9.19) complies with the general definition of configuration extension sets in Section 8.1.

Theorem 9.3 *Solution extension sets in valuation algebras of symmetric, positive definite systems satisfy the property of Definition 8.1, i.e. for all* $\phi = (\mathbf{A}, \mathbf{b}) \in \Phi$ *with* $t \subseteq u \subseteq s = d(\phi)$ *and all t-vectors* $\mathbf{x}$ *we have*

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

Proof: If $\mathbf{x}$ is the partial solution to the system $\phi = (\mathbf{A}, \mathbf{b})$, then the above statement follows directly from Theorem 9.1. In the following proof, we consider an arbitrary t-vector $\mathbf{x}$ and determine a new system $\phi_1 = (\mathbf{A}, \mathbf{b}_1)$ to which $\mathbf{x}$ is the partial solution. The statement then holds for $\mathbf{x}$ with respect to this new system, and if ϕ_1 is chosen in such a way that Lemma 9.2 applies, then the statement also holds for

$\mathbf{x}$ with respect to the original system ϕ. Hence, let $\mathbf{x}$ be an arbitrary t-vector. We consider the new system $\phi_1 = (\mathbf{A}, \mathbf{b}_1)$ with

$$\mathbf{b}^{\downarrow s-t} \quad = \quad \mathbf{b}_1^{\downarrow s-t}$$

and determine $\mathbf{b}_1^{\downarrow t}$ such that $\mathbf{x}$ is the solution to $\phi_1^{\downarrow t} = (\mathbf{A}^{\Downarrow t}, \mathbf{b}_1^{\Downarrow t})$, i.e.

$$\mathbf{A}^{\Downarrow t}\mathbf{x} \; = \; \mathbf{b}_1^{\Downarrow t} \; = \; \mathbf{b}_1^{\downarrow t} - \mathbf{A}^{\downarrow t, s-t}(\mathbf{A}^{\downarrow s-t, s-t})^{-1}\mathbf{b}_1^{\downarrow s-t}$$

and therefore

$$\mathbf{b}_1^{\downarrow t} \quad = \quad \mathbf{A}^{\Downarrow t}\mathbf{x} + \mathbf{A}^{\downarrow t, s-t}(\mathbf{A}^{\downarrow s-t, s-t})^{-1}\mathbf{b}^{\downarrow s-t}.$$

Since $\mathbf{x}$ is the solution to $(\mathbf{A}, \mathbf{b}_1)^{\downarrow t}$, it follows from Theorem 9.1 that $(\mathbf{y}, \mathbf{x})$ with

$$\mathbf{y} \quad = \quad (\mathbf{A}^{\downarrow s-t, s-t})^{-1}\mathbf{b}^{\downarrow s-t} - (\mathbf{A}^{\downarrow s-t, s-t})^{-1}\mathbf{A}^{\downarrow s-t, t}\mathbf{x}$$

is the solution to the system $(\mathbf{A}, \mathbf{b}_1)$. We thus have

$$\mathbf{y} \in W^t_{\phi_1}(\mathbf{x}) \quad = \quad W^t_{\phi}(\mathbf{x})$$

as a consequence of Lemma 9.2. Then, since $(\mathbf{y}, \mathbf{x})$ is the solution to $(\mathbf{A}, \mathbf{b}_1)$, it follows from Theorem 9.1 that $(\mathbf{y}^{\downarrow u-t}, \mathbf{x})$ is the solution to $(\mathbf{A}, \mathbf{b}_1)^{\downarrow u}$. Hence,

$$\mathbf{y}^{\downarrow u-t} \in W^t_{\phi_1^{\downarrow u}}(\mathbf{x}) \quad = \quad W^t_{\phi^{\downarrow u}}(\mathbf{x})$$

by Lemma 9.2. Finally, since $(\mathbf{y}^{\downarrow u-t}, \mathbf{x})$ is the solution to $(\mathbf{A}, \mathbf{b}_1)^{\downarrow u}$ it again follows from Theorem 9.1 that $(\mathbf{y}^{\downarrow u-t}, \mathbf{y}^{\downarrow s-u}, \mathbf{x})$ is the solution to $(\mathbf{A}, \mathbf{b}_1)$. We have

$$\mathbf{y}^{\downarrow s-u} \in W^u_{\phi_1}(\mathbf{y}^{\downarrow u-t}, \mathbf{x}) \quad = \quad W^u_{\phi}(\mathbf{y}^{\downarrow u-t}, \mathbf{x}).$$

Altogether, this show that for an arbitrary t-vector $\mathbf{x}$ we have

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

■

Next, we specialize general solution sets from Definition 8.2 to symmetric, positive definite systems. For $\phi = (\mathbf{A}, \mathbf{b}) \in \Phi$ and $t = \emptyset$, it follows from equation (9.19) that

$$c_{\phi} \;=\; W^{\emptyset}_{\phi}(\diamond) \;=\; \{\mathbf{A}^{-1}\mathbf{b}\}. \tag{9.20}$$

This shows that c_{ϕ} indeed corresponds to the singleton set of the unique solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ to the symmetric, positive definite system $\mathbf{AX} = \mathbf{b}$. We therefore conclude that the generic solution construction procedure of Section 8.2.1 could be applied to build c_{ϕ} based on the results of a multi-query local computation scheme. However, the same is also possible using the results of a single-query architecture only. This follows by verifying the property of Theorem 8.3:

Lemma 9.3 *Symmetric, positive definite systems satisfy the property that for all* $\psi_1, \psi_2 \in \Phi$ *with* $d(\psi_1) = s$, $d(\psi_2) = t$, $s \subseteq u \subseteq s \cup t$ *and* u*-vector* $\mathbf{x}$ *we have*

$$W_{\psi_2}^{u\cap t}(\mathbf{x}^{\downarrow u\cap t}) \quad = \quad W_{\psi_1\otimes\psi_2}^{u}(\mathbf{x}).$$

Proof: We observe the following identities for $\psi_1 = (\mathbf{A}_1, \mathbf{b}_1)$ and $\psi_2 = (\mathbf{A}_2, \mathbf{b}_2)$:

$$(\mathbf{A}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t})^{\downarrow(s\cup t)-u,(s\cup t)-u} \quad = \quad \mathbf{A}_2^{\downarrow(s\cup t)-u,(s\cup t)-u} \quad = \quad \mathbf{A}_2^{\downarrow t-u,t-u}$$

and similarly

$$(\mathbf{b}_1^{\uparrow s\cup t} + \mathbf{b}_2^{\uparrow s\cup t})^{\downarrow(s\cup t)-u} \quad = \quad \mathbf{b}_2^{\downarrow t-u}$$

It then follows that

$$\begin{aligned} W_{\phi\otimes\psi}^{u}(\mathbf{x}) &= \Big\{([\mathbf{A}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t}]^{\downarrow(s\cup t)-u,(s\cup t)-u})^{-1}[\mathbf{b}_1^{\uparrow s\cup t} + \mathbf{b}_2^{\uparrow s\cup t}]^{\downarrow(s\cup t)-u} \\ &- ([\mathbf{A}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t}]^{\downarrow(s\cup t)-u,(s\cup t)-u})^{-1}[\mathbf{A}_1^{\uparrow s\cup t} + \mathbf{A}_2^{\uparrow s\cup t}]^{\downarrow(s\cup t)-u,u}\mathbf{x}\Big\} \\ &= \Big\{(\mathbf{A}_2^{\downarrow t-u,t-u})^{-1}\mathbf{b}_2^{\downarrow t-u} - (\mathbf{A}_2^{\downarrow t-u,t-u})^{-1}\mathbf{A}_2^{\downarrow t-u,u}\mathbf{x}^{\downarrow u\cap t}\Big\} \\ &= W_{\psi}^{u\cap t}(\mathbf{x}^{\downarrow u\cap t}). \end{aligned}$$

■

Hence, given the results of a single-query local computation architecture, we may apply Algorithm 8.4 to build the unique solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ of a factorized system $\phi = (\mathbf{A}, \mathbf{b}) = \phi_1 \otimes \ldots \otimes \phi_m$ with $\phi_i = (\mathbf{A}_i, \mathbf{b}_i)$ for $i = 1, \ldots, m$. At this point, we should remember that the valuation algebra of symmetric, positive definite systems is isomorphic to the valuation algebra of Gaussian potentials and does not provide neutral elements; see Instance 3.5. Hence, the domain $\omega_i = d(\phi_i)$ of node $i \in V$ in the join tree does not necessarily correspond to the node label $\lambda(i)$. We only have $\omega_i \subseteq \lambda(i)$. Consequently, we must choose the generalized collect algorithm from Section 3.10 to explain the message-passing view of this local computation based solution. The run of this algorithm followed by the solution construction process is delineated next, using the notation of equation (3.42) from Section 3.10.

At step $i = 1, \ldots, m-1$, node i sends the message

$$\mu_{i\rightarrow ch(i)} \quad = \quad \left(\mathbf{A}_i^{(i)}, \mathbf{b}_i^{(i)}\right)^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i))}$$

to its child node $ch(i)$, where the message is combined with the current node content

$$\left(\mathbf{A}_{ch(i)}^{(i+1)}, \mathbf{b}_{ch(i)}^{(i+1)}\right) \quad = \quad \mu_{i\rightarrow ch(i)} \otimes \left(\mathbf{A}_{ch(i)}^{(i)}, \mathbf{b}_{ch(i)}^{(i)}\right).$$

The domain of the node content updates to:

$$\omega_{ch(i)}^{(i+1)} = d\left(\mathbf{A}_{ch(i)}^{(i+1)}, \mathbf{b}_{ch(i)}^{(i+1)}\right) = \omega_{ch(i)}^{(i)} \cup \left(\omega_i^{(i)} \cap \lambda(ch(i))\right).$$

For all other nodes $j \neq ch(i)$ we set

$$\left(\mathbf{A}_j^{(i+1)}, \mathbf{b}_j^{(i+1)}\right) = \left(\mathbf{A}_j^{(i)}, \mathbf{b}_j^{(i)}\right) \quad \text{and} \quad \omega_j^{(i+1)} = \omega_j^{(i)}$$

with

$$\left(\mathbf{A}_i^{(1)}, \mathbf{b}_i^{(1)}\right) = \left(\mathbf{A}_i, \mathbf{b}_i\right) \quad \text{and} \quad \omega_i^{(1)} = \omega_i$$

for all $i = 1, \ldots, m$. We now add an additional step that is not part of the general collect algorithm. When node i computes the message for its child node, it has to eliminate the variables

$$\omega_i^{(i)} - \lambda(ch(i)) = \lambda(i) - \lambda(ch(i)) \tag{9.21}$$

from its node content. This equality is due to Lemma 4.2. The eliminated variables satisfy the system

$$\begin{aligned}\mathbf{X}^{\downarrow\lambda(i)-\lambda(ch(i))} = & \left(\mathbf{A}_i^{(i)\ \downarrow\lambda(i)-\lambda(ch(i)),\lambda(i)-\lambda(ch(i))}\right)^{-1} \mathbf{b}^{(i)\ \downarrow\lambda(i)-\lambda(ch(i))} - \\ & \left(\mathbf{A}_i^{(i)\ \downarrow\lambda(i)-\lambda(ch(i)),\lambda(i)-\lambda(ch(i))}\right)^{-1} \\ & \mathbf{A}_i^{(i)\ \downarrow\omega_i^{(i)}\cap\lambda(ch(i)),\omega_i^{(i)}\cap\lambda(ch(i))} \mathbf{X}^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i))}.\end{aligned} \tag{9.22}$$

For later use in the solution construction phase, we store in node i the matrices on the right-hand side of the equation. This is the only modification with respect to the generalized collect algorithm. If we repeat this process up to $i = m$, we obtain on node m the pair

$$\left(\mathbf{A}_m^{(m)}, \mathbf{b}_m^{(m)}\right) = \phi^{\downarrow\lambda(m)}$$

as a consequence of Theorem 3.7. The content of node m at the end of the collect phase represents the symmetric system

$$\mathbf{A}_m^{(m)} \mathbf{X}^{\downarrow\lambda(m)} = \mathbf{b}_m^{(m)}.$$

Solving this system gives us the partial solution set $c_\phi^{\downarrow\lambda(m)}$ as a consequence of Lemma 9.20 and Lemma 8.1. We now follow the solution construction algorithm of Section 8.2.3 starting in the root node m. At step $i = m-1, \ldots, 1$ node $i \in V$ receives the partial solution set

$$c_\phi^{\downarrow\lambda(i)\cap\lambda(ch(i))} = \left\{\mathbf{x}^{\downarrow\lambda(i)\cap\lambda(ch(i))}\right\}$$

from its child node $ch(i)$. Using the matrices stored in (9.22), node i computes

$$\left\{\mathbf{x}^{\downarrow\lambda(i)-\lambda(ch(i))}\right\} = W^{\omega_i\cap\lambda(ch(i))}_{(\mathbf{A}_i^{(i)}, \mathbf{b}_i^{(i)})}\left(\mathbf{x}^{\downarrow\lambda(i)\cap\lambda(ch(i))}\right) \tag{9.23}$$

to obtain the partial solution $\mathbf{x}^{\downarrow\lambda(i)} = (\mathbf{x}^{\downarrow\lambda(i)\cap\lambda(ch(i))}, \mathbf{x}^{\downarrow\lambda(i)-\lambda(ch(i))})$ with respect to its proper node label. At the end of the solution construction process, each node $i \in \lambda(i)$ contains the solution $\mathbf{x}$ to the system $\mathbf{AX} = \mathbf{b}$ projected to its node label, and the complete solution $\mathbf{x}$ can simply be obtained by assembling these partial solutions. This shows how factorized, symmetric, positive definite systems are solved using local computation. The sparsity reflected by the join tree is maintained all the time. A very similar scheme for the computation of solutions in quasi-regular semiring systems will be presented in Section 9.3.2. Here, we next focus on the adaption of $\mathbf{LDR}$-decomposition to symmetric, positive definite systems.

9.2.3 Symmetric Gaussian Elimination

Consider again a system $\mathbf{AX} = \mathbf{b}$, where $\mathbf{A}$ is a symmetric, positive definite $n \times n$ matrix, and $\mathbf{X}$ and $\mathbf{b}$ are vectors with corresponding dimensions. Since positive definite systems are always regular, we could apply the local computation scheme based on the $\mathbf{LDR}$-decomposition from Section 9.1.3, if we have to solve this system multiple times for different vectors $\mathbf{b}$. However, here we are dealing with a different valuation algebra such that parts of this discussion must be revisited. Moreover, the approach presented in this section does not only exploit the sparsity of the system, but also benefits from the symmetry in the matrix $\mathbf{A}$. Because this matrix is regular, we can use any variable elimination sequence in the Gaussian method to solve the system. As in Section 9.1.3, we renumber the variables such that they are eliminated in the order $X_1, \ldots, X_n$. We only need to permute the columns and the rows of the matrix $\mathbf{A}$ accordingly. By the symmetry of the matrix, the same permutation must be used for both columns and rows. In addition, it follows by symmetry that the $\mathbf{LDR}$-decomposition satisfies $\mathbf{R} = \mathbf{L}^T$ such that

$$\mathbf{A} \quad = \quad \mathbf{LDL}^T.$$

For positive definite matrices, $\mathbf{D}$ has positive diagonal entries. It also holds that

$$\mathbf{A} \quad = \quad \mathbf{GG}^T,$$

where $\mathbf{G} = \mathbf{LD}^{1/2}$. This decomposition is due to Choleski (Forsythe & Moler, 1967).

The symmetry of the matrix $\mathbf{A}$ can be exploited for the elimination of variables, where it is now sufficient to store the lower (or upper) half of the matrix $\mathbf{A}$ and all derived matrices. Then, equation (9.9) still applies: For $j, i = k+1, \ldots, n$ we have

$$l_{j,k} \quad = \quad \frac{a_{j,k}^{(k-1)}}{a_{k,k}^{(k-1)}} \quad = \quad d_{k,k}a_{j,k}^{(k-1)} \tag{9.24}$$

and according to equation (9.8) also

$$a_{j,i}^{(k)} \quad = \quad a_{j,i}^{(k-1)} - l_{j,k}a_{k,i}^{(k-1)} \quad \text{and} \quad b_j^{(k)} \quad = \quad b_j^{(k-1)} - l_{j,k}b_k^{(k-1)}.$$

Only the elements $a_{j,i}^{(k)}$ for $k \leq j \leq i$ must be computed due to the symmetry in the matrix $\mathbf{A}$. We again remark that in a symmetric, positive definite matrix the diagonal elements $d_{k,k}$ are positive in each step $k = 1, \ldots, n$. They can thus be used as pivot elements (Schwarz, 1997). Once the decomposition $\mathbf{A} = \mathbf{LDL}^T$ is found, the solution to the system is obtained by solving the triangular systems $\mathbf{LY} = \mathbf{b}$, $\mathbf{DZ} = \mathbf{Y}$ and $\mathbf{L}^T\mathbf{X} = \mathbf{Z}$, or alternatively by solving $\mathbf{LY} = \mathbf{b}$ and $\mathbf{L}^T\mathbf{X} = \mathbf{D}^{-1}\mathbf{Y}$.

We now return to local computation to avoid fill-ins in the set of zero elements Z given in equation (9.18). Since we are still dealing with a valuation algebra without neutral elements, we again assume a covering join tree for the factorization of equation (9.17) and further that the join tree edges are directed towards the root node m. As usual, the nodes are numbered such that if node j is on the path form node i to node m, then $i < j$. Each node $i \in V$ contains a factor ϕ_i with $\omega_i = d(\phi_i) \subseteq \lambda(i)$. If we execute the collect algorithm as in Section 9.2.2, the computation of each message $\mu_{i \rightarrow ch(i)}$ requires to eliminate the variables in

$$\omega_i^{(i)} - \lambda(ch(i)) \quad = \quad \lambda(i) - \lambda(ch(i)).$$

See equation 9.21. This allows us to determine a corresponding variable elimination sequence. Because the matrix $\mathbf{A}$ is regular, we may without loss of generality assume that the variables are numbered such that $X_1, \ldots, X_n$ is a variable elimination sequence that corresponds to the sequence of messages. Note that a possible renumbering of variables matches a permutation of the rows and columns of the matrix $\mathbf{A}$ and the vector $\mathbf{b}$.

We claim that the factors $a_{j,h}^{(k)}$ remain zero for all elimination steps k, if $(j,h) \in Z$.

Theorem 9.4 *For all* $(j,h) \in Z$ *and* $k = 0, \ldots, n-1$ *we have* $a_{j,h}^{(k)} = 0$ *and* $l_{j,h} = 0$.

Proof: We prove the theorem by induction: For $k = 0$ the proposition holds by the definition of Z. We note also that $l_{j,1} = 0$ for all $(j, 1) \in Z$. This follows from equation (9.24). So, let us assume that the proposition holds for $k-1$ and also that $l_{j,k} = 0$ for all $(j,k) \in Z$. From equation (9.8) follows that $a_{j,h}^{(k)} = 0$, if $(j,h) \in Z$. Then, by equation (9.24) it also follows that $l_{j,k+1} = 0$ for $(j, k+1) \in Z$. ■

The theorem states that the non-zero pattern defined by the join tree decomposition of the matrix $\mathbf{A}$ is maintained in the $\mathbf{LDL}^T$ factorization of $\mathbf{A}$.

We next observe that all diagonal, square submatrices of a lower triangular matrix are lower triangular too. So, for $s \subseteq r = \{X_1, \ldots, X_n\}$ the submatrix $\mathbf{L}^{\downarrow s,s}$ is lower triangular as well as $\mathbf{L}^{\downarrow t-s,t-s}$ and $\mathbf{L}^{\downarrow s\cap t, s\cap t}$ with $s \cup t = r$. This situation is schematically represented in Figure 9.4.

Subsequently, we assume that the factors $a_{j,h}^{(k)}$ or the l-elements are still available from an earlier run of the collect algorithm, see equation (9.24). This means that each node $i \in V$ contains the matrix $\mathbf{L}^{\downarrow \omega_i^{(i)}, \omega_i^{(i)}}$ which is lower triangular as remarked above. Using these matrices, we now discuss how the system $\mathbf{AX} = \mathbf{b}$ can be solved

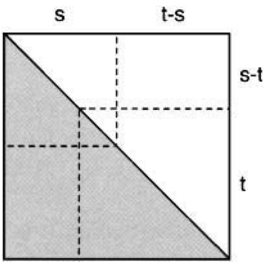


Figure 9.4 Square submatrices of a lower triangular matrix are lower triangular too.

by $\mathbf{LDL}^T$-decomposition and local computation. In a first step, we solve the system $\mathbf{LY} = \mathbf{b}$ by a similar collect phase as in the foregoing Section 9.2.2. Thus, we directly explain the computations performed by node $i \in V$ in the step $i = 1, \ldots, m-1$. At step i of the generalized collect algorithm, node i computes the message to its child node by eliminating the variables

$$\mathbf{Y}^{\downarrow\omega_i^{(i)}-\lambda(ch(i))} = \mathbf{Y}^{\downarrow\lambda(i)-\lambda(ch(i))}$$

(see equation (9.21)) from the system

$$\mathbf{L}^{\downarrow\omega_i^{(i)},\omega_i^{(i)}}\mathbf{Y}^{\downarrow\omega_i^{(i)}} = \mathbf{b}^{(i)\downarrow\omega_i^{(i)}}.$$

From the decomposition

$$\begin{aligned}
\mathbf{L}^{\downarrow\lambda(i)-\lambda(ch(i)),\lambda(i)-\lambda(ch(i))}\mathbf{Y}^{\downarrow\lambda(i)-\lambda(ch(i))} &= \mathbf{b}^{(i)\downarrow\lambda(i)-\lambda(ch(i))},\\
\mathbf{L}^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i)),\lambda(i)-\lambda(ch(i))}\mathbf{Y}^{\downarrow\lambda(i)-\lambda(ch(i))} &+ \\
\mathbf{L}^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i)),\omega_i^{(i)}\cap\lambda(ch(i))}\mathbf{Y}^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i))} &= \mathbf{b}^{(i)\downarrow\omega_i^{(i)}\cap\lambda(ch(i))}
\end{aligned}$$

we obtain

$$\mathbf{y}^{\downarrow\lambda(i)-\lambda(ch(i))} = (\mathbf{L}^{\downarrow\lambda(i)-\lambda(ch(i)),\lambda(i)-\lambda(ch(i))})^{-1}\mathbf{b}^{(i)\downarrow\lambda(i)-\lambda(ch(i))}.$$

Note that due to the triangularity of the involved matrix, it is a simple stepwise process to compute $\mathbf{y}^{\downarrow\lambda(i)-\lambda(ch(i))}$. This solution is introduced into the second part of the system which gives after rearrangement

$$\begin{aligned}
\mathbf{L}^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i)),\omega_i^{(i)}\cap\lambda(ch(i))}\mathbf{Y}^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i))} &= \\
\mathbf{b}^{(i)\downarrow\omega_i^{(i)}\cap\lambda(ch(i))} - \mathbf{L}^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i)),\lambda(i)-\lambda(ch(i))}\mathbf{y}^{\downarrow\lambda(i)-\lambda(ch(i))}&.
\end{aligned}$$

We define the vector

$$\mathbf{b}^{(ch(i))} = \begin{bmatrix} \mathbf{b}^{(i)\downarrow\omega_i^{(i)}\cap\lambda(ch(i))} - \mathbf{L}^{\downarrow\omega_i^{(i)}\cap\lambda(ch(i)),\lambda(i)-\lambda(ch(i))}\mathbf{y}^{\downarrow\lambda(i)-\lambda(ch(i))} \\ \mathbf{b}^{(i)\downarrow u-\omega_i^{(i)}} \end{bmatrix},$$

where u is the union of all $\lambda(k)$ for $k = i, \ldots, m$. So, node i sends the message

$$\mu_{i \to \lambda(ch(i))} \quad = \quad -\mathbf{L}^{\downarrow \omega_i^{(i)} \cap \lambda(ch(i)), \lambda(i) - \lambda(ch(i))} \mathbf{y}^{\downarrow \lambda(i) - \lambda(ch(i))}$$

to its child node. The child node combines the message with its content $\mathbf{b}^{(i) \downarrow \omega_{ch(i)}^{(i)}}$:

$$\mathbf{b}^{(i) \downarrow \omega_{ch(i)}^{(i)}} \otimes \mu_{i \to ch(i)} \quad = \quad \left(\mathbf{b}^{(i) \downarrow \omega_{ch(i)}^{(i)}}\right)^{\uparrow \omega_{ch(i)}^{(i)} \cup (\omega_i^{(i)} \cap \lambda(ch(i)))} - \left(\mathbf{L}^{\downarrow \omega_i^{(i)} \cap \lambda(ch(i)), \lambda(i) - \lambda(ch(i))} \mathbf{y}^{\downarrow \lambda(i) - \lambda(ch(i))}\right)^{\uparrow \omega_{ch(i)}^{(i)} \cup (\omega_i^{(i)} \cap \lambda(ch(i)))}.$$

We thus obtain for the domain of the updated node content:

$$\omega_{ch(i)}^{(i+1)} \quad = \quad \omega_{ch(i)}^{(i)} \cup \left(\omega_i^{(i)} \cap \lambda(ch(i))\right).$$

The content of all other nodes does not change at step i. At the end of the collect algorithm, the root node m contains the system

$$\mathbf{L}^{\downarrow \lambda(m), \lambda(m)} \mathbf{Y}^{\downarrow \lambda(m)} \quad = \quad \mathbf{b}^{(m) \downarrow \lambda(m)}$$

from which we obtain

$$\mathbf{y}^{\downarrow \lambda(m)} \quad = \quad \left(\mathbf{L}^{\downarrow \lambda(m), \lambda(m)}\right)^{-1} \mathbf{b}^{(m) \downarrow \lambda(m)}.$$

This clearly shows how the collect algorithm operates locally on the join tree of the decomposition (9.17). At the end of the collect algorithm, each node $i = 1, \ldots, m-1$ contains the partial solution $\mathbf{y}^{\downarrow \lambda(i) - \lambda(ch(i))}$, and the root node contains $\mathbf{y}^{\downarrow \lambda(m)}$. Remark also that no solution extension phase is necessary, because the involved matrices are all lower triangular. The total solution $\mathbf{y}$ could be obtained from Lemma 8.4, although this is not necessary for the second part of the process.

As in Section 9.2.2 we now solve the system $\mathbf{DL}^T\mathbf{X} = \mathbf{Y}$ by solution extension. The root node m solves the system

$$(\mathbf{DL}^T)^{\downarrow \lambda(m), \lambda(m)} \mathbf{X}^{\downarrow \lambda(m)} \quad = \quad \mathbf{y}^{\downarrow \lambda(m)}$$

and finds the partial solution

$$\mathbf{x}^{\downarrow \lambda(m)} \quad = \quad \left((\mathbf{DL}^T)^{\downarrow \lambda(m), \lambda(m)}\right)^{-1} \mathbf{y}^{\downarrow \lambda(m)}. \tag{9.25}$$

These computations are simple because $\mathbf{DL}^T$ is upper triangular. Assume now that node $i = m-1, \ldots, 1$ obtains the partial solution $\mathbf{x}^{\downarrow \lambda(i) \cap \lambda(ch(i))}$ from its child node. Then, node i computes equation (9.22) with $\mathbf{A}$ replaced by $\mathbf{DL}^T$ and $\mathbf{b}$ by $\mathbf{y}$:

$$\left((\mathbf{DL}^T)^{\downarrow \lambda(i) - \lambda(ch(i)), \lambda(i) - \lambda(ch(i))}\right)^{-1} \mathbf{y}^{\downarrow \lambda(i) - \lambda(ch(i))} \quad - \quad \left((\mathbf{DL}^T)^{\downarrow \lambda(i) - \lambda(ch(i)), \lambda(i) - \lambda(ch(i))}\right)^{-1} \left((\mathbf{DL}^T)^{\downarrow \omega_i^{(m)} \cap \lambda(ch(i)), \omega_i^{(m)} \cap \lambda(ch(i))}\right) \mathbf{x}^{\downarrow \omega_i^{(m)} \cap \lambda(ch(i))} \quad = \quad \mathbf{x}^{\downarrow \lambda(i) - \lambda(ch(i))} \tag{9.26}$$

and obtains

$$\mathbf{x}^{\downarrow\lambda(i)} = \left(\mathbf{x}^{\downarrow\lambda(i)-\lambda(ch(i))}, \mathbf{x}^{\downarrow\lambda(i)\cap\lambda(ch(i))}\right).$$

However, there is actually no need to compute the inverse matrices that occur in this formula explicitly. Instead, $\mathbf{x}^{\downarrow\lambda(i)-\lambda(ch(i))}$ corresponds to the solution to the system

$$\begin{aligned} \left((\mathbf{DL}^T)^{\downarrow\lambda(i)-\lambda(ch(i)),\lambda(i)-\lambda(ch(i))}\right)\mathbf{X}^{\downarrow\lambda(i)-\lambda(ch(i))} &= \\ \mathbf{y}^{\downarrow\lambda(i)-\lambda(ch(i))} &- \left((\mathbf{DL}^T)^{\downarrow\omega_i^{(m)}\cap\lambda(ch(i)),\omega_i^{(m)}\cap\lambda(ch(i))}\right)\mathbf{x}^{\downarrow\omega_i^{(m)}\cap\lambda(ch(i))}, \end{aligned}$$

and solving this system is simple because only triangular matrices are involved. This process is repeated for $i = m-1,\ldots,1$. At the end of the solution construction process, each node $i \in V$ contains the partial solution $\mathbf{x}^{\downarrow\lambda(i)}$ which can then be aggregated to the total solution $\mathbf{x}$ of $\mathbf{AX} = \mathbf{b}$.

We given an example of this process:

Example 9.3 *Consider the join tree of Figure 9.5 where the nodes are labeled with index sets. This is suitable because we are going to solve two systems with different variable vectors* $\mathbf{X}$ *on the same join tree. The variables are already numbered according to an elimination sequence that corresponds to this join tree, if node* 3 *is taken as root. The symmetric, positive definite matrices* $\mathbf{A}_1$, $\mathbf{A}_2$ *and* $\mathbf{A}_3$ *are then written as*

$$\mathbf{A}_1 = \begin{bmatrix} a^1_{1,1} & a^1_{1,2} & a^1_{1,4} \\ a^1_{2,1} & a^1_{2,2} & a^1_{2,4} \\ a^1_{4,1} & a^1_{4,2} & a^1_{4,4} \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} a^2_{3,3} & a^2_{3,4} \\ a^2_{4,3} & a^2_{4,4} \end{bmatrix}, \quad \mathbf{A}_3 = \begin{bmatrix} a^3_{4,4} & a^3_{4,5} & a^3_{4,6} \\ a^3_{5,4} & a^3_{5,5} & a^3_{5,6} \\ a^3_{6,4} & a^3_{6,5} & a^3_{6,6} \end{bmatrix}.$$

Similarly, we define the vectors $\mathbf{b}_1$, $\mathbf{b}_2$ *and* $\mathbf{b}_3$ *associated with the three matrices. This defines three valuation* $\phi_i = (\mathbf{A}_i, \mathbf{b}_i)$ *for* $i = 1,2,3$, *which combine to the total system* $\phi = \phi_1 \otimes \phi_2 \otimes \phi_2$ *representing the valuation* $\phi = (\mathbf{A}, \mathbf{b})$ *with*

$$\mathbf{A} = \mathbf{A}_1^{\uparrow\{1,2,3,4,5,6\}} + \mathbf{A}_2^{\uparrow\{1,2,3,4,5,6\}} + \mathbf{A}_3^{\uparrow\{1,2,3,4,5,6\}}$$

and

$$\mathbf{b} = \mathbf{b}_1^{\uparrow\{1,2,3,4,5,6\}} + \mathbf{b}_2^{\uparrow\{1,2,3,4,5,6\}} + \mathbf{b}_3^{\uparrow\{1,2,3,4,5,6\}}$$

Note also that all join tree nodes are filled in this simple example, i.e. $\omega_i = \lambda(i)$. *We further observe that the matrix* $\mathbf{A}$ *has the following non-zero structure*

$$\mathbf{A} = \begin{bmatrix} x & x & & x & & \\ x & x & & x & & \\ & & x & x & & \\ x & x & x & x & x & x \\ & & & x & x & x \\ & & & x & x & x \end{bmatrix},$$

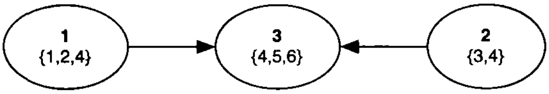


Figure 9.5 The join tree of Example 9.3.

where x denotes a non-zero element.

If we execute the collect algorithm on the join tree of Figure 9.5, we first eliminate the variables X_1 and X_2 from the system in node 1 to obtain the message sent from node 1 to node 3. Then, variable X_3 is eliminated for the message of node 2 to node 3. Finally, on node 3 the variables X_4 and X_5 are eliminated. This gives the lower triangular matrix $\mathbf{L}$ with the following non-zero structure:

$$\mathbf{L} = \begin{bmatrix} 1 & & & & & \\ x & 1 & & & & \\ & & 1 & & & \\ x & x & x & 1 & & \\ & & & x & 1 & \\ & & & x & x & 1 \end{bmatrix}.$$

We see that this matrix maintains the zero-pattern captured by the join tree and exhibited in the matrix $\mathbf{A}$. Note also that the submatrices like $\mathbf{L}^{\downarrow\{1,2,4\},\{1,2,4\}}$ and $\mathbf{L}^{\downarrow\{3,4\},\{3,4\}}$ are still lower triangular.

Let us now consider the computations of the collect phase for solving $\mathbf{LY} = \mathbf{b}$ in more detail. In the first step, we eliminate the variables Y_1 and Y_2 to obtain the message that is sent from node 1 to node 3. We thus have

$$\mathbf{L}^{\downarrow\{1,2\},\{1,2\}}\mathbf{Y}^{\downarrow\{1,2\}} = \mathbf{b}^{\downarrow\{1,2\}}.$$

Solving this system is very simple because the matrix $\mathbf{L}^{\downarrow\{1,2\},\{1,2\}}$ is lower triangular. In fact, from

$$Y_1 = b_1 \quad \textit{and} \quad l_{2,1}Y_1 + Y_2 = b_2$$

we obtain

$$y_1 = b_1 \quad \textit{and} \quad y_2 = b_2 - l_{2,1}b_1.$$

This solution is introduced into the fourth equation on node 3 to obtain

$$l_{3,4}Y_3 + Y_4 = b_4 - l_{4,1}y_1 - l_{4,2}y_2.$$

The message of node 2 to node 3 determines the value of Y_3, which then allows to compute the solution values of variables Y_4, Y_5 and Y_6. This builds up the entire solution $\mathbf{y}$ of the system $\mathbf{LY} = \mathbf{b}$ and concludes the collect phase.

The solution to the system $\mathbf{AX} = \mathbf{b}$ *is then obtained by solving* $\mathbf{DL}^T\mathbf{X} = \mathbf{y}$ *by solution extension. The root node obtains from equation (9.25) the partial solution* $\mathbf{x}^{\downarrow\{X_4,X_5,X_6\}}$ *and sends the message* $\mathbf{x}^{\downarrow\{X_4\}}$ *to the nodes* 1 *and* 2*. The two receiving nodes compute the partial solutions with respect to their proper node label by equation (9.26). For example, node* 1 *solves the system*

$$\left(\mathbf{DL}^{T\ \downarrow\{X_1,X_2\},\{X_1,X_2\}}\right)\mathbf{X}^{\downarrow\{X_1,X_2\}} = \mathbf{y}^{\downarrow\{X_1,X_2\}} - \left(\mathbf{DL}^{T\ \downarrow\{X_4\},\{X_4\}}\right)\mathbf{x}^{\downarrow\{X_4\}}.$$

Finally, the total solution $\mathbf{x}$ *is build from* $\mathbf{x}^{\downarrow\{X_4,X_5,X_6\}}$, $\mathbf{x}^{\downarrow\{X_1,X_2\}}$ *and* $\mathbf{x}^{\downarrow\{X_3\}}$.

At the beginning of Section 9.2.2 we assumed in equation (9.17) a join tree decomposition of the valuation $(\mathbf{A}, \mathbf{b})$ that represents a symmetric, positive definite system. There are important applications where such factorizations occur naturally, as for example in the least squares method that will be discussed in Section 9.2.5 below. If, however, no natural prior factorization of the total system is given, then a corresponding decomposition must be found in order to benefit from local computation. How such decompositions are produced is the topic of the following section.

9.2.4 Symmetric Decompositions and Elimination Sequences

The problem of producing a good decomposition of a symmetric, positive definite system $\phi = (\mathbf{A}, \mathbf{b})$ is closely related to the problem of choosing a good elimination sequence for the construction of join trees that has already been addressed in Section 3.7. Here, we revisit parts of this discussion by following the graphical theory for symmetric, positive definite matrices proposed by (Rose, 1972). Interestingly, this can be done to a large extend without reference to an underlying valuation algebra, but the algebra becomes important for the discussion of applications like the least squares method in Section 9.2.5.

The non-zero pattern in an $n \times n$ matrix $\mathbf{A}$ of the symmetric, positive definite system ϕ can be represented by an undirected graph: The vertices of the graph $G = (V, E)$ correspond to the rows or, equivalently, to the columns of matrix $\mathbf{A}$. The vertex associated with row i is called v_i for $i = 1, \ldots, n$. We further introduce an edge $\{v_i, v_j\} \in E$ for any non-zero element $a_{i,j}$ of the matrix $\mathbf{A}$. This graphical structure reflects the *non-zero structure* of the matrix $\mathbf{A}$. The example below assumes a 7×7 matrix whose graph is shown Figure 9.6. We now apply the theory of Section 3.7.1 to this graph by first constructing a triangulation. In the triangulated graph, we then search for all maximum cliques and form a join tree whose nodes correspond to the identified cliques. Finally, we fix a node in the join tree as root, direct all edges towards it and renumber the nodes of the graph G by $n(i)$, such that the number of rows in a join tree node before another one on the path to the root is smaller. More precisely, if s and t are neighbor nodes with s before t on the path to the root, then $n(i) \leq n(j)$ for any $i \in s - t$ and $j \in t$.

Example 9.4 *The following table shows the non-zero pattern of a* 7×7 *symmetric matrix, where non-zero elements are represented by a symbol x,*

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	x				x		x
v_2		x	x	x	x	x	
v_3		x		x			
v_4		x	x			x	
v_5	x	x					x
v_6		x		x			
v_7	x				x		

The associated graph shown in Figure 9.6 has seven nodes v_1 *to* v_7*. We observe that it is already triangulated with maximal cliques* $\{v_1, v_5, v_7\}$*,* $\{v_2, v_5\}$*,* $\{v_2, v_3, v_4\}$ *and* $\{v_2, v_4, v_6\}$*. These cliques can be arranged in a join tree as shown in Figure 9.7. The node* $\{v_2, v_4, v_6\}$ *is chosen as root node.*

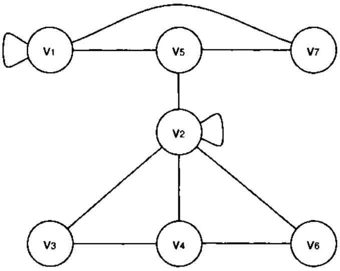


Figure 9.6 The graph representing the non-zero pattern of Example 9.4.

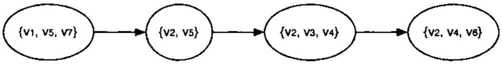


Figure 9.7 The join tree of the triangulated graph in Figure 9.6.

From the left to the right in the join tree of Figure 9.7, the variables are eliminated in the following sequence: first $\{v_1, v_7\}$*, then* $\{v_5\}$ *and* $\{v_3\}$ *until finally the variables* $\{v_2, v_4, v_6\}$ *in the root node remain. Hence, we may choose an elimination sequence from the set* $\{v_1, v_7\} \times \{v_5\} \times \{v_3\} \times \{v_2, v_4, v_6\}$*, for example* $(v_1, v_7, v_5, v_3, v_2, v_4, v_6)$*. This elimination sequence induces the following renumbering that satisfies the above requirement:*

v_1	v_2	v_3	v_4	v_5	v_6	v_7
1	*5*	*4*	*6*	*3*	*7*	*2*

Rearranging the rows and columns of the matrix according to this numbering gives

$$\begin{array}{c|ccccccc}
 & v_1 & v_7 & v_5 & v_3 & v_2 & v_4 & v_6 \\
\hline
v_1 & x & x & x & & & & \\
v_7 & x & & x & & & & \\
v_5 & x & x & & & x & & \\
v_3 & & & & & x & x & \\
v_2 & & & x & x & x & x & x \\
v_4 & & & & x & x & & x \\
v_6 & & & & & x & x &
\end{array}$$

This identifies the block structure exploited in local computation. The zeros outside the blocks will not be filled-in, if the variables are eliminated in the sequence defined by the renumbering. At the end of the collect algorithm applied to the above join tree, we therefore have the following non-zero pattern of the triangular matrix:

$$\begin{array}{c|ccccccc}
 & v_1 & v_7 & v_5 & v_3 & v_2 & v_4 & v_6 \\
\hline
v_1 & x & & & & & & \\
v_7 & x & x & & & & & \\
v_5 & x & x & x & & & & \\
v_3 & & & & x & & & \\
v_2 & & & x & x & x & & \\
v_4 & & & & x & x & x & \\
v_6 & & & & & x & x & x
\end{array}$$

All unmarked matrix entries remain zero during the complete local computation process. Alternatively, we could also select the node $\{2, 5\}$ as root and choose the elimination sequence $(v_1, v_7, v_6, v_3, v_4, v_5, v_2)$. This induces the renumbering:

v_1	v_2	v_3	v_4	v_5	v_6	v_7
1	*7*	*4*	*5*	*6*	*3*	*2*

Rearranging the rows and columns of the matrix according to this numbering gives

$$\begin{array}{c|ccccccc}
 & v_1 & v_7 & v_6 & v_3 & v_4 & v_5 & v_2 \\
\hline
v_1 & x & x & & & & x & \\
v_7 & x & & & & & x & \\
v_6 & & & & & x & & x \\
v_3 & & & & & x & & x \\
v_4 & & & x & x & & & x \\
v_5 & x & x & & & & & x \\
v_2 & & & x & x & x & x & x
\end{array}$$

In this case, variables X_1 and X_7 are eliminated first and expressed by X_5 in the original equations 1 *and* 7*. Then, in the original equations* 2, 4, *and* 6, *the variable X_6 is eliminated, i.e. it is expressed by X_2 and X_4. This process continues for the X_3 and X_4 until two equations for the variables X_2 and X_5 remain in the root node.*

We obtain the following non-zero pattern of the lower triangular matrix:

	v_1	v_7	v_6	v_3	v_4	v_5	v_2
v_1	x						
v_7	x	x					
v_6			x				
v_3				x			
v_4			x	x	x		
v_5	x	x				x	
v_2			x	x	x	x	x

These examples show that different numberings lead to different $\mathbf{LDL}^T$ *decompositions, but the non-zero pattern represented by the join tree is always respected.*

We next present an application where factorizations of symmetric, positive definite systems according to this valuation algebra occur in a natural manner.

9.2.5 An Application: The Least Squares Method

It is often the case in empirical studies that unknown parameters in a functional relationship have to be estimated from observations. Here, we assume a linear relationship between the unknown parameters. In general, there are much more observations than parameters which allows us to account for inevitable errors in the observations. This leads to overdetermined systems which cannot be solved exactly. Statistical methods are therefore needed to obtain estimates of the unknown parameters as for example the method of *least squares* proposed by Gauss. In this method the parameters are determined by minimizing the sum of squared residues in the linear equations. The minimization process leads to linear equations, the so-called *normal equations*, in the unknown parameters with a symmetric, positive definite matrix. So, the theory developed above applies. Moreover, we shall see that a decomposition of the original overdetermined systems leads to combining normal equations according to the combination rule of the valuation algebra introduced in the previous Section 9.2.1. In other words, we face here an important application of the theory developed so far.

■ 9.1 Least Squares Method

Consider a system of linear equations

$$\begin{array}{ccccccccccc} a_{1,1}X_1 & + & a_{1,2}X_2 & + & \cdots & + & a_{1,n}X_n & + & D_1 & = & b_1, \\ a_{2,1}X_1 & + & a_{2,2}X_2 & + & \cdots & + & a_{2,n}X_n & + & D_2 & = & b_2, \\ & & & & & & & & & \vdots & \\ a_{m,1}X_1 & + & a_{m,2}X_2 & + & \cdots & + & a_{m,n}X_n & + & D_m & = & b_m, \end{array} \tag{9.27}$$

where $m \geq n$, i.e. there are more equations than unknowns X_1 to X_n. Also, we have introduced the residues or differences D_i in order to balance the right and left-hand side of the equations. In matrix form, this system is written as $\mathbf{AX} + \mathbf{D} = \mathbf{b}$, where $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{D}$ and $\mathbf{b}$ are m-vectors. We assume that the matrix $\mathbf{A}$ has maximal rank n. The principal idea of the least squares method is to determine the unknowns X_1 to X_n such that the sum of the squares of the residues becomes minimal. This sum can be written as

$$\mathbf{D}^T\mathbf{D} = (\mathbf{b}-\mathbf{AX})^T(\mathbf{b}-\mathbf{AX}) = \mathbf{X}^T\mathbf{A}^T\mathbf{AX} - 2(\mathbf{A}^T\mathbf{b})^T\mathbf{X} + \mathbf{b}^T\mathbf{b}.$$

So, the function to be minimized is a quadratic function of the unknowns $\mathbf{X}$. For simplification we introduce the short-hand notations

$$\mathbf{C} = \mathbf{A}^T\mathbf{A} \quad \text{and} \quad \mathbf{c} = \mathbf{A}^T\mathbf{b},$$

where $\mathbf{C}$ is an $n \times n$ symmetric, positive definite matrix and $\mathbf{c}$ an n-vector. The sum of squares can then be written as

$$F(\mathbf{X}) = \mathbf{D}^T\mathbf{D} = \mathbf{X}^T\mathbf{CX} - 2\mathbf{c}^T\mathbf{X} + \mathbf{b}^T\mathbf{b} = \text{min!} \qquad (9.28)$$

The necessary condition that $F(\mathbf{X})$ takes a minimum is the vanishing of the gradient $\nabla F(\mathbf{X})$, whose i-th component can be determined from equation (9.28) as

$$\frac{\partial F(\mathbf{X})}{\mathbf{X}_i} = 2\sum_{j=1}^{n} c_{i,j}X_j - 2c_i.$$

We divide by 2 and obtain the system of equations $\mathbf{CX} - \mathbf{c} = \mathbf{0}$. These are the *normal equations* of the overdetermined system (9.27). They can also be written as

$$\mathbf{A}^T\mathbf{AX} = \mathbf{A}^T\mathbf{b}.$$

Since the matrix $\mathbf{A}^T\mathbf{A}$ is positive definite, the unknowns are uniquely determined by

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}.$$

This is the so-called *least squares estimate* of the unknown parameters in the system (9.27). If we compute the second order derivatives of $F(\mathbf{X})$ we obtain $2\mathbf{C}$. Since $\mathbf{C}$ is positive definite, we conclude that the solutions of the normal equations indeed minimize the function $F(\mathbf{X})$. Statistical theory justifies this way of determining the parameters, in particular, if certain probabilistic assumptions are made about the residues. We reconsider this in Chapter 10 and derive similar results by a very different method based on an isomorphic valuation algebra.

In order to solve the normal equations we may proceed as described in Section 9.2.3 and use local computation to exploit the sparsity of the normal equations.

In this respect, we prove a remarkable result relating the original overdetermined system to its normal equations. In fact, already the original system may be very sparse. More precisely, we assume that we may capture this sparsity by a covering join tree for the original system. Thus, let $r = \{1,\ldots,n\}$ be the index set of the variables of the system and (V, E, λ, D) a join tree with $D = \mathcal{P}(r)$. We now assume that this join tree covers the system (9.27). If $V = \{1,\ldots,l\}$, then this means that after permuting the rows of the system of equations, its matrix $\mathbf{A}$ decomposes into submatrices $\mathbf{A}_i$ for $i = 1,\ldots,l$, where $\mathbf{A}_i$ is an $m_i \times s_i$ matrix such that $s_i \subseteq \lambda(i)$ and $m_1 + \cdots + m_l = m$. Therefore, every pair $(\mathbf{A}_i, \mathbf{b}_i)$ is covered by some node of the join tree and thus also every equation of the system (9.27). It is not excluded that some of these systems are empty, i.e. $m_i = 0$. The following theorem claims that the corresponding normal system is decomposed into normal systems of the subsystems of the decomposition above and covered by the same join tree. Here, $\otimes$ denotes the combination operation of the valuation algebra of Section 9.2.1.

Theorem 9.5 *Let* $\mathbf{AX}+\mathbf{D} = \mathbf{b}$ *be an overdetermined system and* $\mathbf{A}$ *an* $m\times n$ *matrix of full column rank* n. *Let further* $(V, E, \lambda, \mathcal{P}(r))$ *be a join tree with* $V = \{1,\ldots,l\}$. *If, after permutation of the equations of the system* $\mathbf{AX} + \mathbf{D} = \mathbf{b}$, *the matrix* $\mathbf{A}$ *decomposes into* $m_i \times s_i$ *matrices* $\mathbf{A}_i$ *and* $\mathbf{b}$ *correspondingly into* m_i*-vectors* $\mathbf{b}_i$ *for* $i = 1,\ldots,l$, *such that*

1. $s_i \subseteq \lambda(i)$;
2. $m_1 + \cdots + m_l = m$;
3. *all matrices* $\mathbf{A}_i$ *have column rank* $|s_i|$;

then for $\mathbf{C} = \mathbf{A}^T\mathbf{A}$ *and* $\mathbf{c} = \mathbf{A}^T\mathbf{b}$

$$(\mathbf{C},\mathbf{c}) \quad = \quad \bigotimes_{i=1}^{l}(\mathbf{C}_i,\mathbf{c}_i),$$

where $\mathbf{C}_i = \mathbf{A}_i^T\mathbf{A}_i$ *and* $\mathbf{c}_i = \mathbf{A}_i^T\mathbf{b}_i$.

Proof: The proof goes by induction over the nodes of the join tree. We select node $l \in V$ as a root and direct all edges towards this root. Further, we number the nodes such that $i < j$ if node j is on the path from i to the root l. Then, node 1 is necessarily a leaf. Let $s_1 = s$ to simplify notation and

$$t \quad = \quad \bigcup_{i=2}^{l} s_i,$$

If the system of equations covered by this node is empty, then we may eliminate this node directly and proceed to the next node in the numbering. Otherwise, the first m_1 equations contain only variables from s. We then decompose the columns of the matrix $\mathbf{A}_1$ of the first m_1 equations according to $s - t$ and $s \cap t$,

$$\mathbf{A}_1 \quad = \quad \left[\begin{array}{cc} \mathbf{A}_{1,s-t} & \mathbf{A}_{1,s\cap t} \end{array}\right].$$

The matrix $\mathbf{B}$ of the remaining $m - m_1$ equations contains only variables from the set t. We decompose B according to $s \cap t$ and t into

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} & \mathbf{B}_{s\cap t} & \mathbf{B}_{t-s} \end{bmatrix}.$$

Then the total matrix $\mathbf{A}$ of the system may be decomposed into

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,s-t} & \mathbf{A}_{1,s\cap t} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{s\cap t} & \mathbf{B}_{t-s} \end{bmatrix}.$$

We see that

$$\begin{aligned}
\mathbf{A}^T\mathbf{A} &= \begin{bmatrix} \mathbf{A}_{1,s-t}^T & \mathbf{0} \\ \mathbf{A}_{1,s\cap t}^T & \mathbf{B}_{s\cap t}^T \\ \mathbf{0} & \mathbf{B}_{t-s}^T \end{bmatrix} \begin{bmatrix} \mathbf{A}_{1,s-t} & \mathbf{A}_{1,s\cap t} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{s\cap t} & \mathbf{B}_{t-s} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_{1,s-t}^T\mathbf{A}_{1,s-t} & \mathbf{A}_{1,s-t}^T\mathbf{A}_{1,s\cap t} & \mathbf{0} \\ \mathbf{A}_{1,s\cap t}^T\mathbf{A}_{1,s-t} & \mathbf{A}_{1,s\cap t}^T\mathbf{A}_{1,s\cap t} + \mathbf{B}_{s\cap t}^T\mathbf{B}_{s\cap t} & \mathbf{B}_{s\cap t}^T\mathbf{B}_{t-s} \\ \mathbf{0} & \mathbf{B}_{t-s}^T\mathbf{B}_{t\cap s} & \mathbf{B}_{t-s}^T\mathbf{B}_{t-s} \end{bmatrix} \\
&= (\mathbf{A}_1^T\mathbf{A}_1)^{\uparrow s\cup t} + (\mathbf{B}^T\mathbf{B})^{\uparrow s\cup t}.
\end{aligned}$$

The vector $\mathbf{b}$ is decomposed in the same way with respect to the first m_1 and the remaining $m - m_1$ equations:

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{c} \end{bmatrix}.$$

Then, we see in the same way that

$$\mathbf{A}^T\mathbf{b} = (\mathbf{A}_1^T\mathbf{b}_1)^{\uparrow s\cup t} + (\mathbf{B}^T\mathbf{c})^{\uparrow s\cup t}$$

and finally conclude that

$$\left(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{b}\right) = \left(\mathbf{A}_1^T\mathbf{A}_1, \mathbf{A}_1^T\mathbf{b}_1\right) \otimes \left(\mathbf{B}^T\mathbf{B}, \mathbf{B}^T\mathbf{c}\right).$$

If node 1 in the join tree is eliminated, the remaining tree is still a join tree and node $ch(1)$ is now a leaf. The remaining system of $m - m_1$ equations with matrix $\mathbf{B}$ and right-hand side $\mathbf{c}$ can then be treated as the system above and this process may be repeated until the remaining matrix is $\mathbf{A}_l$ and the right-hand side is $\mathbf{b}_l$. If $t = t_1$ and t_i for $i = 2, \ldots l - 1$ denote the variables in the remaining systems obtained during this process respectively, such that $t_{l-1} = s_l$, we obtain in this way

$$\begin{aligned}
\mathbf{A}^T\mathbf{A} &= (\mathbf{A}_1^T\mathbf{A}_1)^{\uparrow s_1\cup t_1} + ((\mathbf{A}_2^T\mathbf{A}_2)^{\uparrow s_2\cup t_2} \\
&+ (\cdots + (\mathbf{A}_l^T\mathbf{A}_l)^{\uparrow s_{l-1}\cup s_l})^{\uparrow s_{l-2}\cup s_{l-1}} \ldots)^{\uparrow s_2\cup t_2})^{\uparrow s_1\cup t_1} \\
&= (\mathbf{A}_1^T\mathbf{A}_1)^{\uparrow r} + (\mathbf{A}_2^T\mathbf{A}_2)^{\uparrow r} + \cdots + (\mathbf{A}_l^T\mathbf{A}_l)^{\uparrow r},
\end{aligned}$$

since $s_1 \cup t_1 = r$. Similarly, we also obtain

$$\mathbf{A}^T\mathbf{b} = (\mathbf{A}_1^T\mathbf{b}_1)^{\uparrow r} + (\mathbf{A}_2^T\mathbf{b}_2)^{\uparrow r} + \cdots + (\mathbf{A}_l^T\mathbf{b}_l)^{\uparrow r}.$$

But this means that

$$(\mathbf{A}^T\mathbf{A}, \mathbf{A}^T\mathbf{b}) \quad = \quad \bigotimes_{i=1}^{l}(\mathbf{A}_i^T\mathbf{A}_i, \mathbf{A}_i^T\mathbf{b}_i).$$

which proves the theorem. ■

Local computation in the valuation algebra of Section 9.2.1 is thus applicable to the least squares method for linear equations, where a factorization of the normal equation is induced by a join tree decomposition of the original overdetermined system. We shall illustrate this more concretely in the following application. Note also that if some factor in the decomposition of the original system is empty, then the induced normal system corresponds to the identity element, see Section 3.9. Finally, we refer to Chapter 10 where closely related problems are studied from a different perspective. In particular, the assumption of Theorem 9.5 that the matrix of the system has full column rank will be dropped.

■ 9.2 Smoothing and Filtering in Linear Dynamic System

Let $\mathbf{X}_0, \mathbf{X}_1, \mathbf{X}_2$ and $\mathbf{X}_3$ be n-dimensional variable vectors that satisfy

$$\begin{array}{rcrcl} \mathbf{X}_1 & = & \mathbf{A}_0\mathbf{X}_0 & + & \mathbf{D}_0, \\ \mathbf{X}_2 & = & \mathbf{A}_1\mathbf{X}_1 & + & \mathbf{D}_1, \\ \mathbf{X}_3 & = & \mathbf{A}_2\mathbf{X}_2 & + & \mathbf{D}_2. \end{array}$$

The vectors $\mathbf{X}_i$ may be imagined as unknown state vectors of some system, and the equations describe how these states change form time $i-1$ to i, here for $i \in \{1,2,3\}$. The residues $\mathbf{D}_i$ represent unknown disturbances which influence the development of the state over time. The matrices $\mathbf{A}_i$ are all regular $n \times n$ matrices. State vectors may not be observed directly, but only through some measurement devices which are described by the equations

$$\begin{array}{rcrcl} \mathbf{H}_0\mathbf{X}_0 & + & \mathbf{E}_0 & = & \mathbf{y}_0, \\ \mathbf{H}_1\mathbf{X}_1 & + & \mathbf{E}_1 & = & \mathbf{y}_1, \\ \mathbf{H}_2\mathbf{X}_2 & + & \mathbf{E}_2 & = & \mathbf{y}_2. \end{array}$$

The matrices $\mathbf{H}_i$ are $m \times n$ matrices and the vectors $\mathbf{y}_i$, representing observed values, give information about the unknown states. The residues $\mathbf{E}_i$ again represent unknown measurement errors. The task is to estimate the values of the state vectors from these equations and the observations. We first bring the equations into the form of system (9.27):

$$\begin{array}{ccccccccccc} \mathbf{A}_0\mathbf{X}_0 & - & \mathbf{X}_1 & & & & & + & \mathbf{D}_0 & = & \mathbf{0}, \\ \mathbf{H}_0\mathbf{X}_0 & & & & & & & + & \mathbf{E}_0 & = & \mathbf{y}_0, \\ & & \mathbf{A}_1\mathbf{X}_1 & - & \mathbf{X}_2 & & & + & \mathbf{D}_1 & = & \mathbf{0}, \\ & & \mathbf{H}_1\mathbf{X}_1 & & & & & + & \mathbf{E}_1 & = & \mathbf{y}_1, \\ & & & & \mathbf{A}_2\mathbf{X}_2 & - & \mathbf{X}_3 & + & \mathbf{D}_2 & = & \mathbf{0}, \\ & & & & \mathbf{H}_2\mathbf{X}_2 & & & + & \mathbf{E}_2 & = & \mathbf{y}_2. \end{array}$$

We observe that the system can be decomposed into three subsystems, each consisting of two consecutive equations that are covered by the three nodes of the simple join tree in Figure 9.8. In order to apply Theorem 9.5, we assume that the subsystems have full column rank, i.e. that $m \geq n$, and that $\mathbf{H}_i$ have full column rank too. If we switch to the valuation algebra of symmetric, positive definite systems, then by Theorem 9.5 the normal equations of the total system can be decomposed into

$$(\mathbf{C}_0, \mathbf{c}_0) \otimes (\mathbf{C}_1, \mathbf{c}_1) \otimes (\mathbf{C}_2, \mathbf{c}_2),$$

where

$$\mathbf{C}_i = \begin{bmatrix} \mathbf{A}_i & -\mathbf{I} \\ \mathbf{H}_i & \mathbf{0} \end{bmatrix}^T \begin{bmatrix} \mathbf{A}_i & -\mathbf{I} \\ \mathbf{H}_i & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_i^T\mathbf{A}_i + \mathbf{H}_i^T\mathbf{H}_i & -\mathbf{A}_i^T \\ -\mathbf{A}_i & \mathbf{I} \end{bmatrix},$$

and

$$\mathbf{c}_i = \begin{bmatrix} \mathbf{A}_i & -\mathbf{I} \\ \mathbf{H}_i & \mathbf{0} \end{bmatrix}^T \begin{bmatrix} \mathbf{0} \\ \mathbf{y}_i \end{bmatrix} = \begin{bmatrix} \mathbf{H}_i^T\mathbf{y}_i \\ \mathbf{0} \end{bmatrix}$$

for $i \in \{0, 1, 2\}$. This allows us to apply local computation for the solution of the normal equations using the collect algorithm as described in Section 9.2.2 or the fusion algorithm. If we select the rightmost node in Figure 9.8 as the root, we eliminate first $\mathbf{X}_0$ and then $\mathbf{X}_1$, i.e. we express $\mathbf{X}_0$ in terms of $\mathbf{X}_1$ and then $\mathbf{X}_1$ in terms of $\mathbf{X}_2$. The variables $\mathbf{X}_2$ are eliminated in the last node to obtain $\mathbf{X}_3$. Determining the last state, given the past and present measurements is called a *filter solution*. Then, by backward substituting, the least squares estimates of the other states can be computed. This is called the *smoothing solution*.

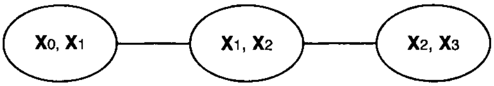


Figure 9.8 The join tree covering the equations of the linear dynamic system.

We examine this procedure a bit more closely. In terms of message-passing, the message from the leftmost node $\mathbf{X}_0, \mathbf{X}_1$ in the join tree of Figure 9.8 corresponds to the elimination of $\mathbf{X}_0$. The message is the projection of $(\mathbf{C}_0, \mathbf{c}_0)$ to the variables $\mathbf{X}_1$ which, according to (9.13) and (9.14), is

$$\left(\Sigma_{0,1}, \nu_{0,1}\right) = \left(\mathbf{A}_0\Sigma_0^{-1}\mathbf{A}_0^T + \mathbf{I}, \mathbf{A}_0\Sigma_0^{-1}\mathbf{H}_0^T\mathbf{y}_0\right),$$

where

$$\Sigma_0 = \mathbf{A}_0^T\mathbf{A}_0 + \mathbf{H}_0^T\mathbf{H}_0.$$

This matrix is the sum of two positive definite matrices and is thus positive definite itself, i.e. it has an inverse. The elimination process for $\mathbf{X}_0$ in the first subsystem leads to the equation

$$\mathbf{X}_0 = \mathbf{\Sigma}_0^{-1}\mathbf{A}_0^T\mathbf{X}_1 + \mathbf{\Sigma}_0^{-1}\mathbf{H}_0^T\mathbf{y}_0,$$

which can later be used for backward substitution, once $\mathbf{X}_1$ is determined. Passing the message to the next node of the join tree and combining it to the node content is equivalent to substituting this expression for $\mathbf{X}_0$ in the second subsystem. More generally, let us define for $i = 0, 1, 2, \ldots$

$$\mathbf{\Sigma}_i = \mathbf{\Sigma}_{i-1,i} + \mathbf{A}_i^T\mathbf{A}_i + \mathbf{H}_i^T\mathbf{H}_i,$$

and

$$\nu_i = \nu_{i-1,i} + \mathbf{H}_i^T\mathbf{y}_i,$$

where $\mathbf{\Sigma}_{-1,0} = \mathbf{0}$ and $\nu_{-1,0} = \mathbf{y}_0$. Since combination (9.15) corresponds to matrix addition, the system on the node $\mathbf{X}_i, \mathbf{X}_{i+1}$ at step i of the collect algorithm is:

$$\begin{bmatrix} \mathbf{\Sigma}_i & -\mathbf{A}_i^T \\ -\mathbf{A}_i & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{X}_i \\ \mathbf{X}_{i+1} \end{bmatrix} = \begin{bmatrix} \nu_i \\ \mathbf{0} \end{bmatrix}.$$

Due to (9.13) and (9.14), the message from node $\mathbf{X}_{i-1}, \mathbf{X}_i$ to node $\mathbf{X}_i, \mathbf{X}_{i+1}$ computed by eliminating $\mathbf{X}_{i-1}$ is

$$\left(\mathbf{\Sigma}_{i-1,i}, \nu_{i-1,i}\right) = \left(\mathbf{A}_{i-1}\mathbf{\Sigma}_{i-1}^{-1}\mathbf{A}_{i-1}^T + \mathbf{I}, \mathbf{A}_{i-1}\mathbf{\Sigma}_{i-1}^{-1}\mathbf{H}_{i-1}^T\mathbf{y}_{i-1}\right)$$

for $i = 1, 2, \ldots$ and the eliminated variables in this step can be expressed as

$$\mathbf{X}_{i-1} = \mathbf{\Sigma}_{i-1}^{-1}\mathbf{A}_{i-1}^T\mathbf{X}_{i-1} + \mathbf{\Sigma}_{i-1}^{-1}\mathbf{H}_{i-1}^T\mathbf{y}_{i-1}.$$

If the sequence stops with $\mathbf{X}_3$ as in our example, then eliminating $\mathbf{X}_2$ in the subsystem on the middle node in join tree of Figure 9.8 gives

$$(-\mathbf{A}_2\mathbf{\Sigma}_2^{-1}\mathbf{A}_2^T + \mathbf{I})\mathbf{X}_3 = \mathbf{A}_2\mathbf{\Sigma}_2^{-1}\nu_2,$$

hence

$$\mathbf{X}_3 = (-\mathbf{A}_2\mathbf{\Sigma}_2^{-1}\mathbf{A}_2^T + \mathbf{I})^{-1}\mathbf{A}_2\mathbf{\Sigma}_2^{-1}\nu_2.$$

This is also called a *one-step predictor* for $\mathbf{X}_3$ and can be used to start backward substitution: first for $\mathbf{X}_2$, then for $\mathbf{X}_1$ and finally for $\mathbf{X}_0$. Such linear dynamic systems will be reconsidered from a more general point of view in Section 10.5. In particular, it will be shown that the assumption that $\mathbf{H}_i$ has full column rank can be dropped. This closes our discussion of ordinary linear systems and we next focus on linear fixpoint equation systems. A third class of linear systems called linear systems with Gaussian disturbances will be studied in Section 10.1.

9.3 SEMIRING FIXPOINT EQUATION SYSTEMS

A linear fixpoint equation system takes the form $\mathbf{MX}+\mathbf{b} = \mathbf{X}$, where $\mathbf{M}$ is an $n \times n$ matrix, $\mathbf{X}$ an n-vector of variables and $\mathbf{b}$ an n-vector. If the matrix $\mathbf{M}$ and the vector $\mathbf{b}$ take values from a field, we may express the fixpoint system as $(\mathbf{I} - \mathbf{M})\mathbf{X} = \mathbf{b}$ and by defining $\mathbf{A} = \mathbf{I} - \mathbf{M}$ we obtain an ordinary linear system $\mathbf{AX} = \mathbf{b}$. This allows us to apply the theory of the previous sections and shows that fixpoint equation systems over fields do not need to be considered separately. However, we pointed out in Section 6.2 that the computation of path problems amounts to the solution of fixpoint equation systems over semirings. Moreover, we identified a particular class of semirings called quasi-regular semirings, where all fixpoint equations provide a solution determined by the star operation. If $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ denotes a quasi-regular semiring, then the set $\mathcal{M}(A,n)$ of $n \times n$ matrices with values from A also forms a quasi-regular semiring with the inductive definition of the star operation given in equation (6.17), see Theorem 6.2. Hence, a solution to the fixpoint equation $\mathbf{MX} + \mathbf{b} = \mathbf{X}$ in a quasi-regular semiring is always given by the element $\mathbf{M}^*\mathbf{b}$. How is this related to fixpoint equations over real numbers and thus to ordinary equation systems? It was shown in Example 6.11 that real numbers $\mathbb{R} \cup \{\infty\}$ form a quasi-regular semiring with the definition

$$a^* \quad = \quad \frac{1}{1-a}$$

for $a \neq 1$ and $a^* = \infty$ for $a = 1$. If we now consider its induced semiring of matrices and further assume that $\mathbf{I} - \mathbf{M}$ is regular, then the fixpoint system $\mathbf{MX} + \mathbf{b} = \mathbf{X}$ has a unique solution that must be equal to the result of the star operation,

$$\mathbf{M}^*\mathbf{b} \quad = \quad (\mathbf{I}-\mathbf{M})^{-1}\mathbf{b}.$$

By applying the star operation to $\mathbf{I} - \mathbf{M}$ we obtain

$$(\mathbf{I}-\mathbf{M})^*\mathbf{b} \;=\; (\mathbf{I}-(\mathbf{I}-\mathbf{M}))^{-1}\mathbf{b} \;=\; \mathbf{M}^{-1}\mathbf{b},$$

which solves the regular system $\mathbf{MX} = \mathbf{b}$. On the other hand, if we directly compute $(\mathbf{I} - \mathbf{M})^*$ without caring about the regularity of $\mathbf{I} - \mathbf{M}$, then the result corresponds to the inverse matrix of $\mathbf{M}$, if $(\mathbf{I} - \mathbf{M})^*$ does not contain the adjoined semiring element ∞. This follows from Theorem 6.18 and the uniqueness of the inverse matrix (Lehmann, 1976). Concluding, we see that fixpoint equation systems over quasi-regular semirings indeed provide an important generalization of ordinary linear systems. Semiring fixpoint equations induce two different types of valuation algebras which will both be used in the remaining parts of this chapter to obtain a local computation based solution to such equations. First, we focus on quasi-regular valuation algebras from Section 6.4 which leads to a solution approach that is similar to local computation with symmetric, positive definite systems studied in Section 9.2.2. The second approach is based on Kleene valuation algebras from Section 6.7 and leads to an interesting compilation approach which is based on the query answering technique for idempotent valuation algebras in Section 4.6. However, computation in both valuation algebras requires evaluating the star operation for matrices over quasi-regular semirings. The following section therefore presents first a well-known algorithm for this task.

9.3.1 Computing Quasi-Inverse Matrices

For a finite index set s, the set of labeled matrices $\mathcal{M}(A,s)$ with values from a quasi-regular semiring $\langle A,+,\times,*,\mathbf{0},\mathbf{1}\rangle$ forms itself a quasi-regular semiring with the inductive definition of the star operation given in equation (6.17). This operation is performed as part of the projection rule in quasi-regular valuation algebras, respectively as part of the combination rule in Kleene valuation algebras. In the latter case, we often use the alternative induction of equation (6.31), but it has also been shown that the two definitions are equivalent for Kleene algebras. It is therefore sufficient for both algebras to dispose of an algorithm that computes a quasi-inverse matrix $\mathbf{M}^*$ for $\mathbf{M} \in \mathcal{M}(A,s)$ that satisfies the fixpoint equation

$$\mathbf{M}^* \;=\; \mathbf{M}\mathbf{M}^* + \mathbf{I} \;=\; \mathbf{M}^*\mathbf{M} + \mathbf{I}. \tag{9.29}$$

The well-known *Warshall-Floyd-Kleene algorithm* proposed by (Roy, 1959; Warshall, 1962; Floyd, 1962; Kleene, 1956) computes quasi-inverses iteratively by the following formula: For $s = \{1,\ldots,n\}$ and $k = 0,\ldots,n$ we define

$$\mathbf{M}^{(k)} \;=\; \mathbf{M}^{(k-1)} + \left(\mathbf{M}^{(k-1)}\right)^{\downarrow s,\{k\}}\left(\mathbf{M}^{(k-1)}(k,k)\right)^*\left(\mathbf{M}^{(k-1)}\right)^{\downarrow\{k\},s} \tag{9.30}$$

with $\mathbf{M}^{(0)} = \mathbf{M}$. Note that the operation of matrix projection is used to refer to the k-th column and row of the matrix $\mathbf{M}$. Also, we again omit the explicit writing of the semiring multiplication symbol. It is proved in (Lehmann, 1976) that $\mathbf{M}^{(n)} + \mathbf{I}$ indeed corresponds to the quasi-inverse of equation (6.17), which implies that

$$\mathbf{M}^* \;=\; \mathbf{M}^{(n)} + \mathbf{I}$$

satisfies equation (9.29) as a consequence of Theorem 6.2. Formula (9.30) therefore provides a general algorithm for the computation of the star operation in $\mathcal{M}(A,s)$ using only the semiring operations in A. The space complexity of this algorithm is $\mathcal{O}(|s|^2)$ because it only needs to store two matrices. Given two indices $i,j \in s$, equation (9.30) is equivalently expressed as

$$\mathbf{M}^{(k)}(i,j) \;=\; \mathbf{M}^{(k-1)}(i,k) + \mathbf{M}^{(k-1)}(i,k)\left(\mathbf{M}^{(k-1)}(k,k)\right)^*\mathbf{M}^{(k-1)}(k,j).$$

This shows that an implementation required three nested loops for the indices i,j and k, which leads to a time complexity of $\mathcal{O}(|s|^3)$. It is important to note that the theory in the subsequent sections is independent on the actual algorithm used to compute the star operation of matrices over quasi-regular semirings and Kleene algebras. Here, we presented the Warshall-Floyd-Kleene algorithm but there are other algorithms as for example the *Gauss-Jordan method*. However, we will use the Warshall-Floyd-Kleene algorithm to reason about the complexity of local computation based approaches to the solution of semiring fixpoint equation systems, but if another algorithm is present for some particular quasi-regular semirings that outperforms the Warshall-Floyd-Kleene algorithm or its implementation, then this algorithm can be used and its gain in efficiency also carries over to the local computation scheme.

Algorithm 9.1 The Warshall-Floyd-Kleene Algorithm

```
input:  $\mathbf{M} \in \mathcal{M}(A,s)$   output:  $\mathbf{M}^*$

begin
  for each $X,Y \in s$ do                                   // Initialization.
    $\mathbf{M}_0(X,Y)$ := $\mathbf{M}(X,Y)$
  end;

  for each $Z = 1 \ldots |s|$ do
    for each $X,Y \in s$ do
      $c$ := $\mathbf{M}_{Z-1}(X,Z)(\mathbf{M}_{Z-1}(Z,Z))^*\mathbf{M}_{Z-1}(Z,Y)$;
      $\mathbf{M}_Z(X,Y)$ := $\mathbf{M}_{Z-1}(X,Y) + c$;
    end;
  end;

  for each $X \in s$ do                            // Addition of unit matrix.
    $\mathbf{M}_Z(X,X)$ := $\mathbf{M}_Z(X,X) + \mathbf{1}$;
  end;
  return $\mathbf{M}_Z$;
```

9.3.2 Local Computation with Quasi-Regular Valuations

We focus in this section on a local computation based solution of fixpoint equation systems $\mathbf{X} = \mathbf{MX} + \mathbf{b}$ defined over a quasi-regular semiring $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$. For variables $X_1, \ldots, X_n$ and the associated index set $r = \{1, \ldots, n\}$, we assume for $s \subseteq r$ that $\mathbf{M} \in \mathcal{M}(A, s)$ is an $s \times s$ matrix of semiring values, $\mathbf{X}$ an s-vector of variables and $\mathbf{b}$ an s-vector of semiring values. It was shown in Section 6.4 that such systems correspond to valuations $\phi = (\mathbf{A}, \mathbf{b})$ with domain $d(\phi) = s$ in the quasi-regular valuation algebra $\langle \Phi, D\rangle$ induced by the semiring A, where $D = \mathcal{P}(r)$. We observe the similar structure with the valuation algebra of symmetric, positive definite systems in Section 9.2.1: in both algebras, valuations are pairs of square matrices and vectors, the operations of combination defined in the equations (6.30) and (9.15) are identical, and also the operations of projection given in the equations (6.23) and (9.14) look very similar.

As usual for local computation, we subsequently assume that $\phi = (\mathbf{M}, \mathbf{b})$ is given as a factorization or decomposition $\phi = \phi_1 \otimes \ldots \otimes \phi_m$ of quasi-regular valuations $\phi_i = (\mathbf{M}_i, \mathbf{b}_i) \in \Phi$ with $i = 1, \ldots, m$. It is important to note that the following local computation based solution solves any such factorized system. But since we motivated semiring fixpoint equations from the perspective of the algebraic path problem in Chapter 6, we subsequently go into this important application field. Moreover, when the system $(\mathbf{M}, \mathbf{b})$ models a path problem in a graph, such factorizations often come from the underlying graph. Imagine, for example, that we want to compute shortest distances between cities of different European countries from road maps that contain only direct distances between neighboring cities, see Instance 6.2. Hence, let s denote the set of all European cities. Instead of a single road map $\mathbf{M}$ that contains all cities of Europe, it is more natural to consider local maps $\mathbf{M}_i$ for each country with some overlapping region to neighboring countries. Since matrix addition corresponds to minimization in the tropical semiring, we clearly have

$$\mathbf{M} \quad = \quad \mathbf{M}_1^{\uparrow s} + \ldots + \mathbf{M}_m^{\uparrow s}. \tag{9.31}$$

Further, we explained at the very end of Section 6.3 that the vector $\mathbf{b}$ serves to specify the distances to be computed. If we are interested in the shortest distances between all possible cities of Europe and a selected target city $T \in s$, we define the vector $\mathbf{b}$ such that $\mathbf{b}(T) = \mathbf{1}$ and $\mathbf{b}(Y) = \mathbf{0}$ for all $Y \in s - \{T\}$. Solving the fixpoint equation $\mathbf{X} = \mathbf{MX} + \mathbf{b}$ then corresponds to the single-target shortest distance problem and its transposed system represents the single-source problem. The vectors $\mathbf{b}_i$ are therefore specified with respect to query and must satisfy

$$\mathbf{b} \quad = \quad \mathbf{b}_1^{\uparrow s} + \ldots + \mathbf{b}_m^{\uparrow s}.$$

This gives us a natural factorization

$$(\mathbf{M}, \mathbf{b}) \quad = \quad (\mathbf{M}_1, \mathbf{b}_1) \otimes \cdots \otimes (\mathbf{M}_m, \mathbf{b}_m)$$

for the shortest distance problem. If $\mathbf{x} = \mathbf{M}^*\mathbf{b}$ is a solution to the system $(\mathbf{M}, \mathbf{b})$, it contains the shortest distances between all European cities and the selected target city $T \in s$. For an arbitrary source city $S \in s$, it follows from Theorem 6.3 that

$$\mathbf{x}^{\downarrow\{S,T\}} \quad = \quad (\mathbf{M}_{\{S,T\}})^* \mathbf{b}_{\{S,T\}}.$$

Hence, the single-pair problem requires to compute

$$\Big(\mathbf{M}, \mathbf{b}\Big)^{\downarrow\{S,T\}} \quad = \quad \Big(\mathbf{M}_{\{S,T\}}, \mathbf{b}_{\{S,T\}}\Big)$$

and we therefore obtain its solution by solving the single-query inference problem

$$\Big(\mathbf{M}, \mathbf{b}\Big)^{\downarrow\{S,T\}} \quad = \quad \Big((\mathbf{M}_1, \mathbf{b}_1) \otimes \cdots \otimes (\mathbf{M}_m, \mathbf{b}_m)\Big)^{\downarrow\{S,T\}}.$$

This can either be done by the fusion or collect algorithm. As for symmetric, positive definite systems in Section 9.2.2, we finally compute the complete solution $\mathbf{x}$ by a step-wise extension of the single-pair solution in a solution construction process. To do so, we show in this section that solutions to quasi-regular fixpoint equations satisfy the requirements for general solutions in valuation algebras imposed in Section 8.1. This allows us to directly apply one of the generic solution construction procedures.

Alternatively, if no natural factorization of the sparse system $\phi = (\mathbf{M}, \mathbf{b})$ exists, we have to produce a decomposition $\phi = \phi_1 \otimes \ldots \otimes \phi_m$. There are many applications of path problems where the adjacency matrix is symmetric. Imagine for example the modelling of air-line distances, which are always symmetric. We then obtain a decomposition by applying the graphical theory of Section 9.2.4. This follows from the similar structure between the two valuation algebras mentioned above. On the other hand, if the matrix is not symmetric, then the construction of the graph for the triangulation process must be modified. We propose this as Exercise I.1 to the reader. By application of local computation, we exploit the sparsity in the system $\phi = (\mathbf{M}, \mathbf{b})$ captured by the factorization or decomposition. As shown below, this is again similar to local computation for symmetric, positive definite systems studied in Section 9.2.2 and also follows the same principle of locality.

We now propose a suitable definition of configuration extension sets for quasi-regular valuation algebras: for $\phi = (\mathbf{M}, \mathbf{b}) \in \Phi$ with $d(\phi) = s$ and $t \subseteq s$ we define the configuration extension set for an arbitrary t-vector $\mathbf{x}$ to ϕ as

$$W_\phi^t(\mathbf{x}) \quad = \quad \left\{(\mathbf{M}^{\downarrow s-t,s-t})^*(\mathbf{M}^{\downarrow s-t,t}\mathbf{x} + \mathbf{b}^{\downarrow s-t})\right\}. \tag{9.32}$$

Again, observe the similarity to the definition of configuration extension sets for symmetric, positive definite systems in equation (9.19). The following theorem shows that this definition also satisfies the requirements for general solution extension sets in valuation algebras imposed in Section 8.1.

Theorem 9.6 *Solution extension sets in quasi-regular valuation algebras satisfy the property of Definition 8.1, i.e. for all $\phi \in \Phi$ with $t \subseteq u \subseteq s = d(\phi)$ and all t-vectors $\mathbf{x}$ we have*

$$W_\phi^t(\mathbf{x}) \quad = \quad \left\{(\mathbf{y}, \mathbf{z}) : \mathbf{y} \in W_{\phi^{\downarrow u}}^t(\mathbf{x}) \textit{ and } \mathbf{z} \in W_\phi^u(\mathbf{x}, \mathbf{y})\right\}.$$

Proof: For an arbitrary t-vector $\mathbf{x}$ let

$$\mathbf{y} \quad = \quad (\mathbf{M}^{\downarrow s-t,s-t})^*(\mathbf{M}^{\downarrow s-t,t}\mathbf{x} + \mathbf{b}^{\downarrow s-t}) \in W_\phi^t(\mathbf{x}).$$

It then follows from Lemma 6.2 and Definition 6.4 that $\mathbf{y}$ is the solution to the system

$$Y \quad = \quad (\mathbf{M}^{\downarrow s-t,s-t})Y + (\mathbf{M}^{\downarrow s-t,t}\mathbf{x} + \mathbf{b}^{\downarrow s-t}).$$

We therefore have

$$\begin{bmatrix} \mathbf{y}^{\downarrow u-t} \\ \mathbf{y}^{\downarrow s-u} \end{bmatrix} \quad = \quad \begin{bmatrix} \mathbf{M}^{\downarrow u-t,u-t} & \mathbf{M}^{\downarrow u-t,s-u} \\ \mathbf{M}^{\downarrow s-u,u-t} & \mathbf{M}^{\downarrow s-u,s-u} \end{bmatrix} \begin{bmatrix} \mathbf{y}^{\downarrow u-t} \\ \mathbf{y}^{\downarrow s-u} \end{bmatrix} + \begin{bmatrix} \mathbf{M}^{\downarrow u-t,t} \\ \mathbf{M}^{\downarrow s-u,t} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{b}^{\downarrow u-t} \\ \mathbf{b}^{\downarrow s-u} \end{bmatrix}$$

and obtain the two systems

$$\begin{aligned} \mathbf{y}^{\downarrow u-t} &= \mathbf{M}^{\downarrow u-t,u-t}\mathbf{y}^{\downarrow u-t} + \mathbf{M}^{\downarrow u-t,s-t}\mathbf{y}^{\downarrow s-u} + \mathbf{M}^{\downarrow u-t,t}\mathbf{x} + \mathbf{b}^{\downarrow u-t} \\ \mathbf{y}^{\downarrow s-u} &= \mathbf{M}^{\downarrow s-u,u-t}\mathbf{y}^{\downarrow u-t} + \mathbf{M}^{\downarrow s-u,s-u}\mathbf{y}^{\downarrow s-u} + \mathbf{M}^{\downarrow s-u,t}\mathbf{x} + \mathbf{b}^{\downarrow s-u}. \end{aligned}$$

From the second equation follows that $\mathbf{y}^{\downarrow s-u}$ is the solution to

$$Y^{\downarrow s-u} \;=\; \mathbf{M}^{\downarrow s-u,s-u}Y^{\downarrow s-u} + (\mathbf{M}^{\downarrow s-u,u-t}\mathbf{y}^{\downarrow u-t} + \mathbf{M}^{\downarrow s-u,t}\mathbf{x} + \mathbf{b}^{\downarrow s-u}).$$

We must therefore have

$$\mathbf{y}^{\downarrow s-u} \;=\; (\mathbf{M}^{\downarrow s-u,s-u})^*(\mathbf{M}^{\downarrow s-u,u-t}\mathbf{y}^{\downarrow u-t} + \mathbf{M}^{\downarrow s-u,t}\mathbf{x} + \mathbf{b}^{\downarrow s-u}). \tag{9.33}$$

We insert this expression into the first equation and obtain after rearranging terms:

$$\begin{aligned}\mathbf{y}^{\downarrow u-t} \;=\; & \Big[(\mathbf{M}^{\downarrow u-t,u-t}) + \mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u-t}\Big]\mathbf{y}^{\downarrow u-t} \; + \\ & \Big[\mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,t}\mathbf{x} \; + \\ & \mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{b}^{\downarrow s-u} \; + \; \mathbf{M}^{\downarrow u-t,t}\mathbf{x} \; + \; \mathbf{b}^{\downarrow u-t}\Big] \end{aligned} \tag{9.34}$$

Next, we observe the following identities:

$$\begin{aligned}\Big(\mathbf{M}_u\Big)^{\downarrow u-t,u-t} &= \Big(\mathbf{M}^{\downarrow u,u} + \mathbf{M}^{\downarrow u,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u}\Big)^{\downarrow u-t,u-t} \\ &= \mathbf{M}^{\downarrow u-t,u-t} + \mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,u-t}, \\ \Big(\mathbf{b}_u\Big)^{\downarrow u-t} &= \Big(\mathbf{b}^{\downarrow u} + \mathbf{M}^{\downarrow u,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{b}^{\downarrow s-u}\Big)^{\downarrow u-t} \\ &= \mathbf{b}^{\downarrow u-t} + \mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{b}^{\downarrow s-u}.\end{aligned}$$

This follows from the definition of projection in the equations (6.24) and (6.25). Likewise, we also have

$$\Big(\mathbf{M}_u\Big)^{\downarrow u-t,t} \;=\; \mathbf{M}^{\downarrow u-t,t} + \mathbf{M}^{\downarrow u-t,s-u}(\mathbf{M}^{\downarrow s-u,s-u})^*\mathbf{M}^{\downarrow s-u,t}.$$

We insert these identities in equation (9.34) and obtain

$$\mathbf{y}^{\downarrow u-t} \;=\; (\mathbf{M}_u)^{\downarrow u-t,u-t}\mathbf{y}^{\downarrow u-t} \;+\; ((\mathbf{b}_u)^{\downarrow u-t} \;+\; (\mathbf{M}_u)^{\downarrow u-t,t}\mathbf{x}).$$

We observe that $\mathbf{y}^{\downarrow u-t}$ is the solution to the system

$$Y^{\downarrow u-t} \;=\; (\mathbf{M}_u)^{\downarrow u-t,u-t}Y^{\downarrow u-t} \;+\; (\mathbf{b}_u)^{\downarrow u-t} \;+\; (\mathbf{M}_u)^{\downarrow u-t,t}\mathbf{x}$$

and it must again hold that

$$\mathbf{y}^{\downarrow u-t} \;=\; \Big((\mathbf{M}_u)^{\downarrow u-t,u-t}\Big)^*\Big((\mathbf{b}_u)^{\downarrow u-t} \;+\; (\mathbf{M}_u)^{\downarrow u-t,t}\mathbf{x}\Big).$$

This shows that $\mathbf{y}^{\downarrow u-t} \in W^t_{\phi^{\downarrow u}}(\mathbf{x})$. Finally, we conclude from equation (9.33)

$$\begin{aligned}\mathbf{y}^{\downarrow s-u} &= (\mathbf{M}^{\downarrow s-u,s-u})^*(\mathbf{M}^{\downarrow s-u,u-t}\mathbf{y}^{\downarrow u-t} + \mathbf{M}^{\downarrow s-u,t}\mathbf{x} + \mathbf{b}^{\downarrow s-u}) \\ &= (\mathbf{M}^{\downarrow s-u,s-u})^*\Big(\mathbf{M}^{\downarrow s-u,u}\begin{bmatrix}\mathbf{y}^{\downarrow u-t} \\ \mathbf{x}\end{bmatrix} + \mathbf{b}^{\downarrow s-u}\Big).\end{aligned}$$

This implies that $\mathbf{y}^{\downarrow s-u} \in W_\phi^u(\mathbf{y}^{\downarrow u-t}, \mathbf{x})$. ■

Configuration extension sets therefore fulfill the requirements for the generic solution construction procedure of Section 8.2.1, that builds the solution set

$$c_\phi \;=\; W_\phi^\emptyset \;=\; \{\mathbf{M}^*\mathbf{b}\}. \tag{9.35}$$

based on the results of a multi-query local computation procedure. However, remember that the combination rule for quasi-regular valuation algebras and symmetric, positive definite systems are identical, see equations (6.30) and (9.15). For symmetric, positive definite systems we know from Lemma 9.3 that they also satisfy the additional condition of Theorem 8.3 for solution construction based on the results of a single-query architecture. This property is only based on the combination operator and therefore also holds for quasi-regular valuation algebras.

Lemma 9.4 *Quasi-regular valuation algebras satisfy the property that for all $\psi_1, \psi_2 \in \Phi$ with $d(\psi_1) = s$, $d(\psi_2) = t$, $s \subseteq u \subseteq s \cup t$ and u-vector $\mathbf{x}$ we have*

$$W_{\psi_2}^{u\cap t}(\mathbf{x}^{\downarrow u \cap t}) \;=\; W_{\psi_1 \otimes \psi_2}^{u}(\mathbf{x}).$$

Hence, we can conclude from Section 8.2.3 that Algorithm 8.4 returns the solution to a factorized, quasi-regular fixpoint equation system based on the results of a single-query local computation architecture. The actual process is very similar to the case of symmetric, positive definite systems, which we considered in detail in Section 9.2.2.

To determine the complexity of solving fixpoint equation systems by local computation, we first remind that projection is the more time consuming operation than combination. For $\phi = (\mathbf{M}, \mathbf{b}) \in \Phi$ with $d(\phi) = s$ and $t \subseteq s$, the projection rule for quasi-regular valuations, given in equation (6.23), is

$$\begin{aligned}(\mathbf{M}, \mathbf{b})^{\downarrow t} \;=\; \Big(&\mathbf{M}^{\downarrow t,t} + \mathbf{M}^{\downarrow t, s-t}(\mathbf{M}^{\downarrow s-t, s-t})^*\mathbf{M}^{\downarrow s-t, t},\\ &\mathbf{b}^{\downarrow t} + \mathbf{M}^{\downarrow t, s-t}(\mathbf{M}^{\downarrow s-t, s-t})^*\mathbf{b}^{\downarrow s-t}\Big)\end{aligned}$$

includes the evaluation of the star operation for matrices and this has a cubic time complexity in the general case, as shown in Section 9.3.1. However, let us now rewrite this operation with variable elimination instead of projection. If $t = s - \{Z\}$ for some $Z \in s$, the matrix component of this operation becomes

$$\mathbf{M}^{\downarrow s-\{Z\}, s-\{Z\}} + \mathbf{M}^{\downarrow s-\{Z\},\{Z\}}\mathbf{M}(Z,Z)^*\mathbf{M}^{\downarrow \{Z\}, s-\{Z\}},$$

which requires computing

$$\mathbf{M}(X,Y) + \mathbf{M}(X,Z)\mathbf{M}(Z,Z)^*\mathbf{M}(Z,Y)$$

for all $X, Y \in s - \{Z\}$. This expression does not apply the star operation to matrices anymore. Since the same holds for the vector component, the operation of variable elimination adopts a quadratic time complexity. This motivates the application of the fusion algorithm instead of collect. Moreover, the size of the valuation stored in a join tree node is bounded by the treewidth ω^* which then also gives an upper bound for the time complexity of the operations of combination and variable elimination. To sum it up, the time and space complexity of applying the fusion algorithm to quasi-regular valuations is

$$\mathcal{O}\Big(m \cdot (\omega^* + 1)^2\Big).$$

The single-target path problem with target node $T \in s$ is represented by the fixpoint equation $\mathbf{X} = \mathbf{M}\mathbf{X} + \mathbf{b}_T$ with $\mathbf{b}(T) = \mathbf{1}$ and $\mathbf{b}(Y) = \mathbf{0}$ for all $Y \in s - \{T\}$. Its solution $\mathbf{x} = \mathbf{M}^*\mathbf{b}_T$ corresponds to the column of the matrix $\mathbf{M}^*$ that corresponds to the variable T. If we solve this problem for all possible target nodes, we completely determine the matrix $\mathbf{M}^*$ and thus solve the all-pairs path problem. This is possible with $|s|$ repetitions of the above local computation procedure using the same matrix $\mathbf{M}$ but different vectors. In other words, we have a prototype application for a compilation approach such as the $\mathbf{LDR}$-decomposition for regular systems in Section 9.1.4 or the $\mathbf{LDL}^T$-decomposition for symmetric, positive definite systems in Section 9.2.3. In fact, it was shown by (Backhouse & Carré, 1975) that $\mathbf{LDR}$-decomposition can also be applied to *regular algebras*, which are quasi-regular semirings with idempotent addition. This approach was later generalized by (Radhakrishnan *et al.*, 1992) to quasi-regular semirings. We propose the development of $\mathbf{LDR}$-decomposition for matrices with values from a quasi-regular semiring as Exercise I.2 to the reader. In the following section, we focus on an alternative approach based on Kleene valuation algebras that computes some selected values of the quasi-inverse matrix $\mathbf{M}^*$ directly. This corresponds to the multiple-pairs algebraic path problem which does not assume specific source or target nodes.

9.3.3 Local Computation with Kleene Valuations

For an index set $r = \{1, \ldots, n\}$ and $s \subseteq r$, the solution to the all-pairs algebraic path problem $\mathbf{M}^*$ defined by the matrix $\mathbf{M} \in \mathcal{M}(A, s)$ is a solution to the fixpoint system $\mathbf{X} = \mathbf{M}\mathbf{X} + \mathbf{I}$. Here, $\mathbf{X}$ is an $s \times s$ matrix of variables, $\mathbf{M}$ an $s \times s$ matrix of semiring values and $\mathbf{I}$ the unit matrix in $\mathcal{M}(A, s)$. We further assume in this section that the semiring $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$ is a Kleene algebra according to Definition 6.6. It then follows from Theorem 6.5 that the set of labeled matrices $\mathcal{M}(A, s)$ also forms a Kleene algebra, and we obtain its induced Kleene valuation algebra $\langle \Phi, D\rangle$ by defining $D = \mathcal{P}(r)$ and

$$\Phi \quad = \quad \{\mathbf{N}^* \mid \mathbf{N} \in \mathcal{M}(A, s) \text{ and } s \in D\}$$

as shown in Theorem 6.6. Consequently, we may consider the solution $\mathbf{M}^*$ to the all-pairs path problem over a Kleene algebra as an element of a valuation algebra. With regard to inference problems, we declare this element as the objective function and assume a knowledgebase $\{\mathbf{M}_1^*, \ldots, \mathbf{M}_n^*\} \subseteq \Phi$ such that

$$\mathbf{M}^* \;=\; \mathbf{M}_1^* \otimes \cdots \otimes \mathbf{M}_m^*.$$

This corresponds to the situation of having an existing factorization for the path problem solution. Alternatively, we could take a factorization of the original matrix $\mathbf{M}$ and start from a set of matrices $\{\mathbf{M}_1, \ldots, \mathbf{M}_m\}$ with $\mathbf{M}_i \in \mathcal{M}(A, s_i)$ and $s_i \subseteq s$ for $i = 1, \ldots, m$ such that

$$\mathbf{M} \;=\; \mathbf{M}_1^{\uparrow s} + \ldots + \mathbf{M}_m^{\uparrow s}. \tag{9.36}$$

This second factorization represents in the example of the previous section the local road maps for each country, whereas the elements of the first factorization correspond to tables of shortest distances that already exist for each country. Although probably more realistic, the second factorization cannot be considered as a knowledgebase directly, because the factors $\mathbf{M}_i$ are generally not contained in Φ. But as this example foreshadows, it proves sufficient to compute the closure of each factor in order to obtain a knowledgebase that factors $\mathbf{M}^*$ as a consequence of Lemma 6.11 and 6.17:

$$\begin{aligned}
\mathbf{M}^* &= \left(\mathbf{M}_1^{\uparrow s} + \cdots + \mathbf{M}_m^{\uparrow s}\right)^* \\
&= \left(((\mathbf{M}_1^{\uparrow s_1 \cup s_2} + \mathbf{M}_2^{\uparrow s_1 \cup s_2})^{\uparrow s_1 \cup s_2 \cup s_3} + \ldots)^{\uparrow s_1 \cup \ldots \cup s_m} + \mathbf{M}_m^{\uparrow s_1 \cup \ldots \cup s_m}\right)^* \\
&= \left(((\mathbf{M}_1^{*\uparrow s_1 \cup s_2} + \mathbf{M}_2^{*\uparrow s_1 \cup s_2})^{*\uparrow s_1 \cup s_2 \cup s_3} + \ldots)^{*\uparrow s_1 \cup \ldots \cup s_m} + \mathbf{M}_m^{*\uparrow s_1 \cup \ldots \cup s_m}\right)^* \\
&= \mathbf{M}_1^* \otimes \cdots \otimes \mathbf{M}_m^*.
\end{aligned}$$

The last equality follows from the definition of combination in Kleene valuation algebras given in equation (6.38). Thus, we may either start from a factorization of the path problem solution, which directly gives a knowledgebase of Kleene valuations, or produce the latter from a factorization of the path problem input matrix. Note that the second case includes the type of factorization that was considered for quasi-regular valuation algebras in equation (9.31). Alternatively, if no natural factorization of either the closure or the input matrix exists, we may exploit the idempotency of semiring addition to produce a decomposition of the matrix $\mathbf{M} \in \mathcal{M}(A, s)$ that satisfies equation (9.36): For $X, Y \in s$ and $X \neq Y$,

- if $\mathbf{M}(X, Y) \neq \mathbf{0}$ or $\mathbf{M}(Y, X) \neq \mathbf{0}$, add $\mathbf{M}^{\downarrow\{X,Y\}}$ to the factorization;
- if $\mathbf{M}(X, X) \neq \mathbf{0}$, add $\mathbf{M}^{\downarrow\{X\}}$ to the factorization.

This creates a factorization of minimal granularity where the domain of each factor contains at most two variables. Note also that the same diagonal value $\mathbf{M}(X, X)$ possibly occurs in multiple factors. This, however, does not matter as a consequence of idempotent semiring addition in a Kleene algebra. Moreover, if the Kleene algebra is a bounded semiring with the property $a + \mathbf{1} = \mathbf{1}$ for all $a \in A$, it follows from equation (9.29) that $\mathbf{M}^*(X, X) = \mathbf{1}$ for all $X \in s$ and independently of the original values of $\mathbf{M}(X, X)$. In this case, we can ignore the diagonal elements in the decomposition process. Also, the semantics of path problems often yield input matrices where most of the diagonal elements are $\mathbf{M}(X, X) = \mathbf{1}$. Then, the decomposition process also leads to factors that share this property, and the following lemma shows that these factors are equal to their closures. This simplifies the process of building a knowledgebase from the decomposition process considerably, as illustrated in Example 9.5.

Lemma 9.5 *In a bounded Kleene algebra we have*

$$\begin{pmatrix} \mathbf{1} & c \\ d & \mathbf{1} \end{pmatrix}^* = \begin{pmatrix} \mathbf{1} & c \\ d & \mathbf{1} \end{pmatrix}. \tag{9.37}$$

Proof: We apply equation (6.32) and remark that the property of boundedness implies $f = b + ce^*d = \mathbf{1}$. It follows from Property (KP3) in Lemma 6.8 that $f = f^* = \mathbf{1}$, and the statement then follows directly. ■

Example 9.5 *Consider the variable set* $s = \{Berlin, Paris, Rome, Berne\}$ *and the following sparse matrix defined over the tropical semiring of non-negative integers* $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0\rangle$ *expressing the path lengths between some selected cities:*

$\mathbf{M}$ =

	Berlin	*Paris*	*Rome*	*Berne*
Berlin	0	1111	1181	∞
Paris	1111	0	∞	436
Rome	1181	∞	0	∞
Berne	∞	436	∞	0

This table might be seen as the direct distances in a road map, where the symbol ∞ *tags unknown distances or the absence of a direct connection in the road map. The matrix* $\mathbf{M}$ *then factorizes as*

$$\mathbf{M} = \left(\mathbf{M}^{\downarrow\{Berlin,Paris\}}\right)^{\uparrow s} + \left(\mathbf{M}^{\downarrow\{Berne,Paris\}}\right)^{\uparrow s} + \left(\mathbf{M}^{\downarrow\{Rome,Berlin\}}\right)^{\uparrow s}$$

with factors

$\mathbf{M}^{\downarrow\{Berlin,Paris\}}$ =

	Berlin	*Paris*
Berlin	0	1111
Paris	1111	0

$\mathbf{M}^{\downarrow\{Berne,Paris\}}$ =

	Berne	*Paris*
Berne	0	436
Paris	436	0

$\mathbf{M}^{\downarrow\{Rome,Berlin\}}$ =

	Rome	*Berlin*
Rome	0	1181
Berlin	1181	0

We further recall from Example 5.10 that the tropical semiring is bounded with the integer 0 *as unit element. By Lemma 9.5 the above factors are equal to their closures* *and therefore directly form a knowledgebase, i.e. a factorization of the yet unknown solution* $\mathbf{M}^*$. *We have*

$$\mathbf{M}^* \quad = \quad \mathbf{M}^{\downarrow\{Berlin,Paris\}} \otimes \mathbf{M}^{\downarrow\{Berne,Paris\}} \otimes \mathbf{M}^{\downarrow\{Rome,Berlin\}}.$$

The second component of an inference problem is the query set. When solving the multiple-pairs path problem, we are interested in the semiring values $\mathbf{M}^*(X, Y)$ for certain pairs of variables (X, Y) from the domain $s = d(\mathbf{M}^*)$ of the objective function. To extract these values, it is in fact not necessary to compute the objective function $\mathbf{M}^*$ completely, but only the marginals $\mathbf{M}^{*\downarrow\{X,Y\}}$ must be available. The queries of the inference problem are therefore given by sets of variables $\{X, Y\}$ for non-diagonal entries and $\{X\}$ for the diagonal entries of the objective function. Depending on the structure of the query set, we retrieve the classical types of path problems: when the query set consists of a single element $\{X, Y\}$, the inference problem corresponds to a *single-pair algebraic path problem.* If the query set if formed by all possible pairs of variables $\{X, Y\}$, we solve the *all-pairs algebraic path problem.* If for a selected variable $X \in s$, the query set contains the pairs $\{X, Y\}$ for all nodes $Y \in s$ with $X \neq Y$, we find the *single-source algebraic path problem.* Note that since we consider sets of variables, this is equivalent to the *single-target algebraic path problem.* Finally, if the non-empty query set has no such structure, we refer to the inference problem as the *multiple-pairs algebraic path problem.* It will later be shown in this section that we can completely pass on the explicit definition of query sets when dealing with Kleene valuation algebras. This, however, requires some complexity considerations with respect to the local computation process. Until then, we assume such an explicit query set as part of the inference problem and next focus on the local computation based solution process.

Given an inference problem with a knowledgebase of Kleene valuations and a query set, its solution can be delegated to one of the generic local computation methods introduced in Chapter 4. Since Kleene valuation algebras are idempotent, we go for the particularly simple idempotent architecture of Section 4.5. We first construct a join tree (V, E, λ, D) that covers the inference problem according to Definition 3.8 and execute the chosen architecture. At then end of the message-passing, each join tree node $i \in V$ contains $\mathbf{M}^{*\downarrow\lambda(i)}$ as a consequence of Theorem 4.4. The result of each query $\{X, Y\}$ is finally obtained by finding a covering node $i \in V$ with $\{X, Y\} \subseteq \lambda(i)$ and performing one last projection per query

$$\left(\mathbf{M}^*\right)^{\downarrow\{X,Y\}} \quad = \quad \left(\mathbf{M}^{*\downarrow\lambda(i)}\right)^{\downarrow\{X,Y\}}. \tag{9.38}$$

This shows how local computation solves multiple-pairs path problems represented as inference problems over Kleene valuation algebras. Let us next consider the complexity of this approach in more detail: since Kleene valuations are labeled matrices, the space requirement to store a valuation $\mathbf{M}^*$ with domain $d(\mathbf{M}^*) = s$ is $\mathcal{O}(|s|^2)$. Concerning the time complexity of the valuation algebra operations, we first remark that combination is the more expensive operation. Projection only drops rows and columns, whereas the combination of two Kleene valuations $\mathbf{M}_1^*$ and $\mathbf{M}_2^*$ with domains $d(\mathbf{M}_1^*) = s$ and $d(\mathbf{M}_2^*) = t$ performs a component-wise addition of the two matrices and computes the closure of the result. Applying the Warshall-Floyd-Kleene algorithm of Section 9.3.1 therefore gives a time-complexity of $\mathcal{O}(|s \cup t|^3)$ for the operation of combination. Based on the time and space complexity of Kleene valuations and the generic complexity analysis for the idempotent architecture in Section 4.5.1, we conclude that the message-passing process adopts a total time and space complexity of

$$\mathcal{O}\Big(|V| \cdot (\omega^* + 1)^3\Big) \quad \text{and} \quad \mathcal{O}\Big(|V| \cdot (\omega^* + 1)^2\Big), \tag{9.39}$$

where ω^* denotes the treewidth of the inference problem and $|V|$ the number of join tree nodes. This may be compared with a direct computation of $\mathbf{M}^*$ that adopts a time and space complexity of $\mathcal{O}(|s|^3)$ and $\mathcal{O}(|s|^2)$ respectively. Since it is always possible to apply the fusion and bucket elimination algorithms to knowledgebases of Kleene valuations, we may assume that $|V| \approx |s|$ when comparing the two approaches. Depending on the treewidth and thus on the sparsity of the input matrix, the difference to the local computation complexity of equation (9.39) may be considerable, as shown in the following example.

Example 9.6 *Assume that we are charged with the task of computing the travelling distances of packets for a European express delivery service. Two different rules apply for national and international shipping: In order to guarantee fast delivery, packets are transported from the source to the target city directly, if both cities are in the same country. Otherwise, packets are first transported to an international distribution center. For simplicity, we assume that only one distribution center exists per country. If we for example send a packet from a Portuguese city to a French city, the packet is first transported to the Portuguese distribution center, then to the Spanish distribution center, then to the French distribution center and finally to the French address. It is furthermore assumed that the delivery service always chooses the shortest travelling distance and that packets are transported by car or train. The data of this example consists of a local map for each country including its distribution center and the 99 largest cities and villages. Hence, if we include* 43 *European countries with a common land frontier to another European country, we have* $|s| = 4300$. *In addition, the local road maps contain only the direct distances between neighboring cities and villages in the corresponding country and the direct distances between its own distribution center and the distribution centers of all neighboring countries with a common land frontier. If we model these maps as valuations over the tropical semiring of non-negative integers, we obtain a knowledgebase of 43 valuations. The* 9 *neighboring countries of Germany are never exceeded in our example, which implies that the German map contains exactly* 9 *international distances. The largest domain of all valuations in the knowledgebase is therefore* 109. *Let us compare different approaches for the computation of the required distances. First, we could unify all local road maps to a single European road map and compute its closure. The complexity of* $\mathcal{O}(|s|^3)$ *for computing closures by the Warshall-Floyd-Kleene algorithm gives us an estimation* *of* $4300^3 = 79'507'000'000$ *operations for this approach. A more intelligent way of solving this problem exploits the observation that it is in fact sufficient to know all shortest distances between the distribution centers. An international distance between two cities can then be computed by adding the shortest distances to their corresponding distribution centers and the distance between the two distribution centers. Hence, we create a European road map that contains only the distribution centers and compute its closure together with the closures of all local road maps for the national distances. Since the largest map has* 109 *cities, the computational effort of this approach is bounded by* $(43 + 1) \cdot 109^3 = 56'981'276$ *operations. This is a very rough estimation such that we can ignore the two additions needed to obtain an international distance. We observe that the second approach is more efficient because it exploits the problem structure. Finally, we could solve this problem by local computation. The OSLA-SC algorithm of Section 3.7 implemented in the NENOK framework (Pouly, 2010) returns a join tree with a treewidth of* $\omega^* + 1 = 109$. *This shows that the join tree identifies the same problem structure and local computation essentially mirrors the computations of the second approach.*

The complexity of solving a path problem by local computation depends on the treewidth of the inference problem, which also includes the query set. Query answering according to equation (9.38) presupposes that for each query $\{X, Y\}$, it is possible to find some join tree node $i \in V$ that covers this query. This important aspect was ignored in Example 9.6. The join tree must therefore ensure that each query is covered by some node, and this blows up the node labels. In other words, the larger the query set is, the larger becomes the treewidth of the inference problem, and the less efficient is local computation. The worst-case scenario is the all-pairs algebraic path problem where the query set is built from all possible pairs of variables. Every covering join tree for such an inference problem contains at least one node whose label equals the domain of the objective function. We thus have $\omega^* + 1 = |s|$ and the message-passing process essentially mirrors the direct computation of the objective function. Conversely, if the query set is empty (or if each query is covered by some knowledgebase factor), the treewidth depends only on the sparsity of the matrix, which results in the best possible performance of local computation. From this point of view, the query set may destroy the gain of the sparsity in the knowledgebase. However, there are two properties of Kleene valuation algebras that allow us to ignore the query set and benefit from a treewidth that only depends on the knowledgebase. These are the polynomial complexity of Kleene valuations and the property of idempotency. Section 4.6 and in particular Algorithm 4.7 propose a general procedure for the computation of uncovered queries based on the idempotent architecture. For a given inference problem, it is thus sufficient to find a covering join tree for the knowledgebase and execute a complete run of the idempotent architecture. This results in a join tree compilation of the path problem, from which the unconsidered queries can be computed by Algorithm 4.7: For each query $\{X, Y\}$, this algorithm searches two join tree nodes whose labels contain the variables X and Y respectively. Then, it computes the query by successively combining and projecting the neighboring node contents on the path between the two nodes. We observed in Section 4.6.1 that the complexity of this algorithm doubles the treewidth. Its time complexity is

$$\mathcal{O}\Big(|V| \cdot (2(\omega^* + 1))^3\Big) \;=\; \mathcal{O}\Big(|V| \cdot 8(\omega^* + 1)^3\Big) \;=\; \mathcal{O}\Big(|V| \cdot (\omega^* + 1)^3\Big).$$

Similarly, we obtain for the space complexity

$$\mathcal{O}\Big((2(\omega^* + 1))^2\Big) \;=\; \mathcal{O}\Big(4(\omega^* + 1)^2\Big) \;=\; \mathcal{O}\Big((\omega^* + 1)^2\Big)$$

since only one such factor has to be kept in memory. Repeating this procedure for each query provides the better option for two reasons: first, we keep the smallest possible treewidth that only depends on the knowledgebase and second, we can answer new incoming queries dynamically without reconstructing the join tree. In addition, this procedure does not presume any structure in the query set which is contrary to many other approaches for the solution of path problems. On the other hand, it is clear that solving the all-pairs algebraic path problem with $|V|^2/2$ different queries still exceeds the effort of computing the objective function directly. Nevertheless, it is an efficient method for the solution of sparse multi-pairs algebraic path problems over Kleene algebras with a moderate number of queries.

A further query answering strategy is pointed out by the compilation algorithm 4.8 for uncovered queries in Section 4.6. Assume, for example, that we want to solve the single-source problem for a knowledgebase of Kleene valuations. We first execute a complete run of the idempotent architecture to obtain the join tree compilation. Then, if $S \in s$ denotes the selected source variable, we search a node $i \in V$ with $S \in \lambda(i)$. Algorithm 4.8 then computes a new factorization such that each node $j \in V$ contains the projection of $\mathbf{M}^*$ relative to $\lambda(i) \cup \lambda(j)$. This corresponds to adding the node label $\lambda(i)$ to each join tree node and is sometimes referred to as *distance tree*. For an arbitrary variable $T \in \lambda(j)$, we thus obtain the two values $\mathbf{M}^*(S, T)$ and $\mathbf{M}^*(T, S)$ from the extended domain of node $j \in V$. In other words, this algorithm solves the single-source and single-target path problem simultaneously with the same time complexity of answering a single query with Algorithm 4.7. In fact, applied to the shortest distance problem, this procedure corresponds to the classical construction of a *shortest path tree* rooted at variable S and confirms the common knowledge that solving a single-source path problem using *Dijkstra's algorithm* (Dijkstra, 1959) or the *Bellman-Ford algorithm* (Ford, 1956; Bellman, 1958) does not take more effort than computing a single-pair path problem. A similar way of deriving a distance tree from a join tree factorization but limited to shortest distances is proposed in (Chaudhuri & Zaroliagis, 1997). Finally, we could tackle the all-pairs algebraic path problem by solving the single-source problem for each variable. This corresponds to $|s| \approx |V|$ repetitions of Algorithm 4.8 on the same propagated join tree that results in a time complexity of

$$\mathcal{O}\Big(|V|^2 \cdot (\omega^* + 1)^3\Big).$$

For extremely sparse matrices, i.e. if the number of non-zero values is $\mathcal{O}(\sqrt[3]{|V|})$, this might be more efficient than the direct computation of the objective function.

On the other hand, it is improbable that a better time complexity can be reached for the all-pairs problem based on Kleene valuation algebras. We need to compute $|V|^2$ different values and the combination rule requires the execution of a closure in each join tree node. However, we have seen in Section 9.3.2 that quasi-regular valuation algebras provide a more efficient way to solve the all-pairs problem. There are still $|V|^2$ values to be computed, but since the closure is part of the projection rule, the corresponding effort can further be reduced.

Finding Paths in Kleene Valuation Algebras

Chapter 6 presented several applications of path problems that are not related to graphs. Here, we return to the very first conception of path problems in weighted, directed graphs represented by an adjacency matrix $\mathbf{M}$ with values from a Kleene algebra $\langle A, +, \times, *, \mathbf{0}, \mathbf{1}\rangle$. For a pair (X, Y) of variables the value $\mathbf{M}^*(X, Y) \in A$ corresponds to the weight of the optimum path between X and Y, where the exact criterion of optimality depends on the Kleene algebra. However, Kleene algebras are idempotent semirings with the canonical partial order of equation (5.4). Property (SP4) in Lemma 5.5 then states that $a + b = \sup\{a, b\}$ for all $a, b \in A$. If furthermore the Kleene algebra is totally ordered, we conclude that $a + b = \max\{a, b\}$. A formal proof of this statement is given in Lemma 8.5. Computing the sum of all path weights between X and Y therefore amounts to choosing the maximum value among them with respect to the canonical semiring order. In other words, the algorithms for the computation of a matrix closure can be interpreted as a search algorithm under this setting. In addition, we also conclude that if $\mathbf{M}^*(X, Y)$ is the weight of the optimum path evaluated by comparing path weights, we can always find a path between X and Y with this total weight. Example 9.7 shows that this does not hold in partially ordered semirings.

Example 9.7 *We know from Section 5.1.1 that multi-dimensional semirings with a component-wise definition of addition and multiplication again form a semiring. We thus consider the two-dimensional semiring with values from the tropical semiring* $\langle \mathbb{N} \cup \{0, \infty\}, \min, +, \infty, 0\rangle$ *of non-negative integers. An example of a directed graph with values from this semiring is shown in Figure 9.9. There are two possible paths connecting the source node S with the terminal T. Evaluating the total path weights by multiplying their edge weights gives:* $(1, 2) \times (6, 2) = (7, 4)$ *and* $(3, 4) \times (1, 5) = (4, 9)$. *Finally, we solve the path problem by adding the path weights and obtain* $(7, 4) + (4, 9) = (4, 4)$. *There is no path between S and T whose weight equals the computed optimum path weight, because the multi-dimensional, tropical semiring is only partially ordered.*

For totally ordered Kleene algebras, the Warshall-Floyd-Kleene algorithm of Section 9.3.1 can be extended to build a so-called *predecessor matrix* simultaneously to the computation of the matrix closure (Aho *et al.*, 1974). For a pair of variables (S, T) the predecessor matrix $\mathbf{P}$ determines the variable $X = \mathbf{P}(S, T)$ that is the immediate predecessor of variable T on the optimum path from S to T. The corresponding extension of Algorithm 9.1 is shown in Algorithm 9.2:

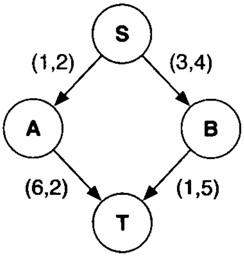


Figure 9.9 A path problem with values from a partially ordered semiring.

Algorithm 9.2 The Warshall-Floyd-Kleene Algorithm with Predecessor Matrix

```
input:  $\mathbf{M} \in \mathcal{M}(A,s)$   output:  $\mathbf{M}^*$, $\mathbf{P}$

begin
  for each $X,Y \in s$ do                              // Initialization.
    $\mathbf{M}_0(X,Y)$ := $\mathbf{M}(X,Y)$
    $\mathbf{P}(X,Y)$ := $X$;
  end;

  for each $Z = 1 \dots |s|$ do
    for each $X,Y \in s$ do
      $c$ := $\mathbf{M}_{Z-1}(X,Z)(\mathbf{M}_{Z-1}(Z,Z))^*\mathbf{M}_{Z-1}(Z,Y)$;
      $\mathbf{M}_Z(X,Y)$ := $\mathbf{M}_{Z-1}(X,Y) + c$;
      if $\mathbf{M}_Z(X,Y) \neq \mathbf{M}_{Z-1}(X,Y)$ do
        $\mathbf{P}(X,Y)$ := $\mathbf{P}(Z,Y)$;
      end;
    end;
  end;

  for each $X \in s$ do                           // Addition of unit matrix.
    if $\mathbf{1} > \mathbf{M}_Z(X,X)$ do
      $\mathbf{M}_Z(X,X)$ := $\mathbf{1}$;
      $\mathbf{P}(X,X)$ := $X$;
    end;
  end;
  return $\mathbf{M}_Z$, $\mathbf{P}$;
end
```

This algorithm still performs the same computations for the closure matrix. Because addition corresponds to maximization with respect to the canonical semiring order, the value $\mathbf{M}_Z(X,Y)$ in the middle block is set to the larger value between $\mathbf{M}_{Z-1}(X,Y)$ and c. In the second case, it means that a better path exists from variable X to variable Y through the intermediate variable Z. We therefore update the predecessor of variable Y according to this new path. Finally, the addition of the matrix $\mathbf{M}_Z$ with the unit matrix $\mathbf{I}_s$ in the last block affects the values $\mathbf{M}_Z(X,X)$ only if $\mathbf{1} > \mathbf{M}_Z(X,X)$. It is easy to see that both variations of the Warshall-Floyd-Kleene algorithm adopt equal time and space complexity. The optimum path can then be constructed from the predecessor matrix by recursively enumerating all predecessors. This is illustrated in Example 9.8.

Example 9.8 *We take the adjacency matrix of Example 6.2 defined over the tropical semiring of non-negative integers. Here, we identify the graph nodes with letters instead of numbers to prevent confusions with the semiring values. We obtain*

$$\mathbf{M} \;=\; \begin{array}{|c|cccc|} \hline & A & B & C & D \\ \hline A & \infty & 9 & 8 & \infty \\ B & \infty & \infty & 6 & \infty \\ C & \infty & \infty & \infty & 7 \\ D & 5 & \infty & \infty & \infty \\ \hline \end{array} \qquad \mathbf{M}^* \;=\; \begin{array}{|c|cccc|} \hline & A & B & C & D \\ \hline A & 0 & 9 & 8 & 15 \\ B & 18 & 0 & 6 & 13 \\ C & 12 & 21 & 0 & 7 \\ D & 5 & 14 & 13 & 0 \\ \hline \end{array}$$

The predecessor matrix obtained from Algorithm 9.2 is

$$\mathbf{P} \;=\; \begin{array}{|c|cccc|} \hline & A & B & C & D \\ \hline A & A & A & A & C \\ B & D & B & B & C \\ C & D & A & C & C \\ D & D & A & A & D \\ \hline \end{array}$$

Let us now look for the shortest path from C to B which, according to the closure matrix, has shortest distance 21. We obtain from the predecessor matrix $\mathbf{P}(C, B) = A$. This means that the last visited node just before reaching B on the path from C to B is A. We proceed recursively and obtain $\mathbf{P}(C, A) = D$ for the predecessor of A. Finally, $\mathbf{P}(C, D) = C$ means that the direct path between C and D also has the shortest distance. Putting things together, the shortest path with distance 21 is

$$p \;=\; (C, D)(D, A)(A, B).$$

Given a single matrix with values from a totally ordered Kleene algebra, the extended version of the Warshall-Floyd-Kleene algorithm computes the closure and its associated predecessor matrix with the same computational effort. Based on the predecessor matrix, it is then possible to recursively enumerate each path (X, Y) in linear time to the total number of variables. However, the advantage of using Kleene valuations and local computation for the solution of path problems is the abdication of the explicitly computation of the total matrix and its closure. When we are interested in paths, we must apply a similar strategy to deduce total paths only from the predecessor matrices of smaller subgraphs. For that purpose, we subsequently assume that Kleene valuations consist of pairs of closure and predecessor matrices and start from a knowledgebase $\{\phi_1, \ldots, \phi_n\}$ with $\phi_i = (\mathbf{M}_i^*, \mathbf{P}_i)$ where $\mathbf{P}_i$ corresponds to the predecessor matrix of $\mathbf{M}_i^*$ obtained from Algorithm 9.2. We define the domain of ϕ_i by $d(\phi_i) = d(\mathbf{M}_i^*)$. The projection rule still corresponds to matrix restriction and is applied to both components separately. Here, it is important to note that the values of the projected predecessor matrix may refer to variables that have been eliminated from the domain of the valuation. The combination is executed in the usual way for the closure component. It consists of a component-wise addition of the two extended matrices, followed by the execution of a closure operation. Since we are dealing with a totally ordered Kleene algebra, the component-wise addition of two matrices corresponds to the component-wise maximum, which allows us to initialize the predecessor matrix of the combination. For $\phi = (\mathbf{M}_1^*, \mathbf{P}_1)$ and $\psi = (\mathbf{M}_2^*, \mathbf{P}_2)$ with $d(\phi) = s$, $d(\psi) = t$ and $X, Y \in d(\phi) \cup d(\psi)$, we compute

$$\mathbf{M}(X,Y) \quad = \quad \mathbf{M}_1^{*\uparrow s \cup t}(X,Y) \; + \; \mathbf{M}_2^{*\uparrow s \cup t}(X,Y)$$

and

$$\mathbf{P}(X,Y) \quad = \quad \begin{cases} \mathbf{P}_1(X,Y) & \text{if } X,Y \in s \cap t \text{ and } \mathbf{M}_1^*(X,Y) \; \geq \; \mathbf{M}_2^*(X,Y), \\ \mathbf{P}_2(X,Y) & \text{if } X,Y \in s \cap t \text{ and } \mathbf{M}_1^*(X,Y) \; < \; \mathbf{M}_2^*(X,Y), \\ \mathbf{P}_1(X,Y) & \text{if } X,Y \in s - t, \\ \mathbf{P}_2(X,Y) & \text{if } X,Y \in t - s, \\ X & \text{otherwise.} \end{cases}$$

The combined valuation $\phi \otimes \psi$ is then obtained by computing the closure of $\mathbf{M}$ with $\mathbf{P}$ replacing the initialization of the predecessor matrix in Algorithm 9.2. We now observe that this produces a correct predecessor matrix: if computing the closure shows no effect, i.e. if $\mathbf{M}^*(X,Y) = \mathbf{M}(X,Y)$, then $\mathbf{P}(X,Y)$ either corresponds to $\mathbf{P}_1(X,Y)$, $\mathbf{P}_2(X,Y)$ or X. In the first two cases, the correctness follows from the assumption that ϕ and ψ are pairs of closures and their corresponding predecessor matrices. The third case is equal to the initialization in Algorithm 9.2. If on the other hand $\mathbf{M}^*(X,Y) \neq \mathbf{M}(X,Y)$, then some better path was found that goes through some variable $Z \in s \cup t$. The algorithm then sets $\mathbf{P}(X,Y) = \mathbf{P}(Z,Y)$ and the correctness follows by induction.

Let us now summarize the complete process of computing an optimum path from S to T by local computation. We start from a factorization of Kleene valuations extended by their predecessor matrices, build a covering join tree for this knowledgebase and execute a complete run of the idempotent architecture. We then answer the single-source problem for the variable S by the compilation approach presented above. If ϕ denotes the objective function, we answer the query

$$\phi^{\downarrow\{S,T\}} \quad = \quad \left(\mathbf{M}^{*\downarrow\{S,T\}}, \; \mathbf{P}^{\downarrow\{S,T\}}\right)$$

from which we obtain the optimum path weight $\mathbf{M}^*(S,T)$ and the predecessor $Z = \mathbf{P}(S,T)$ of T. We proceed recursively by computing the query $\phi^{\downarrow\{S,Z\}}$ and finally obtain the complete path by at most $|V|$ repetitions of this process. It is important to note that once the compilation for the single-source problem is available, no more computations in the valuation algebra are necessary. We thus obtain the path from S to T with a complexity equal to just computing the path weight.

9.4 CONCLUSION

The first part of this chapter deals with sparse, linear systems $\mathbf{AX} = \mathbf{b}$ with values from a field. Exploiting the sparsity of the matrix $\mathbf{A}$ means to limit the number of zero values that change to a non-zero value during the computations. These values are called fill-ins. If no additional structure of the matrix $\mathbf{A}$ is present, then ordinary Gaussian elimination with pivoting is performed. This corresponds to computing in the valuation algebra of affine spaces, where the equations are considered as finite representations of the latter. The equations are distributed over the nodes of a join tree, such that the non-zero values in the matrix $\mathbf{A}$ are covered by the join tree nodes. Since only these values are involved in the computations, fill-ins are controlled by the join tree. Similar considerations for regular systems lead to $\mathbf{LDR}$-decomposition based on local computation. A second valuation algebra, which is isomorphic to Gaussian potentials, was derived for the case of symmetric, positive definite matrices. Here, the valuations are no more ordinary equations, but they are symmetric, positive definite subsystems themselves. It was shown that such factorizations can either be produced from the matrix, or they occur naturally from specific applications as for example for the normal equations in the least squares method. Local computation with symmetric, positive definite systems required replacing the ordinary distribute phase by a solution construction process. The second part of this chapter focused on fixpoint equation systems $\mathbf{X} = \mathbf{AX} + \mathbf{b}$ with values from a quasi-regular semiring. Such equations frequently occur in path problems and their local computation based solution uses quasi-regular valuation algebras, whose structure is very similar to symmetric, positive definite systems. Consequently, we have a similar solution construction process that follows the usual collect phase of local computation. The second approach was based on Kleene valuation algebras and can be applied to fixpoint systems with values from a Kleene algebra. The application of the idempotent architecture to such knowledgebases lead to a compilation of the path problem into a join tree that qualifies for online query answering.

PROBLEM SETS AND EXERCISES

I.1 ★★ We observed in the introduction of Section 9.3.2 that the valuation algebra of linear systems with symmetric, positive definite matrices and the quasi-regular valuation algebra have a very similar structure. Given a symmetric, positive definite system that represents the objective function ϕ, it was shown in Section 9.2.4 how a possible decomposition $\phi = \phi_1 \otimes \ldots \otimes \phi_m$ can be found. Repeat these considerations for quasi-regular valuation algebras, where the matrices are not necessarily symmetric anymore.

I.2 ★★ Section 9.3.2 discusses local computation with quasi-regular valuation algebras. This process is very similar to local computation with symmetric, positive definite systems in Section 9.2.2, from which we derived $\mathbf{LDL}^T$-decomposition in

Section 9.2.3. Show that **LDR**-decomposition based on local computation is also possible for semiring fixpoint equation systems and thus for quasi-regular valuations. Indication: It is shown in (Backhouse & Carré, 1975) that **LDR**-decomposition is possible for quasi-regular semirings with idempotent addition. This approach is generalized to quasi-regular semirings in (Radhakrishnan *et al.*, 1992).

I.3 ★★ In graph related path problems we are often not only interested in the solution of the path problem (e.g. the shortest distance) but also in finding a path that adopts this optimum value (e.g. the shortest paths). At the end of Section 9.3.3 this problem was addressed for Kleene valuation algebras. Provide a similar method that computes paths for quasi-regular valuation algebras.

I.4 ★★ In the least squares method of Section 9.2.5 the residues are often weighed.

a) Derive the normal equations, if residue D_i has weight σ_i such that

$$\sum_{i=1}^{n} \sigma_i^2 D_i^2$$

is to be minimized.

b) More generally, derive the normal equations if $\mathbf{D}^T \Sigma \mathbf{D}$ is to be minimized. The weight matrix Σ is a positive definite, symmetric matrix.

c) Consider a join tree decomposition of the equations. What conditions must Σ satisfy in order that the normal equations can be solved by local computation on the join tree?

I.5 ★★ Reconsider the linear dynamic system from Instance 9.2:

a) Add the equation $\mathbf{H}_3\mathbf{X}_3 = \mathbf{y}_3$ to the equations in Instance 9.2. How does the estimate of $\mathbf{X}_3$ change? This gives the filter solution for $\mathbf{X}_3$.

b) Treat this system in the general case for time $i = 1, \ldots, n$ and derive the one-step prediction and the filter solution for $\mathbf{X}_i$.

c) Assume in the dynamic equations and the measurement equations that the residues $\mathbf{D}_i$ and $\mathbf{E}_i$ are weighed with weight matrices Q_i and R_i (all positive definite and symmetric). Derive the filter and smoothing solutions in this case.

d) Note that the solution derived so far works if all $\mathbf{A}_i$ are regular. No assumption about the rank of $\mathbf{H}_i$ is needed. Can you extend the valuation algebra of Section 9.2.1 for non-negative definite, symmetric matrices? Indication: Take $(\mathbf{A}, \mathbf{Ab})$ as valuations instead of $(\mathbf{A}, \mathbf{b})$. Similar algebras will be studied in Chapter 10.

I.6 ★★ The equations of the linear dynamic system in Instance 9.2 may also be differently assigned to a join tree: add a node $\mathbf{X}_0$ to the left of the join tree in Figure

9.8 and put the equation $\mathbf{H}_0\mathbf{X}_0$ on this node. On the next node $\mathbf{X}_0, \mathbf{X}_1$ put the equations $\mathbf{A}_0\mathbf{X}_0 - \mathbf{X}_1 = \mathbf{0}$ and $\mathbf{H}_1\mathbf{X}_1\mathbf{y}_1$ and so forth.

a) Derive the filter solution with this set-up.

b) Derive the smoothing solutions with this set-up.

I.7 ★★★ The exact algebraic relationship between the quasi-regular valuation algebra of Section 6.4 and the symmetric, positive definite valuation algebra of Section 9.2 is yet unknown. The valuations both have a very similar structure, share the same combination rule and also have a related projection rule. On the other hand, there are also important differences, e.g. the quasi-regular valuation algebra provides neutral elements which is not the case in the other algebra. Try to find a common theory for the two algebras.

CHAPTER 10

GAUSSIAN INFORMATION

In this chapter we look at linear systems with stochastic disturbances. Imagine, for example, that we want to determine a physical state of a certain object by repeated measurement. Then, the observed measurement results are always composed of the real state and some unknown measurement errors. In many important applications, however, these stochastic disturbances may be assumed to have a normal or Gaussian distribution. Together with accompanying observations, such a system forms what we call a *Gaussian information*. We are going to discuss in this chapter how inference from Gaussian information can be carried out. This leads to a compact representation of Gaussian information in the form of *Gaussian potentials*, and it will be shown that these potentials form a valuation algebra. We may therefore apply local computation for the inference process which exploits the structure of the Gaussian information. This chapter is largely based on (Eichenberger, 2009).

Section 10.1 defines an important form of Gaussian information that is used for assumption-based reasoning. This results in a particular stochastic structure, called *precise Gaussian hint*, which is closely related to the Gaussian potentials of Instance 1.6. In fact it provides meaning to these potentials and shows that combination and projection of Gaussian potentials reflect natural operations on the original Gaussian

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

information. This gives sense to the valuation algebra of Gaussian potentials that was introduced in Instance 1.6 on a purely algebraic level. Section 10.2 starts with an illustrative example to motivate a necessary generalization of the concept of Gaussian information and also indicates how inference can be adapted to this more general form. Then, Section 10.3 rigorously carries through the sketched approach, which finally results in an important extension of the valuation algebra of Gaussian potentials. Section 10.2 will also be developed in another direction in Section 10.6. Here, we show that the extended valuation algebra of Gaussian potentials serves to compute Gaussian networks. Finally, Section 10.5 applies the results of this chapter to the important problems of *filtering*, *smoothing* and *prediction* in *Gaussian time-discrete dynamic systems.*

10.1 GAUSSIAN SYSTEMS AND POTENTIALS

In this section, we consider models that explain how certain parameters generate certain output values or observations. In a simple but still very important case, we may assume that they do this in a *linear* manner, which means that the output values are linear combinations of the parameters. Often, there are additional disturbance terms that also influence the output. Let us call the parameters $x_1, \ldots, x_n$, the observed output values $z_1, \ldots, z_m$ and let the term which influences observation z_i be ω_i. The following model describes how the parameters generate the output values

$$\sum_{i=1}^{n} a_{j,i}x_i + \omega_j \quad = \quad z_j,$$

where $j = 1, \ldots, m$. This is often called a *linear functional model.* Then, the problem considered in this section is the following: Given this functional model and m observations $z_1, \ldots, z_m$, try to determine the unknown values of the parameters $x_1, \ldots, x_m$. Since the parameters are unknown, we replace them by variables $X_1, \ldots, X_m$ and obtain for $j = 1, \ldots, m$ the system of linear equations

$$\sum_{i=1}^{n} a_{j,i}X_i + \omega_j \quad = \quad z_j. \tag{10.1}$$

This system rarely has a unique solution for the unknown variables, and it is often assumed that the system is overdetermined, i.e. the number of equations exceeds the number of unknowns, $m \geq n$. The method of *least squares*, presented in Section 9.2.5, provides a popular approach to this problem. It determines the unknowns $x_1, \ldots, x_n$ by minimizing the sum of squares

$$\sum_{j=1}^{m}\Big(z_j - \sum_{i=1}^{n} a_{j,i}X_i\Big)^2.$$

We propose a different approach from least squares that yields more information about the unknowns, but at the price of an additional assumption: we assume that the disturbances ω_j have a Gaussian distribution with known parameters. For that purpose, it is convenient to change into matrix notation. Let $\mathbf{A}$ be the matrix with elements $a_{j,i}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. $\mathbf{X}$ is the vector with components X_i and ω and $\mathbf{z}$ are the vectors with components ω_j and z_j respectively. System (10.1) can then be written as

$$\mathbf{AX} + \omega \quad = \quad \mathbf{z}. \tag{10.2}$$

The vector of disturbances ω is assumed to have a Gaussian distribution as in equation (1.20) with the zero vector $\mathbf{0}$ as mean and an $m \times m$ concentration matrix $\mathbf{K}$. We further assume an overdetermined system where $m \geq n$ and also that matrix $\mathbf{A}$ has full column rank n. This setting is called a *linear Gaussian system* whereas more general systems are considered in Section 10.3. The following example of a linear Gaussian system is taken from (Pearl, 1988), see also (Kohlas & Monney, 2008; Eichenberger, 2009).

Example 10.1 *This example considers the causal model of Figure 10.1 for estimating the wholesale price of a car. There are observations of quantities that influence the wholesale price (i.e. production costs and marketing costs) and quantities that are influenced by the wholesale price (i.e. the selling prices of dealers). Besides, each observation has an associated Gaussian random term that simulates the variation of estimations and profits. We are interested in determining the wholesale price of a car given the final selling prices of dealers and the estimated costs for production and marketing. Let W denote the wholesale price of the car, U_1 the production costs and U_2 the marketing costs. The manufacturer's profit U_3 is influenced by a certain profit variation ω_3. We thus have*

$$W \quad = \quad U_1 + U_2 + U_3 + \omega_3.$$

The selling prices Y_1 and Y_2 of two dealers are composed of the wholesale price and the mean profits Z_1 and Z_2 of each dealer. They are subject to variations ω_1 and ω_2:

$$\begin{aligned} Y_1 &= W + Z_1 + \omega_1, \\ Y_2 &= W + Z_2 + \omega_2. \end{aligned}$$

The manufacturer's production costs U_1 are estimated by two independent experts. Both estimates I_1 and I_2 are affected by estimation errors ν_1 and ν_2:

$$\begin{aligned} I_1 &= U_1 + \nu_1, \\ I_2 &= U_1 + \nu_2. \end{aligned}$$

The marketing costs U_2 are also estimated by two independent marketing experts. Again, both estimates J_1 and J_2 are affected by errors λ_1 and λ_2:

$$\begin{aligned} J_1 &= U_2 + \lambda_1, \\ J_2 &= U_2 + \lambda_2. \end{aligned}$$

We bring this system into the standard form:

$$\begin{array}{ccccccccc}
W & & & & & + & \omega_1 & = & Y_1 - Z_1, \\
W & & & & & + & \omega_2 & = & Y_2 - Z_2, \\
 & & U_1 & & & + & \nu_1 & = & I_1, \\
 & & U_1 & & & + & \nu_2 & = & I_2, \\
 & & & & U_2 & + & \lambda_1 & = & J_1, \\
 & & & & U_2 & + & \lambda_2 & = & J_2, \\
W & - & U_1 & - & U_2 & - & \omega_3 & = & U_3.
\end{array}$$

This is an overdetermined system with $m = 7$ and $n = 3$. The corresponding matrix $\mathbf{A}$ has full column rank.

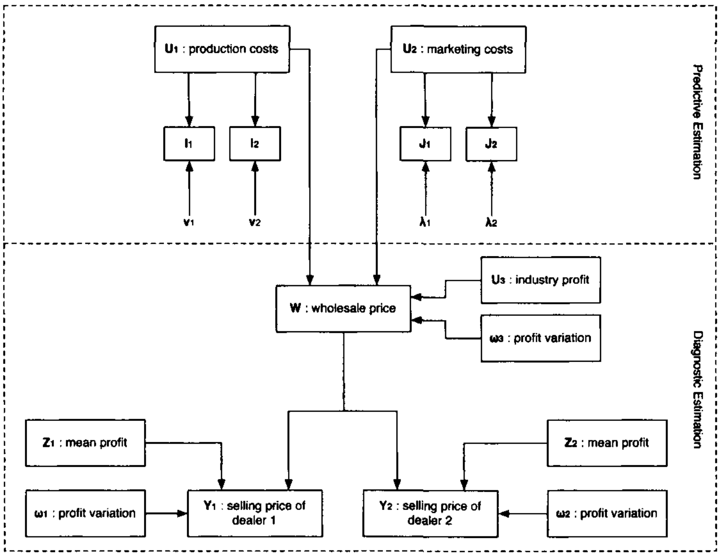


Figure 10.1 A causal model for the wholesale price of a car.

The idea of *assumption-based reasoning* is the following: although ω is unknown in equation (10.2), we may tentatively assume a value for it and see what can be inferred about the unknown parameter $\mathbf{x}$ under this assumption. First we note this system only has a solution if the vector $\mathbf{z} - \omega$ is in the linear space spanned by the column vectors of $\mathbf{A}$. The crucial observation of assumption-based reasoning is that all ω which do not satisfy this condition are excluded by the observation of $\mathbf{z}$, since $\mathbf{z}$ has been generated by some parameter vector $\mathbf{x}$ according to the functional model (10.2). Thus, admissible disturbances ω are only those vectors for which (10.2) has a solution. In other words, the observation $\mathbf{z}$ defines an event

$$v_{\mathbf{z}} \quad = \quad \{\omega \in \mathbb{R}^m : \exists \mathbf{x} \text{ such that } \omega = \mathbf{z} - \mathbf{A}\mathbf{x}\}$$

in the space of disturbances and we have to condition the distribution of ω on this event. The set $v_{\mathbf{z}}$ is clearly an affine subspace of $\mathbb{R}^m$. In order to compute this conditional distribution, we apply a linear transformation to the original system (10.2) which does not change its solutions. Since the rank of matrix $\mathbf{A}$ equals n, there must be n linearly independent rows in $\mathbf{A}$. If necessary, we renumber the equations and can therefore assume that the first n rows are linearly independent. They form a submatrix $\mathbf{A}_1$ of $\mathbf{A}$, and $\mathbf{A}_2$ denotes the submatrix formed by the remaining $m-n$ rows. We then define the $m \times m$ matrix

$$\mathbf{B} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{A}_2 & \mathbf{I}_{m-n} \end{bmatrix}$$

where $\mathbf{I}_{m-n}$ is the $(m-n) \times (m-n)$ identity matrix. Clearly, $\mathbf{B}$ has full rank m and has therefore the inverse

$$\mathbf{T} = \mathbf{B}^{-1} = \begin{bmatrix} \mathbf{A}_1^{-1} & \mathbf{0} \\ -\mathbf{A}_2\mathbf{A}_1^{-1} & \mathbf{I}_{m-n} \end{bmatrix}.$$

Let us now consider the transformed system $\mathbf{TAX} + \mathbf{T}\omega = \mathbf{Tz}$. Since the matrix $\mathbf{T}$ is regular this system has the same solutions as the original. This system can be written as

$$\begin{aligned} \mathbf{X} + \tilde{\omega}_1 &= \mathbf{A}_1^{-1}\mathbf{z}_1 \\ \tilde{\omega}_2 &= -\mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{z}_1 + \mathbf{z}_2 \end{aligned} \tag{10.3}$$

where

$$\begin{aligned} \tilde{\omega}_1 &= \mathbf{A}_1^{-1}\omega_1, \\ \tilde{\omega}_2 &= -\mathbf{A}_2\mathbf{A}_1^{-1}\omega_1 + \omega_2 \end{aligned}$$

and $\mathbf{z}_1$, ω_1 and $\mathbf{z}_2$, ω_2 are the subvectors of the first n and the remaining $m-n$ components of $\mathbf{z}$ and ω respectively. The advantage of the transformed system (10.3) is that the solution is given explicitly as a function of $\tilde{\omega}_1$, whereas $\tilde{\omega}_2$ has a constant value. In this situation, the conditional density of $\tilde{\omega}_1$ given $\tilde{\omega}_2 = -\mathbf{A}_2\mathbf{A}_1^{-1}\omega_1 + \omega_2$ can be determined by classical results of Gaussian distributions (see appendix to this chapter). First, we note that the transformed disturbance vector $\tilde{\omega}$ with the components $\tilde{\omega}_1$ and $\tilde{\omega}_2$ has still a Gaussian density with mean vector $\mathbf{0}$ and concentration matrix

$$\begin{aligned} \tilde{\mathbf{K}} &= \mathbf{B}^T\mathbf{K}\mathbf{B} \\ &= \begin{bmatrix} \mathbf{A}_1^T & \mathbf{A}_2^T \\ \mathbf{0} & \mathbf{I}_{m-n} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{1,1} & \mathbf{K}_{1,2} \\ \mathbf{K}_{2,1} & \mathbf{K}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} \\ \mathbf{A}_2 & \mathbf{I}_{m-n} \end{bmatrix} \\ &= \begin{bmatrix} \begin{array}{c} \mathbf{A}_1^T\mathbf{K}_{1,1}\mathbf{A}_1 + \mathbf{A}_1^T\mathbf{K}_{1,2}\mathbf{A}_2 + \mathbf{A}_2^T\mathbf{K}_{2,1}\mathbf{A}_1 \\ +\mathbf{A}_2^T\mathbf{K}_{2,2}\mathbf{A}_2 \end{array} & \mathbf{A}_1^T\mathbf{K}_{1,2} + \mathbf{A}_2^T\mathbf{K}_{2,2} \\ \mathbf{K}_{2,1}\mathbf{A}_1 + \mathbf{K}_{2,2}\mathbf{A}_2 & \mathbf{K}_{2,2} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}^T\mathbf{K}\mathbf{A} & \mathbf{A}_1^T\mathbf{K}_{1,2} + \mathbf{A}_2^T\mathbf{K}_{2,2} \\ \mathbf{K}_{2,1}\mathbf{A}_1 + \mathbf{K}_{2,2}\mathbf{A}_2 & \mathbf{K}_{2,2} \end{bmatrix}. \end{aligned}$$

The correctness of the first equality will be proved in Appendix J.2. Then, the conditional distribution of $\tilde{\omega}_1$ given $\tilde{\omega}_2 = -\mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{z}_1 + \mathbf{z}_2$ is still Gaussian and has concentration matrix $\mathbf{A}^T\mathbf{K}\mathbf{A}$ (see Appendix J.2) and mean vector

$$(\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}\left(\mathbf{A}_1^T\mathbf{K}_{1,2} + \mathbf{A}_2^T\mathbf{K}_{2,2}\right)\left(\mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{z}_1 - \mathbf{z}_2\right).$$

Note now that according to (10.3) we have

$$\mathbf{X} \quad = \quad \mathbf{A}_1^{-1}\mathbf{z}_1 - \tilde{\omega}_1.$$

Since the stochastic term $\tilde{\omega}_1$ has a Gaussian distribution, the same holds for $\mathbf{X}$. In fact, it has the same concentration matrix $\mathbf{A}^T\mathbf{K}\mathbf{A}$ as $\tilde{\omega}_1$. Its mean vector is

$$\begin{aligned}
\mathbf{A}_1^{-1}\mathbf{z}_1 - (\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}\left(\mathbf{A}_1^T\mathbf{K}_{1,2} + \mathbf{A}_2^T\mathbf{K}_{2,2}\right)\left(\mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{z}_1 - \mathbf{z}_2\right) &= \\
\mathbf{A}_1^{-1}\mathbf{z}_1 - (\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}\begin{bmatrix} \mathbf{0} \\ \mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{z}_1 - \mathbf{z}_2 \end{bmatrix} &= \\
\mathbf{A}_1^{-1}\mathbf{z}_1 - (\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}\begin{bmatrix} \mathbf{A}_1\mathbf{A}_1^{-1}\mathbf{z}_1 - \mathbf{z}_1 \\ \mathbf{A}_2\mathbf{A}_1^{-1}\mathbf{z}_1 - \mathbf{z}_2 \end{bmatrix} &= \\
\mathbf{A}_1^{-1}\mathbf{z}_1 - (\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}(\mathbf{A}^T\mathbf{K}\mathbf{A})\mathbf{A}_1^{-1}\mathbf{z}_1 + (\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}\mathbf{z} &= \\
(\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}\mathbf{z}.
\end{aligned}$$

We thus obtain for the unknown parameter in equation (10.2) a Gaussian density or a Gaussian potential

$$\left((\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}\mathbf{z},\ \mathbf{A}^T\mathbf{K}\mathbf{A}\right). \tag{10.4}$$

Assumption-based reasoning therefore infers from an overdetermined linear Gaussian systems a Gaussian potential. How is this result to be interpreted? The unknown parameter vector $\mathbf{x}$ is finally not really a random variable. Assumption-based analysis simply infers that for a feasible assumption ω about the random distributions, the unknown parameter vector would take the value $\mathbf{A}_1^{-1}\mathbf{z}_1 - \tilde{\omega}$. But this expression has a Gaussian density with parameters as specified in the potential above. So, a hypothesis about $\mathbf{x}$, like $\mathbf{x} \leq \mathbf{0}$ for instance, is satisfied for all feasible assumptions ω satisfying $\mathbf{A}_1^{-1}\mathbf{z}_1 - \tilde{\omega} \leq \mathbf{0}$, and the probability that the assumptions satisfy this condition can be obtained from the Gaussian density above. This is the probability that the hypothesis can be derived from the model. The Gaussian density determined by the Gaussian potential is also called a *fiducial distribution*. The concept of fiducial probability goes back to the statistician Fisher (Fisher, 1950; Fisher, 1935). For its present interpretation in relation to assumption-based reasoning we refer to (Kohlas & Monney, 2008; Kohlas & Monney, 1995).

Example 10.2 *We first execute a* diagnostic estimation *for the linear Gaussian system of Example 10.1. The term diagnostic means that observations are given with respect to the variables that are influenced by the wholesale price. These are the selling prices* *and the mean profits of both dealers, and inference is performed on the disturbance vector* $\omega = \begin{bmatrix} \omega_1 & \omega_2 \end{bmatrix}$. *We thus consider the first two equations of the system:*

$$\begin{aligned} W + \omega_1 &= Y_1 - Z_1, \\ W + \omega_2 &= Y_2 - Z_2. \end{aligned}$$

If $\sigma^2_{\omega_i}$ *denotes the variance of the disturbance* ω_i *associated to* Y_i *we have*

$$\mathbf{A} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \textit{and} \quad \mathbf{K} = \begin{bmatrix} \frac{1}{\sigma^2_{\omega_1}} & 0 \\ 0 & \frac{1}{\sigma^2_{\omega_2}} \end{bmatrix}.$$

With respect to equation (10.4) we compute

$$\mathbf{A}^T\mathbf{KA} = \frac{\sigma^2_{\omega_1} + \sigma^2_{\omega_2}}{\sigma^2_{\omega_1}\sigma^2_{\omega_2}}$$

and

$$\mathbf{A}^T\mathbf{Kz} = \frac{Y_1 - Z_1}{\sigma^2_{\omega_1}} + \frac{Y_2 - Z_2}{\sigma^2_{\omega_2}}.$$

Finally, we obtain the following Gaussian potential for the wholesale price W *given* Y_1, Y_2, Z_1 *and* Z_2 *in the above system:*

$$\mathcal{G}_{W|Y_1,Y_2,Z_1,Z_2} = \left(\frac{\sigma^2_{\omega_1}\sigma^2_{\omega_2}}{\sigma^2_{\omega_1} + \sigma^2_{\omega_2}} \left(\frac{Y_1 - Z_1}{\sigma^2_{\omega_1}} + \frac{Y_2 - Z_2}{\sigma^2_{\omega_2}} \right), \frac{\sigma^2_{\omega_1} + \sigma^2_{\omega_2}}{\sigma^2_{\omega_1}\sigma^2_{\omega_2}} \right).$$

Example 10.3 *Let us next execute a* predictive estimation *for the Gaussian system of Example 10.1. The term predictive means that observations are given with respect to the variables that influence the wholesale price. These are the production costs* U_1 *and the marketing costs* U_2 *which are estimated by two experts, and the industry profit* U_3 *for which we have one estimation. We thus consider the remaining subsystem:*

$$\begin{array}{ccccccccc} & & U_1 & & & + & \nu_1 & = & I_1, \\ & & U_1 & & & + & \nu_2 & = & I_2, \\ & & & & U_2 & + & \lambda_1 & = & J_1, \\ & & & & U_2 & + & \lambda_2 & = & J_2, \\ W & - & U_1 & - & U_2 & - & \omega_3 & = & U_3. \end{array}$$

Observe that the last equation $W = U_1 + U_2 + U_3 + \omega_3$ *defines the wholesale price as a linear combination of the production costs, marketing costs and the industry profit. We may therefore conclude that the same holds for the Gaussian potential of the wholesale price given the estimations for the production costs, marketing costs and the industry profit. We have*

$$\mathcal{G}_{W|I_1,I_2,J_1,J_2,U_3} = \left(\mu_{W|I_1,I_2,J_1,J_2,U_3}, \mathbf{K}_{W|I_1,I_2,J_1,J_2,U_3} \right)$$

where

$$\mu_{W|I_1,I_2,J_1,J_2,U_3} \;=\; \mu_{U_1|I_1,I_2} + \mu_{U_2|J_1,J_2} + U_3$$

and

$$\mathbf{K}_{W|I_1,I_2,J_1,J_2,U_3} \;=\; \mathbf{K}_{U_1|I_1,I_2} + \mathbf{K}_{U_2|J_1,J_2} + \sigma_{\omega_3}^{-2}.$$

A similar analysis as in Example 10.2 gives:

$$\mu_{U_1|I_1,I_2} \;=\; \mathbf{K}_{U_1|I_1,I_2}^{-1}\left(\frac{I_1}{\sigma_{\nu_1}^2} + \frac{I_2}{\sigma_{\nu_2}^2}\right)$$

and

$$\mu_{U_2|J_1,J_2} \;=\; \mathbf{K}_{U_2|J_1,J_2}^{-1}\left(\frac{J_1}{\sigma_{\lambda_1}^2} + \frac{J_2}{\sigma_{\lambda_2}^2}\right)$$

with

$$\mathbf{K}_{U_1|I_1,I_2} = \frac{\sigma_{\nu_1}^2 + \sigma_{\nu_2}^2}{\sigma_{\nu_1}^2\sigma_{\nu_2}^2} \quad \textit{and} \quad \mathbf{K}_{U_2|J_1,J_2} = \frac{\sigma_{\lambda_1}^2 + \sigma_{\lambda_2}^2}{\sigma_{\lambda_1}^2\sigma_{\lambda_2}^2}.$$

Next, we are going to relate the operations of combination and projection in the valuation algebra of Gaussian potentials from Instance 1.6 to linear Gaussian systems. For this purpose, we fix a set r of variables with subsets s and t. So, let $\mathbf{X}$ denote an $(s \cup t)$-tuple of variables. We consider the system of linear equations with Gaussian disturbances composed of two subsystems,

$$\begin{aligned}\mathbf{A}_1\mathbf{X}^{\downarrow s} + \omega_1 &= \mathbf{z}_1,\\ \mathbf{A}_2\mathbf{X}^{\downarrow t} + \omega_2 &= \mathbf{z}_2.\end{aligned}$$

Here, we assume that $\mathbf{A}_1$ is an $m_1 \times s$ matrix, $\mathbf{A}_2$ an $m_2 \times t$ matrix and $\mathbf{z}_1, \omega_1$ and $\mathbf{z}_2, \omega_2$ are m_1 and m_2-vectors respectively. Both matrices are assumed to have full column rank $|s|$ and $|t|$ respectively. Further, we assume that ω_1 has a Gaussian density with mean $\mathbf{0}$ and concentration matrix $\mathbf{K}_1$, whereas ω_2 has a Gaussian density with mean $\mathbf{0}$ and concentration matrix $\mathbf{K}_2$. Both vectors are stochastically independent. Now, we can either determine the Gaussian potentials of the two systems:

$$G_1 \;=\; \left((\mathbf{A}_1^T\mathbf{K}_1\mathbf{A}_1)^{-1}\mathbf{A}_1\mathbf{z}_1,\; \mathbf{A}_1^T\mathbf{K}_1\mathbf{A}_1\right)$$

and

$$G_2 \;=\; \left((\mathbf{A}_2^T\mathbf{K}_2\mathbf{A}_2)^{-1}\mathbf{A}_2\mathbf{z}_2,\; \mathbf{A}_2^T\mathbf{K}_2\mathbf{A}_2\right),$$

or we can determine the Gaussian potential of the combined system

$$\begin{bmatrix} \mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow s\cap t} & \mathbf{A}_1^{\downarrow t-s} \end{bmatrix}\mathbf{X} + \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \;=\; \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}.$$

Here, the disturbance vector has the concentration matrix

$$\begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix}.$$

Remark that the matrix of this combined system has still full column rank $|s \cup t|$. The Gaussian potential of the combined system is $(\mu, \mathbf{K})$ with

$$\begin{aligned}
\mathbf{K} &= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t\;T} & \mathbf{A}_2^{\downarrow s\cap t\;T} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s\;T} \end{bmatrix} \begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow s\cap t} & \mathbf{A}_2^{\downarrow t-s} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T}\mathbf{K}_1\mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s-t\;T}\mathbf{K}_1\mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t\;T}\mathbf{K}_1\mathbf{A}_1^{\downarrow s-t\;T} & \begin{array}{c}\mathbf{A}_1^{\downarrow s\cap t\;T}\mathbf{K}_1\mathbf{A}_1^{\downarrow s\cap t\;T} \\ +\mathbf{A}_2^{\downarrow s\cap t\;T}\mathbf{K}_2\mathbf{A}_2^{\downarrow s\cap t\;T}\end{array} & \mathbf{A}_2^{\downarrow s\cap t\;T}\mathbf{K}_2\mathbf{A}_2^{\downarrow t-s\;T} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s\;T}\mathbf{K}_2\mathbf{A}_2^{\downarrow s\cap t\;T} & \mathbf{A}_2^{\downarrow t-s\;T}\mathbf{K}_2\mathbf{A}_2^{\downarrow t-s} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T}\mathbf{K}_1\mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s-t\;T}\mathbf{K}_1\mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t\;T}\mathbf{K}_1\mathbf{A}_1^{\downarrow s-t\;T} & \mathbf{A}_1^{\downarrow s\cap t\;T}\mathbf{K}_1\mathbf{A}_1^{\downarrow s\cap t\;T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\
&\quad + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow s\cap t\;T}\mathbf{K}_2\mathbf{A}_2^{\downarrow s\cap t\;T} & \mathbf{A}_2^{\downarrow s\cap t\;T}\mathbf{K}_2\mathbf{A}_2^{\downarrow s\cap t\;T} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s\;T}\mathbf{K}_2\mathbf{A}_2^{\downarrow s\cap t\;T} & \mathbf{A}_2^{\downarrow t-s\;T}\mathbf{K}_2\mathbf{A}_2^{\downarrow t-s} \end{bmatrix} \\
&= (\mathbf{A}_1^T\mathbf{K}_1\mathbf{A}_1)^{\uparrow s\cup t} + (\mathbf{A}_2^T\mathbf{K}_1\mathbf{A}_2)^{\uparrow s\cup t}.
\end{aligned}$$

Note that this is exactly the concentration matrix of the combined potential $G_1 \otimes G_2$ given in equation (1.21). For the mean vector μ of the combined system we have:

$$\begin{aligned}
\mu &= \mathbf{K}^{-1} \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t\;T} & \mathbf{A}_2^{\downarrow s\cap t\;T} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s\;T} \end{bmatrix} \begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} \\
&= \mathbf{K}^{-1} \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T}\mathbf{K}_1\mathbf{z}_1 \\ \mathbf{A}_1^{\downarrow s\cap t\;T}\mathbf{K}_1\mathbf{z}_1 + \mathbf{A}_2^{\downarrow s\cap t\;T}\mathbf{K}_2\mathbf{z}_2 \\ \mathbf{A}_2^{\downarrow t-s\;T}\mathbf{K}_2\mathbf{z}_2 \end{bmatrix} \\
&= \mathbf{K}^{-1}\left(\begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T}\mathbf{K}_1\mathbf{z}_1 \\ \mathbf{A}_1^{\downarrow s\cap t\;T}\mathbf{K}_1\mathbf{z}_1 \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{A}_2^{\downarrow s\cap t\;T}\mathbf{K}_2\mathbf{z}_2 \\ \mathbf{A}_2^{\downarrow t-s\;T}\mathbf{K}_2\mathbf{z}_2 \end{bmatrix} \right).
\end{aligned}$$

Let now $\mu_1 = (\mathbf{A}_1^T\mathbf{K}_1\mathbf{A}_1)^{-1}\mathbf{A}_1\mathbf{K}_1\mathbf{z}_1$ and $\mu_2 = (\mathbf{A}_2^T\mathbf{K}_2\mathbf{A}_2)^{-1}\mathbf{A}_2\mathbf{K}_2\mathbf{z}_2$ be the mean vectors of the two potentials G_1 and G_2. We then see that we can rewrite the last expression above as

$$\mathbf{K}^{-1}\Big((\mathbf{A}_1^T\mathbf{K}_1\mathbf{A}_1)^{\uparrow s\cup t}\mu_1^{\uparrow s\cup t} + (\mathbf{A}_2^T\mathbf{K}_2\mathbf{A}_2)^{\uparrow s\cup t}\mu_2^{\uparrow s\cup t}\Big).$$

But this is the mean vector of the combined potential $G_1 \otimes G_2$ as shown in equation (1.22). Thus, we have shown that

$$(\mu, \mathbf{K}) \quad = \quad G_1 \otimes G_2.$$

Therefore, combining two linear Gaussian systems results in the Gaussian potential, which is the combination of the Gaussian potentials of the individual systems. This justifies the combination operation which itself is derived from multiplying the associated Gaussian densities. The combination of Gaussian potentials as defined in Instance 1.6 therefore makes much sense.

Example 10.4 *Let us now assume that all values in the system of Example 10.1 have been observed. Then, inference can either be made for the complete model, or by combining the two potentials derived from the submodels in Example 10.2. We have*

$$G_{W|Y_1,Y_2,Z_1,Z_2,I_1,I_2,J_1,J_2,U_3} \; = \; (\mu, \mathbf{K}) \; = \; G_{W|Y_1,Y_2,Z_1,Z_2} \otimes G_{W|I_1,I_2,J_1,J_2,U_3}$$

The two components of the combined potential are:

$$\mathbf{K} \quad = \quad \mathbf{K}_{W|Y_1,Y_2,Z_1,Z_2} + \mathbf{K}_{W|I_1,I_2,J_1,J_2,U_3}$$

and

$$\mu \; = \; \mathbf{K}^{-1}\Big(\mathbf{K}_{W|Y_1,Y_2,Z_1,Z_2}\mu_{W|Y_1,Y_2,Z_1,Z_2} + \mathbf{K}_{W|I_1,I_2,J_1,J_2,U_3}\mu_{W|I_1,I_2,J_1,J_2,U_3}\Big).$$

Note that since we are dealing with potentials over the same single variable, there is no need for the extensions in the definition of combination. This is an example of solving a rather large system by decomposition and local computation. However, we remark that the theory is not yet general enough since the subsystems for the decompositions do not always have full column rank.

Now, suppose that $\mathbf{X}$ is a vector of variables from a subset $s \subseteq r$ which represents unknown parameters in a linear Gaussian system (10.1). Inference leads to a Gaussian potential $(\mu, \mathbf{K}) = ((\mathbf{A}^T\mathbf{K}_\omega\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}_\omega\mathbf{z}, (\mathbf{A}^T\mathbf{K}_\omega\mathbf{A}))$. If t is a subset of s, and we are only interested in the parameters in $\mathbf{X}^{\downarrow t}$, it seems intuitively reasonable that for this vector the marginal distribution of the associated Gaussian density is the correct fiducial distribution. This marginal is still a Gaussian density with the corresponding potential $(\mu^{\downarrow t}, ((\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1})^{\downarrow t})^{-1})$. It corresponds to the projection of the Gaussian potential $(\mu, \mathbf{K})$ to t as defined in Instance 1.6:

$$(\mu, \mathbf{K})^{\downarrow t} \quad = \quad \Big(\mu^{\downarrow t},\; ((\mathbf{A}^T\mathbf{K}_\omega\mathbf{A})^{-1})^{\downarrow t})^{-1}\Big). \tag{10.5}$$

In fact, this can be justified by the elimination of the variables $\mathbf{X}^{\downarrow s-t}$ in the system (10.1), but it is convenient to consider for this purpose the equivalent system (10.3). The second part of this system does not contain the variables at all, and in the first part we simply have to extract the variables in the set t to get the system

$$\mathbf{X}^{\downarrow t} \quad = \quad (\mathbf{A}_1\mathbf{z}_1)^{\downarrow t} - \tilde{\omega}_1^{\downarrow t}.$$

The distribution of $\tilde{\omega}_1$ has not changed by this operation. This therefore shows that the fiducial density of $\mathbf{X}^{\downarrow t}$ is indeed the marginal of the fiducial density of $\mathbf{X}$ (for the marginal of a Gaussian density see Appendix J.2).

By these considerations, Gaussian potentials are strongly linked to certain Gaussian systems, and the operations of combination and projection in the valuation algebra of Gaussian potentials are explained by corresponding operations on linear Gaussian systems. The valuation algebra of Gaussian potentials therefore obtains a clear significance through this connection. In particular, Gaussian potentials can be used to solve inference problems for large, sparse systems (10.1), since they can often be decomposed into many much smaller systems. This will be illustrated in the subsequent parts of this chapter. However, it will also become evident that an extension of the valuation algebra of Gaussian potentials will be useful and even necessary for some important problems. We finally refer to Section 9.2.5 where a different interpretation of the same algebra based on the least squares method is given.

10.2 GENERALIZED GAUSSIAN POTENTIALS

The purpose of this section is to show that the valuation algebra of Gaussian potentials is not always sufficient for the solution of linear, functional models by local computation. To do so, we present an important instance of such models named *time-discrete, dynamic system with Gaussian disturbances and error terms.*

■ 10.1 Gaussian Time-Discrete Dynamic Systems

Let $X_1, X_2, \ldots$ be state variables where the index refers to the time. The state at time $i+1$ is proportional to the state at time i plus a Gaussian disturbance:

$$\begin{array}{rcl} X_2 & = & aX_1 + \omega_1, \\ X_3 & = & aX_2 + \omega_2, \\ & \vdots & \end{array}$$

The state cannot be observed exactly but only with some measurement error. We denote the observation of the state at time i by z_i such that

$$\begin{array}{rcl} z_1 & = & X_1 + \nu_1, \\ z_2 & = & X_2 + \nu_2, \\ & \vdots & \end{array}$$

We further assume the disturbance terms ω_i and the measurement errors ν_i to be mutually independent. The ω_i are all assumed to have a Gaussian density with mean 0 and variance σ_ω^2, whereas the observation errors ν_i are assumed to have a Gaussian density with mean 0 and variance σ_ν^2. Such a system has an illustrative graphical representation which indicates that state X_{i+1} is influenced by state X_i and observation z_i by state X_i. This is shown in Figure 10.2. It resembles a Bayesian network like the one in Instance 2.1 with the difference that the involved variables are not discrete but continuous. The computational problem is to infer about the states of the system at time $i = 1, 2, \ldots$ given the equations above and the measurements z_i of the states at different points in time.

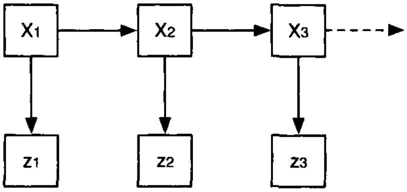


Figure 10.2 The graph of the time-discrete dynamic system.

Let us now consider a manageable size of this instance where z_i for $i = 1, 2, 3$ is given and the states X_1, X_2 and X_3 are to be determined. Then, the equations above can be rewritten in a format corresponding to the linear Gaussian systems introduced in the previous section:

$$\begin{array}{ccccccccc} X_1 & & & & & + & \omega_0 & = & m, \\ aX_1 & - & X_2 & & & + & \omega_1 & = & 0, \\ & & aX_2 & - & X_3 & + & \omega_2 & = & 0, \\ X_1 & & & & & + & \nu_1 & = & z_1, \\ & & X_2 & & & + & \nu_2 & = & z_2, \\ & & & & X_3 & + & \nu_3 & = & z_3. \end{array} \tag{10.6}$$

The first equation has been added to indicate an initial condition on state X_1. If we consider each individual equation of this system as a piece of information, or as a valuation, then we obtain a covering join tree for the system as shown in Figure 10.3. Each equation of the system is on one of the nodes of the join tree. This constitutes a nice decomposition of the total system. If we could derive a Gaussian potential for each node, we could solve the problem specified above by local computation. However, although each equation represents a linear Gaussian system, it does not satisfy the requirement of full column rank imposed in the previous section. In fact, the single equations do not represent overdetermined systems, but rather underdetermined systems. So, unfortunately, we cannot directly apply the results from the previous section and work with Gaussian potentials. This indicates the need of an extension of the theory of inference in linear Gaussian systems.

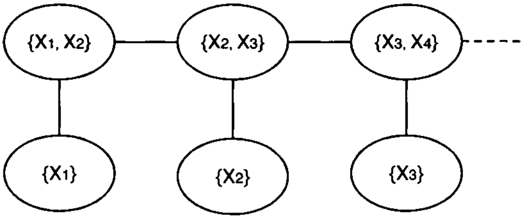


Figure 10.3 A covering join tree for the system (10.6).

In order to introduce the new concepts needed to treat such systems, let us have a closer look at the first two equations of system (10.6):

$$\begin{array}{ccccccc} X_1 & & & + & \omega_0 & = & m, \\ aX_1 & - & X_2 & + & \omega_1 & = & 0 \end{array} \tag{10.7}$$

Its matrix and vector are

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ a & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{z} = \begin{bmatrix} m \\ 0 \end{bmatrix}.$$

The disturbance vector $\omega = \begin{bmatrix} \omega_1 & \omega_2 \end{bmatrix}$ has the concentration matrix

$$\mathbf{K}_\omega = \begin{bmatrix} \frac{1}{\sigma_\omega^2} & 0 \\ 0 & \frac{1}{\sigma_\omega^2} \end{bmatrix}.$$

Finally, if $\mathbf{X}$ denotes the variable vector with components X_1 and X_2, the system (10.7) can be represented in the standard form (10.2) of Section 10.1:

$$\mathbf{AX} + \omega = \mathbf{z}.$$

This system has full column rank 2 and thus fulfills the requirement of assumption-based reasoning imposed in Section 10.1. We derive for $\mathbf{X}$ a Gaussian potential

$$G_{\mathbf{X}} = \left(\mu_{\mathbf{X}}, \mathbf{K}_{\mathbf{X}}\right)$$

with

$$\begin{aligned} \mathbf{K}_{\mathbf{X}} &= \mathbf{A}^T\mathbf{K}_\omega\mathbf{A} \\ &= \begin{bmatrix} 1 & a \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_\omega^2} & 0 \\ 0 & \frac{1}{\sigma_\omega^2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ a & -1 \end{bmatrix} = \begin{bmatrix} \frac{1+a^2}{\sigma_\omega^2} & -\frac{a}{\sigma_\omega^2} \\ -\frac{a}{\sigma_\omega^2} & \frac{1}{\sigma_\omega^2} \end{bmatrix}. \end{aligned}$$

The mean vector of $\mathbf{X}$ can be obtained from system (10.7) directly,

$$\mu_{\mathbf{X}} = \begin{bmatrix} m \\ am \end{bmatrix}, \tag{10.8}$$

or from the results of Section 10.1. The fact that $\mathbf{A}$ has an inverse simplifies this way.

We now look at this result from another perspective. If we consider the second equation of system (10.7) in the form $X_2 = aX_1 + \omega_1$, we can read from it the conditional Gaussian distribution of X_2 given X_1 as a Gaussian density $f_{X_2|X_1}$ with mean ax_1 and variance σ_ω^2. This is the same conditional density as we would get from the Gaussian density of $\mathbf{X}$ above by conditioning on X_1. Hence, this conditional density of X_2 given X_1 can also be obtained as the quotient of the density $f_\mathbf{X}$ of $\mathbf{X}$ divided by the marginal density f_{X_1} of X_1, which is Gaussian with mean m and variance σ_ω^2. If we compute this quotient, we divide two exponential functions, and, neglecting normalization factors, it is sufficient to look at the exponent, which is the difference of the exponent of $f_\mathbf{X}$ and f_{X_1},

$$\begin{bmatrix} x_1 - m & x_2 - am \end{bmatrix} \begin{bmatrix} \frac{1+a^2}{\sigma_\omega^2} & -\frac{a}{\sigma_\omega^2} \\ -\frac{a}{\sigma_\omega^2} & \frac{1}{\sigma_\omega^2} \end{bmatrix} \begin{bmatrix} x_1 - m \\ x_2 - am \end{bmatrix} - (x_1 - m)\frac{1}{\sigma_\omega^2}(x_1 - m).$$

The second term of this expression can be written as

$$\begin{bmatrix} x_1 - m & x_2 - am \end{bmatrix} \mathbf{K}_{X_1}^{\uparrow\{1,2\}} \begin{bmatrix} x_1 - m \\ x_1 - am \end{bmatrix},$$

where $\mathbf{K}_{X_1} = [1/\sigma_\omega^2]$ is the one-element concentration matrix of X_1. If we introduce this in the exponent of the conditional Gaussian density above, we obtain

$$\begin{bmatrix} x_1 - m & x_2 - am \end{bmatrix} \begin{bmatrix} \frac{a^2}{\sigma_\omega^2} & -\frac{a}{\sigma_\omega^2} \\ -\frac{a}{\sigma_\omega^2} & \frac{1}{\sigma_\omega^2} \end{bmatrix} \begin{bmatrix} x_1 - m \\ x_2 - am \end{bmatrix}.$$

This looks almost like a Gaussian density, except that the concentration matrix $\mathbf{K}_{X_2|X_1}$ in this exponent is only non-negative definite and no more positive definite, since it has only rank 1. We remark that this concentration matrix is essentially the difference of the concentration matrices of $f_\mathbf{X}$ and f_{X_1},

$$\mathbf{K}_{X_2|X_1} = \mathbf{K}_\mathbf{X} - \mathbf{K}_{X_1}^{\uparrow\{1,2\}}.$$

This observation leads us to tentatively introduce a generalized Gaussian potential $(\mu, \mathbf{K}_{X_2|X_1})$ where μ is still given by (10.8). It is thought that this potential represents the conditional Gaussian distribution of X_2 given X_1. This will be tested in a moment.

Remark that from the first equation of (10.7), it follows that X_1 has a Gaussian density f_{X_1} with mean m and variance σ_ω^2. This is the marginal obtained from the density$f_\mathbf{X}$ and it is represented by the ordinary Gaussian potential $(m, 1/\sigma_\omega^2)$. The vector $\mathbf{X}$ then has the Gaussian density $f_\mathbf{X} = f_{X_2|X_1} f_{X_1}$. Let us try whether we get this also by formally applying the combination rule to the Gaussian potential for X_1 and the generalized potential of X_2 given X_1. So, let

$$\left(\mu_\mathbf{X}, \mathbf{K}_\mathbf{X}\right) = \left(\mu_{X_2|X_1}, \mathbf{K}_{X_2|X_1}\right) \otimes \left(m, 1/\sigma_\omega^2\right).$$

According to the combination rule for potentials defined in Instance 1.6 we have

$$\begin{aligned}\mathbf{K_X} &= \mathbf{K}_{X_1|X_2} + \mathbf{K}_{X_1}^{\uparrow\{1,2\}} \\ &= \begin{bmatrix} \frac{a^2}{\sigma_\omega^2} & -\frac{a}{\sigma_\omega^2} \\ -\frac{a}{\sigma_\omega^2} & \frac{1}{\sigma_\omega^2} \end{bmatrix} + \begin{bmatrix} \frac{1}{\sigma_\omega^2} & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1+a^2}{\sigma_\omega^2} & -\frac{a}{\sigma_\omega^2} \\ -\frac{a}{\sigma_\omega^2} & \frac{1}{\sigma_\omega^2} \end{bmatrix}.\end{aligned}$$

This is indeed the concentration matrix of $f_{\mathbf{X}}$. Let us see whether this also works for the mean vector. We first determine

$$\mathbf{K_X^{-1}} = \begin{bmatrix} \sigma_\omega^2 & a\sigma_\omega^2 \\ a\sigma_\omega^2 & (1+a^2)\sigma_\omega^2 \end{bmatrix}.$$

According to the combination rule for potentials defined in Instance 1.6 we have

$$\begin{aligned}\mu_{\mathbf{X}} &= \mathbf{K_X^{-1}}\left(\begin{bmatrix} \frac{a^2}{\sigma_\omega^2} & -\frac{a}{\sigma_\omega^2} \\ -\frac{a}{\sigma_\omega^2} & \frac{1}{\sigma_\omega^2} \end{bmatrix}\begin{bmatrix} m \\ am \end{bmatrix} + \begin{bmatrix} \frac{1}{\sigma_\omega^2} & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} m \\ 0 \end{bmatrix}\right) \\ &= \mathbf{K_X^{-1}}\begin{bmatrix} \frac{1+a^2}{\sigma_\omega^2} & -\frac{a}{\sigma_\omega^2} \\ -\frac{a}{\sigma_\omega^2} & \frac{1}{\sigma_\omega^2} \end{bmatrix}\begin{bmatrix} m \\ am \end{bmatrix} = \begin{bmatrix} m \\ am \end{bmatrix}.\end{aligned}$$

This is indeed the mean vector of $f_{\mathbf{X}}$. It seems that the combination operation for ordinary potentials works formally also with generalized Gaussian potentials.

We shall see later that it is even more convenient to define the generalized Gaussian potentials as $(\mathbf{K}\mu, \mathbf{K})$. Indeed, we remark that the generalized Gaussian potential of X_2 given X_1 can also be determined directly from the second equation of system (10.8) which defines the conditional density. Let $\mathbf{A} = \begin{bmatrix} a & -1 \end{bmatrix}$ be the matrix of this equation and $\mathbf{K}_\omega = [1/\sigma_\omega^2]$ its concentration matrix. We consider the exponent of the exponential function defining the conditional density of X_2 given X_1:

$$\begin{aligned}& (x_2 - ax_1)\mathbf{K}_\omega(x_2 - ax_1) \\ = \; & (-a(x_1 - m) + (x_2 - am))\,\mathbf{K}_\omega\,(-a(x_1 - m) + (x_2 - am)) \\ = \; & \begin{bmatrix} x_1 - m & x_2 - am \end{bmatrix} \mathbf{A}^T\mathbf{K}_\omega\mathbf{A} \begin{bmatrix} x_1 - m \\ x_2 - am \end{bmatrix}.\end{aligned}$$

From this we see that indeed

$$\mathbf{K}_{X_2|X_1} = \mathbf{A}^T\mathbf{K}_\omega\mathbf{A} = \begin{bmatrix} \frac{a^2}{\sigma_\omega^2} & -\frac{a}{\sigma_\omega^2} \\ -\frac{a}{\sigma_\omega^2} & \frac{1}{\sigma_\omega^2} \end{bmatrix}.$$

Further, we obtain

$$\begin{aligned}\mathbf{K}_{X_2|X_1}\mu_{X_2|X_1} &= \mathbf{A}^T\mathbf{K}_\omega\mathbf{A}\mu_{X_2|X_1} = \begin{bmatrix} a \\ -1 \end{bmatrix}\mathbf{K}_\omega\begin{bmatrix} a & -1 \end{bmatrix}\begin{bmatrix} m \\ am - m \end{bmatrix} \\ &= \begin{bmatrix} a \\ -1 \end{bmatrix}\mathbf{K}_\omega m = \mathbf{A}^T\mathbf{K}_\omega m.\end{aligned}$$

So, indeed, the generalized Gaussian potential is completely determined by the elements of the defining equation only. This approach can be developed in general, and it will be done so in Section 10.3. For the time being, we claim that the same procedure can be applied to the equations associated with the nodes of the join tree in Figure 10.3. We claim, and show later, that the extended Gaussian potentials form again a valuation algebra, extending the algebra of ordinary Gaussian potentials. The Gaussian potential of the total system (10.6) can therefore be obtained by combining the generalized Gaussian potentials. The fiducial densities of the state variables is then obtained by projecting this combination of potentials to each variable. Thus, the inference problem stated at the beginning of this section can be solved by local computation with very simple data rather than by treating system (10.6) as a whole. This example is a simple instance of a general *filtering, smoothing and projection problem* associated with time-discrete Gaussian dynamic systems. It will be examined in its full generality in Section 10.5. Moreover, can also be seen as an instance of a Gaussian Bayesian network. This perspective will be discussed in Section 10.6.

10.3 GAUSSIAN INFORMATION AND GAUSSIAN POTENTIALS

In this section, we take up the approach sketched in the previous Section in a more systematic and general manner by considering arbitrary underdetermined linear systems with Gaussian disturbances.

10.3.1 Assumption-Based Reasoning

Let us assume a linear system with Gaussian disturbances

$$\mathbf{A}\mathbf{X} + \omega \quad = \quad \mathbf{z}. \tag{10.9}$$

which, in contrast to Section 10.1, is underdetermined. Here, $\mathbf{X}$ is an n-vector, $\mathbf{A}$ an $m \times n$ matrix with $m \leq n$ and $\mathbf{z}$ an m-vector. The stochastic m-vector ω has a Gaussian density with mean vector $\mathbf{0}$ and concentration matrix $\mathbf{K}_\omega$. Further, we assume for the time being that $\mathbf{A}$ has full row rank m. More general cases will be considered later.

We first give a geometric picture of this situation based on the approach of assumption-based reasoning. Note that, under the above assumptions, system (10.9) has a solution for any vector $\omega \in \mathbb{R}^m$. So, in contrast to the situation in Section 10.1 with overdetermined systems, no ω has to be excluded, and for any $\omega \in \mathbb{R}^m$ the solution space is an affine space $\mathcal{L} + y(\omega)$ in $\mathbb{R}^n$. Here, $\mathcal{L}$ is the kernel (see Section 7.2) of the system and $y(\omega)$ any particular solution of the system with a fixed ω. It is evident that the different assumptions lead to disjoint, parallel, affine spaces. Therefore, we sometimes call $\Gamma(\omega) = \mathcal{L} + y(\omega)$ the *focal space* associated with ω. It is also clear that if ω varies over $\mathbb{R}^m$, then their focal spaces cover the whole space $\mathbb{R}^n$. The Gaussian probability distribution of ω induces also an associated Gaussian probability of the focal spaces. Such a structure is also called a *hint*: Each assumption ω out of a set of possible assumptions determines a set $\Gamma(\omega)$ of possible values for the unknown parameter x, and the assumptions are not all equally likely but have a specified probability distribution. We refer to (Kohlas & Monney, 1995) for a general theory of hints and to (Kohlas & Monney, 2008) for the connection of hints with statistical inference. If $\mathbf{B}$ is a regular $m \times m$ matrix, then the transformed system $\mathbf{BAX} + \tilde{\omega} = \tilde{z}$ with $\tilde{\omega} = \mathbf{B}\omega$ and $\tilde{z} = \mathbf{Bz}$ has the same family of parallel focal spaces $\Gamma(\tilde{\omega}) = \Gamma(\mathbf{B}\omega)$ as the original system and also the same Gaussian density over the focal spaces. Such hints, which differ from each other only by a regular transformation, are therefore called *equivalent*. If $m = n$, we have an important special case: the focal spaces $\Gamma(\omega)$ then reduce to single points, i.e. the unique solutions $y(\omega) = \mathbf{A}^{-1}(\mathbf{z} - \omega)$ of the system for a fixed assumption ω. We refer to the system (10.7) and its analysis in the previous Section 10.2 as an example for such a situation. It induces an ordinary Gaussian density with mean vector $\mathbf{A}^{-1}\mathbf{z}$ and concentration matrix $\mathbf{A}^T\mathbf{K}_\omega\mathbf{A}$. This case is also covered in Section 10.1, and the Gaussian density corresponds to the Gaussian potential $(\mathbf{A}^{-1}\mathbf{z}, \mathbf{A}^T\mathbf{K}_\omega\mathbf{A})$ associated with the system. This can indeed be verified from (10.4) in Section 10.1: for the concentration matrix we have the same expression. The mean vector in (10.4) can be developed in this particular case as follows:

$$(\mathbf{A}^T\mathbf{K}_\omega\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}_\omega\mathbf{z} \;=\; \mathbf{A}^{-1}\mathbf{K}_\omega^{-1}\mathbf{A}^{T\;-1}\mathbf{A}^T\mathbf{K}_\omega\mathbf{z} \;=\; \mathbf{A}^{-1}\mathbf{z}.$$

Hints with one-point focal spaces result from assumption-based reasoning according to Section 10.1 from any overdetermined system.

In the following, we usually assume m strictly smaller than n. We show that in this case Gaussian conditional densities can be associated with linear Gaussian systems (10.9) and thus also with generalized Gaussian potentials as exemplified in the previous section. Since $\mathbf{A}$ has full row rank m, there is at least one $m \times m$ regular submatrix of $\mathbf{A}$. By renumbering columns we may always assume that the first m columns of $\mathbf{A}$ are linearly independent. We decompose $\mathbf{A}$ into the regular submatrix $\mathbf{A}_1$ composed of the first m columns and the remaining $m \times (n - m)$ submatrix $\mathbf{A}_2$, that is $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2]$. Let us further decompose the vector of variables $\mathbf{X}$ accordingly into the m-vector $\mathbf{X}_1$ and the $(n - m)$-vector $\mathbf{X}_2$. Then, the system (10.9) can be written as

$$\mathbf{A}_1\mathbf{X}_1 + \mathbf{A}_2\mathbf{X}_2 + \omega \;=\; \mathbf{z}.$$

Applying the regular transformation $\mathbf{B} = \mathbf{A}_1^{-1}$ yields the equivalent system

$$\mathbf{X}_1 + \mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{X}_2 + \tilde{\omega} \;=\; \tilde{\mathbf{z}},$$

where $\tilde{\omega} = \mathbf{A}_1^{-1}\omega$ and $\tilde{\mathbf{z}} = \mathbf{A}_1^{-1}\mathbf{z}$. The stochastic vector $\tilde{\omega}$ still has a Gaussian density with mean $\mathbf{0}$ and concentration matrix $\mathbf{K}_{\tilde{\omega}} = \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1$ as shown in the appendix of this chapter. Using the transformed system, we can express $\mathbf{X}_1$ by $\mathbf{X}_2$,

$$\mathbf{X}_1 \;=\; -\mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{X}_2 + (\tilde{\mathbf{z}} - \tilde{\omega}).$$

If we fix $\mathbf{X}_2$ to some value $\mathbf{x}_2$, we can directly read the Gaussian conditional density of $\mathbf{X}_1$ given $\mathbf{X}_2 = \mathbf{x}_2$ as a Gaussian density with mean-vector $\mu_{\mathbf{X}_1|\mathbf{X}_2}$ and concentration matrix $\mathbf{K}_{\mathbf{X}_1|\mathbf{X}_2}$ obtained as follows:

$$\begin{aligned} \mu_{\mathbf{X}_1|\mathbf{X}_2}(\mathbf{x}_2) &= \mathbf{A}_1^{-1}\mathbf{z} - \mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{x}_2 \\ \mathbf{K}_{\mathbf{X}_1|\mathbf{X}_2} &= \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1. \end{aligned}$$

This conditional Gaussian density of $\mathbf{X}_1$ given $\mathbf{X}_2$ has the exponent

$$(\mathbf{x}_1 + \mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{x}_2 - \mathbf{A}_1^{-1}\mathbf{z})^T\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1(\mathbf{x}_1 + \mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{x}_2 - \mathbf{A}_1^{-1}\mathbf{z}). \tag{10.10}$$

Next, we are going to associate a generalized Gaussian density to this conditional Gaussian density in a way similar to that of the previous Section 10.2: To do so, we introduce two undetermined vectors μ_1 and μ_2 which are linked by the relation

$$\mu_1 = -\mathbf{A}_1^{-1}\mathbf{A}_2\mu_2 + \mathbf{A}_1^{-1}\mathbf{z}.$$

This also means that

$$\mathbf{A}_1\mu_1 + \mathbf{A}_2\mu_2 = \mathbf{z}. \tag{10.11}$$

The vectors μ_1 and μ_2 are only used temporarily and will disappear in the end. We introduce this expression into the exponent of equation (10.10):

$$\begin{aligned} & (\mathbf{x}_1 + \mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{x}_2 - \mathbf{A}_1^{-1}\mathbf{z})^T\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1(\mathbf{x}_1 + \mathbf{A}_1^{-1}\mathbf{A}_2\mathbf{x}_2 - \mathbf{A}_1^{-1}\mathbf{z}) \\ = \ & ((\mathbf{x}_1 - \mu_1) + \mathbf{A}_1^{-1}\mathbf{A}_2(\mathbf{x}_2 - \mu_2))^T\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1((\mathbf{x}_1 - \mu_1) + \mathbf{A}_1^{-1}\mathbf{A}_2(\mathbf{x}_2 - \mu_2)) \\ = \ & \begin{bmatrix} \mathbf{x}_1 - \mu_1 & \mathbf{x}_2 - \mu_2 \end{bmatrix} \begin{bmatrix} \mathbf{I}_m \\ \mathbf{A}_2^T\mathbf{A}_1^{-1\,T} \end{bmatrix} \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1 \\ & \begin{bmatrix} \mathbf{I}_m & \mathbf{A}_1^{-1}\mathbf{A}_2^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \mu_1 & \mathbf{x}_2 - \mu_2 \end{bmatrix}. \end{aligned}$$

This looks like the exponent of a Gaussian density with concentration matrix

$$\begin{aligned} \mathbf{K} &= \begin{bmatrix} \mathbf{I}_m \\ \mathbf{A}_2^T\mathbf{A}_1^{-1\,T} \end{bmatrix} \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1 \begin{bmatrix} \mathbf{I}_m & \mathbf{A}_1^{-1}\mathbf{A}_2^T \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1 & \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_2 \\ \mathbf{A}_2^T\mathbf{K}_\omega\mathbf{A}_1 & \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_2 \end{bmatrix} = \mathbf{A}^T\mathbf{K}_\omega\mathbf{A}. \end{aligned}$$

Since this matrix is only non-negative definite, we call it a *pseudo-concentration matrix*. The conditional Gaussian density above has an undetermined mean vector μ with components μ_1 and μ_2, but this mean vector disappears if we consider

$$\begin{aligned} \nu &= \mathbf{K}\mu = \mathbf{A}^T\mathbf{K}_\omega\mathbf{A}\mu = \mathbf{A}^T\mathbf{K}_\omega \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \\ &= \mathbf{A}^T\mathbf{K}_\omega(\mathbf{A}_1\mu_1 + \mathbf{A}_2\mu_2) = \mathbf{A}^T\mathbf{K}_\omega\mathbf{z}. \end{aligned}$$

The last equality follows from (10.11). To sum it up, we have associated a *generalized Gaussian potential*

$$(\mathbf{A}^T\mathbf{K}_\omega\mathbf{z},\ \mathbf{A}^T\mathbf{K}_\omega\mathbf{A}) \tag{10.12}$$

with the original linear Gaussian system (10.9). We have done this via a conditional density extracted from the system (10.11). Note that there may be several different conditional Gaussian densities that can be extracted from (10.11), depending on which set of m variables we choose to solve for. But the final generalized conditional Gaussian potential is independent of this choice. It only depends on the *Gaussian information* expressed by the system (10.9). Let us verify this assertion by considering an $m \times m$ regular matrix $\mathbf{B}$ and the corresponding transformed system $\mathbf{BAX} + \tilde{\omega} = \tilde{\mathbf{z}}$, where $\mathbf{K}_{\tilde{\omega}} = \mathbf{B}^{T\ -1}\mathbf{K}_{\omega}\mathbf{B}^{-1}$. The generalized potential associated with this transformed system has concentration matrix

$$\tilde{\mathbf{K}} = (\mathbf{BA})^T\mathbf{K}_{\tilde{\omega}}\mathbf{BA} = \mathbf{A}^T\mathbf{B}^T\mathbf{B}^{T\ -1}\mathbf{K}_{\omega}\mathbf{B}^{-1}\mathbf{BA} = \mathbf{A}^T\mathbf{K}_{\omega}\mathbf{A}.$$

Similarly, we obtain for the first part of the generalized potential

$$\tilde{\nu} = (\mathbf{BA})^T\mathbf{K}_{\tilde{\omega}}\mathbf{Bz} = \mathbf{A}^T\mathbf{B}^T\mathbf{B}^{T\ -1}\mathbf{K}_{\omega}\mathbf{B}^{-1}\mathbf{Bz} = \mathbf{A}^T\mathbf{K}_{\omega}\mathbf{z}.$$

So, the transformed equivalent system has the same potential as the original system. In other words: equivalent systems have identical generalized Gaussian potentials.

10.3.2 General Gaussian Information

Linear Gaussian information can be represented in a compact form by generalized Gaussian potentials. How are natural operations with linear Gaussian information reflected by Gaussian potentials? Here, we look at the operations of combination. For subsets s and t of some set of indices of variables let $\mathbf{X}$ be an $(s \cup t)$-vector and consider two pieces of linear Gaussian information, one referring to the group of variables in s

$$\mathbf{A}_1\mathbf{X}^{\downarrow s} + \omega_1 = \mathbf{z}_1 \tag{10.13}$$

where $\mathbf{A}_1$ is an $m_1 \times s$ matrix and ω_1 and $\mathbf{z}_1$ are m_1-vectors. ω_1 has a Gaussian density with mean vector $\mathbf{0}$ and concentration matrix $\mathbf{K}_{\omega_1}$. The second piece of linear Gaussian information refers to the set t of variables,

$$\mathbf{A}_2\mathbf{X}^{\downarrow t} + \omega_2 = \mathbf{z}_2, \tag{10.14}$$

where $\mathbf{A}_2$ is an $m_2 \times t$ matrix and ω_2 and $\mathbf{z}_2$ are m_2-vectors. ω_2 has a Gaussian density with mean vector $\mathbf{0}$ and concentration matrix $\mathbf{K}_{\omega_2}$. Together, the two systems provide linear Gaussian information for the total vector $\mathbf{X}$,

$$\mathbf{AX} + \omega = \mathbf{z}. \tag{10.15}$$

The $(m_1 + m_2)$-vector ω has a Gaussian density with mean vector $\mathbf{0}$. Since we assume that the vectors ω_1 and ω_2 are stochastically independent, its concentration matrix is

$$\mathbf{K}_{\omega} = \begin{bmatrix} \mathbf{K}_{\omega_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{\omega_2} \end{bmatrix}.$$

The $(m_1 + m_2)$-vector $\mathbf{z}$ is composed of the two vectors $\mathbf{z}_1$ and $\mathbf{z}_2$. Finally, the $(m_1 + m_2) \times (s \cup t)$ matrix $\mathbf{A}$ is composed of the matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow s\cap t} & \mathbf{A}_2^{\downarrow t-s} \end{bmatrix}. \tag{10.16}$$

Now, even if the two matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ have full row rank, this is no more guaranteed for the matrix $\mathbf{A}$. Therefore, we need to extend our assumption-based analysis to the case of a general linear Gaussian system

$$\mathbf{A}\mathbf{X} + \omega = \mathbf{z}, \tag{10.17}$$

before we can return to the combined system above. Here, we assume $\mathbf{A}$ to be an $m \times n$ matrix of rank $k \leq m, n$. The number of equations m may be less than, equal to or greater than the number of variables n. So, the following discussion generalizes both particular cases studied so far. The vectors ω and $\mathbf{z}$ have both dimension m, and ω has a Gaussian density with mean $\mathbf{0}$ and $\mathbf{K}_\omega$ as concentration matrix. As usual, we consider the system to be derived from a functional model where the linear, stochastic function $\mathbf{A}\mathbf{x} + \omega$ determines the outcome or the observation $\mathbf{z}$. Then, by the usual approach of assumption-based reasoning, we first remark that, given the observation $\mathbf{z}$, only assumptions ω in the set

$$v_{\mathbf{z}} = \{\omega \in \mathbb{R}^m : \omega = \mathbf{z} - \mathbf{A}\mathbf{x},\ \mathbf{x} \in \mathbb{R}^n\}$$

may have generated the observation $\mathbf{z}$. The set $v_{\mathbf{z}}$ is an affine space in the linear space $\mathbb{R}^m$ and in general different from $\mathbb{R}^m$. These are the admissible assumptions. Then, each admissible assumption $\omega \in v_{\mathbf{z}}$ determines the set of possible parameters $\mathbf{x}$ by

$$\Gamma(\omega) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{z} - \omega\}.$$

As before, these are parallel affine subspaces of $\mathbb{R}^n$ which cover $\mathbb{R}^n$ completely. It is also possible that the affine spaces are single points. Different admissible assumptions ω have disjoint focal spaces $\Gamma(\omega)$.

Next, we need to determine the conditional density of ω, given that ω must be in the affine space $v_{\mathbf{z}}$. As in Section 10.1, we transform the system (10.17) in a convenient way which allows us to easily derive the conditional density. For this purpose, we select an $m \times k$ matrix $\mathbf{B}_1$ whose columns form a basis of the column space of matrix $\mathbf{A}$. For example, we could take k linearly independent columns of $\mathbf{A}$, which is always possible since $\mathbf{A}$ has rank k. Consequently, the matrix $\mathbf{B}_1$ has rank k. Then, there is a $k \times m$ matrix $\mathbf{\Lambda}$ such that $\mathbf{A} = \mathbf{B}_1\mathbf{\Lambda}$. We complete $\mathbf{B}_1$ with an $m \times (m - k)$ matrix $\mathbf{B}_2$ such that $\mathbf{B} = [\mathbf{B}_1\ \mathbf{B}_2]$ is a regular $m \times m$ matrix and define $\mathbf{T} = \mathbf{B}^{-1}$. We decompose

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix}$$

such that $\mathbf{T}_1$ is a $k \times m$ and $\mathbf{T}_2$ an $(m-k) \times m$ matrix. Then, we see that

$$\mathbf{TA} = \begin{bmatrix} \mathbf{T}_1\mathbf{A} \\ \mathbf{T}_2\mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_1\mathbf{A} \\ \mathbf{T}_2\mathbf{B}_1\mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_1\mathbf{A} \\ \mathbf{0} \end{bmatrix}$$

since $\mathbf{TB} = \mathbf{I}$ implies that $\mathbf{T}_2\mathbf{B}_1 = \mathbf{0}$. Now, the matrix $\mathbf{TA}$ has rank k because $\mathbf{T}$ is regular. This implies that the $k \times m$ matrix $\mathbf{T}_1\mathbf{A}$ has full row rank k. The focal spaces for ω can now be described by

$$\Gamma(\omega) = \{\mathbf{x} : \mathbf{T}_1\mathbf{A}\mathbf{x} + \mathbf{T}_1\omega = \mathbf{T}_1\mathbf{z}\}$$

and the admissible assumptions by

$$v_{\mathbf{z}} = \{\omega : \mathbf{T}_2\omega = \mathbf{T}_2\mathbf{z}\}.$$

This corresponds to the transformed system

$$\begin{aligned} \mathbf{T}_1\mathbf{A}\mathbf{X} + \tilde{\omega}_1 &= \tilde{\mathbf{z}}_1, \\ \tilde{\omega}_2 &= \tilde{\mathbf{z}}_2, \end{aligned} \tag{10.18}$$

where $\tilde{\omega}_1 = \mathbf{T}_1\omega$ and $\tilde{\mathbf{z}}_1 = \mathbf{T}_1\mathbf{z}$. The transformed vector $\tilde{\omega} = \mathbf{T}\omega$ is composed of the components $\tilde{\omega}_1$ and $\tilde{\omega}_2$. It is still Gaussian with mean $\mathbf{0}$ and concentration matrix

$$\mathbf{K}_{\tilde{\omega}} = \mathbf{B}^T\mathbf{K}_\omega\mathbf{B} = \begin{bmatrix} \mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1 & \mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2 \\ \mathbf{B}_2^T\mathbf{K}_\omega\mathbf{B}_1 & \mathbf{B}_2^T\mathbf{K}_\omega\mathbf{B}_2 \end{bmatrix}. \tag{10.19}$$

From the system (10.18) it is now easy to determine the conditional Gaussian density of $\tilde{\omega}_1$ given $\tilde{\omega}_2 = \tilde{\mathbf{z}}_2$. According to equation (10.19) it has concentration matrix $\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1$ and mean vector (see Appendix J.2)

$$-(\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1)^{-1}(\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2)\mathbf{z}.$$

Finally, we obtain from system (10.18) the following linear Gaussian system

$$\mathbf{T}_1\mathbf{A}\mathbf{X} + \tilde{\tilde{\omega}} = \mathbf{T}_1\mathbf{z} + (\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1)^{-1}(\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2)\mathbf{z}, \tag{10.20}$$

where

$$\tilde{\tilde{\omega}} = \tilde{\omega}_1 - (\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1)^{-1}(\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2)\mathbf{z}. \tag{10.21}$$

This is now a linear Gaussian system with full row rank k. Therefore, we can apply the results from Section 10.3.1 to derive its generalized potential.

We now argue that this is also the generalized Gaussian potential associated with the original system. In fact, the selection of the basis $\mathbf{B}_1$ was arbitrary, and the resulting system depends on this choice, but the associated generalized Gaussian potential does not, as we show next. A linear Gaussian system like (10.17) is fully determined by the three elements $\mathbf{A}$, $\mathbf{z}$ and $\mathbf{K}_\omega$. Therefore, we represent it by the triplet $(\mathbf{A}, \mathbf{z}, \mathbf{K}_\omega)$ and define the mapping

$$e(\mathbf{A}, \mathbf{z}, \mathbf{K}_\omega) \quad = \quad (\mathbf{A}^T\mathbf{K}_\omega\mathbf{z}, \mathbf{A}^T\mathbf{K}_\omega\mathbf{A}).$$

The image of this mapping looks like a generalized Gaussian potential. In fact, it is one, but it remains to verify that the matrix $\mathbf{A}^T\mathbf{K}_\omega\mathbf{A}$ is non-negative definite. This however is a consequence of the following theorem:

Theorem 10.1 *Let $l = (\mathbf{A}, \mathbf{z}, \mathbf{K}_\omega)$ describe a linear Gaussian system (10.17) with an $m \times n$ matrix $\mathbf{A}$ of rank k, and*

$$h \; : \; \mathbf{T}_1\mathbf{X} + \tilde{\omega} \quad = \quad \mathbf{T}_1\mathbf{z} + (\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1)^{-1}(\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2)\mathbf{z}$$

a system with full row rank derived from it as in (10.20) and (10.21). Then, $e(l) = e(h)$.

Proof: We define

$$\begin{aligned}
\tilde{\mathbf{A}} &= \mathbf{T}_1\mathbf{A}, \\
\tilde{\mathbf{z}} &= \mathbf{T}_1\mathbf{z} + (\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1)^{-1}(\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2)\mathbf{z}, \\
\tilde{\mathbf{K}}_{\tilde{\omega}} &= \mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1.
\end{aligned}$$

It follows that $e(h) = (\tilde{\mathbf{A}}^T\tilde{\mathbf{K}}_{\tilde{\omega}}\tilde{\mathbf{z}}, \tilde{\mathbf{A}}^T\tilde{\mathbf{K}}_{\tilde{\omega}}\tilde{\mathbf{A}})$. But then, if $\mathbf{B}$ is the matrix used in transforming the system $(\mathbf{A}, \mathbf{z}, \mathbf{K}_\omega)$ and $\mathbf{T} = \mathbf{B}^{-1}$

$$\begin{aligned}
\mathbf{A}^T\mathbf{K}_\omega\mathbf{A} &= \mathbf{A}^T\mathbf{T}^T\mathbf{B}^T\mathbf{K}_\omega\mathbf{B}\mathbf{T}\mathbf{A} \;=\; (\mathbf{T}\mathbf{A})^T\mathbf{B}^T\mathbf{K}_\omega\mathbf{B}(\mathbf{T}\mathbf{A}) \\
&= \begin{bmatrix} (\mathbf{T}_1\mathbf{A})^T & \mathbf{0}_{n,m-k} \end{bmatrix} \begin{bmatrix} \mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1 & \mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2 \\ \mathbf{B}_2^T\mathbf{K}_\omega\mathbf{B}_1 & \mathbf{B}_2^T\mathbf{K}_\omega\mathbf{B}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T}_1\mathbf{A} \\ \mathbf{0}_{m-k,n} \end{bmatrix} \\
&= (\mathbf{T}_1\mathbf{A})^T\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1\mathbf{T}_1\mathbf{A} \;=\; \tilde{\mathbf{A}}^T\tilde{\mathbf{K}}_{\tilde{\omega}}\tilde{\mathbf{A}}.
\end{aligned}$$

Further,

$$\begin{aligned}
\mathbf{A}^T\mathbf{K}_\omega\mathbf{z} &= (\mathbf{T}\mathbf{A})^T\mathbf{B}^T\mathbf{K}_\omega\mathbf{B}\mathbf{T}\mathbf{z} \\
&= \begin{bmatrix} (\mathbf{T}_1\mathbf{A})^T & \mathbf{0}_{n,m-k} \end{bmatrix} \begin{bmatrix} \mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1 & \mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2 \\ \mathbf{B}_2^T\mathbf{K}_\omega\mathbf{B}_1 & \mathbf{B}_2^T\mathbf{K}_\omega\mathbf{B}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T}_1\mathbf{z} \\ \mathbf{T}_2\mathbf{z} \end{bmatrix} \\
&= (\mathbf{T}_1\mathbf{A})^T \left(\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1\mathbf{T}_1\mathbf{z} + \mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2\mathbf{T}_2\mathbf{z}\right) \\
&= (\mathbf{T}_1\mathbf{A})^T(\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1)\left(\mathbf{T}_1\mathbf{z} + (\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_1)^{-1}\mathbf{B}_1^T\mathbf{K}_\omega\mathbf{B}_2\mathbf{T}_2\mathbf{z}\right) \\
&= \tilde{\mathbf{A}}^T\tilde{\mathbf{K}}_{\tilde{\omega}}\tilde{\mathbf{z}}.
\end{aligned}$$

So, indeed the generalized Gaussian potential of h equals $e(l)$. ■

Since $e(h)$ is indeed a generalized Gaussian potential, it follows that $e(l)$ is one too. Thus, by the mapping $e(l)$, we associate a uniquely determined generalized

Gaussian potential to each linear Gaussian information of the form (10.17) without any restriction on the rank of the system. We may say that two linear Gaussian systems l and l' are equivalent if they induce equivalent systems h and h' with full row rank. Since we have seen that equivalent systems h and h' generate the same generalized Gaussian potential, this also holds for equivalent Gaussian systems.

10.3.3 Combination of Gaussian Information

We are now able to compute the generalized Gaussian potential $(\nu, \mathbf{K})$ of the combined system (10.15) with the combined matrix (10.16). Note first that the generalized Gaussian potential $(\nu_1, \mathbf{K}_1)$ corresponding to the first subsystem (10.13) is given by

$$\mathbf{K}_1 = \mathbf{A}_1^T \mathbf{K}_{\omega_1} \mathbf{A}_1 = \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s-t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s\cap t} \\ \mathbf{A}_1^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s\cap t} \end{bmatrix} \tag{10.22}$$

and

$$\nu_1 = \mathbf{A}_1^T \mathbf{K}_{\omega_1} \mathbf{z}_1 = \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T} \mathbf{K}_{\omega_1} \mathbf{z}_1 \\ \mathbf{A}_1^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_1} \mathbf{z}_1 \end{bmatrix}. \tag{10.23}$$

For the second subsystem (10.14), we have similar results, essentially replacing $\mathbf{A}_1$ by $\mathbf{A}_2$, $\mathbf{K}_{\omega_1}$ by $\mathbf{K}_{\omega_2}$ and $\mathbf{z}_1$ by $\mathbf{z}_2$. Now, the second part of the potential $(\nu, \mathbf{K})$ of the combined system (10.15) is given by

$$\begin{aligned}
\mathbf{K} &= \mathbf{A}^T \mathbf{K}_\omega \mathbf{A} \\
&= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t\;T} & \mathbf{A}_2^{\downarrow s\cap t\;T} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s\;T} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{\omega_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{\omega_2} \end{bmatrix} \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow s\cap t} & \mathbf{A}_2^{\downarrow t-s} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s-t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s-t} & \begin{matrix} \mathbf{A}_1^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s\cap t} \\ + \mathbf{A}_2^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_2} \mathbf{A}_2^{\downarrow s\cap t} \end{matrix} & \mathbf{A}_2^{\downarrow s\cap t} \mathbf{K}_{\omega_2} \mathbf{A}_2^{\downarrow t-s} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s\;T} \mathbf{K}_{\omega_2} \mathbf{A}_2^{\downarrow t-s} & \mathbf{A}_2^{\downarrow t-s\;T} \mathbf{K}_{\omega_2} \mathbf{A}_2^{\downarrow t-s} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s-t} & \mathbf{A}_1^{\downarrow s-t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s\cap t} & \mathbf{A}_1^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_1} \mathbf{A}_1^{\downarrow s\cap t} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\
&+ \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow s\cap t\;T} \mathbf{K}_{\omega_2} \mathbf{A}_2^{\downarrow s\cap t} & \mathbf{A}_2^{\downarrow s\cap t} \mathbf{K}_{\omega_2} \mathbf{A}_2^{\downarrow t-s} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s\;T} \mathbf{K}_{\omega_2} \mathbf{A}_2^{\downarrow t\cap s} & \mathbf{A}_2^{\downarrow t-s\;T} \mathbf{K}_{\omega_2} \mathbf{A}_2^{\downarrow t-s} \end{bmatrix} = \mathbf{K}_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}.
\end{aligned}$$

The last equality can be concluded from equation (10.22).

Next, we compute the first part of the potential $(\nu, \mathbf{K})$ of the combined system

$$
\begin{aligned}
\nu &= \mathbf{A}^T \mathbf{K}_\omega \mathbf{z} \\
&= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\ T} & \mathbf{0} \\ \mathbf{A}_1^{\downarrow s\cap t\ T} & \mathbf{A}_2^{\downarrow s\cap t\ T} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow t-s\ T} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{\omega_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{\omega_2} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\ T} \mathbf{K}_{\omega_1} \mathbf{z}_1 \\ \mathbf{A}_1^{\downarrow s\cap t\ T} \mathbf{K}_{\omega_1} \mathbf{z}_1 + \mathbf{A}_2^{\downarrow s\cap t\ T} \mathbf{K}_{\omega_2} \mathbf{z}_2 \\ \mathbf{A}_2^{\downarrow t-s\ T} \mathbf{K}_{\omega_2} \mathbf{z}_2 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{A}_1^{\downarrow s-t\ T} \mathbf{K}_{\omega_1} \mathbf{z}_1 \\ \mathbf{A}_1^{\downarrow s\cap t\ T} \mathbf{K}_{\omega_1} \mathbf{z}_1 \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{A}_2^{\downarrow s\cap t\ T} \mathbf{K}_{\omega_2} \mathbf{z}_2 \\ \mathbf{A}_2^{\downarrow t-s\ T} \mathbf{K}_{\omega_2} \mathbf{z}_2 \end{bmatrix} = \nu_1^{\uparrow s\cup t} + \nu_2^{\uparrow s\cup t}.
\end{aligned}
$$

This shows how generalized Gaussian potentials can be combined in order to reflect the combination of the linear Gaussian information. Namely, if the potential $(\nu_1, \mathbf{K}_1)$ has domain s and $(\nu_2, \mathbf{K}_2)$ has domain t,

$$
(\nu_1, \mathbf{K}_1) \otimes (\nu_2, \mathbf{K}_2) = \left(\nu_1^{\uparrow s\cup t} + \nu_2^{\uparrow s\cup t}, \mathbf{K}_1^{\uparrow s\cup t} + \mathbf{K}_2^{\uparrow s\cup t}\right).
$$

10.3.4 Projection of Gaussian Information

After the operation of combination, we examine the operation of information extraction or projection. We start with a linear Gaussian system

$$
l : \mathbf{A}\mathbf{X} + \omega = \mathbf{z} \qquad (10.24)
$$

where $\mathbf{X}$ is an s-vector of variables, $\mathbf{A}$ an $m \times s$ matrix, representing m equations over the variables in s, $\mathbf{z}$ an m-vector and ω a stochastic m-vector having Gaussian density with mean vector $\mathbf{0}$ and concentration matrix $\mathbf{K}_\omega$. We want to extract from this Gaussian information the part of information relating to the variables $\mathbf{X}^{\downarrow t}$ for some subset $t \subseteq s$. To this end, we decompose the linear system according to the variables $\mathbf{X}^{\downarrow s-t}$ and $\mathbf{X}^{\downarrow t}$,

$$
\mathbf{A}_1 \mathbf{X}^{\downarrow s-t} + \mathbf{A}_2 \mathbf{X}^{\downarrow t} + \omega = \mathbf{z}. \qquad (10.25)
$$

Here, $\mathbf{A}_1$ is the $m \times (s - t)$ matrix containing the columns of $\mathbf{A}$ which correspond to the variables in $s - t$, and $\mathbf{A}_2$ the $m \times t$ matrix containing the columns of $\mathbf{A}$ which correspond to the variables in t. Extracting the information bearing on the variables in t means to eliminate the variables in $s - t$ from the linear system above. This operation has already been examined in Chapter 9. But here, we additionally look at the stochastic component ω in the system. To simplify the discussion, we first assume that $\mathbf{A}_1$ has full column rank $|s - t| = k$. This implies that there are k linearly independent rows in the matrix $\mathbf{A}_1$. By renumbering rows, we may assume that the first k rows are linearly independent such that the matrix $\mathbf{A}_1$ decomposes into a regular $k \times k$ submatrix $\mathbf{A}_{1,1}$ composed of the first k rows and the $(m-k) \times k$ submatrix $\mathbf{A}_{2,1}$ containing the remaining rows. If we decompose system (10.25) accordingly into the system of the first k equations and the $m-k$ equations, we obtain

$$\begin{array}{rcl} \mathbf{A}_{1,1}\mathbf{X}^{\downarrow s-t} + \mathbf{A}_{1,2}\mathbf{X}^{\downarrow t} + \omega_1 & = & \mathbf{z}_1, \\ \mathbf{A}_{2,1}\mathbf{X}^{\downarrow s-t} + \mathbf{A}_{2,2}\mathbf{X}^{\downarrow t} + \omega_2 & = & \mathbf{z}_2. \end{array}$$

Here, $\mathbf{A}_{1,2}$ is a $k \times (s \cap t)$ matrix and $\mathbf{A}_{2,2}$ an $(m-k) \times (s \cap t)$ matrix. The vectors ω_1 and $\mathbf{z}_1$ have dimensions k, whereas ω_2 and $\mathbf{z}_2$ have dimension $m-k$. Since $\mathbf{A}_{1,1}$ is regular, we can solve the first part for $\mathbf{X}^{\downarrow s-t}$ and replace these variables in the second part of the system,

$$\begin{array}{rcl} \mathbf{X}^{\downarrow s-t} + \mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2}\mathbf{X}^{\downarrow t} + \mathbf{A}_{1,1}^{-1}\omega_1 & = & \mathbf{A}_{1,1}^{-1}\mathbf{z}_1, \\ (\mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2})\mathbf{X}^{\downarrow t} + \left(-\mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\omega_1 + \omega_2\right) & = & -\mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{z}_1 + \mathbf{z}_2. \end{array}$$

This is a new linear Gaussian system

$$\bar{\mathbf{A}}\mathbf{X} + \bar{\omega} = \bar{\mathbf{z}} \tag{10.26}$$

with

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{I}_k & \mathbf{A}_{1,1}^{-1}\mathbf{A}_{12} \\ \mathbf{0}_{m-k,k} & \mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2} \end{bmatrix} \tag{10.27}$$

and

$$\bar{\omega} = \begin{bmatrix} \bar{\omega}_1 \\ \bar{\omega}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1}^{-1}\omega_1 \\ -\mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\omega_1 + \omega_2 \end{bmatrix},$$

$$\bar{\mathbf{z}} = \begin{bmatrix} \bar{\mathbf{z}}_1 \\ \bar{\mathbf{z}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{1,1}^{-1}\mathbf{z}_1 \\ -\mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{z}_1 + \mathbf{z}_2 \end{bmatrix}.$$

We note that in the second part of the linear system (10.26) the variables in $s-t$ are eliminated. Therefore, this part of the system represents the part of the original Gaussian information (10.24) bearing on the variables in t, that is

$$l^{\downarrow t} : \bar{\mathbf{A}}_{2,2}\mathbf{X}^{\downarrow t} + \bar{\omega}_2 = \bar{\mathbf{z}}_2. \tag{10.28}$$

Here, the matrix $\bar{\mathbf{A}}_{2,2} = \mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2}$ corresponds to the right lower element of the matrix $\bar{\mathbf{A}}$. This linear Gaussian information can be represented by the generalized Gaussian potential $(\bar{\nu}, \bar{\mathbf{C}})$, where

$$\bar{\nu} = \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_{\bar{\omega}_2}\bar{\mathbf{z}}_2 \quad \text{and} \quad \bar{\mathbf{C}} = \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_{\bar{\omega}_2}\bar{\mathbf{A}}_{2,2}. \tag{10.29}$$

In order to fully determine this potential, it only remains to compute $\mathbf{K}_{\bar{\omega}_2}$. For this purpose, note that $\bar{\omega} = \mathbf{T}\omega$ with the matrix

$$\mathbf{T} = \begin{bmatrix} \mathbf{A}_{1,1}^{-1} & \mathbf{0} \\ -\mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1} & \mathbf{I} \end{bmatrix}.$$

The inverse of this matrix is

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{0} \\ \mathbf{A}_{2,1} & \mathbf{I} \end{bmatrix}.$$

From this we obtain the concentration matrix of $\bar{\omega} = \mathbf{T}\omega$,

$$\begin{aligned} \mathbf{K}_{\bar{\omega}} &= \mathbf{T}^{-1\,T}\mathbf{K}_{\omega}\mathbf{T}^{-1} \\ &= \begin{bmatrix} \mathbf{A}_{1,1}^T & \mathbf{A}_{2,1}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{1,1} & \mathbf{K}_{1,2} \\ \mathbf{K}_{2,1} & \mathbf{K}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{0} \\ \mathbf{A}_{2,1} & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}_1^T\mathbf{K}_{\omega}\mathbf{A}_1 & \mathbf{A}_1^T\mathbf{K}_2^T \\ \mathbf{K}_2\mathbf{A}_1 & \mathbf{K}_{2,2} \end{bmatrix}. \end{aligned} \tag{10.30}$$

Here, $\mathbf{A}_1$ is the matrix of $\mathbf{X}^{\downarrow s-t}$ in the system (10.25), whereas

$$\mathbf{K}_2 = \begin{bmatrix} \mathbf{K}_{2,1} & \mathbf{K}_{2,2} \end{bmatrix}$$

denotes the lower row in the decomposition of $\mathbf{K}_{\omega}$. We finally obtain the concentration matrix of the marginal density of $\bar{\omega}$, and it is shown in Appendix J.2 that this also corresponds to the concentration matrix of $\bar{\omega}_2$,

$$\mathbf{K}_{\bar{\omega}_2} = \mathbf{K}_{2,2} - \mathbf{K}_2\mathbf{A}_1(\mathbf{A}_1^T\mathbf{K}_{\omega}\mathbf{A}_1)^{-1}\mathbf{A}_1^T\mathbf{K}_2^T. \tag{10.31}$$

We have completely determined the information for $\mathbf{X}^{\downarrow t}$, once as extracted from the original system and then as the associated generalized Gaussian potential. It is very important to remark that these results do not depend on the particular choice of the regular submatrix $\mathbf{A}_{1,1}$ of $\mathbf{A}_1$, which is used in the variable elimination, since only the matrix $\mathbf{A}_1$ appears in the generalized Gaussian potential.

We want to see whether and how this potential relative to the extracted information can directly be obtained from the generalized Gaussian potential $(\nu, \mathbf{C})$ of the original system (10.24). As we have shown, equivalent systems have identical potentials. So, instead of computing the potential of system (10.24), we can as well compute the potential of the transformed system (10.26) with the matrix $\bar{\mathbf{A}}$ as defined in (10.27). For the pseudo-concentration matrix we have

$$\mathbf{C} = \mathbf{A}^T\mathbf{K}_{\omega}\mathbf{A} = \bar{\mathbf{A}}^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}} = \begin{bmatrix} \bar{\mathbf{A}}_1^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_1 & \bar{\mathbf{A}}_1^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_2 \\ \bar{\mathbf{A}}_2^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_1 & \bar{\mathbf{A}}_2^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_2 \end{bmatrix}.$$

The matrix $\bar{\mathbf{A}}_1$ denotes the left-hand column of the matrix $\bar{\mathbf{A}}$ in (10.27), whereas $\bar{\mathbf{A}}_2$ denotes the right-hand column. Here, $\mathbf{C}$ is decomposed with respect to $s - t$ and t. The blocks of $\mathbf{C}$ are as follows, if the internal structure of matrix $\bar{\mathbf{A}}$ in (10.27) is used:

$$\mathbf{C}_{1,1} = \bar{\mathbf{A}}_1^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}\mathbf{K}_{\bar{\omega}}\begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix}.$$

The last term equals the left upper element of $\mathbf{K}_{\bar{\omega}}$ in (10.30). We thus obtain

$$\mathbf{C}_{1,1} = \mathbf{A}_1^T\mathbf{K}_{\omega}\mathbf{A}_1.$$

If $\bar{\mathbf{A}}_{2,1}$ and $\bar{\mathbf{A}}_{2,2}$ denote the submatrices of $\bar{\mathbf{A}}_2$ corresponding to the decomposition with respect to $s-t$ and t, we obtain for the lower right element of $\mathbf{C}$ using (10.30)

$$\begin{aligned}
\mathbf{C}_{2,2} &= \bar{\mathbf{A}}_2^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_2 \\
&= \begin{bmatrix} \bar{\mathbf{A}}_{2,1}^T & \bar{\mathbf{A}}_{2,2}^T \end{bmatrix} \begin{bmatrix} \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1 & \mathbf{A}_1^T\mathbf{K}_2^T \\ \mathbf{K}_2\mathbf{A}_{2,1} & \mathbf{K}_{2,2} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{A}}_{2,1} \\ \bar{\mathbf{A}}_{2,2} \end{bmatrix} \\
&= \bar{\mathbf{A}}_{2,1}^T(\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1)\bar{\mathbf{A}}_{2,1} + \bar{\mathbf{A}}_{2,1}^T\mathbf{A}_1^T\mathbf{K}_2^T\bar{\mathbf{A}}_{2,2} \\
&+ \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_2\mathbf{A}_{2,1}\bar{\mathbf{A}}_{2,1} + \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_{2,2}\bar{\mathbf{A}}_{2,2}.
\end{aligned}$$

Next, we compute the lower left element of $\mathbf{C}$:

$$\begin{aligned}
\mathbf{C}_{2,1} &= \bar{\mathbf{A}}_2^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_1 \\
&= \begin{bmatrix} \bar{\mathbf{A}}_{2,1}^T & \bar{\mathbf{A}}_{2,2}^T \end{bmatrix}\mathbf{K}_{\bar{\omega}} \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} = \bar{\mathbf{A}}_{2,1}^T(\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1) + \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_2\mathbf{A}_1.
\end{aligned}$$

By the symmetry of the matrix $\mathbf{C}$ it finally follows that

$$\mathbf{C}_{1,2} = \mathbf{C}_{2,1}^T = (\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1)\bar{\mathbf{A}}_{2,1} + \mathbf{A}_1^T\mathbf{K}_2^T\bar{\mathbf{A}}_{2,2}.$$

These elements are sufficient to establish the relation between the concentration matrix $\mathbf{C}$ in the potential of the original system (10.24) and the concentration matrix $\bar{\mathbf{C}}$ in the potential of the reduced system (10.28). In fact, we conjecture that the same projection rule applies as for ordinary Gaussian potentials (10.5) in its transformed form $(\mathbf{C}\mu, \mathbf{C})$, i.e.

$$\mathbf{C}^{\Downarrow t} = ((\mathbf{C}^{-1})^{\downarrow t})^{-1} = \mathbf{C}_{2,2} - \mathbf{C}_{2,1}\mathbf{C}_{1,1}^{-1}\mathbf{C}_{1,2},$$

see Appendix J.2. This can easily be verified by using the above results for the submatrices $\mathbf{C}_{i,j}$ to obtain

$$\begin{aligned}
((\mathbf{C}^{-1})^{\downarrow t})^{-1} &= \mathbf{C}_{2,2} - \mathbf{C}_{2,1}\mathbf{C}_{1,1}^{-1}\mathbf{C}_{1,2} \\
&= \bar{\mathbf{A}}_{2,2}^T\left(\mathbf{K}_{2,2} - \mathbf{K}_2\mathbf{A}_1(\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1)^{-1}\mathbf{A}_1^T\mathbf{K}_2^T\right)\bar{\mathbf{A}}_{2,2} \\
&= \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_{2,2} = \bar{\mathbf{C}},
\end{aligned}$$

see equation (10.29). Similarly, we examine the first part ν of the potential,

$$\begin{aligned}
\nu &= \mathbf{A}^T\mathbf{K}_\omega\mathbf{z} = \bar{\mathbf{A}}^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{z}} \\
&= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \bar{\mathbf{A}}_{2,1}^T & \bar{\mathbf{A}}_{2,2}^T \end{bmatrix} \begin{bmatrix} \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1 & \mathbf{A}_1^T\mathbf{K}_2^T \\ \mathbf{K}_2\mathbf{A}_1 & \mathbf{K}_{2,2} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{z}}_1 \\ \bar{\mathbf{z}}_2 \end{bmatrix}.
\end{aligned}$$

We find for its two components

$$\begin{aligned}
\nu_1 &= \mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1\bar{\mathbf{z}}_1 + \mathbf{A}_1^T\mathbf{K}_2^T\bar{\mathbf{z}}_2, \\
\nu_2 &= \bar{\mathbf{A}}_{2,1}^T(\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1)\bar{\mathbf{z}}_1 + \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_2\mathbf{A}_1\bar{\mathbf{z}}_1 + \bar{\mathbf{A}}_{2,1}^T\mathbf{A}_1^T\mathbf{K}_2^T\bar{\mathbf{z}}_2 + \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_{2,2}\bar{\mathbf{z}}_2.
\end{aligned}$$

We next want to apply the projection rule for ordinary Gaussian potentials from equation (10.5) to the first part of the potential. This requires first transforming the rule to the new form of the potential. From $\nu = \mathbf{C}\mu$ we derive

$$\begin{aligned}\nu_1 &= \mathbf{C}_{1,1}\mu_1 + \mathbf{C}_{1,2}\mu_2,\\ \nu_2 &= \mathbf{C}_{2,1}\mu_1 + \mathbf{C}_{2,2}\mu_2.\end{aligned}$$

From this we obtain

$$\nu_2 - \mathbf{C}_{2,1}\mathbf{C}_{1,1}^{-1}\nu_1 \;=\; (\mathbf{C}_{2,2} - \mathbf{C}_{2,1}\mathbf{C}_{1,1}^{-1}\mathbf{C}_{1,2})\mu_2 \;=\; \mathbf{C}^{\downarrow t}\mu^{\downarrow t}.$$

So, the conjecture is

$$\nu^{\Downarrow t} \;=\; \nu_2 - \mathbf{C}_{2,1}\mathbf{C}_{1,1}^{-1}\nu_1.$$

In fact,

$$\begin{aligned}\nu_2 - \mathbf{C}_{2,1}\mathbf{C}_{1,1}^{-1}\nu_1 &= \bar{\mathbf{A}}_{2,2}^T\left(\mathbf{K}_{2,2} - \mathbf{K}_2\mathbf{A}_1(\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1)^{-1}\mathbf{A}_1^T\mathbf{K}_2^T\right)\mathbf{z}_2\\ &= \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{z}}_2 \;=\; \bar{\nu},\end{aligned}$$

see equation (10.29). Putting things together, we conclude that generalized and ordinary Gaussian potentials essentially share the same projection rule

$$\begin{aligned}(\nu,\mathbf{C})^{\downarrow t} \;=\; (\nu^{\Downarrow t},\mathbf{C}^{\Downarrow t}) \;=\; &\Big(\nu^{\downarrow t} - \mathbf{C}^{\downarrow t,s-t}(\mathbf{C}^{\downarrow s-t,s-t})^{-1}\nu^{\downarrow s-t},\\ &\mathbf{C}^{\downarrow t,s-t}(\mathbf{C}^{\downarrow s-t,s-t})^{-1}\mathbf{C}^{\downarrow s-t,t}\Big).\end{aligned}$$

This, however, does not yet cover the general case, since we assumed for the extraction of information that the matrix $\mathbf{A}_1$ has full column rank, i.e. that all columns of $\mathbf{A}_1$ are linearly independent, which is not always the case. Fortunately, it is now no more difficult to treat the general case. So, we return to the original system (10.24) together with its decomposition (10.25) according to the variables in $s - t$ and t. But this time we allow that the rank k of $\mathbf{A}_1$ may be less than the number of columns, $k < |s - t|$. Then, there are k linearly independent columns in $\mathbf{A}_1$ and we again assume that these are the k first columns. Denote the matrix formed by these k linearly independent columns by $\mathbf{A}_1'$ and the matrix formed by the remaining $|s - t| - k$ columns by $\mathbf{A}_1''$. The columns in the second matrix can be expressed as linear combinations of the first k columns. This means that there is a $k \times (m - k)$ matrix Λ such that

$$\mathbf{A}_1'' \;=\; \mathbf{A}_1'\Lambda.$$

Within the matrix $\mathbf{A}_1'$ of these first k columns there is a regular $k \times k$ submatrix. Now, if we decompose the matrix $\mathbf{A}$ of the system (10.24) according to k, $|s - t| - k$ and $|t|$ columns and the rows according to the first k and the remaining $m - k$ rows, we have

$$\mathbf{A} \;=\; \begin{bmatrix}\mathbf{A}_{1,1} & \mathbf{A}_{1,1}\Lambda & \mathbf{A}_{1,2}\\ \mathbf{A}_{2,1} & \mathbf{A}_{2,1}\Lambda & \mathbf{A}_{2,2}\end{bmatrix}.$$

The system (10.24) then decomposes accordingly, if x denotes the set of variables associated with the first k columns of $\mathbf{A}_1$ and y the set of variables associated with its remaining columns such that $x \cup y = s - t$, we have

$$\begin{aligned} \mathbf{A}_{1,1}\mathbf{X}^{\downarrow x} + \mathbf{A}_{1,1}\mathbf{\Lambda}\mathbf{X}^{\downarrow y} + \mathbf{A}_{1,2}\mathbf{X}^{\downarrow t} + \omega_1 &= \mathbf{z}_1, \\ \mathbf{A}_{2,1}\mathbf{X}^{\downarrow x} + \mathbf{A}_{2,1}\mathbf{\Lambda}\mathbf{X}^{\downarrow y} + \mathbf{A}_{2,2}\mathbf{X}^{\downarrow t} + \omega_2 &= \mathbf{z}_2. \end{aligned}$$

As before, the first part of this system can be solved for $\mathbf{X}^{\downarrow x} + \mathbf{\Lambda}\mathbf{X}^{\downarrow y}$ which gives

$$\begin{aligned} \mathbf{X}^{\downarrow x} + \mathbf{\Lambda}\mathbf{X}^{\downarrow y} + \mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2}\mathbf{X}^{\downarrow t} + \mathbf{A}_{1,1}^{-1}\omega_1 &= \mathbf{A}_{1,1}^{-1}\mathbf{z}_1, \\ (\mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2})\mathbf{X}^{\downarrow t} + (-\mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\omega_1 + \omega_2) &= -\mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{z}_1 + \mathbf{z}_2. \end{aligned}$$

We observe that ω is transformed into $\bar{\omega} = \mathbf{T}\omega$ just as before and consequently, $\bar{\omega}$ still has the concentration matrix $\mathbf{K}_{\bar{\omega}}$ as computed in (10.30). It follows from equation (10.31) that the same holds for $\mathbf{K}_{\bar{\omega}_2}$. As above we define the matrix

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{I}_k & \mathbf{\Lambda} & \mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2} \\ \mathbf{0}_{m-k,k} & \mathbf{0}_{m-k,|s-t|-k} & \mathbf{A}_{2,2} - \mathbf{A}_{2,1}^T\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2} \end{bmatrix}$$

where the submatrices $\bar{\mathbf{A}}_1$ (left-hand column), $\bar{\mathbf{A}}_{2,1}$ and $\bar{\mathbf{A}}_{2,2}$ (right upper and lower submatrices) are still the same. As a consequence, the generalized Gaussian potential of the projected system

$$l^{\downarrow t} : \bar{\mathbf{A}}_{2,2}\mathbf{X}^{\downarrow t} + \bar{\omega}_2 = \bar{\mathbf{z}}_2$$

does not change and equation (10.29) still holds. In contrast, the potential of the original system changes somewhat. If we define

$$\mathbf{B}^T = \begin{bmatrix} \mathbf{\Lambda}^T & \mathbf{0}_{m-k,|s-t|-k} \end{bmatrix},$$

we obtain

$$\mathbf{C} = \mathbf{A}^T\mathbf{K}_\omega\mathbf{A} = \bar{\mathbf{A}}^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}} = \begin{bmatrix} \bar{\mathbf{A}}_1^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_1 & \bar{\mathbf{A}}_1^T\mathbf{K}_{\bar{\omega}}\mathbf{B} & \bar{\mathbf{A}}_1^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_2 \\ \mathbf{B}^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_1 & \mathbf{B}^T\mathbf{K}_{\bar{\omega}}\mathbf{B} & \mathbf{B}^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_2 \\ \bar{\mathbf{A}}_2^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_1 & \bar{\mathbf{A}}_2^T\mathbf{K}_{\bar{\omega}}\mathbf{B} & \bar{\mathbf{A}}_2^T\mathbf{K}_{\bar{\omega}}\bar{\mathbf{A}}_2 \end{bmatrix}.$$

Writing $\mathbf{C}_{i,j}$ for the submatrices above and $i, j = 1, 2, 3$, we remark that $\mathbf{C}_{1,1}$ is the old $\mathbf{C}_{1,1}$ computed above, $\mathbf{C}_{3,3}$ is the old $\mathbf{C}_{2,2}$, $\mathbf{C}_{3,1}$ and $\mathbf{C}_{1,3}$ are the old $\mathbf{C}_{2,1}$ and $\mathbf{C}_{1,2}$. It only remains to compute the new $\mathbf{C}_{2,2}$, $\mathbf{C}_{2,1}$, $\mathbf{C}_{3,2}$ and the symmetric $\mathbf{C}_{1,2}$ and $\mathbf{C}_{2,3}$. From the matrix above one easily obtains

$$\begin{aligned} \mathbf{C}_{2,2} &= \mathbf{\Lambda}^T(\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1)\mathbf{\Lambda}, \\ \mathbf{C}_{2,1} &= \mathbf{\Lambda}^T(\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1), \\ \mathbf{C}_{3,2} &= \bar{\mathbf{A}}_{2,1}^T(\mathbf{A}_1^T\mathbf{K}_\omega\mathbf{A}_1)\mathbf{\Lambda} + \bar{\mathbf{A}}_{2,2}^T\mathbf{K}_2\mathbf{A}_1\mathbf{\Lambda}. \end{aligned}$$

The remaining matrices $\mathbf{C}_{1,2}$ and $\mathbf{C}_{2,3}$ are the transposes of $\mathbf{C}_{2,1}$ and $\mathbf{C}_{3,2}$. As before, it follows from these remarks that

$$\begin{aligned} \mathbf{C}_{3,3} - \mathbf{C}_{3,1}\mathbf{C}_{1,1}^{-1}\mathbf{C}_{1,3} &= \bar{\mathbf{C}}, \\ \nu_3 - \mathbf{C}_{3,1}\mathbf{C}_{1,1}^{-1}\nu_1 &= \bar{\nu}. \end{aligned}$$

More generally, it is now easy to verify that

$$\begin{bmatrix} \mathbf{C}_{2,2} & \mathbf{C}_{2,3} \\ \mathbf{C}_{3,2} & \mathbf{C}_{3,2} \end{bmatrix} - \begin{bmatrix} \mathbf{C}_{2,1} \\ \mathbf{C}_{3,1} \end{bmatrix} \mathbf{C}_{1,1}^{-1} \begin{bmatrix} \mathbf{C}_{1,2} & \mathbf{C}_{1,3} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \bar{\mathbf{C}} \end{bmatrix}$$

and also that

$$\begin{bmatrix} \nu_2 \\ \nu_3 \end{bmatrix} - \begin{bmatrix} \mathbf{C}_{2,1} \\ \mathbf{C}_{3,1} \end{bmatrix} \mathbf{C}_{1,1}^{-1}\nu_1 = \begin{bmatrix} 0 \\ \bar{\nu}_3 \end{bmatrix}.$$

This clarifies the general situation concerning extraction of information or projection. The following theorem summarizes this important result:

Theorem 10.2 *Consider the linear Gaussian system $l : \mathbf{AX} + \omega = \mathbf{z}$ for the set s of variables and the system $l^{\downarrow t}$ for $t \subseteq s$ obtained by eliminating the variables in $s - t$. If $(\nu, \mathbf{C}) = e(l)$ is the generalized Gaussian potential for the system l, then there is a subset $x \subseteq y = s - t$ such that $\mathbf{C}^{\downarrow x}$ is regular of rank $rank(\mathbf{C}^{\downarrow x}) = rank(\mathbf{C}^{\downarrow y})$. For each such subset x,*

$$e(l^{\downarrow t}) = \left(\nu^{\downarrow t} - \mathbf{C}^{\downarrow t,x}(\mathbf{C}^{\downarrow x})^{-1}\nu^{\downarrow x},\ \mathbf{C}^{\downarrow t} - \mathbf{C}^{\downarrow t,x}(\mathbf{C}^{\downarrow x})^{-1}\mathbf{C}^{\downarrow x,t}\right).$$

There may be several subsets $x \subseteq s - t$ for which the statement of the theorem holds, and for each such subset the generalized potential $e(l^{\downarrow t})$ is the same. In practice, this projected potential will be computed from the original potential $(\nu, \mathbf{C})$ by successive variable elimination. Gaussian potentials summarize the information contained in a linear Gaussian system. In the following section, we show that these potentials form a valuation algebra that later enables the application of local computation for the efficient solution of large, linear Gaussian systems. Illustrations are given in Section 10.5 and 10.6.

10.4 VALUATION ALGEBRA OF GAUSSIAN POTENTIALS

In the previous section, we defined combination of generalized Gaussian potentials, representing union of the underlying linear Gaussian systems, and projection of generalized Gaussian potentials, representing extraction of information in the corresponding linear Gaussian system. In this section, we show that the generalized Gaussian potentials form a valuation algebra under these operations and that the valuation algebra of ordinary Gaussian potentials introduced in Instance 1.6 essentially is a subalgebra of this valuation algebra. For this purpose, we show that hints form a valuation algebra isomorphic to the algebra of generalized Gaussian potentials.

To start, we formally define the algebra under consideration. Let r be a set of variables and D the subset lattice of r. For any non-empty subset $s \subseteq r$ we consider pairs $(\nu, \mathbf{C})$ where $\mathbf{C}$ is a symmetric, non-negative definite $s \times s$ matrix and ν an s-vector in the column space $\mathcal{C}(\mathbf{C})$ of $\mathbf{C}$. Denote by Φ_s the set of all such pairs. For the empty set we define $\Phi_\emptyset = \{(\diamond, \diamond)\}$. Finally, let

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

to be the set of all pairs $(\nu, \mathbf{C})$ with domain subset of r. The elements of Φ are called generalized Gaussian potentials, or shortly, potentials. Note that ordinary Gaussian potentials in the form $(\mathbf{K}\mu, \mathbf{K})$ also belong to Φ. Within Φ we define the operations of a valuation algebra as follows:

1. *Labeling:* $d(\nu, \mathbf{C}) = s$ if $(\nu, \mathbf{C}) \in \Phi_s$.

2. *Combination:* If $d(\nu_1, \mathbf{C}_1) = s$ and $d(\nu_2, \mathbf{C}_2) = t$ then

$$(\nu_1, \mathbf{C}_1) \otimes (\nu_2, \mathbf{C}_2) \quad = \quad \left(\nu_1^{\uparrow s \cup t} + \nu_2^{\uparrow s \cup t},\ \mathbf{C}_1^{\uparrow s \cup t} + \mathbf{C}_2^{\uparrow s \cup t}\right).$$

3. *Projection:* If $d(\nu, \mathbf{C}) = s$ and $t \subseteq s$ then

$$(\nu, \mathbf{C})^{\downarrow t} \quad = \quad \left(\nu^{\downarrow t} - \mathbf{C}^{\downarrow t,x}(\mathbf{C}^{\downarrow x})^{-1}\nu^{\downarrow x},\ \mathbf{C}^{\downarrow t} - \mathbf{C}^{\downarrow t,x}(\mathbf{C}^{\downarrow x})^{-1}\mathbf{C}^{\downarrow x,t}\right),$$

where $x \subseteq s - t$ is selected such that $\mathbf{C}^{\downarrow x}$ is regular with rank $rank(\mathbf{C}^{\downarrow x}) = rank(\mathbf{C}^{\downarrow s-t})$. According to Theorem 10.2 such a subset always exists.

Instead of directly verifying the valuation algebra axioms for these operations, it is easier to examine a corresponding algebra of hints. Indeed, we know that potentials faithfully represent the associated operations with linear Gaussian systems, in particular with Gaussian hints. It turns out to be easier to verify the axioms of a valuation algebra with hints rather than with potentials. We remember however that operations with hints are not uniquely defined. For example, depending on how the variables in $s - t$ are eliminated in a hint $h\ :\ \mathbf{AX} + \omega = \mathbf{z}$ on variables s, the reduced systems $h^{\downarrow t}$ are different, but equivalent. Therefore, rather than considering individual hints h, we must look at classes of equivalent hints $[h]$. So, as with potentials, we define Ψ_s to be the family of all classes $[h]$ where h are hints relating to the subset $s \subseteq r$ of variables and

$$\Psi \quad = \quad \bigcup_{s \subseteq r} \Psi_s.$$

As above, we represent hints h by triplets $(\mathbf{A}, \mathbf{z}, \mathbf{K})$ where $\mathbf{A}$ is an $m \times s$ matrix with full row rank, hence $m \leq |s|$, $\mathbf{z}$ is an m-vector and $\mathbf{K}$ a symmetric, positive definite $s \times s$ matrix. In order to define the basic operations of labeling, combination and projection with hints we denote the operation of the union of two linear systems $l_1 = (\mathbf{A}_1, \mathbf{z}_1, \mathbf{K}_1)$ and $l_2 = (\mathbf{A}_2, \mathbf{z}_2, \mathbf{K}_2)$ on variables s and t respectively by

$$l_1 \oplus l_2 = \left(\begin{bmatrix} \mathbf{A}_1^{\uparrow s \cup t} \\ \mathbf{A}_2^{\uparrow s \cup t} \end{bmatrix}, \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 \end{bmatrix} \right).$$

Similarly, for a system $l = (\mathbf{A}_1, \mathbf{z}_1, \mathbf{K}_1)$ on variables in s and $t \subseteq s$, let

$$l^{\downarrow t} = (\bar{\mathbf{A}}_{2,2}, \bar{\mathbf{z}}_2, \mathbf{K}_{\bar{\omega}})$$

denote a system like (10.28) were the variables in $s - t$ have been eliminated. Finally, if l is a linear Gaussian system, then let $h(l)$ denote the equivalence class of hints derived from l. We are now able to define the basic operations in Ψ:

1. *Labeling:* $d([h]) = s$ if $[h] \in \Psi_s$.

2. *Combination:* For $h_1 \in [h_1]$ and $h_2 \in [h_2]$

$$[h_1] \otimes [h_2] = h(h_1 \oplus h_2).$$

3. *Projection:* If $d([h]) = s$ and $t \subseteq s$ then for $h \in [h]$

$$[h]^{\downarrow t} = h([h^{\downarrow t}]).$$

These operation are all well-defined, since equivalent systems have equivalent unions and projections. The very technical proof of the following theorem can be found in Appendix J.1.

Theorem 10.3 *The algebra of hints $\langle \Psi, D \rangle$ with labeling, combination and projection satisfies the axioms of a valuation algebra.*

We show that the algebra of generalized Gaussian potentials $\langle \Phi, D \rangle$, which consists of the potentials $e(h) \in \Phi$ associated to the hints $h \in \Psi$, also forms a valuation algebra. We first recall that the mapping $e : \Psi \to \Phi$, defined by $e([h]) = e(h)$, respects labeling, combination and projection

$$d(e([h])) = d([h]), \quad e([h]_1 \otimes [h]_2) = e([h]_1) \otimes e([h]_2), \quad e([h]^{\downarrow t}) = e([h])^{\downarrow t}.$$

This has been shown in Section 10.3. Further, we claim that the mapping is onto. In fact, let $(\nu, \mathbf{C})$ be an element of Φ. $\mathbf{C}$ is therefore a symmetric, non-negative definite $s \times s$ matrix and ν an s-vector in the column space $\mathcal{C}(\mathbf{C})$ of $\mathbf{C}$. For any symmetric non-negative definite $s \times s$ matrix $\mathbf{C}$ of rank $k \leq |s|$, there exists a $k \times s$ matrix $\mathbf{A}$ of rank k such that $\mathbf{C} = \mathbf{A}^T\mathbf{A}$. This is a result from (Harville, 1997). The columns of $\mathbf{C}$ span the same linear space as $\mathbf{A}^T$ and therefore, there is a k-vector $\mathbf{z}$ such that

$$\nu = \mathbf{A}^T\mathbf{z}.$$

Consequently, $h = (\mathbf{A}, \mathbf{z}, \mathbf{I}_k)$ is a hint on s such that

$$e(h) \;=\; e([h]) \;=\; (\mathbf{A}^T\mathbf{I}_k\mathbf{z}, \mathbf{A}\mathbf{I}_k\mathbf{A}^T) \;=\; (\mathbf{A}^T, \mathbf{A}^T\mathbf{A}) \;=\; (\nu, \mathbf{C}).$$

The fact that $\langle \Psi, D \rangle$ is a valuation algebra then implies that $\langle \Phi, D \rangle$ is a valuation algebra, too, and that both algebras are isomorphic.

We have seen in Section 10.1 that Gaussian hints $h = (\mathbf{A}, \mathbf{z}, \mathbf{K})$, where $\mathbf{A}$ has full column rank, are represented by ordinary Gaussian potentials

$$\Big((\mathbf{A}^T\mathbf{K}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{K}\mathbf{z}, \mathbf{A}^T\mathbf{K}\mathbf{A}\Big),$$

that in turn determine Gaussian densities. Such a potential can also be one-to-one transformed into a generalized Gaussian potential $(\mathbf{A}^T\mathbf{K}\mathbf{z}, \mathbf{A}^T\mathbf{K}\mathbf{A})$, and combination as well as projection of ordinary Gaussian potentials correspond to the operations on generalized Gaussian potentials. Thus, the original valuation algebra of ordinary Gaussian potentials can be embedded into the valuation algebra of generalized Gaussian potentials and can therefore be considered as a subalgebra of the latter. These insights are summarized in the following theorem:

Theorem 10.4 *Both algebras of hints $\langle \Psi, D \rangle$ and generalized Gaussian potentials $\langle \Phi, D \rangle$ are valuation algebras and $e : \Psi \rightarrow \Phi$ is an isomorphism between them. The valuation algebra of ordinary Gaussian potentials is embedded into $\langle \Phi, D \rangle$.*

Consequently, we may perform all computations with hints also with generalized Gaussian potentials. This will be exploited in the remaining sections of this chapter.

10.5 AN APPLICATION: GAUSSIAN DYNAMIC SYSTEMS

Linear dynamic systems with Gaussian disturbances are used in many fields, for instance in control theory (Kalman, 1960) or in coding theory (MacKay, 2003). The basic scenario is the following: the state of a system changes in time in a linear way and is also influenced by Gaussian disturbances. However, the state cannot be observed directly, but only through some linear functions, again disturbed by Gaussian noise. The problem is to reconstruct the unknown present, past or future states from the available temporal series of measurements. This is a very well-known and much studied problem, usually from the perspective of least squares estimation or of maximum likelihood estimation. Here, we shall look at this problem from the point of view of local computation in the valuation algebra of generalized Gaussian potentials. Therefore, this section will be an illustration and application of ideas, concepts and results introduced in previous parts of this chapter. Although the main results will not be new, the approach is dramatically different from the usual ways of treating the problem. This results in new insights into the well-known problem which is a value in itself. We should, however, credit (Dempster, 1990a; Dempster, 1990b) and also refer to (Kohlas, 1991; Monney, 2003; Kohlas & Monney, 2008) for applying the theory of Gaussian hints to this problem. But the use of generalized Gaussian potentials, as proposed here, is different and more straightforward. We also refer to Instance 9.2 for a simplified version of a similar problem.

10.2 Kalman Filter: Filtering Problem

The basic model of a time-discrete, linear dynamic system with additive Gaussian noise is as follows:

$$\begin{aligned} \mathbf{X}_{k+1} &= \mathbf{A}_k\mathbf{X}_k + \omega_k, & (10.32)\\ \mathbf{Y}_k &= \mathbf{H}_k\mathbf{X}_k + \nu_k, & (10.33)\\ \mathbf{Y}_k &= \mathbf{y}_k, & (10.34)\\ \mathbf{X}_0 &= \omega_0, & (10.35) \end{aligned}$$

for $k = 1, 2, \ldots$ Here, $\mathbf{X}_k$ are n-dimensional vectors, the matrices $\mathbf{A}_k$ are $n \times n$ real-valued matrices, the vectors $\mathbf{Y}_k$ are m-vectors, $\mathbf{y}_k \in \mathbb{R}^m$ and $\mathbf{H}_k$ denote $m \times n$ real-valued matrices. The n-vector disturbances ω_k and m-vector disturbances ν_k are distributed normally with mean vectors $\mathbf{0}$ and variance-covariance-matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ respectively. Equation (10.32) defines a *first order Markov state evolution process.* Further, equation (10.33) defines the measurement process for state variables, namely how the measurement $\mathbf{Y}_k$ will be determined from the state $\mathbf{X}_k$. In equation (10.34) the actually observed value $\mathbf{y}_k$ is introduced into the system. Finally, equation (10.35) fixes the initial condition of the process. Although this is usually introduced as part of the system, this information is not really needed in our approach. We come back to this point later in the development. Such a linear Gaussian system is called a *Kalman filter model* after the inventor of the filter solution associated with the system. We can easily transform this system into the standard form of a linear Gaussian system,

$$\begin{aligned} \begin{bmatrix} \mathbf{A}_k & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{X}_k \\ \mathbf{X}_{k+1} \end{bmatrix} + \omega_k &= \mathbf{0}, & (10.36)\\ \mathbf{H}_k\mathbf{X}_k + \nu_k &= \mathbf{y}_k, & (10.37)\\ -\mathbf{X}_0 + \omega_0 &= \mathbf{0}. & (10.38) \end{aligned}$$

Note that we substituted the observation equation (10.34) directly into (10.33) as suggested in the previous Section 10.6. This has the effect that the variables $\mathbf{Y}_k$ disappear from the model. In fact, the Kalman filter model is also an instance of a *Gaussian Bayesian network* as discussed in the following Section 10.6. Figure 10.4 represents the system graphically. Alternatively, if conditional Gaussian densities instead of the linear Gaussian systems are given in the graph of Figure 10.4, we speak of a *hidden Markov chain.* It will be shown in Section 10.6 that we may reduce this model to a Kalman filter model.

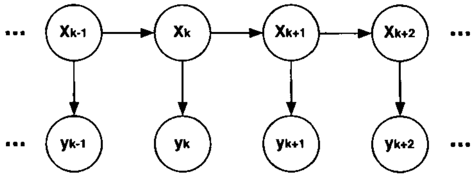


Figure 10.4 The direct influence among variable groups in the Kalman filter model.

If we consider the standard form of the Kalman filter model given above, but with the variables $\mathbf{Y}_k$ eliminated, we see that we may cover this system by the join tree of Figure 10.5. Each subsystem (10.36) is assigned to the nodes labeled with $\mathbf{X}_k, \mathbf{X}_{k+1}$. We point out that node labels in join trees usually correspond to either sets of variables or indices. Here, it is more convenient to refer to the node labels by the vectors directly. So, the subsystems (10.37) are affected to the nodes with label $\mathbf{X}_k$, and the initial condition (10.38) is on the node labeled with $\mathbf{X}_0$. To each of these linear Gaussian subsystems belongs a corresponding generalized Gaussian potential. Let $\phi_{k,k+1}$ for $k = 1, 2, \ldots$ denote the potential on node $\mathbf{X}_k, \mathbf{X}_{k+1}$, ϕ_0 the potential on node $\mathbf{X}_0$ and ψ_k are the potentials on the nodes $\mathbf{X}_k$. According to equation (10.12) these potentials are

$$\phi_0 = \left(\mathbf{0}, \mathbf{Q}_0^{-1}\right), \quad \phi_{k,k+1} = \left(\mathbf{0}, \begin{bmatrix} \mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k & -\mathbf{A}_k^T\mathbf{Q}_k^{-1} \\ -\mathbf{Q}_k^{-1}\mathbf{A}_k & \mathbf{Q}_k^{-1} \end{bmatrix}\right) \tag{10.39}$$

and

$$\psi_k = \left(\mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{y}_k, \mathbf{H}_k^T\mathbf{R}_k^{-1}\mathbf{H}_k\right). \tag{10.40}$$

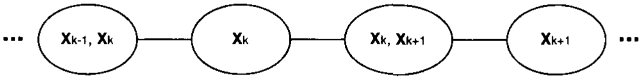


Figure 10.5 A covering join tree for the Kalman filter model.

Assume now that measurements $\mathbf{y}_k$ are given for $k = 1, \ldots, h$. The information about the whole system (10.32), (10.33), (10.34) and (10.35) for $k = 1, \ldots, h$ is then represented by the potential

$$\chi_h = \phi_0 \otimes \left(\bigotimes_{k=0}^{h-1} \phi_{k,k+1}\right) \otimes \left(\bigotimes_{k=1}^{h} \psi_k\right).$$

Computing $\chi_h^{\downarrow h}$ means to infer about the state $\mathbf{X}_h$ given the Kalman filter model and state measurements for $k = 1$ up to $k = h$. This is called the *filter problem* for time h. It can easily be solved by orienting the join tree of the Kalman filter model in

Figure 10.5 towards node $\mathbf{X}_h$ and applying the collect algorithm from Section 3.8. In fact, due to the linear structure of the join tree in Figure 10.5, after the messages are passed up to node $\mathbf{X}_{h-1}$, this node contains the valuation $\chi_{h-1}^{\downarrow h-1}$. Then, the message $\chi_{h-1}^{\downarrow h-1}$ is sent from node $\mathbf{X}_{h-1}$ to node $\mathbf{X}_{h-1}, \mathbf{X}_h$ where it is combined with the node content $\phi_{h-1,h}$. Finally, the result is projected to h and sent to the root node $\mathbf{X}_h$ where it is again combined to the node content. The root node then contains

$$\chi_h^{\downarrow h} \quad = \quad \left(\chi_{h-1}^{\downarrow h-1} \otimes \phi_{h-1,h}\right)^{\downarrow h} \otimes \psi_h. \tag{10.41}$$

This is a *recursive solution* of the filtering problem. The recursion starts with the initial condition $\chi_0^{\downarrow 0} = \phi_0$ and is applied iteratively for each time step $h = 1, 2, \ldots$ incorporating the new state measurements at time h. The message

$$\mu_{(h-1,h)\to h} \quad = \quad \left(\chi_{h-1}^{\downarrow h-1} \otimes \phi_{h-1,h}\right)^{\downarrow h} \tag{10.42}$$

sent from node $\mathbf{X}_{h-1}, \mathbf{X}_h$ to node $\mathbf{X}_h$ is also called the *one-step forward prediction*. It corresponds to the inference about the state variable $\mathbf{X}_h$ before the measurement $\mathbf{y}_h$ of this state becomes available. Note that these results do not yet depend on the actual valuation algebra instance, but they only reflect the linear form of the join tree.

The recursive computation of the potentials (10.41) can be translated into corresponding matrix operations for the elements involved in the potentials, if we use the definitions of combination and projection of the algebra of generalized Gaussian potentials given by (10.39) and (10.40). The recursion (10.41) can now be transformed step-wise into the corresponding operations on the associated generalized Gaussian potentials. To start with let

$$\chi_k^{\downarrow k} \quad = \quad (\nu_k, \mathbf{C}_k),$$

where for $k = 0$ we have the initial condition

$$\chi_0^{\downarrow 0} \; = \; \phi_0 \; = \; (\nu_0, \mathbf{Q}_0^{-1}).$$

We now apply the valuation algebra operations for generalized Gaussian potentials defined in Section 10.4. If follows from the rule of combination that

$$\begin{aligned}
\chi_k^{\downarrow k} \otimes \phi_{k,k+1} \quad &= \quad (\nu_k, \mathbf{C}_k) \otimes \left(\mathbf{0}, \begin{bmatrix} \mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k & -\mathbf{A}_k^T\mathbf{Q}_k^{-1} \\ -\mathbf{Q}_k^{-1}\mathbf{A}_k & \mathbf{Q}_k^{-1} \end{bmatrix}\right) \\
&= \quad \left(\begin{bmatrix} \nu_k \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k + \mathbf{C}_k & -\mathbf{A}_k^T\mathbf{Q}_k^{-1} \\ -\mathbf{Q}_k^{-1}\mathbf{A}_k & \mathbf{Q}_k^{-1} \end{bmatrix}\right)
\end{aligned}$$

From this we compute the projection or one-step forward prediction (10.42),

$$\mu_{(k,k+1)\to k+1} \quad = \quad \Big(\mathbf{Q}_k^{-1}\mathbf{A}_k(\mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k + \mathbf{C}_k)^{-1}\nu_k, \tag{10.43}$$

$$\mathbf{Q}_k^{-1} - \mathbf{Q}_k^{-1}\mathbf{A}_k(\mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k + \mathbf{C}_k)^{-1}\mathbf{A}_k^T\mathbf{Q}_k^{-1}\Big) \tag{10.44}$$

Here, we tacitly assume that the matrix $\mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k+\mathbf{C}_k$ is regular. This is surely the case if either $\mathbf{A}_k$ or $\mathbf{C}_k$ is regular, since then either $\mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k$ or $\mathbf{C}_k$ is symmetric, positive definite. If this is not the case, then Theorem 10.2 must be applied to compute the projection. We further define

$$\mu_{(k,k+1)\rightarrow k+1} \quad = \quad \Big(\nu_{k,k+1},\ \mathbf{C}_{k,k+1}\Big),$$

with

$$\begin{aligned} \nu_{k,k+1} &= \mathbf{Q}_k^{-1}\mathbf{A}_k(\mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k+\mathbf{C}_k)^{-1}\nu_k & (10.45)\\ \mathbf{C}_{k,k+1} &= \mathbf{Q}_k^{-1}-\mathbf{Q}_k^{-1}\mathbf{A}_k(\mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k+\mathbf{C}_k)^{-1}\mathbf{A}_k^T\mathbf{Q}_k^{-1} & (10.46)\end{aligned}$$

This determines the one-step forward prediction. Finally, we obtain the filter potential (10.41) for $\mathbf{X}_{k+1}$, given the observation $\mathbf{y}_1,\ldots,\mathbf{y}_{k+1}$, again by the combination rule for generalized Gaussian potentials:

$$\begin{aligned} \chi_{k+1}^{\downarrow k+1} &= \mu_{(k,k+1)\rightarrow k+1}\otimes\psi_{k+1}\\ &= \Big(\nu_{k,k+1}+\mathbf{H}_{k+1}^T\mathbf{R}_{k+1}^{-1}\mathbf{y}_{k+1},\ \mathbf{C}_{k,k+1}+\mathbf{H}_{k+1}^T\mathbf{R}_{k+1}^{-1}\mathbf{H}_{k+1}\Big).\end{aligned}$$

So, we have

$$\begin{aligned} \nu_{k+1} &= \nu_{k,k+1}+\mathbf{H}_{k+1}^T\mathbf{R}_{k+1}^{-1}\mathbf{y}_{k+1}, & (10.47)\\ \mathbf{C}_{k+1} &= \mathbf{C}_{k,k+1}+\mathbf{H}_{k+1}^T\mathbf{R}_{k+1}^{-1}\mathbf{H}_{k+1}. & (10.48)\end{aligned}$$

The equations (10.45), (10.46), (10.47) and (10.48) define a general recursion scheme for the filtering solution of a linear dynamic system with Gaussian disturbances. Note that in this general solution no assumptions about the initial condition of $\mathbf{X}_0$ are needed. In fact, it is not necessary that ϕ_0 is an ordinary Gaussian potential as assumed above. Further, no regularity assumptions about the matrices $\mathbf{A}_k$ are needed; provided that for the computation of the one-step forward projection the general projection rule of Theorem 10.2 is used. However, in the usual Kalman filter applications as discussed in the literature, see for instance (Roweis & Ghahramani, 1999), the initial condition ϕ_0 in the form of an ordinary Gaussian potential is assumed and the matrices $\mathbf{A}_k$ are assumed to be regular. Consequently, all the filter potentials $\chi_k^{\downarrow k}$ are ordinary Gaussian potentials, which of course simplifies the interpretation of the filter solution as well as the computations. That the potentials $\chi_k^{\downarrow k}$ are ordinary Gaussian potentials in this case follows by induction over k. Indeed, $\chi_0^{\downarrow 0}=\phi_0$ is an ordinary Gaussian potential by assumption. Assume then that this holds also for $\chi_k^{\downarrow k}$ which means that in the generalized Gaussian potential $\chi_k^{\downarrow k}=(\nu_k,\mathbf{C}_k)$ the matrix $\mathbf{C}_k$ is positive definite, hence regular. But then, as remarked above, the same holds for $\mathbf{C}_{k,k+1}$ and by the same argument it then also holds for $\mathbf{C}_k$ defined in (10.48).

We propose in this case to compute directly with the mean vector and the variance-covariance matrix of the different potentials. If $\chi_k^{\downarrow k}=(\nu_k,\mathbf{C}_k)$ is an ordinary

Gaussian potential, i.e. the matrix $\mathbf{C}_k$ is regular and $\mathbf{C}_k^{-1} = \mathbf{\Sigma}_k$ is the variance-covariance matrix of the filter solution at time k, then

$$\chi_k^{\downarrow k} \otimes \phi_{k,k+1}$$

is an ordinary Gaussian potential too. It represents a two-component vector, where the first component is the filter solution at time k and the second component the one-step forward prediction to time $k+1$. It has the variance-covariance matrix

$$\begin{bmatrix} \mathbf{A}_k^T\mathbf{Q}_k^{-1}\mathbf{A}_k & -\mathbf{A}_k^T\mathbf{Q}_k^{-1} \\ -\mathbf{Q}_k^{-1}\mathbf{A}_k & \mathbf{Q}_k^{-1} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{\Sigma}_k & \mathbf{\Sigma}_k\mathbf{A}_k^T \\ \mathbf{A}_k\mathbf{\Sigma}_k & \mathbf{A}_k\mathbf{\Sigma}_k\mathbf{A}_k^T + \mathbf{Q}_k \end{bmatrix}. \qquad (10.49)$$

This can easily be verified. Then, since $\nu_k = \mathbf{C}_k\mu_k$ if μ_k is the mean value of the ordinary potential representing the filter solution $\chi_k^{\downarrow k}$ at time k, we conclude that

$$\begin{bmatrix} \mu_k \\ \mu_{k,k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{\Sigma}_k & \mathbf{\Sigma}_k\mathbf{A}_k^T \\ \mathbf{A}_k\mathbf{\Sigma}_k & \mathbf{A}_k\mathbf{\Sigma}_k\mathbf{A}_k^T + \mathbf{Q}_k \end{bmatrix} \begin{bmatrix} \mathbf{C}_k\mu_k \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mu_k \\ \mathbf{A}_k\mu_k \end{bmatrix},$$

where $\mu_{k,k+1} = \mathbf{C}_{k,k+1}^{-1}\nu_{k,k+1}$ is the mean vector of the one-step forward prediction. It follows that

$$\mu_{k,k+1} = \mathbf{A}_k\mu_k.$$

This, by the way, is a result which could also be derived from elementary probability considerations. For the variance-covariance matrix of the one-step forward prediction we read directly from equation (10.49) that

$$\mathbf{\Sigma}_{k,k+1} = \mathbf{A}_k\mathbf{\Sigma}_k\mathbf{A}_k^T + \mathbf{Q}_k.$$

Next, we go for the filter $\chi_{k+1}^{\downarrow k+1} = (\nu_{k+1}, \mathbf{C}_{k+1})$. According to equation (10.48), the inverse of $\mathbf{C}_{k+1}$ is the variance-covariance matrix of the filter solution at time $k+1$,

$$\begin{aligned} \mathbf{\Sigma}_{k+1} &= (\mathbf{C}_{k,k+1} + \mathbf{H}_{k+1}^T\mathbf{R}_{k+1}^{-1}\mathbf{H}_{k+1})^{-1} \\ &= \mathbf{\Sigma}_{k,k+1} - \mathbf{\Sigma}_{k,k+1}\mathbf{H}_{k+1}^T(\mathbf{H}_{k+1}\mathbf{\Sigma}_{k,k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_{k+1})^{-1}\mathbf{H}_{k+1}\mathbf{\Sigma}_{k,k+1}. \end{aligned}$$

Here, we use a well-known and general identity from matrix algebra (Harville, 1997):

$$(\mathbf{A} + \mathbf{B}^T\mathbf{C}^{-1}\mathbf{B})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}^T(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T + \mathbf{C})^{-1}\mathbf{B}\mathbf{A}^{-1}$$

and $\Sigma_{k,k+1} = \mathbf{C}_{k,k+1}^{-1}$. This result together with equation (10.47) can be used to compute the mean value μ_{k+1} of the filter solution,

$$\begin{aligned}
\mu_{k+1} &= \mathbf{C}_{k+1}^{-1}\nu_{k+1} \\
&= \Sigma_{k+1}(\Sigma_{k,k+1}^{-1}\mu_{k,k+1} + \mathbf{H}_{k+1}^{T}\mathbf{R}_{k+1}^{-1}\mathbf{y}_{k+1}) \\
&= (\Sigma_{k,k+1} - \Sigma_{k,k+1}\mathbf{H}_{k+1}^{T}(\mathbf{H}_{k+1}\Sigma_{k,k+1}\mathbf{H}_{k+1}^{T} + \mathbf{R}_{k+1})^{-1}\mathbf{H}_{k+1}\Sigma_{k,k+1}) \\
&\quad (\Sigma_{k,k+1}^{-1}\mu_{k,k+1} + \mathbf{H}_{k+1}^{T}\mathbf{R}_{k+1}^{-1}\mathbf{y}_{k+1}) \\
&= \mu_{k,k+1} - \Sigma_{k,k+1}\mathbf{H}_{k+1}^{T}(\mathbf{H}_{k+1}\Sigma_{k,k+1}\mathbf{H}_{k+1}^{T} + \mathbf{R}_{k+1})^{-1} \\
&\quad [-(\mathbf{H}_{k+1}\Sigma_{k,k+1}\mathbf{H}_{k+1}^{T} + \mathbf{R}_{k+1})\mathbf{R}_{k+1}^{-1}\mathbf{y}_{k+1} + \mathbf{H}_{k+1}\mu_{k,k+1} + \\
&\quad \mathbf{H}_{k+1}\Sigma_{k,k+1}\mathbf{H}_{k+1}^{T}\mathbf{R}_{k+1}^{-1}\mathbf{y}_{k+1}] \\
&= \mu_{k,k+1} - \Sigma_{k,k+1}\mathbf{H}_{k+1}^{T}(\mathbf{H}_{k+1}\Sigma_{k,k+1}\mathbf{H}_{k+1}^{T} + \mathbf{R}_{k+1})^{-1} \\
&\quad [\mathbf{H}_{k+1}\mu_{k,k+1} - \mathbf{y}_{k+1}].
\end{aligned}$$

This determines the classical Kalman filter for linear dynamic systems with Gaussian noise. Often, the *gain matrix*

$$\mathbf{K}_k = \Sigma_{k,k+1}\mathbf{H}_{k+1}^{T}(\mathbf{H}_{k+1}\Sigma_{k,k+1}\mathbf{H}_{k+1}^{T} + \mathbf{R}_{k+1})^{-1}$$

is introduced and the filter equations become

$$\mu_{k+1} = \mu_{k,k+1} - \mathbf{K}_k(\mathbf{H}_{k+1}\mu_{k,k+1} - \mathbf{y}_{k+1}) \tag{10.50}$$

$$\Sigma_{k+1} = \Sigma_{k,k+1} - \mathbf{K}_k\mathbf{H}_{k+1}\Sigma_{k,k+1} \tag{10.51}$$

We finally remark that these recursive equations are usually derived by the least squares or by maximum likelihood methods as estimators for the unknown state variables. In contrast, they result from assumption-based reasoning in our approach and the potentials represent fiducial probability distributions. Although the results are formally identical, they differ in their derivation and interpretation.

10.6 AN APPLICATION: GAUSSIAN BAYESIAN NETWORKS

Graphical models such as the Bayesian networks of Instance 2.1 are popular tools for probabilistic modeling, in particular using discrete probability distributions. These models serve to generate a factorization of a probability distribution into a product of conditional distributions, which is then amenable to local computation. A similar scheme applies also to Gaussian densities.

■ 10.3 Gaussian Bayesian Networks

We recall that a Bayesian network is a directed, acyclic graph $G = (V, E)$ with a set V of nodes representing variables and a set E of directed edges. We number the nodes from 1 to $m = |V|$ and denote the variable associated with node i by X_i. The parent set $pa(i)$ of node i contains all the nodes j with a directed edge from j to i, i.e. $pa(i) = \{j \in V : (j,i) \in E\}$. Nodes without parents are called roots. Let $\mathbf{X}_{pa(i)}$ denote the vector of variables associated to the parents of node i. Then, a *conditional Gaussian density* function $f_{X_i|\mathbf{X}_{pa(i)}}$ is associated with every node $i \in V$. If i is a root, this is simply an ordinary Gaussian density f_{X_i}. The graph gives rise to a function f of m variables which is the product of the conditional Gaussian densities associated with the nodes,

$$f(x_1,\ldots,x_m) = \prod_{i\in V} f_{X_i|\mathbf{X}_{pa(i)}}(x_i|\mathbf{x}_{pa(i)}).$$

Theorem 10.5 *The function $f(x_1,\ldots,x_m)$ is a Gaussian density.*

Proof: It is always possible to number the nodes in the graph G in such a way that any parent j of a node i has a smaller number than i, i.e. $j < i$ if $j \in pa(i)$. Such an order of the nodes is called a *construction order* (Shafer, 1996). Let us now assume that the nodes of G are numbered in a construction order. We then have

$$f(x_1,\ldots,x_m) = $$
$$f_{X_1}(x_1)f_{X_1|X_2}(x_1|x_2)f_{X_3|\mathbf{X}_{pa(3)}}(x_3|\mathbf{x}_{pa(3)})\cdots f_{X_m|\mathbf{X}_{pa(m)}}(x_m|\mathbf{x}_{pa(m)}).$$

Clearly, each subproduct from $j = 1$ up to some $i \leq m$ is a Gaussian density, hence this holds for the whole product. ■

The factorization of a Gaussian density $f(x_1,\ldots,x_m)$ into a product of conditional Gaussian densities as induced by a Bayesian network is very particular. We may have much more general factorizations of Gaussian densities,

$$f(x_1,\ldots,x_m) = \prod_{(h,t)\in\mathcal{J}} f_{\mathbf{X}_h|\mathbf{X}_t}(\mathbf{x}_h|\mathbf{x}_t) \tag{10.52}$$

where $\mathcal{J}$ is a family of pairs (h,t) of subsets h,t of the index set $\{1,\ldots,m\}$ of the variables $X_1,\ldots,X_m$. The set h is called the *head* and the set t the *tail* of the conditional density $f_{\mathbf{X}_h|\mathbf{X}_t}(\mathbf{x}_h|\mathbf{x}_t)$. The symbols $\mathbf{X}_h$ and $\mathbf{X}_t$ represent the vectors of variables over the index sets h and t respectively. Similarly, $\mathbf{x}_h$ and $\mathbf{x}_t$ represent the corresponding real-valued vectors of the variable vectors. Of course, the collection of conditional Gaussian densities $f_{\mathbf{X}_h|\mathbf{X}_t}(\mathbf{x}_h|\mathbf{x}_t)$ must satisfy some conditions to turn $f(x_1,\ldots,x_m)$ into a Gaussian density. In fact, it is necessary and sufficient that the conditionals can be brought in such an order (h_j,t_j) for $j = 1,\ldots,n$ that the following conditions are satisfied:

1. t_1 is empty;

2. $t_i \subseteq d_1 \cup \ldots \cup d_{i-1}$ and h_i is disjoint from $d_1 \cup \ldots \cup d_{i-1}$ where $d_i = h_i \cup t_i$.

Theorem 10.6 *If these conditions above are satisfied, the product*

$$\prod_{j=1}^{i} f_{\mathbf{X}_{h_j}|\mathbf{X}_{t_j}}(\mathbf{x}_{h_j}|\mathbf{x}_{t_j})$$

is a Gaussian density for all $i = 1, \ldots, n$.

Proof: We proceed by induction: The theorem holds for $i = 1$ since

$$f_{\mathbf{X}_{h_1}|\mathbf{X}_{t_1}}(\mathbf{x}_{h_1}|\mathbf{x}_{t_1}) \quad = \quad f_{\mathbf{X}_{h_1}}(\mathbf{x}_{h_1})$$

is a Gaussian density. Assume that the theorem holds for $i - 1$, i.e.

$$f(x_1, \ldots, x_k) \quad = \quad \prod_{j=1}^{i-1} f_{\mathbf{X}_{h_j}|\mathbf{X}_{t_j}}(\mathbf{x}_{h_j}|\mathbf{x}_{t_j}),$$

where $\{1, \ldots, k\} = d_1 \cup \cdots \cup d_{i-1}$ is a Gaussian density. Then

$$f(x_1, \ldots, x_k) f_{\mathbf{X}_{h_{i+1}}|\mathbf{X}_{t_{i+1}}}(\mathbf{x}_{h_{i+1}}|\mathbf{x}_{t_{i+1}})$$

is a Gaussian density too, because t_{i+1} is a subset of the variables 1 to k of the prior $f(x_1, \ldots, x_k)$ and h_i represents variables outside the prior (see Appendix J.2). ■

A sequence of pairs (h_j, t_j) satisfying these conditions is called a *construction sequence* (Shafer, 1996) for the Gaussian density of equation (10.52).

Example 10.5 *Consider the linear dynamic Gaussian system discussed in the previous section. The dynamic equations (10.32) induce conditional Gaussian densities with head* $\mathbf{X}_i$ *and tail* $\mathbf{X}_{i-1}$ *for* $i = 1, 2, \ldots$ *The measurement equations (10.33) can be represented by conditional Gaussian densities with head* $\mathbf{X}_i$ *and tail* $\mathbf{Y}_i$. *If we further assume ordinary Gaussian densities for* $\mathbf{X}_0$ *and* $\mathbf{Y}_i$ *(interpreted as "conditional" densities without tail) then a possible construction sequence is*

$$(\mathbf{X}_0, \emptyset), (\mathbf{Y}_1, \emptyset), (\mathbf{X}_1, \mathbf{Y}_1), (\mathbf{X}_1, \mathbf{X}_0), (\mathbf{Y}_2, \emptyset), (\mathbf{X}_2, \mathbf{Y}_2), (\mathbf{X}_2, \mathbf{X}_1), \ldots$$

There are other ones. Construction sequences are generally not unique.

The decomposition of a Gaussian density into a product of conditional densities is a problem of modeling. Bayesian networks may help. Subsequently, we assume a given factorization and focus on computational aspects. For local computation of marginals of such a factorized Gaussian density (10.52) we need a join tree that covers the domains $d = h \cup t$ of the head-tail pairs $(h, t) \in \mathcal{J}$ as explained in Chapter 3. Let (V, E, λ, D) be such a join tree, where D is the lattice of subsets of the index set $\{1, \ldots, m\}$ of the variables of the Gaussian density $f(x_1, \ldots, x_m)$. So we have $d \subseteq \lambda(j)$ for every conditional $(h, t) \in \mathcal{J}$ for some node $j \in V$. Assume that we want to compute the marginals of $f(x_1, \ldots, x_m)$ for all variables X_1 to X_m. For that purpose, it suffices to compute the marginals of $f(x_1, \ldots, x_m)$ for all node labels $\lambda(j)$ in the join tree. The one-variable marginals follow then by one last projection per variable. This can be achieved by the Shenoy-Shafer architecture of Section 4.1. As a preparation, we transform all the conditionals $f_{\mathbf{X}_h|\mathbf{X}_t}(\mathbf{x}_h|\mathbf{x}_t)$ into generalized Gaussian potentials $(\nu_{h,t}, \mathbf{C}_{h,t})$. To do so, we represent the conditional Gaussian density $f_{\mathbf{X}_h|\mathbf{X}_t}(\mathbf{x}_h|\mathbf{x}_t)$ with mean vector $\mathbf{z}_{h,t} + \mathbf{A}_{h,t}\mathbf{x}_t$ and concentration matrix $\mathbf{K}_{h,t}$ by a linear Gaussian system

$$\mathbf{X}_h \quad = \quad \mathbf{z}_{h,t} + \mathbf{A}_{h,t}\mathbf{X}_t + \omega_{h,t} \tag{10.53}$$

with concentration matrix $\mathbf{K}_{\omega_{h,t}} = \mathbf{K}_{h,t}$. This transforms into the standard form of a linear Gaussian system

$$\begin{bmatrix} \mathbf{I} & -\mathbf{A}_{h,t} \end{bmatrix} \begin{bmatrix} \mathbf{X}_h \\ \mathbf{X}_t \end{bmatrix} - \omega_{h,t} \quad = \quad \mathbf{z}_{h,t} \tag{10.54}$$

and then into the generalized Gaussian potential

$$\left(\begin{bmatrix} \mathbf{K}_{h,t}\mathbf{z}_{h,t} \\ -\mathbf{A}_{h,t}^T\mathbf{K}_{h,t}\mathbf{z}_{h,t} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{h,t} & -\mathbf{K}_{h,t}\mathbf{A}_{h,t} \\ -\mathbf{A}_{h,t}^T\mathbf{K}_{h,t} & \mathbf{A}_{h,t}^T\mathbf{K}_{h,t}\mathbf{z}_{h,t} \end{bmatrix} \right) \tag{10.55}$$

according to equation (10.12) in Section 10.3. We then select an arbitrary node $r \in V$ of the join tree as root node and execute the collect phase of the Shenoy-Shafer architecture with the generalized Gaussian potentials on the nodes of the join tree. At the end of the message-passing, the Gaussian potential of the marginal of $f(x_1, \ldots, x_m)$ to the root node label $\lambda(r)$ can be obtained from the root node. This corresponds to an ordinary Gaussian potential since this marginal is a Gaussian density. After the distribute phase, the marginal of $f(x_1, \ldots, x_m)$ to each node label $\lambda(i)$ can be obtained from the corresponding join tree node $i \in V$, and these marginals are all ordinary Gaussian potentials.

As long as we remain within the valuation algebra of the generalized Gaussian potentials, we have to use the Shenoy-Shafer architecture due to the absence of a division operator in this algebra. However, we may embed this valuation algebra into the larger valuation algebra of potentials $(\nu, \mathbf{C})$, where $\mathbf{C}$ is only assumed to be symmetric, but no more non-negative definite. In this algebra, division is defined and the application of local computation architectures with division such as the Lauritzen-Spiegelhalter or the HUGIN architecture become possible for factorizations of Gaussian densities into conditional Gaussian densities. For details see (Eichenberger, 2009).

An essential aspect missing so far concerns *observations*. Assume that we have observed the values of some variables and want to condition the Gaussian density $f(x_1, \ldots, x_m)$ on these observations and again compute the marginals of the conditional density to all remaining variables. This amounts to adding simple equations like $X_i = \bar{x}_i$ to the system. More generally, we may need to add general linear systems without disturbances. This gives a combination of linear Gaussian systems with ordinary linear systems of equations. We shall not treat this general case and refer to (Eichenberger, 2009) where an extension of the valuation algebra of generalized Gaussian potentials to a valuation algebra covering both ordinary linear equations and linear Gaussian systems is developed. Here, we treat only the simpler case of observations of single variables. Consider a Gaussian density $f(x_1, \ldots, x_m)$ defined by a factorization (10.52) and assume that the values of a subset of variables X_i

with $i \in o$ has been observed such that the equations $X_i = \bar{x}_i$ for $i \in o$ hold. We decompose the index set $\{1, \ldots, m\}$ into o and its complement o^c and let $\mathbf{X}_o$ and $\mathbf{X}_{o^c}$ denote the corresponding variable vectors. Then we want to compute marginals of the conditional density

$$f_{\mathbf{X}_{o^c}|\mathbf{X}_o = \bar{\mathbf{x}}_o}(\mathbf{x}_{o^c}|\bar{\mathbf{x}}_o) \quad = \quad \frac{f(\bar{\mathbf{x}}_o, \mathbf{x}_{o^c})}{f^{\downarrow o}(\bar{\mathbf{x}}_o)},$$

where $f^{\downarrow o}(\mathbf{x}_o)$ is the marginal of f with respect to o. This is clearly again a Gaussian density. In fact, it is simply the nominator $f(\mathbf{x}_o, \bar{\mathbf{x}}_{o^c})$ renormalized to one. In view of the representation of Gaussian and conditional Gaussian densities by linear Gaussian systems and their associated generalized Gaussian potentials, the density $f(x_1, \ldots, x_m)$ is represented by the totality of the linear Gaussian systems associated with the conditional Gaussian densities into which $f(x_1, \ldots, x_m)$ is factorized. So, we may simply add the observation equations $\mathbf{X}_o = \bar{\mathbf{x}}_o$ to this system. Or, we may simply replace all variables X_i with $i \in o$ by their values $\bar{x}_i$ in all these equations. This is exactly what we did in the linear dynamic Gaussian system discussed in the previous Section 10.5.

To be a little more explicit, consider a conditional $f_{\mathbf{X}_h|\mathbf{X}_t}(\mathbf{x}_h|\mathbf{x}_t)$ of the factorization (10.52) and its associated linear Gaussian system (10.54). Define $h_1 = h \cap o$, the set of observed head variables, $h_2 = h - h_1$ and similarly $t_1 = t \cap o$, the set of observed tail variables, $t_2 = t - t_1$. Decompose the matrix $\mathbf{A}_{h,t}$ according to theses subsets of h and t into

$$\mathbf{A}_{h,t} \quad = \quad \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix},$$

where $\mathbf{A}_{1,1}$ is a $h_1 \times t_1$ matrix, $\mathbf{A}_{1,2}$ is $h_1 \times t_2$, $\mathbf{A}_{2,1}$ is $h_2 \times t_1$ and $\mathbf{A}_{2,2}$ is a $h_2 \times t_2$ matrix. Decompose also the variable vector $\mathbf{X}$ accordingly, whereas ω and $\mathbf{z}$ are decomposed according to the head variables h_1 and h_2 of the conditional density. Then the system (10.54) reads as

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{A}_{1,1} & -\mathbf{A}_{1,2} \\ \mathbf{0} & \mathbf{I} & -\mathbf{A}_{2,1} & -\mathbf{A}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{h_1} \\ \mathbf{X}_{h_2} \\ \mathbf{X}_{t_1} \\ \mathbf{X}_{t_2} \end{bmatrix} + \begin{bmatrix} \omega_{h_1} \\ \omega_{h_2} \end{bmatrix} \quad = \quad \begin{bmatrix} \mathbf{z}_{h_1} \\ \mathbf{z}_{h_2} \end{bmatrix}.$$

If we introduce the observed values $\mathbf{X}_{h_1} = \bar{\mathbf{x}}_{h_1}$ and $\mathbf{X}_{t_1} = \bar{\mathbf{x}}_{t_1}$ into these equations and rearrange the equations to the standard form, we get the system

$$\begin{bmatrix} \mathbf{0} & -\mathbf{A}_{1,2} \\ \mathbf{I} & -\mathbf{A}_{2,2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{h_2} \\ \mathbf{X}_{t_2} \end{bmatrix} + \begin{bmatrix} \omega_{h_1} \\ \omega_{h_2} \end{bmatrix} \quad = \quad \begin{bmatrix} \mathbf{z}_{h_1} \\ \mathbf{z}_{h_2} \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{x}}_{h_1} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{1,1} \\ \mathbf{A}_{2,1} \end{bmatrix} \bar{\mathbf{x}}_{t_1}.$$

This permits to compute the associated generalized Gaussian potential and then to carry out local computation as above to get the marginals of the conditional Gaussian density $f(x_1, \ldots, x_m)$, given the observation $X_i = \bar{x}_i$ for $i \in o$.

To conclude this section, we emphasize again that there is an alternative approach, which considers ordinary linear equations as information in the same way as conditional Gaussian densities or the associated linear Gaussian systems and which is based on an extension of both the valuation algebra of generalized Gaussian potentials and the information algebra of linear systems (Eichenberger, 2009). This approach has the advantage that more general linear system than simple observations can be considered. Further, observation can be added as they arrive by using update versions of local computation (Schneuwly, 2007). This approach is also more in line with least squares estimation. We further mention that mixed discrete Gaussian systems have been considered in connection to local computation (Cowell *et al.*, 1999). These systems however do not fit anymore into the algebraic structure of valuation algebras.

10.7 CONCLUSION

The valuation algebra of Gaussian potentials was introduced in Chapter 1 in a pure algebraic context. In the first section of this chapter we gave meaning to this algebra. By dint of a technique called assumption-based reasoning, we showed that a Gaussian potential can be associated with an overdetermined, linear system with Gaussian disturbances, whose matrix has full column rank. Combination and projection for Gaussian potentials then mirror the operation of union of equations and variable elimination in the associated linear system. However, we then observed by studying certain time-discrete, dynamic systems that the requirement of an overdetermined system with full column rank is too strong for many interesting applications. This inspired a generalization of Gaussian potentials that correspond to conditional Gaussian distributions and that can be associated with arbitrary linear, Gaussian systems. This finally allows us to solve arbitrary linear systems with Gaussian disturbances by local computation. Two typical models of such systems and their local computation based solution were studied in the final part of this chapter. These models are Kalman filters and Gaussian Bayesian networks.

Appendix:

J.1 VALUATION ALGEBRA PROPERTIES OF HINTS

We are going to verify the axioms of a valuation algebra for the algebra $\langle\Psi, D\rangle$ of hints following (Kohlas & Monney, 2008). This supplies the missing poof of Theorem 10.3. We directly observe that the labeling (A2), projection (A3) and domain (A6) axioms follow directly from the definitions of labeling and projection. The remaining axioms are verified as follows:

(A1) *Commutative Semigroup:* Commutativity of combination follows from the equivalence of the linear systems $h_1 \oplus h_2$ and $h_2 \oplus h_1$. Therefore,

$$[h_1] \otimes [h_2] \;=\; h(h_1 \oplus h_2) \;=\; h(h_2 \oplus h_1) \;=\; [h_2] \otimes [h_1].$$

For the associative law of combination, we note that

$$([h_1] \otimes [h_2]) \otimes [h_3] \quad = \quad h(h' \oplus h_3),$$

where $h' \in h(h_1 \oplus h_2)$. But then h' is equivalent to $h_1 \oplus h_2$ and further $h' \oplus h_3$ is equivalent to $((h_1 \oplus h_2) \oplus h_3 = (h_1 \oplus (h_2 \oplus h_3))$. This in turn is equivalent to $h_1 \oplus h''$ for a $h'' \in h(h_2 \oplus h_3)$. So, finally $h' \oplus h_3$ and $h_1 \oplus h''$ are equivalent. This implies that

$$([h_1] \otimes [h_2]) \otimes [h_{3]} \;=\; h(h' \oplus h_3) \;=\; h(h_1 \oplus h'') \;=\; [h_1] \otimes ([h_2] \otimes [h_3]).$$

This proves associativity of combination.

(A4) *Transitivity:* This axiom needs a somewhat longer reflection and is the most difficult part of the proof. We first generalize the approach to variable elimination, which is used to define projection. If $h = (\mathbf{A}, \mathbf{z}, \mathbf{K})$ is a hint on s and $t \subseteq s$, we decompose the linear systems into

$$\mathbf{A}_1\mathbf{X}^{\downarrow s-t} + \mathbf{A}_2\mathbf{X}^{\downarrow t} + \omega \quad = \quad \mathbf{z}.$$

Now, if $\mathbf{A}_1$ is an $m \times |s-t|$ matrix with rank $k \leq m, |s-t|$, then there is an $m \times k$ matrix $\mathbf{B}_1$ which spans the column space $\mathcal{C}(\mathbf{A}_1)$ such that $\mathbf{A}_1 = \mathbf{B}_1\Lambda$. Further, there is an $m \times (m-k)$ matrix $\mathbf{B}_2$ such that $\mathbf{B} = [\mathbf{B}_1\ \mathbf{B}_2]$ is a regular $m \times m$ matrix. Let then $\mathbf{T} = \mathbf{B}^{-1}$ and

$$\mathbf{T} \quad = \quad \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix},$$

where $\mathbf{T}_1$ is a $k \times m$ matrix and $\mathbf{T}_2$ an $(m-k) \times m$ matrix. The original system can be transformed into the equivalent system

$$h' \;: \mathbf{TAX} + \mathbf{T}\omega \quad = \quad \mathbf{Tz}.$$

Here, $\bar{\omega} = \mathbf{T}\omega$ has concentration matrix $\mathbf{B}^T\mathbf{KB}$ or variance-covariance matrix $\mathbf{TKT}^{-1}$. In a decomposed form, the matrix $\mathbf{TA}$ is written as

$$\begin{aligned}\mathbf{TA} \quad &= \quad \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix} \quad = \quad \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} \begin{bmatrix} \mathbf{B}_1\Lambda & \mathbf{A}_2 \end{bmatrix} \\ &= \quad \begin{bmatrix} \mathbf{T}_1\mathbf{B}_1\Lambda & \mathbf{T}_1\mathbf{A}_2 \\ \mathbf{T}_2\mathbf{B}_1\Lambda & \mathbf{T}_2\mathbf{A}_2 \end{bmatrix} \quad = \quad \begin{bmatrix} \mathbf{T}_1\mathbf{A}_1 & \mathbf{T}_1\mathbf{A}_2 \\ \mathbf{0}_{m-k,k} & \mathbf{T}_2\mathbf{A}_2 \end{bmatrix}.\end{aligned}$$

and the transformed system becomes

$$\begin{aligned}\mathbf{T}_1\mathbf{A}_1\mathbf{X}^{\downarrow s-t} + \mathbf{T}_1\mathbf{A}_2\mathbf{X}^{\downarrow t} + \mathbf{T}_1\omega \quad &= \quad \mathbf{T}_1\mathbf{z}, \\ \mathbf{T}_2\mathbf{A}_2\mathbf{X}^{\downarrow t} + \mathbf{T}_2\omega \quad &= \quad \mathbf{T}_2\mathbf{z}.\end{aligned}$$

In the second part of this system,

$$\mathbf{T}_2\mathbf{A}_2\mathbf{X}^{\downarrow t} + \bar{\omega}_2 \quad = \quad \mathbf{T}_2\mathbf{z}, \tag{J.1}$$

the variables in $s-t$ have been eliminated. The stochastic vector $\bar{\omega}_2 = \mathbf{T}_2\omega$ has a Gaussian density with concentration matrix $(\mathbf{T}_2\mathbf{K}^{-1}\mathbf{T}_2^T)^{-1}$ as the marginal with respect to t of the Gaussian vector $\bar{\omega} = \mathbf{T}\omega$. Since the matrix $\mathbf{T}$ is regular and $\mathbf{A}$ has full row rank, it follows that $\mathbf{TA}$ has full row rank and therefore, the matrix $\mathbf{T}_2\mathbf{A}_2$ has full row rank too. Hence,

$$\left(\mathbf{T}_2\mathbf{A}_2,\ \mathbf{T}_2\mathbf{z},\ (\mathbf{T}_2\mathbf{K}^{-1}\mathbf{T}_2^T)^{-1}\right)$$

is a hint. It represents $[h]^{\downarrow t}$ because the solution space of (J.1) is a projection of the solution space of the original system for every disturbance ω. This is a generalization of former approaches to compute a projection of a linear Gaussian system, see Section 10.3.4.

An $(m-k) \times m$ matrix like $\mathbf{T}_2$ which has the property

1. of full row rank $m-k$ and
2. $\mathbf{T}_2\mathbf{A}_1 = \mathbf{0}_{m-k,k}$

is called a *projection matrix* for $\mathbf{A}$ to t. A matrix $\mathbf{A}$ may have different projection matrices to t, but they all lead to equivalent systems. In fact, if $\mathbf{T}$ and $\bar{\mathbf{T}}$ are projection matrices for $\mathbf{A}$ to t, then $\bar{\mathbf{T}}\mathbf{A}_1 = \mathbf{T}\mathbf{A}_1 = \mathbf{0}$, then also $\mathbf{A}_1^T\bar{\mathbf{T}}^T = \mathbf{A}_1^T\mathbf{T}^T = \mathbf{0}$, which shows that the rows of both matrices $\bar{\mathbf{T}}$ and $\mathbf{T}$ are bases of the null space of $\mathbf{A}_1^T$. This implies that there is a regular $(m-k) \times (m-k)$ matrix $\mathbf{C}$ such that $\bar{\mathbf{T}}^T = \mathbf{T}^T\mathbf{C}$. Hence, the system $\bar{\mathbf{T}}\mathbf{A}_2\mathbf{X}^{\downarrow t} + \bar{\mathbf{T}}\omega = \bar{\mathbf{T}}\mathbf{z}$ is the same as the system $\mathbf{C}^T\mathbf{T}\mathbf{A}_2\mathbf{X}^{\downarrow t} + \mathbf{C}^T\mathbf{T}\omega = \mathbf{C}^T\mathbf{T}\mathbf{z}$, which is equivalent to the system $\mathbf{T}\mathbf{A}_2\mathbf{X}^{\downarrow t} + \mathbf{T}\omega = \mathbf{T}\mathbf{z}$.

Further, elimination of the same variables in equivalent hints results in equivalent hints. Indeed, if $h_1 = (\mathbf{A}, \mathbf{z}, \mathbf{K})$ and $h_2 = (\bar{\mathbf{A}}, \bar{\mathbf{z}}, \bar{\mathbf{K}})$ are equivalent hints, then both $\mathbf{A}$ and $\bar{\mathbf{A}}$ are $m \times s$ matrices for some m and s and there is a regular $m \times m$ matrix $\mathbf{T}$ such that

$$\bar{\mathbf{A}} = \mathbf{TA}, \quad \bar{\mathbf{z}} = \mathbf{Tz}, \quad \bar{\mathbf{K}} = (\mathbf{TK}^{-1}\mathbf{T}^T)^{-1}.$$

For $t \subseteq s$, decompose the matrix of the first system for the elimination of the variables in $s-t$ into

$$\mathbf{A} \quad = \quad \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_1\Lambda & \mathbf{A}_2 \end{bmatrix},$$

where $\mathbf{A}_1$ is an $m \times k$ matrix of rank $k \leq |s-t|$ spanning the column space of the submatrix $\mathbf{A}^{\downarrow s-t}$. The matrix $\mathbf{A}_2$ has dimension $m \times t$. We decompose

$$\mathbf{T} \quad = \quad \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix}$$

such that $\mathbf{T}_1$ is a $k \times m$ and $\mathbf{T}_2$ an $(m-k) \times m$ matrix and obtain

$$\mathbf{TA} = \begin{bmatrix} \mathbf{T}_1\mathbf{A}_1 & \mathbf{T}_1\mathbf{A}_1\Lambda & \mathbf{T}_1\mathbf{A}_2 \\ \mathbf{T}_2\mathbf{A}_1 & \mathbf{T}_2\mathbf{A}_1\Lambda & \mathbf{T}_2\mathbf{A}_2 \end{bmatrix} = \bar{\mathbf{A}}.$$

The matrix $\mathbf{T}_1\mathbf{A}_1$ has rank k and is regular, because $\mathbf{T}$ is regular, hence $\mathbf{T}_1$ has rank k as well as $\mathbf{A}_1$. It follows that the matrix

$$\mathbf{P} = \begin{bmatrix} -\mathbf{T}_2\mathbf{A}_1(\mathbf{T}_1\mathbf{A}_1)^{-1} & \mathbf{I}_{m-k} \end{bmatrix},$$

is a projection matrix for $\bar{\mathbf{A}}$ to t. Indeed, we have

$$\begin{aligned} \mathbf{P}\bar{\mathbf{A}}_1 &= \mathbf{PTA}_1 = \begin{bmatrix} -\mathbf{T}_2\mathbf{A}_1(\mathbf{T}_1\mathbf{A}_1)^{-1} & \mathbf{I}_{m-k} \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 \\ \mathbf{T}_2 \end{bmatrix} \mathbf{A}_1 \\ &= -\mathbf{T}_2\mathbf{A}_1(\mathbf{T}_1\mathbf{A}_1)^{-1}\mathbf{T}_1\mathbf{A}_1 + \mathbf{T}_2\mathbf{A}_1 = \mathbf{0}. \end{aligned}$$

The system reduced with projection matrix $\mathbf{P}$ is $\mathbf{P}\bar{\mathbf{A}}\mathbf{X} + \mathbf{P}\bar{\omega} = \mathbf{P}\bar{\mathbf{z}}$ or also $\mathbf{PTAX} + \mathbf{PT}\omega = \mathbf{PTz}$. But the last system is the reduced system of $\mathbf{AX} + \omega = \mathbf{z}$ with projection matrix $\mathbf{PT}$. This proves the equivalence of the reduced systems, hence of projected hints h_1 and h_2, stated in the following lemma. We remark for later reference in the proof of the combination axiom that these considerations can be generalized to arbitrary linear Gaussian systems. We have proved the following lemma:

Lemma J.1 *Projections of equivalent systems are equivalent.*

These results about projection of hints with the aid of projection matrices finally permit to prove the transitivity axiom of projection. Let $h = (\mathbf{A}, \mathbf{z}, \mathbf{K})$ be a hint with label x and $s \subseteq t \subseteq x$. Select a projection matrix P_1 for $\mathbf{A}$ to t. Decompose the matrix $\mathbf{A}$ into $\mathbf{A} = [\mathbf{A}^{\downarrow x-t}\ \mathbf{A}^{\downarrow t-s}\ \mathbf{A}^{\downarrow s}]$. We then obtain for the hint h projected to t,

$$[h]^{\downarrow t} = \left[\left([\mathbf{P}_1\mathbf{A}^{\downarrow t-s}\ \mathbf{P}_1\mathbf{A}^{\downarrow s}],\ \mathbf{P}_1\mathbf{z},\ (\mathbf{P}_1\mathbf{K}^{-1}\mathbf{P}_1^T)^{-1}\right)\right].$$

According to the discussion above, this unambiguously defines the projection of hint $[h]$. Further, let P_2 be a projection matrix for $\mathbf{P}_1\mathbf{A}$ to s, such that

$$([h]^{\downarrow t})^{\downarrow s} = \left[\left(\mathbf{P}_2\mathbf{P}_1\mathbf{A}^{\downarrow s}, \mathbf{P}_2\mathbf{P}_1\mathbf{z}, (\mathbf{P}_2\mathbf{P}_1\mathbf{K}^{-1}\mathbf{P}_1^T\mathbf{P}_2^T)^{-1}\right)\right]. \quad \text{(J.2)}$$

The matrix $\mathbf{P}_1\mathbf{P}_2$ is indeed a projection matrix, since $\mathbf{P}_1\mathbf{A}^{\downarrow x-t} = \mathbf{0}$ and $\mathbf{P}_2\mathbf{P}_1\mathbf{A}^{\downarrow t-s} = \mathbf{0}$, hence

$$\mathbf{P}_2\mathbf{P}_1 \begin{bmatrix} \mathbf{A}^{\downarrow x-t} & \mathbf{A}^{\downarrow t-s} \end{bmatrix} = \mathbf{P}_2 \begin{bmatrix} \mathbf{0} & \mathbf{P}_2\mathbf{A}^{\downarrow t-s} \end{bmatrix} = \mathbf{0}.$$

Both matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ have full row rank which implies that $\mathbf{P}_2\mathbf{P}_1$ has full row rank too. More precisely, $\mathbf{P}_1$ is an $(m - rank(\mathbf{A}^{\downarrow x-t})) \times m$ matrix and

$\mathbf{P}_2$ an $(m - rank(\mathbf{A}^{\downarrow x-t}) - rank(\mathbf{P}_1\mathbf{A}^{\downarrow t-s})) \times (m - rank(\mathbf{A}^{\downarrow x-t}))$ matrix. It holds that

$$rank(\mathbf{A}^{\downarrow x-s}) \quad = \quad rank(\mathbf{A}^{\downarrow x-t}) + rank(\mathbf{P}_1\mathbf{A}^{\downarrow t-s}).$$

Thus, $rank(\mathbf{P}_2) = rank(\mathbf{P}_2\mathbf{P}_1)$. But this implies that $\mathbf{P}_2\mathbf{P}_1$ is a projection matrix for $\mathbf{A}$ to s. Therefore, $\mathbf{P}_2\mathbf{P}_1$ is also the projection of $[h]$ to s, hence

$$([h]^{\downarrow t})^{\downarrow s} \quad = \quad [h]^{\downarrow s}$$

and transitivity holds.

(A5) *Combination:* The results about projection matrices also help to verify the combination axiom. Let $h_1 = (\mathbf{A}_1, \mathbf{z}_1, \mathbf{K}_1)$ and $h_2 = (\mathbf{A}_2, \mathbf{z}_2, \mathbf{K}_2)$ be two hints with $\mathbf{A}_1$ an $m_1 \times x$ matrix and $\mathbf{A}_2$ an $m_2 \times y$ matrix. The vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ have dimension m_1 and m_2 respectively, whereas $\mathbf{K}_1$ and $\mathbf{K}_2$ are $m_1 \times m_1$ and $m_2 \times m_2$ matrices. Accordingly, we have $d([h_1]) = x$ and $d([h_2]) = y$ and assume $x \subseteq z \subseteq x \cup y$. Select a projection matrix $\mathbf{P}$ for $\mathbf{A}_2$ to $y \cap z$ such that $\mathbf{P}\mathbf{A}_2^{\downarrow y-z} = \mathbf{0}$ with rank $rank(\mathbf{P}) = m_2 - rank(\mathbf{A}_2^{\downarrow y-z})$. We then have

$$[h_2]^{\downarrow y\cap z} \quad = \quad \left[\left(\mathbf{P}\mathbf{A}_2^{\downarrow y\cap z},\ \mathbf{P}\mathbf{z}_2,\ (\mathbf{P}\mathbf{K}_2^{-1}\mathbf{P}^T)^{-1}\right)\right].$$

The combined linear system $h_1 \oplus h_2^{\downarrow y\cap z}$ of the two hints h_1 and $h_2^{\downarrow y\cap z}$ is

$$\left(\begin{bmatrix} \mathbf{A}_1^{\downarrow x-y} & \mathbf{A}_1^{\downarrow x\cap y} & \mathbf{0} \\ \mathbf{0} & (\mathbf{P}\mathbf{A}_2)^{\downarrow x\cap y} & (\mathbf{P}\mathbf{A}_2)^{\downarrow z-x} \end{bmatrix}, \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{P}\mathbf{z}_2 \end{bmatrix},\right.$$
$$\left.\begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0} & (\mathbf{P}\mathbf{K}_2^{-1}\mathbf{P}^T)^{-1} \end{bmatrix}\right).$$

On the other hand, we define the matrix

$$\bar{\mathbf{P}} \quad = \quad \begin{bmatrix} \mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \mathbf{P} \end{bmatrix},$$

and it follows that

$$\begin{aligned} \bar{\mathbf{P}}(\mathbf{A}_1 \oplus \mathbf{A}_2) \quad &= \quad \bar{\mathbf{P}} \begin{bmatrix} \mathbf{A}_1^{\downarrow x-y} & \mathbf{A}_1^{\downarrow x\cap y} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{\downarrow x\cap y} & \mathbf{A}_2^{\downarrow z-x} & \mathbf{A}_2^{\downarrow y-z} \end{bmatrix} \\ &= \quad \begin{bmatrix} \mathbf{A}_1^{\downarrow x-y} & \mathbf{A}_1^{\downarrow x\cap y} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}\mathbf{A}_2^{\downarrow x\cap y} & \mathbf{P}\mathbf{A}_2^{\downarrow z-x} & \mathbf{0} \end{bmatrix}. \end{aligned}$$

We note that for the rank of $\bar{\mathbf{P}}$ we have

$$rank(\bar{\mathbf{P}}) \quad = \quad m_1 + rank(\mathbf{P}) = m_1 + (m_2 - rank(\mathbf{A}_2^{\downarrow y-z}))$$

and also $rank(h_1 \oplus h_2^{\downarrow y\cap z}) = rank(\mathbf{A}_2^{\downarrow y-z})$. Therefore, $\bar{\mathbf{P}}$ is a projection matrix for $h_1 \oplus h_2$ to z. Finally, this shows that

$$[(h_1 \oplus h_2)^{\downarrow z}] \quad = \quad [h_1 \oplus h_2^{\downarrow y\cap z}] \quad = \quad [h_1] \otimes [h_2]^{\downarrow y\cap z}$$

but it also holds by definition of combination and projection that

$$[(h_1 \oplus h_2)]^{\downarrow z} = [(h_1 \oplus h_2)^{\downarrow z}] = ([h_1] \otimes [h_2])^{\downarrow z} = [h_1] \otimes [h_2]^{\downarrow y \cap z}.$$

This proves the combination axiom.

J.2 GAUSSIAN DENSITIES

Let $\mathbf{x} \in \mathbb{R}^n$ be an n-vector, μ an n-vector and $\mathbf{K}$ a symmetric, positive definite $n \times n$ matrix, then the function

$$f(\mathbf{x}) = \sqrt{\frac{|\det(\mathbf{K})|}{(2\pi)^n}}\, e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{K}(\mathbf{x}-\mu)},$$

is the *Gaussian density* with parameters μ and $\mathbf{K}$. The matrix $\mathbf{K}$ is called the *concentration matrix* of the density and the square root of the fraction in front of the exponential is the normalization factor, which ensures that the integral over the Gaussian density equals 1. The vector μ is the *expected value* or *mean* of the density and $\mathbf{\Sigma} = \mathbf{K}^{-1}$ is the *variance-covariance matrix* of a random n-vector $\mathbf{X}$ with the Gaussian density above. The matrix $\mathbf{\Sigma}$ is also symmetric and positive definite.

Let $\mathbf{Y} = \mathbf{c} + \mathbf{BX}$ be an *affine transformation* of the random vector $\mathbf{X}$ having a Gaussian density with expected value $\mu_{\mathbf{X}}$ and variance-covariance matrix $\mathbf{\Sigma}_{\mathbf{X}}$, where $\mathbf{c}$ is an m-vector and $\mathbf{B}$ an $m \times n$ matrix. Then, $\mathbf{Y}$ still has a Gaussian distribution with expected value $\mathbf{c} + \mathbf{B}\mu$ and variance-covariance matrix $\mathbf{B\Sigma B}^T$. In the particular case that $m = n$ and the matrix $\mathbf{B}$ is regular with inverse $\mathbf{T}$ the Gaussian density of the random vector $\mathbf{Y}$ has the expected value

$$\mu_{\mathbf{Y}} = \mathbf{c} + \mathbf{B}\mu_{\mathbf{X}}$$

and concentration matrix

$$\mathbf{K}_{\mathbf{Y}} = (\mathbf{B\Sigma_X B}^T)^{-1} = \mathbf{T}^T\mathbf{K_X T}.$$

Let s be a subset of $\{1, \ldots, n\}$. The marginal of the Gaussian density with respect to s is still a Gaussian density with expected value $\mu^{\downarrow s}$ and variance-covariance matrix $\mathbf{\Sigma}^{\downarrow s,s}$. Its concentration matrix is

$$\left((\mathbf{K}^{-1})^{\downarrow s,s}\right)^{-1}.$$

if $\mathbf{K} = \mathbf{\Sigma}^{-1}$. If $\mathbf{K}$ is decomposed according to the sets s and $t = \{1, \ldots, n\} - s$,

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}^{\downarrow s,s} & \mathbf{K}^{\downarrow s,t} \\ \mathbf{K}^{\downarrow t,s} & \mathbf{K}^{\downarrow t,t} \end{bmatrix},$$

the concentration matrix of the marginal can also be written as

$$\left((\mathbf{K}^{-1})^{\downarrow s,s}\right)^{-1} = \mathbf{K}^{\downarrow s,s} - \mathbf{K}^{\downarrow s,t}(\mathbf{K}^{\downarrow t,t})^{-1}\mathbf{K}^{\downarrow t,s}.$$

The conditional distribution of a random vector $\mathbf{X}$ with Gaussian distribution is still a Gaussian distribution. Let again s be a subset of $\{1,\ldots,n\}$, $t = \{1,\ldots,n\} - s$ and the concentration matrix $\mathbf{K}$ be decomposed according to s and t as above. Then, the *conditional Gaussian density* of the random vector $\mathbf{X}^{\downarrow s}$ given $\mathbf{X}^{\downarrow t} = \mathbf{x}^{\downarrow t}$ has expected value

$$\mu^{\downarrow s} - (\mathbf{K}^{\downarrow s,s})^{-1}\mathbf{K}^{\downarrow s,t}(\mathbf{x}^{\downarrow t} - \mu^{\downarrow t}) \quad = \quad \mu^{\downarrow s} + \mathbf{\Sigma}^{\downarrow s,t}(\mathbf{\Sigma}^{\downarrow t,t})^{-1}(\mathbf{x}^{\downarrow t} - \mu^{\downarrow t})$$

and concentration matrix

$$\mathbf{K}^{\downarrow s,s} \quad = \quad \left(\mathbf{\Sigma}^{\downarrow s,s} - \mathbf{\Sigma}^{\downarrow s,t}(\mathbf{\Sigma}^{\downarrow t,t})^{-1}\mathbf{\Sigma}^{\downarrow t,s}\right)^{-1}.$$

If f denotes a Gaussian density over variables $\mathbf{X} \in \mathbb{R}^n$, $f^{\downarrow t}$ the marginal density with respect to the subset $t \subseteq \{1,\ldots,n\}$ and $f_{\mathbf{X}^{\downarrow s}|\mathbf{X}^{\downarrow t}}$ the conditional density of $\mathbf{X}^{\downarrow s}$ given $\mathbf{X}^{\downarrow t}$, then

$$f(\mathbf{x}) \quad = \quad f_{\mathbf{X}^{\downarrow s}|\mathbf{X}^{\downarrow t}}(\mathbf{x}^{\downarrow s}|\mathbf{x}^{\downarrow t})f^{\downarrow t}(\mathbf{x}^{\downarrow t}).$$

More generally, if f is a Gaussian density over a set of variables s and g a conditional Gaussian density over variables in a set h given variables in a set t, such that $t \subseteq s$ and $h \cap s = \emptyset$, we claim that for $\mathbf{x} \in \mathbb{R}^s$,

$$f(\mathbf{x})g(\mathbf{x}^{\downarrow h}|\mathbf{x}^{\downarrow t})$$

is a Gaussian density. To verify this, consider the generalized Gaussian potentials $(\nu_1, \mathbf{K}_1)$ and $(\nu_2, \mathbf{K}_2)$ of f and g. Then, the product above is associated with the generalized Gaussian potential $(\nu_1, \mathbf{K}_1) \otimes (\nu_2, \mathbf{K}_2) = (\nu, \mathbf{K})$ and in particular

$$\mathbf{K} \quad = \quad \mathbf{K}_1^{\uparrow s \cup h} + \mathbf{K}_2^{\uparrow s \cup h}.$$

Now, assume the conditional Gaussian density g is associated with a system of linear equations like (10.53) having the concentration matrix $\mathbf{K}_{h,t}$. Then, the associated concentration matrix is given according to (10.55) by

$$\mathbf{K}_2 \quad = \quad \begin{bmatrix} \mathbf{K}_{h,t} & -\mathbf{K}_{h,t}\mathbf{A}_{h,t} \\ -\mathbf{A}_{h,t}^T\mathbf{K}_{h,t} & \mathbf{A}_{h,t}^T\mathbf{K}_{h,t}\mathbf{A}_{h,t} \end{bmatrix}$$

From this we deduce that

$$\mathbf{K} \quad = \quad \begin{bmatrix} \mathbf{K}_{h,t} & -\mathbf{K}_{h,t}\mathbf{A}_{h,t} & \mathbf{0} \\ -\mathbf{A}_{h,t}^T\mathbf{K}_{h,t} & \mathbf{K}_1^{\downarrow t\ t} + \mathbf{A}_{h,t}^T\mathbf{K}_{h,t}\mathbf{A}_{h,t} & \mathbf{K}_1^{\downarrow t\ s-t} \\ \mathbf{0} & \mathbf{K}_1^{\downarrow s-t,t} & \mathbf{K}_1^{\downarrow s-t,s-t} \end{bmatrix}$$

This matrix is positive definite, since $\mathbf{K}_1$ and $\mathbf{K}_{h,t}$ are both positive definite. Hence, the generalized potential $(\nu, \mathbf{K})$ associated with the product $f(\mathbf{x})g(\mathbf{x}^{\downarrow h}|\mathbf{x}^{\downarrow t})$ is indeed an ordinary Gaussian potential and therefore this product is a Gaussian density.

PROBLEM SETS AND EXERCISES

J.1 ★ Develop the prediction for $k+1, k+2, \ldots$ in the Kalman filter model, given observations up to time k, by continuing the collect algorithm beyond the node for time k in the join tree.

J.2 ★★ Develop a general smoothing algorithm for nodes $i = 1, \ldots, k-1$ in the Kalman filter model given observations up to time k. Take node i as the root of the join tree for the collect algorithm and show that the smoothing solution at time i is given as the combination of potentials for the filtering solution at time i and a backward filtering from time k backwards to i.

J.3 ★★ Develop the backward filtering found in the preceding exercise in terms of generalized Gaussian potentials.

J.4 ★★ Develop the backward filtering in terms of mean vectors and variance-covariance matrices assuming that the potential ψ_k at time k is ordinary Gaussian and that the matrices $\mathbf{A}_k$ are regular. Solve the smoothing problem in this case.

J.5 ★★ Study the updating of the smoothing solution if a new measurement at time $k+1$ is added.

J.6 ★★ Solve the filtering, prediction and smoothing problem for the discrete state hidden Markov chain model.

J.7 ★★★ Solve the filtering, prediction and smoothing problem for a general semiring valuation algebra.

J.8 ★★★ In this chapter, we studied linear systems with Gaussian disturbances and showed that the corresponding operations of combination and variable elimination are mirrored by the operations for Gaussian potentials in Instance 1.6. Parts of this theory for linear systems with disturbances can be developed without the assumption that the disturbances are Gaussian (Kohlas & Monney, 2008). Abstract the theory of this chapter in such a way that other distributions can be assumed for the disturbances. Does this induce valuation algebras based on other classes of densities? This question is closely related to Exercise A.7 of Chapter 1.

References

Aho, A., Hopcroft, J.E., & Ullman, J. 1974. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company.

Aji, S.M. 1999. *Graphical Models and Iterative Decoding.* Ph.D. thesis, California Institute of Technology.

Aji, S.M., & McEliece, R.J. 2000. The Generalized Distributive Law. *IEEE Trans. on Information Theory*, **46**(2), 325–343.

Allen, D., & Darwiche, A. 2002. *On the Optimality of the min-fill Heuristic.* Tech. rept. University of California, Los Angeles.

Allen, D., & Darwiche, A. 2003. Optimal Time-Space Tradeoff in Probabilistic Inference. *Pages 969–975 of: IJCAI'03: Proc. of the 18th Int. Joint Conference on Artif. Intell.* Morgan Kaufmann Publishers, Inc.

Almond, R., & Kong, A. 1991. Optimality Issues in Constructing a Markov Tree from Graphical Models. *Research Report A-3. Department of Statistics, Harvard University.*

Apt, K.R. 2003. *Principles of Constraint Programming.* Cambridge University Press.

Arnborg, S. 1985. Efficient Algorithms for Combinatorial Problems with Bounded Decomposability – A Survey. *BIT*, **25**(1), 2–23.

Arnborg, S., Corneil, D., & Proskurowski, A. 1987. Complexity of Finding Embeddings in a k-Tree. *SIAM J. of Algebraic and Discrete Methods*, **8**, 277–284.

Bacchus, F., & Adam, G. 1995. Graphical Models for Preference and Utility. *Pages 3–10 of: UAI'95: Proc. of the 11th Conference on Uncertainty in Artif. Intell.* Morgan Kaufmann Publishers, Inc.

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas

Bacchus, F., Dalmao, S., & Pitassi, T. 2003. Value Elimination: Bayesian Interence via Backtracking Search. *Pages 20–28 of: UAI'03: Proc. of the 19th Conference on Uncertainty in Artif. Intell.*

Backhouse, R.C., & Carré, B.A. 1975. Regular Algebra Applied to Path-Finding Problems. *J. of the Institute for Mathematics and its Applications*, **15**, 161–186.

Barwise, J., & Seligman, J. 1997. *The Logic of Distributed Systems.* Cambridge University Press.

Beauchamp, K.G. 1984. *Applications of Walsh and related Functions.* Academic Press, Inc.

Beeri, C., Fagin, R., Maier, D., & Yannakakis, M. 1983. On the Desirability of Acyclic Database Schemes. *J. of the ACM*, **30**(3), 479–513.

Bellman, R. 1958. On a Routing Problem. *Quarterly of Applied Mathematics*, **16**(1), 87–90.

Bellman, R.E. 1957. *Dynamic Programming.* Princeton University Press.

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

Bétréma, J. 1982. Topologies sur des Espaces Ordonnés. *Informatique Théorique et Applications*, **16**(2),165–182.

Biacino, L. 2007. Fuzzy Subsethood and Belief Functions of Fuzzy Events. *Fuzzy Sets and Systems*, **158**(1), 38–49.

Bistarelli, S., & Rossi, F. 2008. Semiring-Based Soft Constraints. *Pages 155–173 of:* Degano, P., Nicola, R. De, & Meseguer, J. (eds), *Concurrency, Graphs and Models: Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday.* Springer-Verlag.

Bistarelli, S., Montanari, U., & Rossi, F. 1997. Semiring-Based Constraint Satisfaction and Optimization. *J. of the ACM*, **44**(2), 201–236.

Bistarelli, S., Montanari, U., Rossi, F., Verfaillie, G., & Fargier, H. 1999. Semiring-based CSPs and Valued CSPs: Frameworks, Properties and Comparison. *Constraints*, **4**(3).

Bistarelli, S., Frühwirth, T., Marte, M., & Rossi, F. 2002. Soft Constraint Propagation and Solving in Constraint Handling Rules. *Pages 1–5 of: Proc. of the ACM Symposium on Applied Computing.*

Blau, H., Kyburg, J.R., & Henry, E. 1994. *Ploxoma: Testbed for Uncertain Inference.* Tech. rept. University of Rochester.

Bodlaender, H. 1993. A Linear Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *Pages 226–234 of: STOC'93: Proc. of the 25th Annual ACM Symposium on Theory of Computing.* ACM Press.

Bodlaender, H. 1998. Treewidth: Algorithmic Techniques and Results. *Pages 19–36 of: Mathematical Foundations of Computer Science.* Springer-Verlag.

Broder, A.Z., & Mayr, E.W. 1997. Counting Minimum Weight Spanning Trees. *J. of Algorithms*, **24**(1), 171–176.

Cano, A., & Moral, S. 1995. Heuristic Algorithms for the Triangulation of Graphs. *Pages 166–171 of:* Bouchon-Meunier, R.R. Yager, & Zadeh, L.A. (eds), *Proc. of the 5th IPMU Conference.* Springer-Verlag.

Chang, C.L., & Lee, R.C.T. 1973. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, Inc.

Chaudhuri, S., & Zaroliagis, C.D. 1997. Shortest Paths in Digraphs of Small Treewidth. Part I: Sequential Algorithms. *Algorithmica*, **27**, 212–226.

Chvátal, V. 1983. *Linear Programming.* W.H. Freeman.

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

Conway, J.H. 1971. *Regular Algebra and Finite Machines.* Chapman and Hall Mathematics Series. Chapman and Hall.

Cooley, J.W., & Tukey, J.W. 1965. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, **19**(90), 297–301.

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

Croisot, R. 1953. Demi-Groupes Inversifs et Demi-Groupes Réunions de Demi-Groupes Simples. *Ann. Sci. Ecole norm. Sup.*, **79**(3), 361–379.

Darwiche, A. 2001. Decomposable Negation Normal Form. *J. of the ACM*, **48**(4), 608–647.

Darwiche, A., & Marquis, P. 2001. A Perspective on Knowledge Compilation. *Pages 175–182 of: IJCAI'01: Proc. of the 17th Int. Joint Conference on Artif. Intell.* San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc.

Darwiche, A., & Marquis, P. 2002. A Knowledge Compilation Map. *J. of Artif. Intell. Research*, **17**, 229–264.

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

de Kleer, J., Brown, J., & Seely, J. 1986. Theories of Causal Ordering. *Artif. Intell.*, **29**(1), 33–61.

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

Dechter, R. 2003. *Constraint Processing.* Morgan Kaufmann Publishers, Inc.

Dechter, R. 2006. Tractable Structures for Constraint Satisfaction Problems. *In:* Rossi, F., van Beek, P., & Walsh, T. (eds), *Handbook of Constraint Programming.* Elsevier.

Dechter, R., & Mateescu, R. 2007. AND/OR Search Spaces for Graphical Models. *Artif. Intell.*, **171**(2-3), 73–106.

Dechter, R., & Pearl, J. 1987. Network-Based Heuristics for Constraint-Satisfaction Problems. *Artif. Intell.*, **34**(1), 1–38.

Dechter, R., & Pearl, J. 1989. Tree Clustering for Constraint Networks. *Artif. Intell.*, **38**(3), 353–366.

Dechter, R., & Rish, R. 2003. Mini-Buckets: A General Scheme for Bounded Inference. *J. of the ACM*, **50**(2), 107–153.

Dempster, A.P. 1968. A Generalization of Bayesian Inference. *J. Royal Stat. Soc. B*, **30**, 205–247.

Dempster, A.P. 1990a. Construction and Local Computation Aspects of Network Belief Functions. *In:* Smith, J.Q., & Oliver, R.M. (eds), *Influence Diagrams, Belief Nets and Decision Analysis*. John Wiley & Sons, Inc.

Dempster, A.P. 1990b. *Normal Belief Functions and the Kalman Filter*. Tech. rept. Harvard University, USA.

Dijkstra, E.W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, **1**, 269–271.

Downey, R., & Fellows, M. 1999. *Parameterized Complexity*. Springer-Verlag.

Eichenberger, Ch. 2009. *Algebras of Gaussian Linear Information*. Ph.D. thesis, Dept. of Computer Science, University of Fribourg.

Fargier, H., & Marquis, P. 2007. On Valued Negation Normal Form Formulas. *Pages 360–365 of: IJCAI'07: Proc. of the 20th Int. Joint Conference on Artif. Intell.*

Fargier, H., & Vilarem, M. 2004. Compiling CSPs into Tree-Driven Automata for Interactive Solving. *Constraints*, **9**(4), 263–287.

Fishburn, P.C. 1974. Lexicographic Orders, Utilities and Decision Rules: A Survey. *Management Science*, **20**, 1442–1471.

Fisher, R.A. 1935. The Fiducial Argument in Statistical Inference. *Ann. Eugen.*, **9**, 391–398.

Fisher, R.A. 1950. *Contributions to Mathematical Statistics*. John Wiley & Sons, Inc.

Floyd, R.W. 1962. Algorithm 97: Shortest Path. *Comm. ACM*, **5**, 345.

Ford, L.R. 1956. *Network Flow Theory*. Tech. rept. The RAND Corperation, Santa Moncia, California.

Forsythe, G., & Moler, C.B. 1967. *Computer Solution of Linear Algebra Systems*. Prentice Hall.

Gallager, R.C. 1963. *Low Density Parity Check Codes*. MIT Press.

Garey, M.R., & Johnson, D.S. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W.H. Freeman.

Golan, J. S. 1999. *Semirings and their Applications*. Kluwer, Dordrecht.

Golan, J.S. 2003. *Semirings and Affine Equations over them: Theory and Applications*. Kluwer, Dordrecht.

Gondran, M., & Minoux, M. 2008. *Graphs, Dioids and Semirings: New Models and Algorithms*. Operations Research Computer Science Interfaces Series. Springer-Verlag.

Gonzalez, R.C., & Woods, R.E. 2006. *Digital Image Processing*. 3 edn. Prentice Hall.

Gordon, J., & Shortliffe, E.H. 1985. A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space. *Artif. Intell.*, **26**(3), 323–357.

Grabisch, M. 2009. Belief Functions on Lattices. *Int. J. Intell. Syst.*, **24**(1), 76–95.

Grätzer, G. 1978. *General Lattice Theory*. Pure and Applied Mathematics, vol. 75. Academic Press, Inc.

Green, J.A. 1951. On the Structure of Semigroups. *Ann. of Math.*, **54**, 163–172.

Haenni, R. 2004. Ordered Valuation Algebras: A Generic Framework for Approximating Inference. *Int. J. Approx. Reasoning*, **37**(1), 1–41.

Haenni, R., & Lehmann, N. 1999. *Efficient Hypertree Construction.* Tech. rept. 99-3. Department of Informatics, University of Fribourg.

Haenni, R., Kohlas, J., & Lehmann, N. 2000. Probabilistic Argumentation Systems. *Pages 221–287 of:* Kohlas, J., & Moral, S. (eds), *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 5: Algorithms for Uncertainty and Defeasible Reasoning.* Kluwer, Dordrecht.

Harville, D.A. 1997. *Matrix Algebra from a Statistician's Perspective.* Springer-Verlag.

Hewitt, E., & Zuckerman, H.S. 1956. The l_1–Algebra of a Commutative Semigroup. *Amer. Math. Soc.*, **83**, 70–97.

Hopkins, M., & Darwiche, A. 2002. A Practical Relaxation of Constant-Factor Treewidth Approximation Algorithms. *In: Proc. of the 1st European Workshop on Probabilistic Graphical Models.*

Inoue, K. 1992. Linear Resolution for Consequence Finding. *Artif. Intell.*, **56**(2-3), 301–353.

Jensen, F.V. 1988. *Junction Trees and Decomposable Hypergraphs.* Tech. rept. Aalborg University, Denmark.

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

Jonczy, J. 2009. *Generic Frameworks for the Analysis of Dependable Systems: Algebraic Path Problems, Reliability, and Diagnostics.* Ph.D. thesis, University of Berne, Switzerland.

Kalman, R.E. 1960. A new Approach to Linear Filtering and Predictive Problems. *Trans. of the ASME – J. of Basic Engineering*, **82**, 34–45.

Kanal, L., & Kumar, V. (eds). 1988. *Search in Artificial Intelligence.* Springer-Verlag.

Kask, K., Dechter, R., Larrosa, J., & Fabio, G. 2001. Bucket-Tree Elimination for Automated Reasoning. *Artif. Intell.*, **125**, 91–131.

Kemeny, J.G., Mirkil, H., Snell, J.L., & Thompson, G.L. 1960. *Finite Mathematical Structures.* Prentice Hall.

Kleene, S. 1956. *Representation of Events in Nerve Nets and Finite Automata.* Princeton University Press.

Kohlas, J. 1991. Describing Uncertainty in Dynamical Systems by Uncertain Restrictions. *Pages 210–223 of:* Masi, G.B. Di, Gombani, A., & Kurzhansky, A.B. (eds), *Modeling, Estimation and Control of Systems with Uncertainty.* Birkhäuser, Boston.

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

Kohlas, J., & Monney, P.-A. 2008. *Statistical Information. Assumption-Based Statistical Inference.* Sigma Series in Stochastics, vol. 3. Heldermann.

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.

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

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

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

Kozen, D. 1990. On Kleene Algebras and Closed Semirings. *Pages 26–47 of: Lecture Notes in Computer Science.* Springer-Verlag.

Kozen, D. 1994. A Completeness Theorem for Kleene Algebras and the Algebra of regular Events. *Information and Computing*, **110**(2), 366–390.

Kozlov, A.V., & Singh, J.P. 1994. A Parallel Lauritzen-Spiegelhalter Algorithm for Probabilistic Inference. *Pages 320–329 of: Supercomputing '94: Proc. of the 1994 ACM/IEEE Conference on Supercomputing.* ACM Press.

Kozlov, A.V., & Singh, J.P. 1996. Parallel Implementations of Probabilistic Inference. *Computer*, **29**(12), 33–40.

Kschischang, F.R., Frey, B.J., & Loeliger, H.A. 2001. Factor Graphs and the Sum-Product Algorithm. *IEEE Trans. on Information Theory*, **47**(2), 498–519.

Langel, J. 2010. *Logic and Information: A Unifying Approach to Semantic Information Theory.* Ph.D. thesis, Department of Informatics, University of Fribourg.

Larrosa, J., & Dechter, R. 2003. Boosting Search with Variable Elimination in Constraint Optimization and Constraint Satisfaction Problems. *Constraints*, **8**(3), 303–326.

Larrosa, J., & Morancho, E. 2003. Solving 'Still Life' with Soft Constraints and Bucket Elimination. *Pages 466–479 of: CP'03: Proc. of the 9th Int. Conference on Principles and Practice of Constraint Programming.*

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

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

Lehmann, D.J. 1976. *Algebraic Structures for Transitive Closure.* Tech. rept. Department of Computer Science, University of Warwick.

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

Lepar, V., & Shenoy, P.P. 1998. A Comparison of Lauritzen-Spiegelhalter, Hugin and Shenoy-Shafer Architectures for Computing Marginals of Probability Distributions. *Pages 328–337 of:* Cooper, G., & Moral, S. (eds), *UAI'98: Proc. of the 14th Conference on Uncertainty in Artif. Intell.* Morgan Kaufmann Publishers, Inc.

MacKay, D.J.C. 2003. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press.

Maier, D. 1983. *The Theory of Relational Databases.* Pitman.

Marinescu, R., & Dechter, R. 2009a. AND/OR Branch-and-Bound Search for Combinatorial Optimization in Graphical Models. *Artif. Intell.*, **173**(16-17), 1457–1491.

Marinescu, R., & Dechter, R. 2009b. Memory Intensive AND/OR Search for Combinatorial Optimization in Graphical Models. *Artif. Intell.*, **173**(16–17), 1492–1524.

Menger, K. 1942. Statistical Metrics. *Proc. of the National Academy of Sciences of the United States of America*, **28**, 535–537.

Meyn, S.P., & Tweedie, R.L. 1993. *Markov Chains and Stochastic Stability*. Springer-Verlag.

Mitten, L.G. 1964. Composition Principles for the Synthesis of Optimal Multi-Stage Processes. *Operations Research*, **12**.

Mohri, M. 2002. Semiring Frameworks and Algorithms for shortest-distance Problems. *J. Autom. Lang. Comb.*, **7**(3), 321–350.

Monney, P.-A. 2003. *A Mathematical Theory of Arguments for Statistical Evidence*. Contributions to Statistics. Physica-Verlag.

Namasivayam, V.N., & Prasanna, V.K. 2006. Scalable Parallel Implementation of Exact Inference in Bayesian Networks. *Pages 143–150 of: ICPADS '06: Proc. of the 12th Int. Conference on Parallel and Distributed Systems*, vol. 1. Washington, DC, USA: IEEE Computer Society.

Nicholson, R., Bridge, D., & Wilson, N. 2006. Decision Diagrams: Fast and Flexible Support for Case Retrieval and Recommendation. *Pages 136–150 of: Proc. of the 8th European Conference on Case-Based Reasoning*. LNAI 4106. Springer-Verlag.

Nilsson, N.J. 1982. *Principles of Artificial Intelligence*. Springer-Verlag.

Norman, N.M., & Pollard, S. 1996. *Closure Spaces and Logic*. Springer-Verlag.

Norris, J.R. 1998. *Markov Chains (Cambridge Series in Statistical and Probabilistic Mathematics)*. Cambridge University Press.

Paskin, K., Guestrin, C., & McFadden, J. 2005. A Robust Architecture for Distributed Inference in Sensor Networks. *In: IPSN'05: Proc. of the 4th Int. Symposium on Information Processing in Sensor Networks*.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Publishing Company.

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

Pichon, F., & Denoeux, T. 2008. T-Norm and Uninorm-Based Combination of Belief Functions. *In: NAFIPS'08: Annual Meeting of the North American Fuzzy Information Processing Society*.

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

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

Pouly, M., & Kohlas, J. 2005. *Minimizing Communication Costs of Distributed Local Computation*. Tech. rept. 05-20. Department of Informatics, University of Fribourg.

Pralet, C., Verfaillie, G., & Schiex, T. 2007. An Algebraic Graphical Model for Decision with Uncertainties, Feasibilities and Utilities. *J. of Artif. Intell. Research*, **29**, 421–489.

Radhakrishnan, V., Hunt, H.B., & Stearns, R.E. 1992. Efficient Algorithms for Solving Systems of Linear Equations and Path Problems. *Pages 109–119 of: STACS'92: Proc. of the 9th Annual Symposium on Theoretical Aspects of Computer Science*. Springer-Verlag.

Robertson, N., & Seymour, P.D. 1983. Graph Minors I: Excluding a Forest. *J. Comb. Theory, Ser. B*, **35**(1), 39–61.

Robertson, N., & Seymour, P.D. 1984. Graph Minors III: Planar Tree-Width. *J. Comb. Theory, Ser. B*, **36**(1), 49–64.

Robertson, N., & Seymour, P.D. 1986. Graph Minors II: Algorithmic Aspects of Tree-Width. *J. of Algorithms*, **7**(3), 309–322.

Rose, D. 1972. A Graph-Theoretic Study of the Numerical Solution of Sparse Positive Definite Systems of Linear Equations. *In:* Read, R. (ed), *Graph Theory and Computing*. Academic Press, Inc.

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

Rossi, F., van Beek, P., & Walsh, T. 2006. *Handbook of Constraint Programming*. Foundations of Artif. Intell. Elsevier Science, Inc.

Rote, G. 1990. Path Problems in Graphs. *Computing Suppl*, **7**, 155–198.

Roweis, S.T., & Ghahramani, Z. 1999. A Unifying Review of Linear Gaussian Models. *Neural Computation*, **11**(2), 305–345.

Roy, B. 1959. Transitivité et connexité. *C. R. Acad. Sci. Paris*, **249**, 216–218.

Schiex, Th. 1992. Possibilistic Constraint Satisfaction Problems or "How to handle Soft Constraints?". *Pages 268–275 of: UAI'92: Proc. of the 8th Conference on Uncertainty in Artif. Intell.* Morgan Kaufmann Publishers, Inc.

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

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

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

Schwarz, H.R. 1997. *Numerische Mathematik*. 4 edn. Teubner.

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

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

Shafer, G. 1982. Belief Functions and Parametric Models. *J. Royal Stat. Soc. B*, **44**(3), 322–352.

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*. Society for Industrial and Applied Mathematics.

Shafer, G., & Shenoy, P. 1988. *Local Computation in Hypertrees*. Tech. rept. School of Business, University of Kansas.

Shafer, G., Shenoy, P.P., & Mellouli, K. 1987. Propagating Belief Functions in qualitative Markov trees. *Int. J. Approx. Reasoning*, **1**(4), 349–400.

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

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

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

Shenoy, P.P. 1997. Binary Join Trees for Computing Marginals in the Shenoy-Shafer Architecture. *Int. J. Approx. Reasoning*, **17**, 239–263.

Shenoy, P.P., & Shafer, G. 1990. Axioms for Probability and Belief-Function Propagation. *Pages 575–610 of:* Shafer, G., & Pearl, J. (eds), *Readings in Uncertain Reasoning*. Morgan Kaufmann Publishers, Inc.

Smets, Ph. 2000. *Belief Functions and the Transferable Belief Model*.

Smets, Ph., & Kennes, R. 1994. The Transferable Belief Model. *Artif. Intell.*, **66**(2), 191–234.

Smith, S.W. 1999. *The Scientist and Engineer's Guide to Digital Signal Processing*. 2 edn. California Technical Publishing.

Spohn, W. 1988. Ordinal Conditional Functions: A Dynamic Theory of Epistemic States. *Pages 105–134 of:* Harper, W.L., & Skyrms, B. (eds), *Causation in Decision, Belief Change, and Statistics*, vol. 2. Dordrecht, Netherlands.

Spohn, W. 1990. A General Non-Probabilistic Theory of Inductive Reasoning. *Pages 149–158 of: UAI'88: Proc. of the 4th Conference on Uncertainty in Artif. Intell.* Amsterdam, The Netherlands: North-Holland Publishing Co.

Tamura, T., & Kimura, N. 1954. On Decompositions of a Commutative Semigroup. *Kodai Math. Sem. Rep.*, 109–112.

Tarjan, R.E., & Yannakakis, M. 1984. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. *SIAM J. Comput.*, **13**(3), 566–579.

Ullman, J.D. 1988. *Principles of Database and Knowledge-Base Systems Volume I*. Computer Science Press.

van Leeuwen, J. 1990. Graph Algorithms. *Pages 525–631 of:* van Leeuwen, J. (ed), *Handbook of Theoretical Computer Science: Algorithms and Complexity*, vol. A. MIT Press.

Vempaty, N.R. 1992. Solving Constraint Satisfaction Problems using Finite State Automata. *Pages 453–458 of: AAAI'92: Proc. of the 10th National Conference on Artif. Intell.* AAAI Press.

Wachter, M. 2008. *Knowledge Compilation Map - Theory and Application*. Ph.D. thesis, Philosophisch-Naturwissenschaftlichen Fakultät, University of Berne.

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 Int. Conference on Principles of Knowledge Representation and Reasoning*. AAAI Press.

Wachter, M., Haenni, R., & Pouly, M. 2007. Optimizing Inference in Bayesian Networks and Semiring Valuation Algebras. *Pages 236–247 of:* Gelbukh, A., & Kuri Morales, A. F. (eds), *MICAI'07: 6th Mexican Int. Conference on Artif. Intell.* LNAI 4827.

Warshall, S. 1962. A Theorem on Boolean Matrices. *J. of the ACM*, **9**(1), 11–12.

Wiberg, N. 1996. *Codes and Decoding on General Graphs.* Ph.D. thesis, Linkoping University, Sweden.

Wilson, N. 2005. Decision Diagrams for the Computation of Semiring Valuations. *Pages 331–336 of: IJCAI'05: Proc. of the 19th Int. Joint Conference on Artif. Intell.* Morgan Kaufmann Publishers, Inc.

Wilson, N., & Mengin, J. 2001. Embedding Logics in the Local Computation Framework. *J. of Applied Non-Classical Logics*, **11**(3-4), 239–261.

Winston, W.L., & Venkataramanan, M. 2002. *Introduction to Mathematical Programming: Applications and Algorithms.* Vol. 1. Duxbury Press.

Wojcicki, R. 1988. *Theory of Logical Calculi: Basic Theory of Consequence Operations.* Kluwer, Dordrecht.

Yannakakis, M. 1981. Computing the Minimum Fill-in is NP-Complete. *SIAM J. of Algebraic and Discrete Methods*, **2**(1), 77–79.

Yinglong, X., & Prasanna, V.K. 2008. Junction Tree Decomposition for Parallel Exact Inference. *Pages 1–12 of: IPDPS'08: IEEE Int. Symposium on Parallel and Distributed Processing.*

Zadeh, L.A. 1978. Fuzzy Sets as a Basis for a Theory of Possibility. *Fuzzy Sets and Systems*, **1**, 3–28.

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

INDEX

Generic Inference: A Unifying Theory for Automated Reasoning
First Edition. By Marc Pouly and Jürg Kohlas