Aus dem Institut für Informatik
Universität Freiburg (Schweiz)

Probabilistic Model-Based Diagnostics

Inaugural-Dissertation

zur Erlangung der Würde eines *Doctor scientiarum informaticarum*
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität Freiburg in der Schweiz

vorgelegt von

Bernhard Anrig

aus

Sargans SG und Thayngen SH

Nr. 1271
Druckerei Zofinger Tagblatt AG
2000

Von der Mathematisch-Naturwissenschaftlichen Fakultät der Universität Freiburg in der Schweiz angenommen, auf Antrag von Prof. Dr. Jürg Kohlas (Universität Freiburg), Prof. Dr. Frank Beichelt (University of the Witwatersrand, Johannesburg, Südafrika) und Prof. Dr. Robert F. Stärk (ETH Zürich).

Freiburg, im April 2000

Der Leiter der Doktorarbeit:

Prof. Dr. Jürg Kohlas

Der Dekan:

Prof. Dr. Béat Hirsbrunner

Acknowledgements

I would like to thank everyone who has given me support during the last few years, especially my wife Simone for the wonderful years we have spent together, and that she endured my mathematical talks and ideas and forced me to explain things more comprehensibly. My parents who supported my work during many years. My friends and collaborators at the institute of informatics, especially Dritan Berzati, Roman Bissig, Rolf Haenni, Patrick Hertelendy, Reto Jud, Norbert Lehmann, and Vasilica Stärk-Lepar who helped me to understand the topic better by critical discussions. Paul-André Monney from the Seminar of Statistics for his interesting co-operation at many occasions and also Pascal Murer for his support.

Especially, I would like to thank the director of my thesis, Prof. Jürg Kohlas, for the support during my work in his research group and for a lot of ideas in many discussions; Prof. Frank Beichelt and Prof. Robert F. Stärk for being referees of my thesis.

Abstract

This thesis presents the concept of general argumentation systems, a framework for representing uncertain knowledge using information algebras and information systems as well as probability algebras. Argumentation systems are a generalization of assumption-based systems and propositional argumentation systems and can deal with very general formalisms. We show also that an argumentation system itself is a special case of an information system.

We focus on argumentation systems as a tool for doing model-based diagnostics of complex systems built of components. Given a system and observations, this knowledge is modeled in the framework of an argumentation system. If the observations are not as predicted from the specification of the system, we have a diagnostic problem. Then, using concepts which are built on top of the argumentation system (such as generalized hints and allocations of support), conflicts and diagnoses, arguments, etc. for the system can be computed. Diagnoses of a system which does not work as it is supposed to, can then be used to define repair or replacement strategies for the components of the system. The set of diagnoses is often too big to be computed explicitly, but we define a logical framework for the description of sets of diagnoses.

A main ingredient of an argumentation system is probabilistic knowledge of some parts of the available information, essentially knowledge about the modes of the components. This probabilistic knowledge allows to define allocations of probability and allocations of belief on top of the argumentation systems. Using these allocations, the symbolical results are weighed and discrimination between them is possible. Further, we present algorithms for efficient computation of the probability of logical representations of sets of arguments without explicitly computing the sets.

The local computation framework of Shenoy & Shafer can be applied to argumentation systems. We show also how additional information can be added to an argumentation system, i.e. how argumentation systems are combined. Combination of argumentation systems can also be replaced by combination of the generalized hints, the allocations of arguments or belief defined on top of them.

Further, we address the problem of how additional information about the system can be obtained, especially where further measurements should take place in order to obtain a maximal expected gain of information in the argumentation system; this is useful for a sequential process of discrimination of diagnoses in the system.

Finally, we present the language ABEL, an implementation of a special type of argumentation system. Several examples are presented in order to show the strength of argumentation systems.

Zusammenfassung

In dieser Arbeit führen wir das Konzept von Argumentations-Systemen ein, die ein Hilfsmittel zur Repräsentation von unsicherer Information mit Hilfe von Informations-Algebren und -Systemen sowie Wahrscheinlichkeits-Algebren darstellen. Argumentations-Systeme sind eine Verallgemeinerung von Annahmen-basierten System und propositionellen Argumentations-Systemen und können sehr allgemeine Formalismen handhaben. Wir zeigen auch, dass Argumentations-Systeme ein Spezialfall von Informations-Systemen darstellen.

Schwerpunkt dieser Arbeit ist die Verwendung von Argumentations-Systemen als geeignete Hilfsmittel für Modell-Basierte Diagnostik von komplexen Systemen. Wir betrachten Systeme, welche aus verschiedenen Bausteinen aufgebaut sind. Das System und die Beobachtungen werden zuerst in einem Argumentations-System modelliert. Wenn die Beobachtungen nicht dem Verhalten des Systems gemäss der Spezifikation entsprechen, so sind wir mit einem Diagnose-Problem konfrontiert. Dann verwenden wir Konzepte (z.Bsp. verallgemeinerte Hinweise, Allokationen von Support), welche auf dem Argumentations-System aufbauen, um Konflikte, Diagnosen, Argumente, etc. für das System zu berechnen. Die Diagnosen eines Systems, dessen Verhalten nicht den Spezifikationen entspricht, können für die Ausarbeitung von Reparatur- oder Austauschstrategien für die Komponenten des Systems verwendet werden. Die Menge der Diagnosen ist oft zu gross um explizit berechnet zu werden, aber wir definieren eine logische Sprache für die Repräsentation von Mengen von Diagnosen.

Ein weiterer Hauptbestandteil eines Argumentations-Systems sind probabilistische Aussagen über Teile der vorhandenen Information, hauptsächlich Aussagen über die verschiedenen Arbeits-Modi der Komponenten. Diese Aussagen werden dazu verwendet, Wahrscheinlichkeits- und Belief-Allokationen auf einem Argumentations-System aufzubauen. Wir brauchen dann solche Allokationen, um die symbolischen Resultate zu bewerten und zu unterscheiden. Des weiteren definieren wir Algorithmen zur effizienten Berechnung der Wahrscheinlichkeit von logischen Repräsentation von Mengen von Diagnosen, wobei die Mengen nicht explizit berechnet werden müssen.

Die Methoden zur lokalen Berechnung von Shenoy & Shafer können auch auf Argumentations-Systeme angewandt werden. Wir zeigen weiter, wie neu auftauchende Information in ein Argumentations-System integriert werden kann, d.h. wie Argumentations-Systeme kombiniert werden. Diese Kombination kann auch auf den darauf aufbauenden Konzepten wie verallgemeinerte Hinweise, Allokationen von Support oder Wahrscheinlichkeits-Allokationen durchgeführt werden.

Im weiteren behandeln wir das Problem der Beschaffung von zusätzlicher Information über das System; speziell interessiert uns die Auswahl eines Messpunkts im System, so dass die Messung einen maximalen Erwartungswert von zusätzlicher Information liefert; dies ist nützlich in sequentiellen Verfahren zur Ausscheidung von Diagnosen.

Zum Schluss geben wir eine Einführung in die Sprache ABEL, in welcher Systeme beschrieben werden können, welche auf einem speziellen Typus eines Argumentations-Systems beruhen. Verschiedene Beispiele zeigen die Mächtigkeit von Argumentations-Systemen auf.

Contents

1 Introduction

Information processing and automated reasoning on the computer have to deal with several problems, one of them being uncertain information. In the field of Artificial Intelligence, several different concepts and methods have been developed for dealing with uncertain information, but also with imprecise, unreliable, and inconsistent information. In the presents work, we focus especially on the management of uncertain information with respect to diagnostic processes, at both the qualitative and the quantitative level.

1.1 Motivation and Purpose

Consider a system built of components and described in some modeling language, together with some observations of its behavior. If the observations are in conflict with the values predicted from the description of the system, we have a diagnosis problem, that is, we have to detect the faulty parts of the system. Reiter (1987) introduced the basic theory of diagnosis from first principles and showed how a set of abnormally working components can be used to explain the malfunctioning of the system, that is the discrepancy between the actual observations and the predicted values. Clearly, there are several different sets of possibly abnormally working components, and we do not know which is the right one. Further interesting work in this area has been done by Davis (1984), de Kleer (1976), de Kleer & Williams (1987), Genesereth (1984), and Reggia *et al.* (1983; 1985), but here we will especially focus on the ideas presented in (Kohlas *et al.*, 1998).

The main goal of this thesis is a definition of a general framework for diagnostics called argumentation systems. They generalize the so-called assumption-based systems and propositional argumentation systems (Kohlas & Monney, 1993; Kohlas & Monney, 1995; Haenni, 1996; Kohlas *et al.*, 2000), which are mainly based on propositional logic and probabilities. Argumentation systems in our terminology can deal with information represented by much more general structures than propositional logic. Moreover, algorithms are presented which use this structure to do model-based diagnostics and compute diagnoses, minimal diagnoses, and conflicts, but also supporting or refuting arguments for general hypotheses.

The problem in the approach presented in (Kohlas *et al.*, 1998) is that the knowledge about the system is represented in two languages, a general language $\mathcal{L}$ and a propositional language $\mathcal{S} \subseteq \mathcal{L}$, which contains the so-called assumptions, i.e. the unary predicates $AB(c_i)$ describing the functioning of the components c_i. The only restriction on $\mathcal{L}$ is that there is an operator defined on "instantiated" formulas in $\mathcal{L}$, i.e. on formulas where occurrences of $AB(c_i)$ are replaced by an assignment of true and false, where the answer of the operator is either that the instantiated formula is consistent or that it is inconsistent. But nothing more is known about this operator. Using the concepts of information systems and algebras (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b) together with a generalization of the concept of hints (Kohlas & Monney, 1995; Kohlas & Monney, 1993), more structure is included in the formulation of the model of the system; particularly, the two types of variables are "separated". Model-based diagnostics using (normal) hints have already been discussed in (Kohlas *et al.*, 1995), the approach presented hereafter is also a generalization of that work.

Kohlas *et al.* (1998) show that their approach is also closely related to the general theory of evidence introduced by Shafer (1976) which extends work of Dempster (1967). Further influences from this field are (Kohlas, 1995; Kohlas, 1997a; Kohlas & Brachinger, 1995; Kohlas & Monney, 1994).

Additionally, probabilities are defined on argumentation systems. De Kleer & Williams introduced probabilities into their assumption-based truth maintenance systems (De Kleer & Williams, 1987; De Kleer, 1993; De Kleer, 1986a; De Kleer, 1986b). But the approach presented here is based on a different and correct method of Kohlas *et al.* (1998). So we show how probabilities of conflicts or diagnoses as well as any formula in a language can be computed, based on a special case presented in (Anrig *et al.*, 1996). The resulting information with respect to a hypothesis is a so-called belief function (Shafer, 1976), and thus argumentation systems may also help to formalize the notion of "evidential corpus" in the terminology of Smets (Smets & Kennes, 1994; Smets, 1999). Furthermore, numerical supports and doubts in hypotheses about the system are of special interest. Efficient algorithms for computing numerical results from the symbolical ones are presented, but when only numerical results are needed, there is also the possibility to use numerical propagation algorithms, see for example (Lehmann, 2000) for a discussion.

Probabilities are really an additional information in this system. For example, they can be used to weight (minimal) diagnoses and discriminate between them. Another application is the computation of best next measurements, a method for discrimination between diagnoses using more information in the system. Probabilistic information together with symbolic information can then help the user or an agent to make decisions based on the information presented in the argumentation system, so here two types of results, numerical and symbolical, are presented to the user. Moreover, probabilities can be used to build the system's repair strategies.

1.2 Overview

Chapter 2 presents the mathematical fundaments on which this work is constructed: the theory of information algebras developed by Kohlas and Stärk (1996a; 1996b) for representing pieces of information in a well specified framework; probability algebras for representing belief, especially about a state space; and allocations of probabilities (Kohlas, 1997b) for linking pieces of information of an information algebra with elements of the probability algebra. It is interesting to see that the allocations themselves build also an information algebra. Belief functions then associate the pieces of information directly to numerical degrees of belief (Shafer, 1976). In the last section of the chapter, we focus on the construction of belief functions from allocations of probability and vice versa, the first one being a generalization of well known results, the second using results of (Kohlas & Stärk, 1996b).

In chapter 3, first the standard concept of hints (Kohlas & Monney, 1995) is introduced. The main part of a hint is a multivalued mapping from the possible interpretations to the frame of discernment. Typically, a hint reflects information like "if component one is working but component two is not working then the output of the device is five", i.e. implications where the premiss is not sure to be true. We generalize hints to the present framework, i.e. the frames of discernment are replaced by more general information algebras. We show how generalized hints can be combined and transported, and how allocations of of probability and of quasi-support can be defined on top of generalized hints. Further, also the allocations of quasi-support form themselves an information algebra.

Information algebras are often not easy to handle, because in general they have an infinite number of elements. So in chapter 4 information system are introduced following (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b). An information system is a way to represent an information algebra. For example, linear manifolds which consist in general of an infinite number of points can alternatively be represented by finite sets of linear equalities.

In chapter 5, we introduce the concepts of finite set constraints (FSC) following (Anrig *et al.*, 1997c), a generalization of binary variables. We show that FSC's are also an information system and we construct the corresponding, in fact well-known information algebra. Further, we generalize the ideas of arguments, conflicts, quasi-supports, ..., to this framework.

On the basis of the foregoing chapters, in chapter 6 we define the concept of argumentation systems. An argumentation system is a FSC language and an information system with a partial mapping between them. It is our basic structure for representing knowledge about systems, and we show how a generalized hint can be deduced from it. Additionally, probabilistic information may be available for the literals of the FSC language; in that case, the argumentation system is called probabilistic. Further, we show that an argumentation system is itself also an information system, such that we are able to construct the information algebra on top of it. Then, we show how this information algebra is in fact related to an allocation of arguments constructed on top of the hint which is induced by the argumentation system.

The main topic of this work, model-based diagnostics using argumentation systems, will be the subject of chapter 7, where the fundaments presented in the previous chapters and especially the concepts of argumentation systems are used to develop the theory based on ideas of (Reiter, 1987) and especially (Kohlas *et al.*, 1998). Several examples illustrate the theory.

A well-known architecture can be used for computing conflicts, diagnoses and arguments in argumentation systems, i.e. local propagation in valuation networks (Lauritzen & Spiegelhalter, 1988; Shenoy, 1989). In chapter 8, we adapt these concepts to information systems, and therefore also to argumentation systems, because they are just a special case of information systems.

Given a probabilistic argumentation system, the conflicts and also other arguments can be weighted using the probability function. In chapter 9, we show how prior probabilities of conflicts or diagnoses and posterior probabilities of arguments are computed using efficient algorithms.

If additional information becomes available, it has to be added to that which is already known. In chapter 10 the problem of adding information is addressed, which can also be seen as a combination of argumentation systems. We especially focus on the computation using hypertrees. There, one of the problems is the selection of the node where the new information has to be put. This problem is not so hard to solve, but in some cases, the new information might not "fit" on any edge of the hypertree because of its structure, and so we also present some generalized and some new algorithms for changing the structure of the hypertree efficiently.

Often an argumentation system does not contain enough information to make a decision, for example to replace a component or not. In chapter 11, we show how according to the knowledge contained in an argumentation system, one can compute optimal points for taking measurements, i.e. how additional information can be obtained in an optimal way. Due to the complexity of the algorithms, we present also approximation techniques for this purpose, so-called one-step lookahead methods, some of which are based on (De Kleer *et al.*, 1992b).

Finally in chapter 12, ABEL is presented, a software system which implements some special kinds of argumentation systems and allows to do model-based diagnostics. We present this using several well-known examples. This language has been designed and a solver is being implemented at the Institute of Informatics iiUF at the University of Fribourg, Switzerland. This work has started as part of the Esprit BRA project DRUMS II (Defeasible Reasoning and Uncertainty Management Systems).

1.3 An Introductory Example

In this section, we present an introductory example to clarify the concepts of an argumentation system. We use a well known example of a simple electronic circuit.

Example 1.1: Three Serial Inverters

This example is described in detail in section 7.3, example 7.1; here we give only a overview of the problem.

Consider a simple digital circuit built out of three serial inverters and connected as in fig. 1.1.

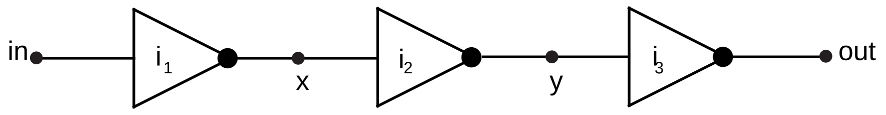


Figure 1.1: Three serial inverters.

Suppose that every component has two working modes. If an inverter i_j is in its correct working mode, then it inverts the incoming signal, i.e. its output signal is the negation of its input signal; this mode will be denoted by *ok* and the signals will be modeled by binary variables. If an inverter is in its faulty mode ($faulty = \neg ok$) then nothing is said about its behavior, so every combination of in- and output signals is possible, and one of these combinations is the correct but unknown one. Assume that the initial probability of a failure of an inverter is 0.01. The set of components is $C = \{i_1, i_2, i_3\}$, and the variables for the connectors are in, x, y, and out. This information represents an argumentation system. Usually, we represent the non-probabilistic part of this information by a relation "$\rightarrowtail$" between information about the components (which is uncertain) and information about variables:

$$\begin{array}{rcl} (i_1 = ok) & \rightarrowtail & (x = \neg in) \\ (i_2 = ok) & \rightarrowtail & (y = \neg x) \\ (i_3 = ok) & \rightarrowtail & (out = \neg y) \end{array}$$

In this example, the relation "$\rightarrowtail$" can be interpreted as a rule, that is *if* $i_1 = ok$ *then* $x = \neg in$.

In addition, suppose that the input and output values are measured as 1. The variables *in* and *out* are binary propositional variables, so we have to state the fact that *in* and *out* are true independently of the state of the systems. This is done using the symbol $\top$ representing the tautology, therefore in the same notation as above:

$$\begin{array}{rcl} \top & \rightarrowtail & in \\ \top & \rightarrowtail & out \end{array}$$

These observations form a second argumentation system which can be combined with the first one to form a new, combined argumentation system. In this example, the non-probabilistic information of the combined system is represented by all five "rules" together.

Apparently, the observations imply that the system is not functioning correctly, because the predicted output ($\neg out$) of the system, that is the behavior of the system when all components are working correctly, is in conflict with the observed one (out) if the input is in.

The combined argumentation system can then be used to compute minimal diagnoses. There are three of them in this example, namely $i_1 \neq ok$ or $i_2 \neq ok$ or $i_2 \neq ok$, where the "or" is not exclusive, which means that at least one of the components must be faulty. Using the prior probability of the working modes of a component, the probabilities of the minimal diagnoses can be computed. Furthermore, symbolic or numeric arguments for any hypothesis can be computed, so for example for the hypothesis $i_1 = ok$.

The three minimal diagnoses in this example appear to be equiprobable, so an additional measurement can get new information to discriminate between the diagnoses. The question is then: where should this measurement be made in order to get as much information as possible? $\ominus$

2
Information and Uncertainty

In order to allow computers to deal with information, it has to be modeled in a precise formalism. For our present purpose, by a piece of information we generally understand statements describing some parts of a world, each piece of information can or cannot be true. The uncertainty of the information being true or not will be considered as not contained in the information itself, but expressed within a second formalism. Thus we make a clear distinction between the description of information and the description of uncertainty of a piece of information. Clearly, the two concepts will be linked.

In this chapter we introduce the main concepts of representing information by information algebras, probability algebras and allocations of probability. These concepts are then the main ingredients for first generalizing hints (chapter 3) and then constructing argumentation systems (chapter 6).

In the present chapter, we introduce the concepts for describing information by information algebras in section 2.1 and for describing uncertainty by probability algebras in section 2.2. Then these concepts will be linked by allocations of probability in section 2.3, and we show that these allocations build themselves again an information algebra (sections 2.4 and 2.5). Finally, in section 2.6 we show that allocations of probability can be used to construct belief functions and vice versa.

The concepts and ideas for sections 2.2 to 2.6.1 are mainly taken from (Kohlas, 1997c).

2.1 Information Algebras

The concept of an information algebra was introduced by Kohlas & Stärk (1996a; 1996b) as a general theory of information processing in computer science. Its motivation comes originally from the field of uncertainty calculi in artificial intelligence, where many different calculi have in fact a common generic inference theory; the calculi are models of a certain algebraic structure similar to information algebras (Shenoy & Shafer, 1990; Shafer, 1991). However, most of those calculi do not respect the axiom of idempotency (see (A6) below) of information algebras (Kohlas & Stärk, 1996b). The following short introduction to unmarked information algebras and to marked information algebras in subsection 2.1.2 follows (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b). In subsection 2.1.3 we show that under some circumstances both concepts are equivalent.

2.1.1 Unmarked Information Algebra

In general, information may consist of different pieces or elements. The set Φ will represent the pieces of information, and a piece of information is also just called an information. Every element of Φ contains information relative to a question. Possible answers to such questions are represented in structures called **frames** or **domains**. We assume that there is a partial order $\geq$ on the set of domains D which represents the precision level of the frame. We assume that D is a lattice such that for every $x, y \in D$ the meet $x \wedge y$, representing the finest frame coarser than x and y, as well as the join $x \vee y$, representing the coarsest frame finer than x and y, are contained in D. Further, there is a top element $\top$ in the lattice, that is $\top \geq x$ for all $x \in D$.

Two or more pieces of information can be combined by a combination operation. The result is again a piece of information which contains the "sum" of all information of the pieces. This operation should clearly be associative (A1) and commutative (A2), that is the order in which pieces of information are combined should not matter. There is an "empty information" which, combined with any other information, does not add anything to it. This is called the neutral element of the combination (A3).

A second operation is called focusing of information. It focuses information to a domain in order to answer a possible question expressed in this domain. The focusing is transitive, that is if an information is transported to one domain and then to a second one, then the result is the same using just one transport of the information to the finest frame coarser than both frames (A4).

Another restriction put on the two operations is called the "combination axiom" (A5): It means that if an information is focused to a domain and combined with a second information, and the combination then focused to the same domain, then this is indeed the same as the combination of the two pieces of information previously focused on that domain; this expresses a certain distributivity of combination and focusing. This axiom is not so evident but essential for the concept of local computations developed in chapter 8. This axiom is fulfilled by a lot of interesting examples.

The transport of a piece of information to another domain does not add anything to it, but often some parts of the information get lost during the transportation. So we have the axiom of idempotency (A6).

A transport of an information to the top element $\top$ of the lattice does not change the information at all; this is called the "support" axiom (A7).

Formally this leads to the definition of an information algebra:

Definition 2.1 *Let D be a lattice and Φ a set. For $\phi_1, \phi_2 \in \Phi$ and $x \in D$, let $(\phi_1, \phi_2) \mapsto \phi_1 \oplus \phi_2$ be the combination operation with neutral element $e \in \Phi$ and $(\phi_1, x) \mapsto {\phi_1}^{\Rightarrow x}$ the focusing operation. The system (Φ, D) is called* ***(unmarked) information algebra*** *if the axioms (A1) to (A7) are fulfilled:*

(A1) Associativity: $(\phi_1 \oplus \phi_2) \oplus \phi_3 = \phi_1 \oplus (\phi_2 \oplus \phi_3)$.

(A2) Commutativity: $\phi_1 \oplus \phi_2 = \phi_2 \oplus \phi_1$.

(A3) Neutral element: $\phi \oplus e = \phi$.

(A4) Transitivity: $(\phi^{\Rightarrow x})^{\Rightarrow y} = \phi^{\Rightarrow x \wedge y}$.

(A5) Combination: $({\phi_1}^{\Rightarrow x} \oplus \phi_2)^{\Rightarrow x} = {\phi_1}^{\Rightarrow x} \oplus {\phi_2}^{\Rightarrow x}$.

(A6) Idempotency: $\phi \oplus \phi^{\Rightarrow x} = \phi$.

(A7) Support: For every $\phi \in \Phi$ *we have* $\phi^{\Rightarrow \top} = \phi$.

Φ is therefore a commutative semigroup with respect to the combination operation.

An element x of the lattice D is called a **support** for $\phi \in \Phi$ if $\phi^{\Rightarrow x} = \phi$. Axiom (A7) implies that every element has at least one support, namely $\top$.

An element z is called **null element** of the information algebra if it satisfies $z \oplus \phi = z$ for every $\phi \in \Phi$. This element represents the information which is never true. Note that if such a z exists, then in general it is not necessarily equal to $z^{\Rightarrow x}$ for every $x \in D$. In some contexts it makes sense to formulate the equality as an additional restriction; however we do not do this here. If there is no null element contained in the information algebra, a new element z with the properties of a null element can always be added to the information algebra (because Φ is a commutative semigroup) by defining $\phi \oplus z := z$ and $z^{\Rightarrow x} := z$ for $\phi \in \Phi$ and $x \in D$. So in the sequel, we always assume that the information algebra has a null element; in a concrete situation an explicit representation of this element is usually available.

If $e = z$ then the information algebra is called **inconsistent**; this case is not of interest here, because an inconsistent information algebra consists of only one element, that is $\Phi = \{z\}$. Otherwise, i.e. if $e \neq z$, the information algebra is called **consistent**.

Two elements $\phi_1, \phi_2 \in \Phi$ are called **contradicting** if their combination is the impossible information, that is if $\phi_1 \oplus \phi_2 = z$. This means that if ϕ_1 is actually true then ϕ_2 cannot be true at the same time (and vice versa); but there can also be situations in which both ϕ_1 and ϕ_2 are not true. Therefore this does not define a negation in the information algebra, but it can under some conditions be used to define so-called pseudo-complements, see (Kohlas, 1995) for further details.

Elements of (Φ, D) may be compared based on the information they contain. We say that ϕ_1 contains less information than ϕ_2 if it can be combined with it and the result is still ϕ_2. Formally, this leads to a partial ordering "$\leq$": For elements ϕ_1, ϕ_2 of the information algebra (Φ, D) we define

$$\phi_1 \leq \phi_2 \quad \text{if and only if} \quad \phi_1 \oplus \phi_2 = \phi_2. \tag{2.1}$$

Lemma 2.2 *(Kohlas & Stärk, 1996a) The relation $\leq$ defined in (2.1) is a partial ordering on Φ.*

Corollary 2.3 *e is the bottom and z the top element of Φ with respect to the partial ordering "$\leq$", that is $e \leq \phi \leq z$ for any $\phi \in \Phi$.*

Lemma 2.4 *$e = e^{\Rightarrow x}$ for any $x \in D$.*

Proof of lemma 2.4 (A3) implies $e^{\Rightarrow x} \oplus e = e^{\Rightarrow x}$ and (A5) implies $e \oplus e^{\Rightarrow x} = e$. Together with (A2), this implies the lemma. □

Note however that if there is a null element z in the information algebra, in general z is not equal to $z^{\Rightarrow x}$ for $x \in D$.

Examples of unmarked information algebras can be found in various fields. In section 5.1 the unmarked information algebra of finite set constraints is presented. Finite set constraints are a generalization of usual propositional logic, therefore the example shows also that propositional logic can be seen as an unmarked information algebra. Further, in section 6.3, another information algebra is presented. Kohlas & Stärk (1996b) show the link to constraint logic programming.

2.1.2 Marked Information Algebra

The idea of a marked information algebra is that every element has a label attached to it which specifies explicitly the domain in which the information is expressed. This structure is very similar to the structure introduced in the last subsection. In subsection 2.1.3 we show how an unmarked information algebra can be transformed into a marked one and vice versa. The new structure will be useful especially for computational purposes, as will be shown in chapter 8.

The formal definition was specified in (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b):

Definition 2.5 *Let D be a lattice, Φ a set, and $d : \Phi \to D$ a labeling function. For $\phi_1, \phi_2 \in \Phi$ let $(\phi_1, \phi_2) \mapsto \phi_1 \oplus \phi_2$ be the combination operation with neutral element $e_x \in \Phi$ for every $x \in D$. For $\phi \in \Phi$ and $x \leq d(\phi)$ $(x \in D)$ let $(\phi, x) \mapsto \phi^{\downarrow x}$ be the marginalization operation. The system $\langle \Phi, D \rangle$ is called **marked information algebra** if the axioms (M1) to (M10) are fulfilled:*

(M1) *Associativity:* $(\phi_1 \oplus \phi_2) \oplus \phi_3 = \phi_1 \oplus (\phi_2 \oplus \phi_3)$.

(M2) *Commutativity:* $\phi_1 \oplus \phi_2 = \phi_2 \oplus \phi_1$.

(M3) *Labeling:* $d(\phi_1 \oplus \phi_2) = d(\phi_1) \vee d(\phi_2)$.

(M4) *Label of the neutral element:* $d(e_x) = x$ *for every* $x \in D$.

(M5) *Neutral element: If* $d(\phi) = x$, *then* $\phi \oplus e_x = \phi$.

(M6) *Label of the marginal: If* $x \leq d(\phi)$, *then* $d(\phi^{\downarrow x}) = x$.

(M7) *Transitivity: If* $x \leq y \leq d(\phi)$, *then* $(\phi^{\downarrow y})^{\downarrow x} = \phi^{\downarrow x}$.

(M8) *Combination: If* $d(\phi_1) = x_1$ *and* $d(\phi_2) = x_2$, *then* $(\phi_1 \oplus \phi_2)^{\downarrow x_1} = \phi_1 \oplus (\phi_2{}^{\downarrow x_1 \wedge x_2})$.

(M9) *Idempotency: If* $x \leq d(\phi)$, *then* $\phi \oplus \phi^{\downarrow x} = \phi$.

(M10) *Stability: If* $x \leq y$, *then* $(e_y)^{\downarrow x} = e_x$.

An information ϕ with label x will usually be written as $\langle \phi, x \rangle$ in order to emphasize the domain x.

An information with label x has sometimes to be "extended" to a superior domain $y \geq x$ in order to answer a question formulated with respect to y. This extension does not add any information and is therefore called the vacuous extension:

Definition 2.6 *The* ***vacuous extension*** *of an information* $\phi \in \Phi$ *with support* x *to a domain* $y \geq x$ *is defined by*

$$\phi^{\uparrow y} := \phi \oplus e_y. \tag{2.2}$$

Clearly, $d(\phi^{\uparrow y}) = d(\phi \oplus e_y) = x \vee y = y$ because $y \geq x$.

The next lemma shows that this definition is consistent, that is the vacuous extension of the empty information e_x with label x to a finer frame y must result in the empty information with label y, that is e_y.

Lemma 2.7 *(Kohlas & Stärk, 1996b) If* $x \leq y$ *then* $e_x \oplus e_y = e_y$.

The vacuous extension has several properties:

Lemma 2.8 *(Kohlas & Stärk, 1996b)*

1. $x \leq y$ *implies* $e_x{}^{\uparrow y} = e_y$.

2. $e_x \oplus e_y = e_{x \vee y}$.

3. *If $d(\phi) = x$, then $\phi \oplus e_y = \phi^{\uparrow x \vee y}$.*
4. *If $d(\phi) = x$, then $\phi^{\uparrow x} = \phi$.*
5. *If $d(\phi) = x$ and $y \leq x$, then $\phi \oplus e_y = \phi$.*
6. *If $d(\phi) = x$ and $x \leq y \leq z$, then $\left(\phi^{\uparrow y}\right)^{\uparrow z} = \phi^{\uparrow z}$.*
7. *If $d(\phi_1) \leq z$, $d(\phi_2) \leq z$, then $(\phi_1 \oplus \phi_2)^{\uparrow z} = {\phi_1}^{\uparrow z} \oplus {\phi_2}^{\uparrow z}$.*
8. *If $d(\phi_1) = x_1$ and $d(\phi_2) = x_2$, then $\phi_1 \oplus \phi_2 = {\phi_1}^{\uparrow x_1 \vee x_2} \oplus {\phi_2}^{\uparrow x_1 \vee x_2}$.*

The stability (M10) implies a stability property which holds for any information ϕ in $\langle \Phi, D \rangle$:

Lemma 2.9 *(Kohlas & Stärk, 1996a) If $d(\phi) = x$ and $x \leq y \in D$ then*

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

This shows that the vacuous extension does indeed not add any information, therefore ϕ and $\phi^{\uparrow y}$ contain essentially the same information.

A relation "$\leq$" can be defined in this information algebra similar to the one introduced in the unmarked case. For $\langle \phi_1, x_1 \rangle, \langle \phi_2, x_2 \rangle$ in $\langle \Phi, D \rangle$ define

$$\langle \phi_1, x_1 \rangle \leq \langle \phi_2, x_2 \rangle \quad \text{if and only if} \quad \phi_1 \oplus \phi_2 = {\phi_2}^{\uparrow x_1 \vee x_2} \tag{2.4}$$

Often, $\langle \phi_1, x_1 \rangle \leq \langle \phi_2, x_2 \rangle$ will be abbreviated as $\phi_1 \leq \phi_2$.

Lemma 2.10 *"$\leq$" defined by (2.4) is a partial ordering.*

Proof of lemma 2.10 Reflexivity: For $\langle \phi, x \rangle \in \langle \Phi, D \rangle$ we have $\phi = \phi \oplus \phi = \phi^{\uparrow x \vee x} = \phi$ and therefore $\langle \phi, x \rangle \leq \langle \phi, x \rangle$.

Transitivity: Let $\langle \phi_i, x_i \rangle \in \langle \Phi, D \rangle$ for $i = 1, 2, 3$, $\langle \phi_1, x_1 \rangle \leq \langle \phi_2, x_2 \rangle$ and $\langle \phi_2, x_2 \rangle \leq \langle \phi_3, x_3 \rangle$. Using the definitions, this is equal to

$$\phi_1 \oplus \phi_2 = {\phi_2}^{\uparrow x_1 \vee x_2} \tag{2.5}$$
$$\phi_2 \oplus \phi_3 = {\phi_3}^{\uparrow x_2 \vee x_3} \tag{2.6}$$

Now

$$\begin{aligned}
\phi_1 \oplus \phi_3 &= \left((\phi_1 \oplus \phi_3)^{\uparrow x_1 \vee x_2 \vee x_3}\right)^{\downarrow x_1 \vee x_3} && \text{by lemma 2.9} \\
&= (\phi_1 \oplus \phi_3 \oplus e_{x_2})^{\downarrow x_1 \vee x_3} && \text{by definition} \\
&= \left(\phi_1 \oplus {\phi_3}^{\uparrow x_2 \vee x_3}\right)^{\downarrow x_1 \vee x_3} && \\
&= (\phi_1 \oplus \phi_2 \oplus \phi_3)^{\downarrow x_1 \vee x_3} && \text{by (2.6)} \\
&= \left({\phi_2}^{\uparrow x_1 \vee x_2} \oplus \phi_3\right)^{\downarrow x_1 \vee x_3} && \text{by (2.5)}
\end{aligned}$$

$$\begin{aligned}
&= (\phi_2 \oplus e_{x_1} \oplus \phi_3)^{\downarrow x_1 \vee x_3} \\
&= \left({\phi_3}^{\uparrow x_2 \vee x_3} \oplus e_{x_1}\right)^{\downarrow x_1 \vee x_3} && \text{by (2.6)} \\
&= \left({\phi_3}^{\uparrow x_1 \vee x_2 \vee x_3}\right)^{\downarrow x_1 \vee x_3} \\
&= \left(\left({\phi_3}^{\uparrow x_1 \vee x_3}\right)^{\uparrow x_1 \vee x_2 \vee x_3}\right)^{\downarrow x_1 \vee x_3} \\
&= {\phi_3}^{\uparrow x_1 \vee x_3} && \text{by lemma 2.9}
\end{aligned}$$

which is the definition of $\langle \phi_1, x_1 \rangle \leq \langle \phi_3, x_3 \rangle$. This proves the lemma. □

Lemma 2.11 *For a marked statement $\langle \phi, x \rangle$ and $y \geq x$*

$$\langle \phi, x \rangle^{\uparrow y} \geq \langle \phi, x \rangle \qquad \textit{as well as} \qquad \langle \phi, x \rangle^{\uparrow y} \leq \langle \phi, x \rangle.$$

Proof of lemma 2.11 By definition, the left hand side is equivalent to $\langle \phi, x \rangle \oplus \langle \phi, x \rangle^{\uparrow y} = \left(\langle \phi, x \rangle^{\uparrow y}\right)^{\uparrow x \vee y}$. Because $y \geq x$ this is equivalent to $\langle \phi, x \rangle \oplus \langle \phi, x \rangle^{\uparrow y} = \langle \phi, x \rangle^{\uparrow y}$ (which is in fact also the definition of the right-hand side of lemma 2.11). The idempotency axiom (M9) implies that $\langle \phi, x \rangle \oplus \langle \phi, x \rangle^{\uparrow y} = \langle \phi, x \rangle \oplus \langle \phi, x \rangle \oplus \langle e_y, y \rangle = \langle \phi, x \rangle \oplus \langle e_y, y \rangle = \langle \phi, x \rangle^{\uparrow y}$ and this proves the lemma. □

Kohlas & Stärk (1996b) discuss the concept of marked information algebras within relational databases and module algebras. In sections 5.1 and 6.3, marked information algebras are constructed from unmarked ones (or vice versa) using techniques presented in the next subsection.

2.1.3 Linking Unmarked and Marked Information Algebras

To every unmarked information algebra we can construct the corresponding marked information algebra. Kohlas & Stärk (1996a; 1996b) show also the inverse path from the marked to the unmarked information algebra. They also show that if one starts with an unmarked information algebra, then constructs the marked one, and finally from the marked one constructs the unmarked one, the latter is isomorphic to the starting algebra if every element therein has a support.

First, we start with an unmarked information algebra and construct the corresponding marked one. So let (Φ, D) be a unmarked information algebra. Consider the set

$$M = \{\langle \phi, x \rangle : \phi^{\Rightarrow x} = \phi\}. \tag{2.7}$$

For two elements $\langle \phi_1, x_1 \rangle$, $\langle \phi_2, x_2 \rangle$ from M define their combination by

$$\langle \phi_1, x_1 \rangle \oplus \langle \phi_2, x_2 \rangle := \langle \phi_1 \oplus \phi_2, x_1 \vee x_2 \rangle \tag{2.8}$$

and for $y \leq x_1$ the marginalization by

$$\langle \phi_1, x_1 \rangle^{\downarrow y} := \langle {\phi_1}^{\Rightarrow y}, y \rangle. \tag{2.9}$$

Then define $d(\langle \phi_1, x_1 \rangle := x_1$ and $e_x := \langle e, x \rangle$. It is easy to check that $\langle M, D \rangle$ satisfies the axioms (M1) to (M10) and is a marked information algebra.

In the sequel, we usually denote by $\langle \Phi, D \rangle$ the marked information algebra constructed from the unmarked information algebra (Φ, D).

The null element z from (Φ, D) carries over to $\langle \Phi, D \rangle$ as follows: For every label x the element $z_x := \langle z, x \rangle$ satisfies $z_x \oplus \phi = z_x$ for every ϕ with label $d(\phi) = x$.

The combination of a marked information with such an element results then still in a null element:

Lemma 2.12 *(Kohlas & Stärk, 1996b) If $d(\phi) = x$ then $\phi \oplus z_y = z_{x \vee y}$.*

Now, to consider the inverse path, we start with a marked information algebra and construct the corresponding unmarked one. Let $\langle \Phi, D \rangle$ be an marked information algebra. Two pieces of information $\langle \phi, x \rangle$ and $\langle \psi, y \rangle$ in $\langle \Phi, D \rangle$ are called **equivalent** $\phi \sim \psi$ if their extensions to $d(\phi) \vee d(\psi) = x \vee y$ are equal, that is if $\psi \oplus e_{x \vee y} = \phi \oplus e_{x \vee y}$. The relation $\phi \sim \psi$ is equivalent to $\phi \leq \psi$ *and* $\psi \leq \phi$, i.e. "$\sim$" is the symmetry relation defined by the partial ordering "$\leq$".

Lemma 2.13 *$\sim$ is an equivalence relation.*

Proof of lemma 2.13 Reflexivity and symmetry follows from the definition. Let now $\langle \phi, x \rangle$, $\langle \psi, y \rangle$ and $\langle \chi, z \rangle$ be three marked statements with $\phi \sim \psi$ and $\psi \sim \chi$. By definition, this is $\phi \oplus e_y = \psi \oplus e_x$ and $\psi \oplus e_z = \chi \oplus e_y$. This implies

$$\phi \oplus e_z \oplus e_y = \psi \oplus e_x \oplus e_z = \chi \oplus e_x \oplus e_y \tag{2.10}$$

Marginalizing the left-hand side of (2.10) to $x \vee z$ then gives

$$\begin{aligned} (\phi \oplus e_z \oplus e_y)^{\downarrow x \vee z} &= (\phi \oplus e_z) \oplus {e_y}^{\downarrow (x \vee z) \wedge y} &&= (\phi \oplus e_z) \oplus e_{(x \vee z) \wedge y} \\ &= (\phi \oplus e_z) \oplus e_{x \vee z} \oplus e_{(x \vee z) \wedge y} &&= (\phi \oplus e_z) \oplus e_{x \vee z} \\ &= \phi \oplus e_z \end{aligned}$$

and, similar, marginalizing the right-hand side of (2.10) to $x \vee z$ gives

$$(\chi \oplus e_x \oplus e_y)^{\downarrow x \vee z} = \chi \oplus e_x,$$

such that finally $\phi \oplus e_z = \chi \oplus e_x$ which, by definition, is $\phi \sim \chi$ and this shows the transitivity of $\sim$. □

Let now $[\phi]$ be the equivalence class to which $\langle \phi, x \rangle$ belongs and $[\Phi]$ the set of all these equivalence classes. The combination of two elements $[\phi]$ and $[\psi]$ of

$[\Phi]$ is defined by $[\phi] \oplus [\psi] := [\phi \oplus \psi]$. Focussing an element $[\phi]$ to a domain y is defined by $[\phi]^{\Rightarrow y} := [(\phi \oplus e_y)^{\downarrow y}]$. Kohlas & Stärk (1996b) show that $[\Phi]$, together with these two operations of combination and focussing, defines indeed an unmarked information algebra.

The next lemma shows that the orders defined on the marked and the unmarked information algebra coincide:

Lemma 2.14 $\langle \phi_1, x_1 \rangle \leq \langle \phi_2, x_2 \rangle$ *if and only if* $[\phi_1] \leq [\phi_2]$.

Proof of lemma 2.14 $\langle \phi_1, x_1 \rangle \leq \langle \phi_2, x_2 \rangle$ means, by definition, $\phi_1 \oplus \phi_2 = {\phi_2}^{\uparrow x_1 \vee x_2}$. But this is equivalent to $[\phi_1] \oplus [\phi_2] = [\phi_2]$ in the corresponding unmarked information algebra, which, again by definition, means $[\phi_1] \leq [\phi_2]$. $\square$

In the sequel, we can work either with the unmarked or the corresponding marked information algebra and switch between the two representations depending on which of them is more appropriate to the context.

2.2 Probability Algebras

An element of an information algebra can be believed to be true of not. But it can also be believed only partly. In order to capture this idea, a **system of partial beliefs** $\mathcal{B}$ is introduced. An element of $\mathcal{B}$ is called a belief. If a belief b is part of another belief b' then this is denoted by $b \leq b'$. We suppose that "$\leq$" is a partial ordering on $\mathcal{B}$ and has a top element $\top$ representing the total belief, and a bottom element $\perp$ representing the void belief. We assume further that the complement b^c of any belief $b \in \mathcal{B}$ is also a belief in $\mathcal{B}$ as well as infimum $\bigwedge_{i \in I} b_i$ and supremum $\bigvee_{i \in I} b_i$ of any countable family of beliefs $\{b_i : i \in I\} \subseteq \mathcal{B}$. Therefore, the system of partial beliefs $\mathcal{B}$ is a Boolean σ-algebra. Further we assume that the countable chain condition (Halmos, 1963) holds, which in a Boolean σ-algebra can be expressed in the following way: If b_i, $i \in I$ is strictly ascending, that is $b_i < b_j$ for $i < j \in I$, then I is countable. Halmos (1963) shows that this implies that the Boolean σ-algebra is **complete**, that arbitrary subsets of $\mathcal{B}$ have infimum as well as supremum, furthermore the countable chain condition implies that for any subset $B \subseteq \mathcal{B}$ there is a countable subset $B' \subseteq B$ such that $\bigwedge B = \bigwedge B'$.

Two beliefs b and b' are called **disjoint** if their infimum is the bottom element, that is $b \wedge b' = \perp$.

Lemma 2.15 *For a complete Boolean algebra* $\mathcal{B}$ *and* $b, b_i, c_j \in \mathcal{B}$ *for* $i \in I$, $j \in J$ *we have*

- *De Morgan's Law:*

$$\left(\bigvee_{i \in I} b_i\right)^c = \bigwedge_{i \in I} {b_i}^c, \qquad \left(\bigwedge_{i \in I} b_i\right)^c = \bigvee_{i \in I} {b_i}^c. \tag{2.11}$$

- *Associative Law: For* $I = \bigcup_{j\in J} I_j$,

$$\bigvee_{j\in J}\left(\bigvee_{i\in I_j} b_i\right) = \bigvee_{i\in I} b_i. \tag{2.12}$$

- *Distributive Law:*

$$b \wedge \left(\bigvee_{i\in I} b_i\right) = \bigvee_{i\in I} (b \wedge b_i) \tag{2.13}$$

$$b \vee \left(\bigwedge_{i\in I} b_i\right) = \bigwedge_{i\in I} (b \vee b_i) \tag{2.14}$$

$$\left(\bigvee_{i\in I} b_i\right) \wedge \left(\bigvee_{j\in J} c_j\right) = \bigvee_{i\in I,\, j\in J} (b_i \wedge c_j) \tag{2.15}$$

$$\left(\bigwedge_{i\in I} b_i\right) \vee \left(\bigwedge_{j\in J} c_j\right) = \bigwedge_{i\in I,\, j\in J} (b_i \vee c_j)\,. \tag{2.16}$$

Proof of lemma 2.15 For (2.11) to (2.14) see (Sikorski, 1960). Let $b = \bigvee_{i\in I} b_i$, then

$$\begin{aligned}
\left(\bigvee_{i\in I} b_i\right) \wedge \left(\bigvee_{j\in J} c_j\right) &= b \wedge \left(\bigvee_{j\in J} c_j\right) = \bigvee_{j\in J} (b \wedge c_j) && \text{by (2.13)}\\
= \bigvee_{j\in J}\left(\left(\bigvee_{i\in I} b_i\right) \wedge c_j\right) &= \bigvee_{j\in J}\left(\bigvee_{i\in I} (b_i \wedge c_j)\right) && \text{by (2.13)}\\
= \bigvee_{i\in I,\, j\in J} (b_i \wedge c_j) & && \text{by (2.12)}
\end{aligned}$$

and this proves (2.15); (2.16) is proved analogously. □

The elements of the system of partial beliefs are then weighted, so a measure $\mu : \mathcal{B} \longrightarrow [0,1]$ assigns a weight to every belief $b \in \mathcal{B}$ representing the degree of belief. The most probable belief, the element $\top$, has a degree of 1, the most improbable belief $\perp$ has the degree 0. When a belief $b \in \mathcal{B}$ is decomposed into a family of disjoint beliefs $\{b_i : i \in I\}$, then the countable chain condition implies that this family is countable, and the sum of the beliefs of the parts, $\sum_{i\in I} \mu(b_i)$, is equal to the belief of the whole, $\mu(b)$. Formally, the measure μ satisfies:

- For any family $\{b_i : i \in I\} \subseteq \mathcal{B}$ of disjoint beliefs,

$$\mu\left(\bigvee_{i\in I} b_i\right) = \sum_{i\in I} \mu(b_i). \tag{2.17}$$

- For any belief $b \in \mathcal{B}$, $\mu(b) \geq 0$.
- For any belief $b \in \mathcal{B}$, if $\mu(b) = 0$ then $b = \bot$.
- $\mu(\top) = 1$.

A measure which respects these conditions is called a **positive probability measure**.

A system $(\mathcal{B}, \mu)$ consisting of a complete Boolean algebra $\mathcal{B}$ together with a positive probability measure μ is called a **probability algebra**, see also fig. 2.1.

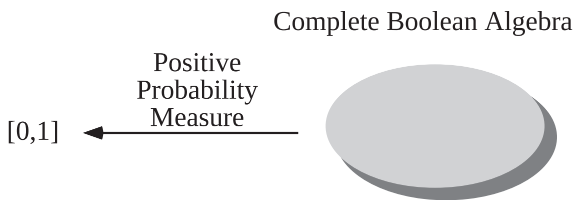


Figure 2.1: Probability algebra

An important property of probability algebras can be expressed using the concepts of up- and downward directed sets:

A subset B of $\mathcal{B}$ is called **downward directed** if for every pair of elements b', b'' in B there is an element $b \in B$ such that $b \leq b' \wedge b''$. Similar, a subset B of $\mathcal{B}$ is called **upward directed** if for every pair of elements b', b'' in B there is an element $b \in B$ such that $b \geq b' \wedge b''$.

Lemma 2.16 *For any subset $B \subseteq \mathcal{B}$*

$$\mu\left(\bigwedge_{b \in B} b\right) \leq \inf_{b \in B} \mu(b) \quad \textit{and} \quad \mu\left(\bigvee_{b \in B} b\right) \geq \sup_{b \in B} \mu(b) \tag{2.18}$$

Additionally, if the set B is downward directed, then

$$\mu\left(\bigwedge_{b \in B} b\right) = \inf_{b \in B} \mu(b), \tag{2.19}$$

if the set B is upward directed, then

$$\mu\left(\bigvee_{b \in B} b\right) = \sup_{b \in B} \mu(b). \tag{2.20}$$

Proof of lemma 2.16 (2.18) is trivial. For a proof of (2.19) and (2.20) see lemma 3.4 of (Kohlas, 1995). □

2.3 Allocations of Probability

Let (Φ, D) be a unmarked information algebra. Additional knowledge may now permit to decide which pieces of information in Φ actually hold and which not. But in general, such statements cannot be made with certitude, but only beliefs can be assigned to the pieces of information. A consistency condition is that if an information ϕ_1 is stated to be true, then also every information $\phi_2 \leq \phi_1$ must be true. If ϕ_1 and ϕ_2 are stated to be true, then also their combination $\phi_1 \oplus \phi_2$. These conditions mean that the pieces of information which are stated to hold must form an ideal.

We represent such a belief in a piece of information $\phi \in \Phi$ by an element $\rho(\phi)$ of a probability algebra $\mathcal{B}$, therefore the function

$$\rho : \quad \Phi \rightarrow \mathcal{B} \tag{2.21}$$

assigns a belief to every element of the information algebra (fig. 2.2). Generalizing the remarks above, ρ has to satisfy two consistency conditions:

(R1) $\rho(e) = \top$.

(R2) For $\phi_1, \phi_2 \in \Phi$: $\rho(\phi_1 \oplus \phi_2) = \rho(\phi_1) \wedge \rho(\phi_2)$.

(R1) means that the total belief $\top$ is assigned to the empty information e. (R2) says that largest partial belief which is part both of $\rho(\phi_1)$ and of $\rho(\phi_2)$, that is the infimum $\rho(\phi_1) \wedge \rho(\phi_2)$, is assigned to the combination $\phi_1 \oplus \phi_2$.

A mapping $\rho : \Phi \rightarrow \mathcal{B}$ which satisfies (R1) and (R2) is called an **allocation of probability** on the information algebra (Φ, D). The notion of allocation of probability has already been studied in a slightly less general context by Shafer (1979).

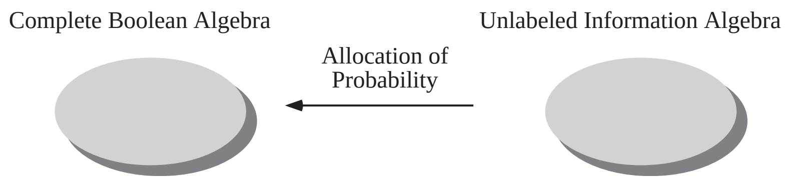


Figure 2.2: Allocation of probability.

The tuple $(\Phi, D, \mathcal{B}, \rho)$ is also called a **body of arguments** by (Kohlas, 1995). Note nevertheless that in the original definition of a body of arguments the structure demanded on Φ is a lattice, whereas in our situation, Φ is an information algebra and therefore in general only a semi-lattice with meet defined by the combination. But this difference is not essential in the present context. In chapter 3 we will introduce a method for constructing allocations of probability from so-called hints, and examples for allocations of probability will be given there.

The next lemma shows that the allocation is indeed monotone:

Lemma 2.17 *Let $\phi_1, \phi_2 \in \Phi$. If $\phi_1 \leq \phi_2$ then $\rho(\phi_1) \geq \rho(\phi_2)$.*

Proof of lemma 2.17 $\phi_1 \leq \phi_2$ is, by definition, $\phi_2 = \phi_1 \oplus \phi_2$. By (R2) we have $\rho(\phi_2) = \rho(\phi_1 \oplus \phi_2) = \rho(\phi_1) \wedge \rho(\phi_2)$ and this shows that $\rho(\phi_1) \geq \rho(\phi_2)$. □

An allocation is called **normalized** if no belief is allocated to the null element z, that is if

(R3) $\rho(z) = \perp$.

An allocation ρ which is not normalized can always be normalized by conditioning like in probability theory. Consider the family of beliefs of the form $b \wedge \rho^c(z)$ for $b \in \mathcal{B}$. This family is denoted by $\mathcal{B}' = \mathcal{B} \wedge \rho^c(z)$. It represents the non-contradictory part of $\mathcal{B}$, and it is still a complete Boolean algebra. If $\mu(\rho(z)) \neq 1$ then a probability measure μ' on $\mathcal{B}$ is defined by

$$\mu'(b \wedge \rho^c(z)) \quad := \quad \frac{\mu(\rho(b \wedge \rho^c(z)))}{\mu(\rho^c(z))}, \tag{2.22}$$

such that finally $\rho'(\phi) := \rho(\phi) \wedge \rho^c(z)$ is a normalized allocation of probability from Φ to $\mathcal{B}'$ derived from ρ.

A special case of a normalized allocation is the so called **vacuous allocation** ρ_v, defined by $\rho_v(e) = \top$ and $\rho_v(\phi) = \perp$ for all $\phi \in \Phi$ with $\phi \neq e$. ρ_v allocates the total belief to the neutral element e and no belief to any other element; therefore it contains no information and represents complete ignorance.

If ρ is an allocation of probability on the information algebra (Φ, D), then denote by $\rho(\Phi)$ the image of Φ under ρ, and by $\mathcal{B}(\rho(\Phi))$ the smallest complete subalgebra of $\mathcal{B}$ which contains $\rho(\Phi)$; such a smallest complete subalgebra always exists, because the family of complete subalgebras is an intersection system, that is the intersection of any family of complete subalgebras of $\mathcal{B}$ is again a complete subalgebra of $\mathcal{B}$. Denote by μ_ρ the restriction of μ to $\mathcal{B}(\rho(\Phi))$.

Let now ρ_1 and ρ_2 be two allocations of probability from (Φ, D) to $\mathcal{B}$. Then $\mathcal{B}(\rho_1(\Phi))$ and $\mathcal{B}(\rho_2(\Phi))$ are called **independent subalgebras**, if for every $m_1 \in \mathcal{B}(\rho_1(\Phi))$ and $m_2 \in \mathcal{B}(\rho_2(\Phi))$ we have $m_1 \wedge m_2 \neq \perp$. If additionally $\mu(m_1 \wedge m_2) = \mu(m_1)\mu(m_2)$, then $(\mathcal{B}(\rho_1(\Phi)), \mu_{\rho_1})$ and $(\mathcal{B}(\rho_2(\Phi)), \mu_{\rho_2})$ are called **independent probability algebras**. Hence, under the same conditions, ρ_1 and ρ_2 are called **independent allocations of probability**.

The concept of independent allocations is used in section 2.6 to compute the belief of the combination of allocations.

In section 3.2 we will show how to construct an allocation of probability from the information contained in a generalized hint.

2.4 Unmarked Algebra of Allocations

In this section, we show that the set P_Φ of allocations from an unmarked information algebra (Φ, D) towards a probability algebra $(\mathcal{B}, \mu)$ form themselves also an unmarked information algebra, so axioms (A1) to (A7) from definition 2.1 are fulfilled, and the whole computational theory which will be developed in chapter 8 can be applied to them as well.

2.4.1 Combination of Allocations

Consider the situation where different sources of information are available, where each of them defines an allocation of probability with respect to the same information algebra (Φ, D) and the same probability algebra $(\mathcal{B}, \mu)$. Then these allocations have to be combined in order to get an allocation which reflects the whole available information about the belief of elements of the information algebra.

Consider two allocations of probability $\rho_i : \Phi \to \mathcal{B}$, $i = 1, 2$. These allocations should now be combined into an allocation ρ in order to represent the combined information.

Let $\phi \in \Phi$ be an information and $\phi_1, \phi_2 \in \Phi$ such that the combination $\phi_1 \oplus \phi_2$ contains more information than ϕ, that is $\phi \leq \phi_1 \oplus \phi_2$. Any belief which is allocated ϕ_1 **and also** to ϕ_2 by ρ_1 and ρ_2 respectively is also a belief for the combined information $\phi_1 \oplus \phi_2$, therefore we require

$$\rho(\phi) \geq \rho_1(\phi_1) \wedge \rho_2(\phi_2) \tag{2.23}$$

It is then natural to define the belief accorded to the information ϕ as the smallest upper bound of all such beliefs, that is

$$\rho(\phi) = (\rho_1 \oplus \rho_2)(\phi) := \bigvee_{\phi \leq \phi_1 \oplus \phi_2} (\rho_1(\phi_1) \wedge \rho_2(\phi_2)) \tag{2.24}$$

and the following theorem shows that this defines indeed an allocation of probability:

Theorem 2.18 *(2.24) defines a combination operation on the set P_Φ of allocations from (Φ, D) to $(\mathcal{B}, \mu)$.*

Proof of theorem 2.18 Following theorem 1.4 of (Kohlas, 1995).

We have to prove that (2.24) defines an allocation of probability, therefore we have to prove that ρ satisfies the axioms (R1) and (R2) on page 18.

$$\rho(e) = \bigvee_{e \leq \phi_1 \oplus \phi_2} (\rho_1(\phi_1) \wedge \rho_2(\phi_2)) = \rho_1(e) \wedge \rho_2(e) = \top,$$

because of the monotonicity of ρ_1 and ρ_2, and therefore (R1) is satisfied.

Let now $\psi_1, \psi_2 \in \Phi$. Then, by definition,

$$\rho(\psi_1 \oplus \psi_2) = \bigvee_{\psi_1 \oplus \psi_2 \leq \phi_1 \oplus \phi_2} (\rho_1(\phi_2) \wedge \rho_2(\phi_2)). \tag{2.25}$$

For $i = 1, 2$, $\psi_i \leq \psi_1 \oplus \psi_2$ implies

$$\bigvee_{\psi_1 \oplus \psi_2 \leq \phi_1 \oplus \phi_2} (\rho_1(\phi_2) \wedge \rho_2(\phi_2)) \leq \bigvee_{\psi_i \leq \phi_1 \oplus \phi_2} (\rho_1(\phi_2) \wedge \rho_2(\phi_2)),$$

such that $\rho(\psi_1 \oplus \psi_2) \leq \rho(\psi_i)$. This implies then $\rho(\psi_1 \oplus \psi_2) \leq \rho(\psi_1) \wedge \rho(\psi_2)$.

On the other hand,

$$\begin{aligned} &\{(\phi_1, \phi_2) : \psi_1 \oplus \psi_2 \leq \phi_1 \oplus \phi_2\} \\ &\supseteq \{(\phi_1, \phi_2) : \phi_1 = \phi'_1 \oplus \phi''_1,\ \phi_2 = \phi'_2 \oplus \phi''_2,\ \psi_1 \leq \phi'_1 \oplus \phi'_2,\ \psi_2 \leq \phi''_1 \oplus \phi''_2\}. \end{aligned}$$

This set inclusion together with (R2) for ρ_1, ρ_2 and lemma 2.15 implies then

$$\begin{aligned} \rho(\psi_1 \oplus \psi_2) &\geq \bigvee_{\substack{\psi_1 \leq \phi'_1 \oplus \phi'_2 \\ \psi_2 \leq \phi''_1 \oplus \phi''_2}} (\rho(\phi'_1 \oplus \phi''_1) \wedge \rho(\phi'_2 \oplus \phi''_2)) \\ &= \bigvee_{\substack{\psi_1 \leq \phi'_1 \oplus \phi'_2 \\ \psi_2 \leq \phi''_1 \oplus \phi''_2}} (\rho_1(\phi'_1) \wedge \rho_1(\phi''_1) \wedge \rho_2(\phi'_2) \wedge \rho_2(\phi''_2)) \\ &= \bigvee_{\substack{\psi_1 \leq \phi'_1 \oplus \phi'_2 \\ \psi_2 \leq \phi''_1 \oplus \phi''_2}} (\rho_1(\phi'_1) \wedge \rho_2(\phi'_2)) \wedge (\rho_1(\phi''_1) \wedge \rho_2(\phi''_2)) \\ &= \left(\bigvee_{\psi_1 \leq \phi'_1 \oplus \phi'_2} (\rho_1(\phi'_1) \wedge \rho_2(\phi'_2)) \right) \wedge \left(\bigvee_{\psi_2 \leq \phi''_1 \oplus \phi''_2} (\rho_1(\phi''_1) \wedge \rho_2(\phi''_2)) \right) \\ &= \rho(\psi_1) \wedge \rho(\psi_2). \end{aligned}$$

So finally $\rho(\psi_1 \oplus \psi_2) = \rho(\psi_1) \wedge \rho(\psi_2)$, therefore (R2) is fulfilled and the theorem proved. □

Theorem 2.19 *The set P_Φ is a semi-lattice under combination, that is P_Φ is a commutative, idempotent semi-group with the vacuous allocation ρ_ν as neutral element.*

Proof of theorem 2.19 Commutativity follows from the definition (2.24).

Let $\phi \in \Phi$, then using lemma 2.15

$$\begin{aligned} ((\rho_1 \oplus \rho_2) \oplus \rho_3)(\phi) &= \bigvee_{\phi \leq \phi' \oplus \phi_3} ((\rho_1 \oplus \rho_2)(\phi') \wedge \rho_3(\phi_3)) \\ &= \bigvee_{\phi \leq \phi' \oplus \phi_3} \left(\left(\bigvee_{\phi' \leq \phi_1 \oplus \phi_2} (\rho_1(\phi_1) \wedge \rho_2(\phi_2)) \right) \wedge \rho_3(\phi_3) \right) \\ &= \bigvee_{\phi \leq \phi_1 \oplus \phi_2 \oplus \phi_3} (\rho_1(\phi_1) \wedge \rho_2(\phi_2) \wedge \rho_3(\phi_3)) \end{aligned}$$

and, by symmetry, this proves the associativity.

For idempotency, consider

$$(\rho \oplus \rho)(\phi) = \bigvee_{\phi \leq \phi_1 \oplus \phi_2} (\rho(\phi_1) \wedge \rho(\phi_2)) = \bigvee_{\phi \leq \phi_1 \oplus \phi_2} \rho(\phi_1 \oplus \phi_2) = \rho(\phi),$$

where the supremum in the last equation is obtained for $\phi = \phi_1 = \phi_2$.

The vacuous allocation ρ_ν is the neutral element, because for any allocation ρ we have

$$(\rho \oplus \rho_\nu)(\phi) = \bigvee_{\phi \leq \phi_1 \oplus \phi_2} (\rho(\phi_1) \wedge \rho_\nu(\phi_2)) = \bigvee_{\phi \leq \phi_1} \rho(\phi_1) = \rho(\phi).$$

This proves the theorem. □

2.4.2 Focusing of Allocations

In the information algebra (Φ, D) we have defined an operation of focusing, an element $\phi \in \Phi$ can be focused to a domain $x \in D$. The focusing operation carries over to allocation of probability. Let $\rho : \Phi \to \mathcal{B}$ be an allocation and x a domain in D. Focusing the information contained in ρ means now to select the part of ρ which allocates beliefs to those pieces of information whose support is x. Consider an information $\psi \in \Phi$ with support x, that is $\psi^{\Rightarrow x} = \psi$, and which contains more information than ϕ, $\psi \geq \phi$. The allocation ρ focused on x allocates then a belief to ϕ which is a least $\rho(\psi)$, therefore

$$\rho^{\Rightarrow x}(\phi) \quad \geq \quad \rho(\psi). \tag{2.26}$$

The belief allocated to ϕ is then defined to be the least upper bound of all these beliefs, that is

$$\rho^{\Rightarrow x}(\phi) \quad := \quad \bigvee_{\psi = \psi^{\Rightarrow x} \geq \phi} \rho(\psi). \tag{2.27}$$

The following theorem shows that this defines indeed an allocation of probability:

Theorem 2.20 *(2.27) defines a focusing operation in the set P_Φ with respect to D, i.e. $\rho^{\Rightarrow x}$ is an allocation of probability.*

Proof of theorem 2.20 We have to prove that (2.27) defines an allocation of probability, therefore we have to prove that $\rho^{\Rightarrow x}$ satisfies the axioms (R1) and (R2) on page 18. By definition,

$$\rho^{\Rightarrow x}(e) \quad = \quad \bigvee_{\psi = \psi^{\Rightarrow x} \geq e} \rho(\psi) \quad = \quad \rho(e) \quad = \quad \top$$

because of the monotonicity of ρ, and this proves (R1).

For (R2) by definition,

$$\rho^{\Rightarrow x}(\phi_1 \oplus \phi_2) \quad = \quad \bigvee_{\psi = \psi^{\Rightarrow x} \geq \phi_1 \oplus \phi_2} \rho(\psi).$$

For $i = 1, 2$ we have $\phi_i \leq \phi_1 \oplus \phi_2$, therefore $\rho^{\Rightarrow x}(\phi_1 \oplus \phi_2) \leq \rho^{\Rightarrow x}(\phi_i)$, such that $\rho^{\Rightarrow x}(\phi_1 \oplus \phi_2) \leq \rho^{\Rightarrow x}(\phi_1) \wedge \rho^{\Rightarrow x}(\phi_2)$.

On the other hand,

$$\begin{aligned}&\{\psi : \psi^{\Rightarrow x} = \psi \geq \phi_1 \oplus \phi_2\} \\ &\qquad \supseteq \{\psi = \psi_1 \oplus \psi_2 : {\psi_1}^{\Rightarrow x} = \psi_1 \geq \phi_1,\, {\psi_2}^{\Rightarrow x} = \psi_2 \geq \phi_2\}.\end{aligned}$$

This set inclusion together with (R2) for ρ and lemma 2.15 implies then

$$\begin{aligned}\rho^{\Rightarrow x}(\phi_1 \oplus \phi_2) \quad &\geq \quad \bigvee_{\substack{\psi_1 = {\psi_1}^{\Rightarrow x} \geq \phi_1 \\ \psi_2 = {\psi_2}^{\Rightarrow x} \geq \phi_2}} \rho(\psi_1 \oplus \psi_2) \quad = \quad \bigvee_{\substack{\psi_1 = {\psi_1}^{\Rightarrow x} \geq \phi_1 \\ \psi_2 = {\psi_2}^{\Rightarrow x} \geq \phi_2}} (\rho(\psi_1) \wedge \rho(\psi_2)) \\ &= \left(\bigvee_{\psi_1 = {\psi_1}^{\Rightarrow x} \geq \phi_1} \rho(\psi_1) \right) \wedge \left(\bigvee_{\psi_2 = {\psi_2}^{\Rightarrow x} \geq \phi_2} \rho(\psi_2) \right) \\ &= \quad \rho^{\Rightarrow x}(\phi_1) \wedge \rho^{\Rightarrow x}(\phi_2).\end{aligned}$$

So finally $\rho^{\Rightarrow x}(\phi_1 \oplus \phi_2) = \rho^{\Rightarrow x}(\phi_1) \wedge \rho^{\Rightarrow x}(\phi_2)$, therefore (R2) is fulfilled and the theorem proved. □

The following lemma shows that every allocation has a support in D. Intuitively, this means that for every allocation, there exists a domain in D in which it can be described.

Lemma 2.21 *$\rho^{\Rightarrow \top} = \rho$ for every $\rho \in P_\Phi$.*

Proof of lemma 2.21 Let $\phi \in \Phi$. By definition,

$$\rho^{\Rightarrow \top}(\phi) = \bigvee_{\psi = \psi^{\Rightarrow \top} \geq \phi} \rho(\psi) = \bigvee_{\psi \geq \phi} \rho(\psi) = \rho(\phi)$$

because $\top$ is a support for every $\psi \in \Phi$ (cf. (A7)), i.e. $\psi^{\Rightarrow \top} = \psi$, and ρ is monotone. □

A straightforward result is that focussing does not add any information to an allocation:

Lemma 2.22 *Let ρ be an allocation, then*

- *$\rho^{\Rightarrow x}(\phi) \leq \rho(\phi)$ for all $\phi \in \Phi$.*
- *$\phi = \phi^{\Rightarrow x}$ implies $\rho^{\Rightarrow x}(\phi) = \rho(\phi)$.*

Proof of lemma 2.22 Follows from the definition (2.27). □

2.4.3 Unmarked Information Algebra of Allocations

The set of allocations P_Φ together with the operations of combination and focusing forms now itself an unmarked information algebra:

Theorem 2.23 *(P_Φ, D) is an unmarked information algebra.*

Proof of theorem 2.23 (A1), (A2), and (A3) have been proved in theorem 2.19, (A7) in lemma 2.21. So we have to prove (A4), (A5) and (A6).

(A4) Transitivity: For $\rho \in P_\Phi$, $x, y \in D$: $(\rho^{\Rightarrow x})^{\Rightarrow y} = \rho^{\Rightarrow x \wedge y}$.

Using the definition (2.27) together with lemma 2.15,

$$\begin{aligned}(\rho^{\Rightarrow x})^{\Rightarrow y}(\phi) &= \bigvee_{\psi = \psi^{\Rightarrow y} \geq \phi} \rho^{\Rightarrow x}(\psi) = \bigvee_{\psi = \psi^{\Rightarrow y} \geq \phi} \left\{ \bigvee_{\xi = \xi^{\Rightarrow x} \geq \psi} \rho(\xi) \right\} \\ &= \bigvee_{\xi = \xi^{\Rightarrow x} \geq \psi = \psi^{\Rightarrow y} \geq \phi} \rho(\xi)\end{aligned}$$

and

$$\rho^{\Rightarrow x \wedge y}(\phi) = \bigvee_{\psi = \psi^{\Rightarrow x \wedge y} \geq \phi} \rho(\psi) = \bigvee_{\psi = (\psi^{\Rightarrow x})^{\Rightarrow y} \geq \phi} \rho(\psi).$$

Because $\psi = (\psi^{\Rightarrow x})^{\Rightarrow y}$ implies $\psi^{\Rightarrow y} = ((\psi^{\Rightarrow x})^{\Rightarrow y})^{\Rightarrow y} = (\psi^{\Rightarrow x})^{\Rightarrow y} = \psi$ and, similar, $\psi = \psi^{\Rightarrow x}$ we have

$$\{\psi : \psi = (\psi^{\Rightarrow x})^{\Rightarrow y} \geq \phi\} \subseteq \{\xi : \xi = \xi^{\Rightarrow x} \geq \psi = \psi^{\Rightarrow y} \geq \phi\}$$

and therefore $(\rho^{\Rightarrow x})^{\Rightarrow y}(\phi) \geq \rho^{\Rightarrow x \wedge y}(\phi)$.

$\xi = \xi^{\Rightarrow x} \geq \psi = \psi^{\Rightarrow y} \geq \phi$ implies $\xi^{\Rightarrow y} = (\xi^{\Rightarrow x})^{\Rightarrow y} \geq \psi^{\Rightarrow y} \geq \phi$. This together with $(\xi^{\Rightarrow x})^{\Rightarrow y} = (\xi^{\Rightarrow y})^{\Rightarrow x}$ gives then

$$(\rho^{\Rightarrow x})^{\Rightarrow y}(\phi) \leq \bigvee_{\xi^{\Rightarrow y} = (\xi^{\Rightarrow y})^{\Rightarrow x} \geq \phi} \rho(\xi) \leq \bigvee_{\xi^{\Rightarrow y} = (\xi^{\Rightarrow y})^{\Rightarrow x} \geq \phi} \rho(\xi^{\Rightarrow y}) \tag{2.28}$$

since $\xi^{\Rightarrow y} \leq \xi$ implies $\rho(\xi) \leq \rho(\xi^{\Rightarrow y})$ by lemma 2.17. Replacing $\xi^{\Rightarrow y}$ by ψ in (2.28), such that $\psi^{\Rightarrow y} = (\xi^{\Rightarrow y})^{\Rightarrow y} = \xi^{\Rightarrow y} = \psi$, we get

$$(\rho^{\Rightarrow x})^{\Rightarrow y}(\phi) \leq \bigvee_{\psi = \psi^{\Rightarrow y} = (\psi^{\Rightarrow y})^{\Rightarrow x} \geq \phi} \rho(\psi) = \bigvee_{\psi = \psi^{\Rightarrow x \wedge y} \geq \phi} \rho(\psi) = \rho^{\Rightarrow x \wedge y}(\phi),$$

such that finally $(\rho^{\Rightarrow x})^{\Rightarrow y} = \rho^{\Rightarrow x \wedge y}$ and this proves the transitivity.

(A5) Combination: For $\rho_1, \rho_2 \in P_\Phi$, $x \in D$: $({\rho_1}^{\Rightarrow x} \oplus \rho_2)^{\Rightarrow x} = {\rho_1}^{\Rightarrow x} \oplus {\rho_2}^{\Rightarrow x}$.

Using the definitions (2.24) and (2.27) together with lemma 2.15,

$$\begin{aligned}
({\rho_1}^{\Rightarrow x} \oplus {\rho_2}^{\Rightarrow x})(\phi) &= \bigvee_{\phi \leq \phi_1 \oplus \phi_2} ({\rho_1}^{\Rightarrow x}(\phi_1) \wedge {\rho_2}^{\Rightarrow x}(\phi_2)) \\
&= \bigvee_{\phi \leq \phi_1 \oplus \phi_2} \left(\bigvee_{\psi_1 = {\psi_1}^{\Rightarrow x} \geq \phi_1} \rho_1(\psi_1) \right) \wedge \left(\bigvee_{\psi_2 = {\psi_2}^{\Rightarrow x} \geq \phi_2} \rho_2(\psi_2) \right) \\
&= \bigvee_{\substack{\psi_1 = {\psi_1}^{\Rightarrow x} \geq \phi_1 \\ \psi_2 = {\psi_2}^{\Rightarrow x} \geq \phi_2 \\ \phi \leq \phi_1 \oplus \phi_2}} (\rho_1(\psi_1) \wedge \rho_1(\psi_2)) \quad = \bigvee_{\substack{\psi_1 = {\psi_1}^{\Rightarrow x} \\ \psi_2 = {\psi_2}^{\Rightarrow x} \\ \phi \leq \psi_1 \oplus \psi_2}} (\rho_1(\psi_1) \wedge \rho_1(\psi_2)),
\end{aligned}$$

and, also by definition,

$$({\rho_1}^{\Rightarrow x} \oplus \rho_2)(\phi) = \bigvee_{\phi \leq \phi_1 \oplus \phi_2} ({\rho_1}^{\Rightarrow x}(\phi_1) \wedge \rho_2(\phi_2)).$$

Using lemma 2.15, this gives

$$\begin{aligned}
({\rho_1}^{\Rightarrow x} \oplus \rho_2)^{\Rightarrow x}(\phi) &= \bigvee_{\phi \leq \psi = \psi^{\Rightarrow x}} \left(\bigvee_{\psi \leq \phi_1 \oplus \phi_2} ({\rho_1}^{\Rightarrow x}(\phi_1) \wedge \rho_2(\phi_2)) \right) \\
&= \bigvee_{\phi \leq \psi = \psi^{\Rightarrow x} \leq \phi_1 \oplus \phi_2} ({\rho_1}^{\Rightarrow x}(\phi_1) \wedge \rho_2(\phi_2)) \\
&= \bigvee_{\phi \leq \psi = \psi^{\Rightarrow x} \leq \phi_1 \oplus \phi_2} \left(\left(\bigvee_{\phi_1 \leq \psi_1 = {\psi_1}^{\Rightarrow x}} \rho_1(\psi_1) \right) \wedge \rho_2(\phi_2) \right) \\
&= \bigvee_{\substack{\phi_1 \leq \psi_1 = {\psi_1}^{\Rightarrow x} \\ \phi \leq \psi = \psi^{\Rightarrow x} \leq \phi_1 \oplus \phi_2}} (\rho_1(\psi_1) \wedge \rho_2(\phi_2)) \quad = \bigvee_{\substack{\psi_1 = {\psi_1}^{\Rightarrow x} \\ \phi \leq \psi = \psi^{\Rightarrow x} \leq \psi_1 \oplus \psi_2}} (\rho_1(\psi_1) \wedge \rho_2(\psi_2))
\end{aligned}$$

Now

$$\begin{aligned}
&\{(\psi_1, \psi_2) : \psi_1 = {\psi_1}^{\Rightarrow x},\ \psi_2 = {\psi_2}^{\Rightarrow x},\ \phi \leq \psi_1 \oplus \psi_2\} \\
&\quad \subseteq \{(\psi_1, \psi_2) : \psi_1 = {\psi_1}^{\Rightarrow x},\ \phi \leq \psi = \psi^{\Rightarrow x} \leq \psi_1 \oplus \psi_2\}
\end{aligned}$$

because if $\psi_1 = {\psi_1}^{\Rightarrow x}$, $\psi_2 = {\psi_2}^{\Rightarrow x}$ and $\phi \leq \psi_1 \oplus \psi_2$ then, setting $\psi := \psi_1 \oplus \psi_2$, we have $\psi^{\Rightarrow x} = (\psi_1 \oplus \psi_2)^{\Rightarrow x} = {\psi_1}^{\Rightarrow x} \oplus {\psi_2}^{\Rightarrow x} = \psi_1 \oplus \psi_2 = \psi$ and $\phi \leq \psi = \psi^{\Rightarrow x} \leq \psi_1 \oplus \psi_2$. This shows that

$$({\rho_1}^{\Rightarrow x} \oplus {\rho_2}^{\Rightarrow x})(\phi) \leq ({\rho_1}^{\Rightarrow x} \oplus \rho_2)^{\Rightarrow x}(\phi). \tag{2.29}$$

On the other hand, if $\psi_1 = {\psi_1}^{\Rightarrow x}$, $\phi \leq \psi = \psi^{\Rightarrow x} \leq \psi_1 \oplus \psi_2$, then $\psi = \psi^{\Rightarrow x} \leq (\psi_1 \oplus \psi_2)^{\Rightarrow x} = {\psi_1}^{\Rightarrow x} \oplus {\psi_2}^{\Rightarrow x} = \psi_1 \oplus {\psi_2}^{\Rightarrow x}$ and because $\rho_2(\psi_2) \leq \rho_2({\psi_2}^{\Rightarrow x})$, for every element ψ_2 we have $\rho_1(\psi_1) \wedge \rho_2(\psi_2) \leq \rho_1(\psi_1) \wedge \rho_2({\psi_2}^{\Rightarrow x})$, therefore

$$\begin{aligned}
({\rho_1}^{\Rightarrow x} \oplus \rho_2)^{\Rightarrow x} &\leq \bigvee_{\substack{\psi_1 = {\psi_1}^{\Rightarrow x} \\ \phi \leq \psi = \psi^{\Rightarrow x} \leq \psi_1 \oplus {\psi_2}^{\Rightarrow x}}} (\rho_1(\psi_1) \wedge \rho_2({\psi_2}^{\Rightarrow x})) \\
&= \bigvee_{\substack{\psi_1 = {\psi_1}^{\Rightarrow x} \\ \psi_2 = {\psi_2}^{\Rightarrow x} \\ \phi \leq \psi = \psi^{\Rightarrow x} \leq \psi_1 \oplus \psi_2}} (\rho_1(\psi_1) \wedge \rho_2(\psi_2)) \quad = \bigvee_{\substack{\psi_1 = {\psi_1}^{\Rightarrow x} \\ \psi_2 = {\psi_2}^{\Rightarrow x} \\ \phi \leq \psi_1 \oplus \psi_2}} (\rho_1(\psi_1) \wedge \rho_2(\psi_2)),
\end{aligned}$$

where the last equation follows from the fact that we may put $\psi = \psi_1 \oplus \psi_2$ which satisfies then $\psi^{\Rightarrow x} = (\psi_1 \oplus \psi_2)^{\Rightarrow x} = \psi_1 \oplus \psi_2 = \psi$. Therefore we have

$$({\rho_1}^{\Rightarrow x} \oplus \rho_2)^{\Rightarrow x} \leq ({\rho_1}^{\Rightarrow x} \oplus {\rho_2}^{\Rightarrow x})(\phi). \tag{2.30}$$

This proves the combination.

(A6) Idempotency: For $\rho \in P_\Phi$, $x \in D$: $\rho \oplus \rho^{\Rightarrow x} = \rho$.

By definition,

$$\begin{aligned}(\rho \oplus \rho^{\Rightarrow x})(\phi) &= \bigvee_{\phi \leq \phi_1 \oplus \phi_2} (\rho(\phi_1) \wedge \rho^{\Rightarrow x}(\phi_2)) \\ &\geq \rho(\phi) \wedge \rho^{\Rightarrow x}(e) \quad = \quad \rho(\phi).\end{aligned}$$

On the other hand,

$$\begin{aligned}(\rho \oplus \rho^{\Rightarrow x})(\phi) &= \bigvee_{\phi \leq \phi_1 \oplus \phi_2} \left(\rho(\phi_1) \wedge \left(\bigvee_{\phi_2 \leq \psi = \psi^{\Rightarrow x}} \rho(\psi)\right)\right) \\ &= \bigvee_{\substack{\phi_2 \leq \psi = \psi^{\Rightarrow x} \\ \phi \leq \phi_1 \oplus \phi_2}} (\rho(\phi_1) \wedge \rho(\psi)) \quad = \quad \bigvee_{\substack{\psi = \psi^{\Rightarrow x} \\ \phi \leq \phi_1 \oplus \psi}} \rho(\phi_1 \oplus \psi) \quad \leq \quad \rho(\phi)\end{aligned}$$

because $\phi \leq \phi_1 \oplus \psi$ implies $\rho(\phi) \geq \rho(\phi_1 \oplus \psi)$. This proves the idempotency. □

The information algebra (Φ, D) can be naturally embedded into (P_Φ, D) by identifying every element $\phi \in \Phi$ with a deterministic allocation $\rho_\phi : \Phi \rightarrow \mathcal{B}$ defined by

$$\rho_\phi(\psi) := \begin{cases} \top & \text{if } \psi \leq \phi, \\ \bot & \text{otherwise.} \end{cases} \tag{2.31}$$

It is easy to verify that ρ_ϕ is in P_Φ and for $\phi_1, \phi_2 \in \Phi$, $x \in D$,

$$\begin{aligned}\rho_{\phi_1} \oplus \rho_{\phi_2} &= \rho_{\phi_1 \oplus \phi_2} \\ (\rho_{\phi_1})^{\Rightarrow x} &= \rho_{({\phi_1}^{\Rightarrow x})}.\end{aligned}$$

2.5 Marked Algebra of Allocations

There are two different ways to define a marked algebra of allocations, either by defining marked allocations of probability directly, which is presented in the next subsection, or by applying the technique from subsection 2.1.3 (for constructing a marked from an unmarked algebra) to the unmarked algebra of allocations, which is presented in subsection 2.5.2. In subsection 2.5.3 we show that the results of the two approaches are essentially the same.

2.5.1 Marked Allocations of Probability

Consider the semigroup $\Phi_x := \{\phi \in \Phi : d(\phi) = x\}$ of pieces of information with label x in the marked information algebra $\langle \Phi, D \rangle$. This semigroup contains the neutral element e_x and (eventually) the null element z_x. An allocation of probability $\rho : \Phi_x \to \mathcal{B}$ which satisfies

(R1′) $\rho(e_x) = \top$.

(R2′) For $\phi_1, \phi_2 \in \Phi_x$, $\rho(\phi_1 \oplus \phi_2) = \rho(\phi_1) \wedge \rho(\phi_2)$

is called a **marked allocation of probability** with label $d(\rho) := x$ and is written $\langle \rho, x \rangle$.

A special case is the so-called **vacuous** allocation $\langle \nu_x, x \rangle$ which satisfies

$$\nu_x(\phi) = \begin{cases} \bot & \text{if } \phi \in \Phi_x,\ \phi \neq e_x \\ \top & \text{if } \phi = e_x. \end{cases} \tag{2.32}$$

The combination of two marked allocations is defined analogous to (2.24). Let $\langle \rho, x \rangle$ and $\langle \rho', y \rangle$ be two marked allocations, then the combination $\langle \rho \oplus \rho', x \vee y \rangle$ is defined by

$$(\rho \oplus \rho')(\phi) := \bigvee_{\substack{\phi \leq \phi_x \oplus \phi_y \\ d(\phi_x) = x,\, d(\phi_y) = y}} (\rho(\phi_x) \wedge \rho'(\phi_y)) \tag{2.33}$$

for $\phi \in \Phi_{x \vee y}$.

Let $\langle \rho, x \rangle$ be a marked allocation. The **marginal** $\langle \rho^{\downarrow y}, y \rangle$ of $\langle \rho, x \rangle$ with respect to a domain $y \in D$ with $y \leq x$ is defined by

$$\rho^{\downarrow y}(\phi) := \rho(\phi^{\uparrow x}) \tag{2.34}$$

for $\phi \in \Phi_y$.

Lemma 2.24 *$\langle \rho \oplus \rho', x \vee y \rangle$ and $\langle \rho^{\downarrow y}, y \rangle$ defined by (2.33) and (2.34) respectively are marked allocations of probability with domain $x \vee y$ and y respectively.*

Proof of lemma 2.24 First we have to prove that $\langle \rho \oplus \rho', x \vee y \rangle$ satisfies the axioms (R1′) and (R2′). But $d(\rho \oplus \rho') = x \vee y$, and the proof is analogous to the one of theorem 2.18.

Second we have to prove that $\langle \rho^{\downarrow y}, y \rangle$ satisfies the axioms (R1′) and (R2′). By definition and lemma 2.8, $\rho^{\downarrow y}(e_y) = \rho({e_y}^{\uparrow x}) = \rho(e_x) = \top$, and therefore (R1′) is satisfied. For $\phi', \phi'' \in \Phi_y$, we have by lemma 2.8 and (R2′) for ρ,

$$\begin{aligned} \rho^{\downarrow y}(\phi' \oplus \phi'') &= \rho\left((\phi' \oplus \phi'')^{\uparrow x}\right) &&= \rho\left({\phi'}^{\uparrow x} \oplus {\phi''}^{\uparrow x}\right) \\ &= \rho({\phi'}^{\uparrow x}) \wedge \rho({\phi''}^{\uparrow x}) &&= \rho^{\downarrow y}(\phi') \wedge \rho^{\downarrow y}(\phi'') \end{aligned}$$

and this proves (R2′).

□

The vacuous extension $\langle \rho^{\uparrow y}, y \rangle$ of an allocation of probability $\langle \rho, x \rangle$ to a finer domain $y \geq x$ is defined analogous to definition 2.6:

$$\rho^{\uparrow y} := \rho \oplus \nu_y. \tag{2.35}$$

By lemma 2.24 the extension is still a marked allocation of probability. The definition of the combination (2.33) implies then that for $\phi \in \Phi_y$ we have

$$\rho^{\uparrow y}(\phi) = \bigvee \{\rho(\psi) : \psi \in \Phi_x,\, \phi \leq \psi^{\uparrow y}\}. \tag{2.36}$$

The interpretation of this equation is the following: The supremum of all beliefs allocated to pieces of information ψ, which have label x and which imply ϕ, is the belief allocated to ϕ by $\rho^{\uparrow y}$.

Denote by P'_Φ the set of marked allocations of probability defined on the information algebra $\langle \Phi, D \rangle$ as above. The next theorem shows that $\langle P'_\Phi, D \rangle$ is indeed a marked information algebra:

Theorem 2.25 *$\langle P'_\Phi, D \rangle$ is a marked information algebra where ν_x is the neutral element for the combination operation within the semigroup Φ_x of allocations with label $x \in D$.*

Proof of theorem 2.25 We have to show that the axioms (M1) to (M10) from definition 2.5 are fulfilled. Commutativity (M2), Labeling (M3), Label of the neutral element (M4), and Label of the marginal (M6) follow directly from the definitions (2.32), (2.33) and (2.34).

(M1) Associativity. Let $\langle \rho_x, x \rangle$, $\langle \rho_y, y \rangle$ and $\langle \rho_z, z \rangle$ three marked allocations and $\phi \in \Phi_{x \vee y \vee z}$, then using lemma 2.15

$$\begin{aligned}
((\rho_x \oplus \rho_y) \oplus \rho_z)(\phi) &= \bigvee_{\substack{\phi \leq \phi_{xy} \oplus \phi_z \\ d(\phi_{xy}) = x \vee y,\, d(\phi_z) = z}} ((\rho_x \oplus \rho_y)(\phi_{xy}) \wedge \rho_z(\phi_z)) \\
&= \bigvee_{\substack{\phi \leq \phi_{xy} \oplus \phi_z \\ d(\phi_{xy}) = x \vee y,\, d(\phi_z) = z}} \left(\left(\bigvee_{\substack{\phi_{xy} \leq \phi_x \oplus \phi_y \\ d(\phi_x) = x,\, d(\phi_y) = y}} (\rho_x(\phi_x) \wedge \rho_y(\phi_y)) \right) \wedge \rho_z(\phi_z) \right) \\
&= \bigvee_{\substack{\phi \leq \phi_x \oplus \phi_y \oplus \phi_z \\ d(\phi_x) = x,\, d(\phi_y) = y,\, d(\phi_z) = z}} (\rho_x(\phi_x) \wedge \rho_y(\phi_y) \wedge \rho_z(\phi_z))
\end{aligned}$$

and, by symmetry, this proves the associativity.

(M5) Neutral element. For an allocation $\langle \rho, x \rangle$, the vacuous allocation $\langle \nu_x, x \rangle$, and $\phi \in \Phi_x$, we have

$$(\rho \oplus \nu_x)(\phi) = \bigvee_{\substack{\phi \leq \phi' \oplus \phi'' \\ d(\phi') = d(\phi'') = x}} (\rho(\phi') \wedge \nu_x(\phi'') = \bigvee_{\phi \leq \phi',\, d(\phi') = x} \rho(\phi') \quad = \quad \rho(\phi).$$

This proves (M5).

(M7) Transitivity. Let $\langle \rho, z \rangle$ be an allocation, $x \leq y \leq z$ all in D, and $\phi \in \Phi_x$, then, by definition,

$$(\rho^{\downarrow y})^{\downarrow x}(\phi) = \rho^{\downarrow y}(\phi^{\uparrow y}) = \rho((\phi^{\uparrow y})^{\uparrow z}) = \rho(\phi^{\uparrow z}) = \rho^{\downarrow x}(\phi),$$

and this proves (M7).

(M8) Combination. Let $\langle \rho', x \rangle$ and $\langle \rho'', y \rangle$ be two allocations, then, for $\phi \in \Phi_x$,

$$(\rho' \oplus \rho'')^{\downarrow x}(\phi) = (\rho' \oplus \rho'')(\phi^{\uparrow x \vee y}) = \bigvee_{\substack{d(\phi')=x,\, d(\phi'')=y,\\ \phi^{\uparrow x \vee y} \leq \phi' \oplus \phi''}} (\rho'(\phi') \wedge \rho''(\phi''))$$

and also

$$(\rho' \oplus \rho''^{\downarrow x \wedge y})(\phi) = \bigvee_{\substack{d(\phi')=x,\, d(\phi'')=x \wedge y,\\ \phi \leq \phi' \oplus \phi''}} (\rho'(\phi') \wedge \rho''(\phi''^{\uparrow y})).$$

If $\phi^{\uparrow x \vee y} \leq \phi' \oplus \phi''$ then $\phi \leq (\phi' \oplus \phi'')^{\downarrow x} = \phi' \oplus \phi''^{\downarrow x \wedge y}$ and this implies then $(\rho' \oplus \rho'')^{\downarrow x}(\phi) \leq (\rho' \oplus \rho''^{\downarrow x \wedge y})(\phi)$. On the other hand, if $d(\phi') = x$, $d(\phi'') = x \wedge y$ and $\phi \leq \phi' \oplus \phi'' \leq \phi' \oplus \phi''^{\uparrow y}$ (lemma 2.11) then $\phi^{\uparrow x \vee y} = \phi \oplus e_{x \vee y} \leq \phi' \oplus \phi''^{\uparrow y} \oplus e_{x \vee y} = \phi' \oplus \phi''^{\uparrow y}$ and this implies then $(\rho' \oplus \rho'')^{\downarrow x}(\phi) \geq (\rho' \oplus \rho''^{\uparrow x \wedge y})(\phi)$. Therefore finally $(\rho' \oplus \rho'')^{\downarrow x}(\phi) = (\rho' \oplus \rho''^{\uparrow x \wedge y})(\phi)$ and this proves (M8).

(M9) Idempotency. Let $\langle \rho, y \rangle$ be an allocation, $\phi \in \Phi_y$ and $x \leq y$, then

$$\begin{aligned}
(\rho \oplus \rho^{\downarrow x})(\phi) &= \bigvee_{\substack{d(\phi')=y,\, d(\phi'')=x,\\ \phi \leq \phi' \oplus \phi''}} (\rho(\phi') \wedge \rho^{\downarrow x}(\phi'')) = \bigvee_{\substack{d(\phi')=y,\, d(\phi'')=x,\\ \phi \leq \phi' \oplus \phi''}} (\rho(\phi') \wedge \rho(\phi''^{\uparrow y})) \\
&= \bigvee_{\substack{d(\phi')=y,\, d(\phi'')=x,\\ \phi \leq \phi' \oplus \phi''}} (\rho(\phi' \oplus \phi''^{\uparrow y})) \leq \bigvee_{\substack{d(\phi')=y,\, d(\phi'')=x,\\ \phi \leq \phi' \oplus \phi''^{\uparrow y}}} (\rho(\phi' \oplus \phi''^{\uparrow y})) \\
&\leq \rho(\phi).
\end{aligned}$$

On the other hand, $\phi = \phi \oplus \phi^{\downarrow x} = \phi \oplus (\phi^{\downarrow x})^{\uparrow y}$, which implies $\rho(\phi) \leq (\rho \oplus \rho^{\downarrow x})(\phi)$. So finally $\rho(\phi) = (\rho \oplus \rho^{\downarrow x})(\phi)$ and this proves (M9).

(M10) Stability. Let $\phi \in \Phi_x$ and $x \leq y$. If $\phi \neq e_x$ then it follows that $\phi^{\uparrow y} \neq e_y$. Therefore, by definition,

$${\nu_y}^{\downarrow x}(\phi) = \nu_y(\phi^{\uparrow y}) = \perp.$$

Otherwise, if $\phi = e_x$, then ${e_x}^{\uparrow y} = e_y$, therefore $(\nu_y)^{\downarrow x}(e_x) = \nu_y({e_x}^{\uparrow y}) = \nu_y(e_y) = \top$. This show that $(\nu_y)^{\downarrow x} = \nu_x$ and proves therefore (M10). □

2.5.2 From the Unmarked to the Marked Algebra of Allocations

In section 2.1.3, we showed how to construct a marked from an unmarked information algebra. This can be applied to the unmarked information algebra

(P_Φ, D) as well. So we consider the marked information algebra $\langle P_\Phi, D\rangle$, with elements

$$\langle \rho, x\rangle \quad \text{such that } \rho \in P_\Phi \text{ and } x \in D \text{ is a support of } \rho. \tag{2.37}$$

Combination and marginalization are then defined by (2.8) and (2.9) respectively. So $\langle P_\Phi, D\rangle$ is clearly a marked information algebra. The question is now: What is the difference between this information algebra and the one defined in the previous subsection?

2.5.3 A Comparison of the Approaches

We compare now the two algebras $\langle P'_\Phi, D\rangle$ and $\langle P_\Phi, D\rangle$. They are clearly not equal because a typical element $\langle \rho, x\rangle$ of $\langle P_\Phi, D\rangle$ is not a marked allocation of probability as defined in subsection 2.5.1, therefore not contained in $\langle P'_\Phi, D\rangle$. But to an element $\langle \rho, x\rangle \in \langle P_\Phi, D\rangle$ we can associate a marked allocation

$$\rho_x(\langle \phi, x\rangle) := \rho(\phi) \quad \text{for } \langle \phi, x\rangle \in \langle \Phi, D\rangle. \tag{2.38}$$

Vice versa, to a marked allocation $\langle \rho', x\rangle$ in $\langle P'_\Phi, D\rangle$ we can associate an element $\langle (\rho')^*, x\rangle$ in $\langle P_\Phi, D\rangle$ by

$$(\rho')^*(\phi) := \bigvee_{\psi = \psi^{\Rightarrow x} \geq \phi} \rho'(\langle \psi, x\rangle) \quad \text{for } \phi \in (\Phi, D). \tag{2.39}$$

These two mappings define indeed an isomorphism between the two marked information algebras:

Theorem 2.26 *The information algebras $\langle P_\Phi, D\rangle$ and $\langle P'_\Phi, D\rangle$ are isomorphic. An isomorphism is given by the mappings defined in (2.38) and (2.39), that is for $\langle \rho, x\rangle \in \langle P_\Phi, D\rangle$*

$$(\rho_x)^* = \rho \tag{2.40}$$

and for $\langle \rho', x\rangle \in \langle P'_\Phi, D\rangle$

$$((\rho')^*)_x = \rho', \tag{2.41}$$

and combination as well as focusing carry over: if $\langle \rho, x\rangle$ and $\langle \xi, y\rangle$ belong to $\langle P_\Phi, D\rangle$ and $z \leq x$, $z \in D$, then

$$\rho_x \oplus \xi_y = (\rho \oplus \xi)_{x \vee y} \tag{2.42}$$

$$(\rho^{\Rightarrow z})_z = {\rho_x}^{\downarrow z}. \tag{2.43}$$

Proof of theorem 2.26 First we prove (2.40), so let $\langle \rho, x\rangle \in \langle P_\Phi, D\rangle$ and $\phi \in (\Phi, D)$. Then, by definition,

$$(\rho_x)^*(\phi) = \bigvee_{\psi = \psi^{\Rightarrow x} \geq \phi} \rho_x(\langle \psi, x\rangle) = \bigvee_{\psi = \psi^{\Rightarrow x} \geq \phi} \rho(\psi) = \rho^{\Rightarrow x}(\phi) = \rho(\phi),$$

where the last equation follows from the fact that x is a support for ρ, $\rho^{\Rightarrow x} = \rho$. This proves (2.40).

Now, we show (2.41), so let $\langle \rho', x\rangle \in \langle P'_\Phi, D\rangle$ and $\langle \phi, x\rangle \in \langle \Phi, D\rangle$. Then, by definition,

$$((\rho')^*)_x(\langle \phi, x\rangle) = (\rho')^*(\phi) = \bigvee_{\psi=\psi^{\Rightarrow x}\geq\phi} \rho'(\langle \psi, x\rangle) = \rho'(\langle \phi, x\rangle),$$

where the last equation follows from the fact that $\langle \phi, x\rangle$ implies $\phi^{\Rightarrow x} = \phi$. This proves (2.41).

Now we prove (2.42). Let $\langle \phi, x \vee y\rangle \in \langle \Phi, D\rangle$. By assumption, x is a support for ρ and y a support for ξ, therefore for all $\phi \in (\Phi, D)$

$$\rho(\phi) = \bigvee_{\phi\leq\psi_x=\psi_x{}^{\Rightarrow x}} \rho(\psi_x), \qquad \xi(\phi) = \bigvee_{\phi\leq\psi_y=\psi_y{}^{\Rightarrow y}} \rho(\psi_y),$$

and this implies

$$\begin{aligned}
(\rho \oplus \xi)(\phi) &= \bigvee_{\phi\leq\phi'\oplus\phi''} \left(\bigvee_{\phi'\leq\psi_x=\psi_x{}^{\Rightarrow x}} \rho(\psi_x) \right) \wedge \left(\bigvee_{\phi''\leq\psi_y=\psi_y{}^{\Rightarrow y}} \xi(\psi_y) \right) \\
&= \bigvee_{\substack{\phi\leq\phi'\oplus\phi'' \\ \phi'\leq\psi_x=\psi_x{}^{\Rightarrow x} \\ \phi''\leq\psi_y=\psi_y{}^{\Rightarrow y}}} (\rho(\psi_x) \wedge \xi(\psi_y)) = \bigvee_{\substack{\phi\leq\psi_x\oplus\psi_y \\ \psi_x=\psi_x{}^{\Rightarrow x} \\ \psi_y=\psi_y{}^{\Rightarrow y}}} (\rho(\psi_x) \wedge \xi(\psi_y)) \\
&= (\rho_x \oplus \xi_y)(\langle \phi, x \vee y\rangle).
\end{aligned}$$

This proves (2.42).

Let now $\langle \rho, x\rangle \in \langle P_\Phi, D\rangle$, $z \leq x$ and $\phi' = \langle \phi, z\rangle \in \langle \Phi, D\rangle$. Then

$$\rho^{\Rightarrow z}(\phi) = \bigvee_{\phi\leq\psi=\psi^{\Rightarrow z}} \rho(\psi) = \rho(\phi),$$

since $\phi = \phi^{\Rightarrow z}$, and this implies $(\rho^{\Rightarrow z})_z(\phi') = \rho(\phi)$. On the other hand, if $\psi' = \langle \psi, x\rangle$, then $\rho_x(\psi') = \rho(\psi)$ and therefore

$$(\rho_x)^{\downarrow z}(\phi') = \rho_x(\phi'^{\uparrow x}) = \rho(\phi^{\Rightarrow x}) = \rho(\phi),$$

since $\phi = \phi^{\Rightarrow z} = \phi^{\Rightarrow x \wedge z} = (\phi^{\Rightarrow z})^{\Rightarrow x} = \phi^{\Rightarrow x}$. So finally $(\rho^{\Rightarrow z})_z = (\rho_x)^{\downarrow z}$ and this proves (2.43). □

2.6 Belief Functions

In this section, we present the connections between an allocation of probability and a belief function, that is how one concept can be constructed from the other one.

2.6.1 Constructing Belief Functions from Allocations of Probability

Given an allocation of probability ρ defined on an information algebra (Φ, D) into a probability algebra $(\mathcal{B}, \mu)$, we can define the **degree of credibility** of an information in Φ by considering its image under $\mu \circ \rho$. That means the information is just as credible as the belief allocated to it. Formally, for $\phi \in \Phi$, we define the mapping

$$\begin{array}{rccl} bel: & \Phi & \to & [0,1] \\ & \phi & \mapsto & bel(\phi) := \mu(\rho(\phi)), \end{array} \tag{2.44}$$

(see fig. 2.3).

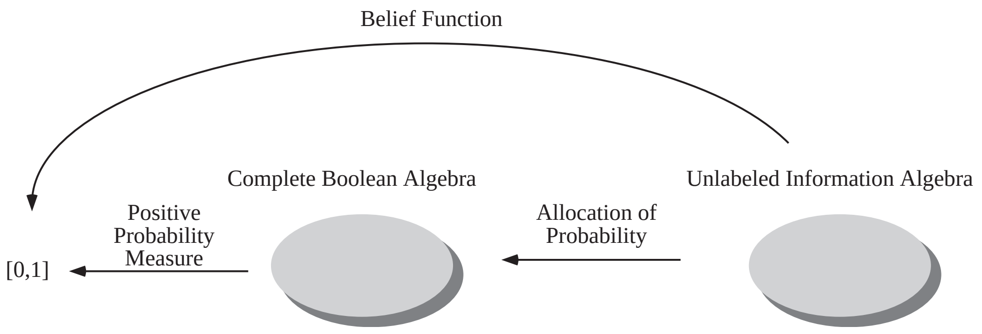

Belief Function
Complete Boolean Algebra
Unlabeled Information Algebra
Positive Probability Measure
Allocation of Probability
[0,1]

Figure 2.3: Belief function.

This mapping satisfies the axioms stated by Shafer (1976):

Theorem 2.27 *The mapping bel defined in (2.44) is a* ***belief function****, that is it satisfies the conditions (B1) and (B2):*

(B1) $bel(e) = 1$.

(B1′) $bel(z) = 0$, *if the allocation ρ is normalized.*

(B2) *Monotonicity of order ∞: If $\phi \in \Phi$, $n \geq 1$, and $\phi \leq \phi_i$, $\phi_i \in \Phi$ for $i = 1, \ldots, n$, then*

$$bel(\phi) \quad \geq \quad \sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} (-1)^{|I|+1} bel\left(\bigoplus_{i \in I} \phi_i\right). \tag{2.45}$$

Proof of theorem 2.27 Adapted from (Kohlas, 1995). By definition (2.44) and (R1),

$$bel(e) = \mu(\rho(e)) = \mu(\top) = 1,$$

and this proves (B1).

Now we prove (B2). $\phi \leq \phi_i$ implies $\rho(\phi) \geq \rho(\phi_i)$ for $i = 1, \ldots, n$, therefore $\rho(\phi) \geq \rho(\phi_1) \vee \cdots \vee \rho(\phi_n)$. Now, using the exclusion-inclusion formula from probability theory and (R2),

$$\begin{aligned} bel(\phi) &= \mu(\rho(\phi)) \geq \mu\left(\bigvee_{i=1}^{n} \rho(\phi_i)\right) \\ &= \sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} (-1)^{|I|+1} \mu\left(\bigwedge_{i \in I} \rho(\phi_i)\right) = \sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} (-1)^{|I|+1} \mu\left(\rho\left(\bigoplus_{i \in I} \phi_i\right)\right) \\ &= \sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} (-1)^{|I|+1} bel\left(\bigoplus_{i \in I} \phi_i\right), \end{aligned}$$

and this proves (B2).

Finally we prove (B1′). Let ρ be a normalized allocation. Then, using (R1′),

$$bel(z) = \mu(\rho(z)) = \mu(\perp) = 0$$

and this shows (B1′). □

This theorem shows how an allocation of probability on an information algebra induces a belief function. For the inverse case, Kohlas (1993a) gives an affirmative answer for the construction of an allocation of probability from a belief function, but only in a less general context, where the reference system is not an information algebra, but only a power set.

A belief function is called **vacuous**, if $bel(e) = 1$ and $bel(\phi) = 0$ for every $\phi \neq e$. This means that only the empty information e is believed, but nothing else. Clearly, a vacuous allocation of probability does induce a vacuous belief function.

If an information ϕ holds, then a **deterministic** belief function allocates the total belief to ϕ, and also to every information $\psi \leq \phi$, that is $bel(\psi) = 1$ for every $\psi \leq \phi$.

Theorem 2.27 induces a monotonicity property for belief functions:

Lemma 2.28 *The belief function is monotone, that is if $\phi_1 \leq \phi_2$ then $bel(\phi_1) \geq bel(\phi_2)$.*

Proof of lemma 2.28 Let $\phi_1 \leq \phi_2$, then because ρ is monotone (lemma 2.17) it follows that $\rho(\phi_1) \geq \rho(\phi_2)$ and because μ is a probability measure on $\mathcal{B}$ we have therefore $bel(\phi_1) = \mu \circ \rho(\phi_1) \geq \mu \circ \rho(\phi_2) = bel(\phi_2)$. □

In the important case of independent allocations of probability, the belief function constructed from the combined allocations can be computed directly from the two belief functions constructed from the individual allocations:

Theorem 2.29 *(Kohlas, 1995) Let bel_1 and bel_2 be two belief functions induced by two independent allocations of probability (section 2.3). Then the belief function bel of the combined allocations of probability satisfies*

$$bel(\phi) = \sup \left\{ \sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} (-1)^{|I|+1} bel_1 \left(\bigoplus_{i \in I} \phi_{1i} \right) bel_2 \left(\bigoplus_{i \in I} \phi_{2i} \right) \right\}, \tag{2.46}$$

where the supremum has to be taken over all finite sets $\{(\phi_{i1}, \phi_{i2}) : i = 1, \ldots, n;$ $n \geq 1\}$ such that $\phi \leq \phi_{i1} \oplus \phi_{i2}$.

This combination generalizes Dempster's rule in the Dempster-Shafer theory of evidence (Dempster, 1967; Shafer, 1976).

2.6.2 Belief Functions Induce Allocations of Probability

Given a belief function on an information algebra as defined in section 2.6, a corresponding allocation of probability can always be constructed using a fundamental result of Shafer (1979), see also (Kohlas, 1993a).

Theorem 2.30 *Given an unlabeled information algebra (Φ, D) where every element has a support, and a belief function $bel : (\Phi, D) \to [0, 1]$, there exist a probability algebra $(\mathcal{B}, \mu)$ and an allocation of probability $\rho : (\Phi, D) \to (\mathcal{B}, \mu)$ such that*

$$bel \quad = \quad \mu \circ \rho. \tag{2.47}$$

Note that an element ϕ of an unlabeled information algebra (Φ, D) has a support if there is a $x \in D$ such that $\phi^{\Rightarrow x} = \phi$.

The theorem is proved in the following subsection.

2.6.3 Proof of Theorem 2.30

Consider the unlabeled information algebra (Φ, D) where every element has a support, and a belief function $bel : (\Phi, D) \to [0, 1]$. Using techniques presented above, the corresponding marked information algebra $\langle \Phi, D \rangle$ is constructed. The belief function bel induces a belief function $bel_M : \langle \Phi, D \rangle \to [0, 1]$ by

$$bel_M(\langle \phi, x \rangle) \quad := \quad bel(\phi) \tag{2.48}$$

for every element $\langle \phi, x \rangle$ in $\langle \Phi, D \rangle$.

The marked information algebra $\langle \Phi, D \rangle$ can now be embedded in a so-called tuple system following results presented in (Kohlas & Stärk, 1996b).

Definition 2.31 *(Kohlas & Stärk, 1996b) A* ***tuple system*** *is a quadruple $\langle D, F, d, \cdot[\cdot] \rangle$ where D is a lattice, F a set, $d : F \to D$ and $\cdot[\cdot] : F \times D \to F$ functions which satisfy the following axioms for $f, g \in F$, and $x, y \in D$:*

(1) If $x \leq d(f)$, then $d(f[x]) = x$.

(2) If $x \leq y \leq d(f)$, then $f[y][x] = f[x]$.

(3) If $d(f) = x$, then $f[x] = f$.

(4) If $d(f) = x$, $d(y) = y$, and $f[x \wedge y] = g[x \wedge y]$, then there exists an $h \in F$ such that $d(h) = x$, $h[x] = f$, and $h[y] = g$.

Kohlas & Stärk (1996b) show that if $\langle \Phi, D \rangle$ is a marked information algebra, then $\langle \Phi, D, d, ^{\downarrow} \rangle$ is a tuple system. They show further how a tuple system generates an information algebra. Consider the tuple system $\langle \Phi, D, d, ^{\downarrow} \rangle$. A **relation** is a pair $[R, x]$ such that $R \subseteq \Phi$, $x \in D$, and $d(\phi) = x$ for every $\phi \in R$. For a relation $[R, x]$ and $y \leq x$, the **projection** of it onto y is defined by $\pi([R, x]) := \{\phi^{\downarrow y} : \phi \in R\}$. The **join** of two relations $[R, x]$ and $[S, y]$ is defined by $[R, x] \bowtie [S, y] := \{\phi \in \Phi : d(\phi) = x \vee y,\ \phi^{\downarrow x} \in R,\ \phi^{\downarrow y} \in S\}$. Further define $d([R, x]) := x$.

Let now R_Φ denote the set of all relations over Φ. Then $\langle R_\Phi, D \rangle$ is a marked information algebra generated by the tuple system $\langle \Phi, D, d, ^{\downarrow} \rangle$.

We started with the marked information algebra $\langle \Phi, D \rangle$. Kohlas & Stärk show that it can be embedded into $\langle R_\Phi, D \rangle$ using the following mapping I

$$\begin{array}{rcccl} I & : & \langle \Phi, D \rangle & \rightarrow & \langle R_\Phi, D \rangle \\ & & \langle \phi, x \rangle & \mapsto & I(\langle \phi, x \rangle) := \langle \{\psi \in \Phi : \phi \leq \psi,\ d(\psi) = x\}, x \rangle \end{array} \tag{2.49}$$

which satisfies

a) $d(\langle \phi, x \rangle) = d(I(\langle \phi, x \rangle))$

b) $I(\langle \phi, x \rangle \oplus \langle \psi, y \rangle) = I(\langle \phi, x \rangle) \bowtie I(\langle \psi, y \rangle)$.

c) If $y \leq x$, then $I(\langle \phi, x \rangle^{\downarrow y}) = \pi_y(I(\langle \phi, x \rangle))$.

d) $I(\phi) = I(\psi)$ implies $\phi = \psi$.

Define $I(\langle \Phi, D \rangle) := \{I(\langle \phi, x \rangle) : \langle \phi, x \rangle \in \langle \Phi, x \rangle\}$. The injectivity of I allows to transport the belief function bel_M to $I(\langle \Phi, D \rangle)$, which results in the function

$$bel_I(I(\langle \phi, x \rangle)) \quad := \quad bel_M(\langle \phi, x \rangle) \tag{2.50}$$

for $I(\langle \phi, x \rangle) \in I(\langle \Phi, D \rangle)$. Clearly, bel_I is again a belief function.

Lemma 2.32 *$I(\langle \Phi, D \rangle)$ is a multiplicative subclass of Φ with respect to the multiplication $\bowtie$, that is*

a) $I(\langle \Phi, D \rangle) \subseteq 2^{\langle \Phi, D \rangle}$, and

b) If $R_1, R_2 \in I(\langle \Phi, D \rangle)$, then $R_1 \bowtie R_2 \in I(\langle \Phi, D \rangle)$.

Proof of lemma 2.32 a) follows from the definition of I. For b), let $R_1, R_2 \in I(\langle\Phi, D\rangle)$. For $i = 1, 2$, there exists $\langle\phi_i, x_i\rangle \in \langle\Phi, D\rangle$ with $R_i = I(\langle\phi_i, x_i\rangle)$. The information algebra $\langle\Phi, D\rangle$ is closed under combination, hence we have $\langle\phi_1, x_1\rangle \oplus \langle\phi_1, x_2\rangle \in \langle\Phi, D\rangle$ and therefore, by the definition of I, $I(\langle\phi_1, x_1\rangle \oplus \langle\phi_2, x_2\rangle) \in I(\langle\Phi, D\rangle)$. Using the properties of I (page 35), this implies that $I(\langle\phi_1, x_1\rangle) \bowtie I(\langle\phi_2, x_2\rangle) = R_1 \bowtie R_2 \in I(\langle\Phi, D\rangle)$. □

Shafer (1979) shows how a belief function defined on a subclass induces an allocation of probability. We will not go into the details here, but use only the following result:

Theorem 2.33 *(Shafer, 1979) Suppose f is a belief function on a multiplicative subclass $\mathcal{E}$. Then there exists an allocation of probability $\rho : \mathcal{E} \to \mathcal{M}$ such that $f = \mu \circ \rho$, where μ is the measure associated with the probability algebra $\mathcal{M}$.*

In the present situation, this theorem can be applied to the multiplicative subclass $I(\langle\Phi, D\rangle)$ and the belief function bel_I, and we get

- a probability algebra $(\mathcal{B}, \mu)$, and
- an allocation of probability $\rho_I : I(\langle\Phi, D\rangle) \to (\mathcal{B}, \mu)$ such that $bel_I = \mu \circ \rho_I$.

The resulting allocation of probability ρ_I can be lifted back to the initial information algebra in two steps:

1. Lifting back to the marked information algebra $\langle\Phi, D\rangle$, that is for $\langle\phi, x\rangle \in \langle\Phi, D\rangle$ define
$$\rho_M(\langle\phi, x\rangle) \quad := \quad \rho_I(I(\langle\phi, x\rangle)). \tag{2.51}$$
2. Lifting back to the unmarked information algebra (Φ, D), that is for $\phi \in (\Phi, D)$ define
$$\rho(\phi) \quad := \quad \rho_M(\langle\phi, x\rangle) \tag{2.52}$$
for any support $x \in D$ of ϕ, that is for any x which satisfies $\phi^{\Rightarrow x} = \phi$.

The latter definition makes sense only if it is independent of the support of the information. This is proved in the following lemma:

Lemma 2.34 *ρ defined by (2.52) is an allocation of probability.*

Proof of lemma 2.34 $\rho(e) = \rho_M(\langle e, x\rangle) = \top$ because ρ_M is an allocation of probability. Further, for $\phi \in \Phi$ and $x, y \in D$ with $\phi = \phi^{\Rightarrow x} = \phi^{\Rightarrow y}$ we have $\rho_M(\langle\phi, x\rangle) = \rho_M(\langle\phi, x\rangle) \wedge \rho_M(\langle e, y\rangle) = \rho_M(\langle\phi, x\rangle \oplus \langle e, y\rangle) = \rho(\langle\phi, x \vee y\rangle) = \rho_M(\langle e, x\rangle \oplus \langle\phi, y\rangle) = \rho_M(\langle e, x\rangle) \wedge \rho_M(\langle\phi, y\rangle) = \rho_M(\langle\phi, y\rangle)$ and therefore $\rho(\phi)$ is well defined. Let $\phi_i \in (\Phi, D)$ with ${\phi_i}^{\Rightarrow x_i} = \phi_i$ for $i = 1, 2$. Then $\rho(\phi_1 \oplus \phi_2) = \rho_M(\langle\phi_1, x_1\rangle \oplus \langle\phi_2, x_2\rangle) = \rho_M(\langle\phi_1, x_1\rangle) \wedge \rho_M(\langle\phi_2, x_2\rangle) = \rho(\phi_1) \wedge \rho(\phi_2)$. □

Lemma 2.35 *$\mu \circ \rho = bel$.*

Proof of lemma 2.35 Let $\phi \in (\Phi, D)$ and $x \in D$ with $\phi^{\Rightarrow x} = \phi$. Then it follows from the previous results in this subsection that

$$\begin{aligned} \mu(\rho(\phi)) &= \mu(\rho_M(\langle \phi, x \rangle)) = \mu(\rho_I(I(\langle \phi, x \rangle))) \\ &= bel_I(I(\langle \phi, x \rangle)) = bel_M(\langle \phi, x \rangle) = bel(\phi). \end{aligned}$$

□

This concludes the prove of theorem 2.30.

3 Hints

In this chapter, the concept of a hint is introduced and also generalized. This concept will be a further ingredient for argumentation systems, a concept which is defined in chapter 6.

Note that in this whole chapter, we will work with unmarked information algebras as defined in section 2.1. Yet equivalent concepts can also be developed for marked information algebras using the results of the previous chapter. In the following chapters we will then use the theory of hints straightforwardly together with marked and unmarked information algebras.

In the first section we introduce the classical definition of a hint following (Kohlas & Monney, 1995), in section 3.2 this concept is extended to generalized hints. In section 3.3, the combination of generalized hints is discussed.

Hints will then be used to construct allocations of support in section 3.5. In section 3.6, we show how to combine and focus the allocations of support such that they form also an information algebra themselves. Finally, allocations of support are then used to construct allocations of probability in section 3.7.

3.1 Hints

The concept of a hint has been introduced in (Kohlas & Monney, 1995; Kohlas, 1995) based on Dempster's concept of multivalued mappings (Dempster, 1967), and we will here first introduce their definition. Let Θ be a set with the interpretation that every element therein represents a possible answer to a given question; Θ is also called the **frame of discernment**. A set Ω, finite or infinite, represents all possible interpretations of a given information, but it is unknown which of these interpretations is the right one. For every interpretation $\omega \in \Omega$ there is a set $\Gamma(\omega) \subseteq \Theta$ representing the subset in which the correct answer to the given question must be if the interpretation ω is the correct one. The elements of Ω are usually not equally likely, and this information is represented by a probability space $(\Omega, \mathcal{A}, P)$.

Definition 3.1 *A **hint** is a tuple $\mathcal{H} = (\Omega, \Gamma, \Theta, \mathcal{A}, P)$ where*

- *Ω is the set of possible interpretations (not necessarily finite),*
- *Θ is a set, called the frame of discernment,*
- *Γ is a multivalued mapping, called **focal mapping**, from Ω to Θ, and*
- *$(\Omega, \mathcal{A}, P)$ is a probability space.*

Different hints can then be combined to a new hint which contains all the information, and mechanisms are developed to answer questions (Kohlas & Monney, 1995), but here we will first focus on a generalization of the structure of a hint before we will come back to these question in the more general context. In order to distinguish different types of hints, we will say that a hint which respects the definition above is a *classical* hint.

3.2 Generalized Hints

In a classical hint, a focal set is an element of the power set 2^Θ. Based on hints, an abstract theory of argumentation has been developed by Monney (1994) by replacing the set of interpretations as well as 2^Θ by two complete lattices. Here, we go one step further by replacing 2^Θ by an unmarked information algebra (Φ, D), but leaving the set of interpretations unchanged. Therefore, the mapping $\Gamma : \Omega \to \Phi$ is not a multivalued mapping but a normal function, in contrast to the theory of hints as introduced in the previous section. If an interpretation $\omega \in \Omega$ is the correct one, then $\Gamma(\omega) \in \Phi$ is the corresponding information which is true, and every information $\phi \leq \Gamma(\omega)$ must clearly be true too.

Definition 3.2 *A **generalized hint** is a tuple $\mathcal{H} = (\Omega, \Gamma, \Phi, D, \mathcal{A}, P)$ where*

- *Ω is the set of possible interpretations (not necessarily finite),*
- *Γ is a mapping, called **focal mapping**, from Ω to the unmarked information algebra (Φ, D), and*
- *$(\Omega, \mathcal{A}, P)$ is a probability space.*

In some situations, we will consider hints without a probability space; such a structure will also be called a hint and denoted by a quadruple $\mathcal{H} = (\Omega, \Gamma, \Phi, D)$.

The images of Γ are called **focal information** of the hint. An interpretation $\omega \in \Omega$ is called **contradictory**, if its image is the null element z of the information algebra; such an interpretation cannot really be the correct one. A hint is called **normalized** if it does not contain any contradictory interpretation, that is $\Gamma(\omega) \neq z$ for every $\omega \in \Omega$. Non-normalized hints can always be turned into

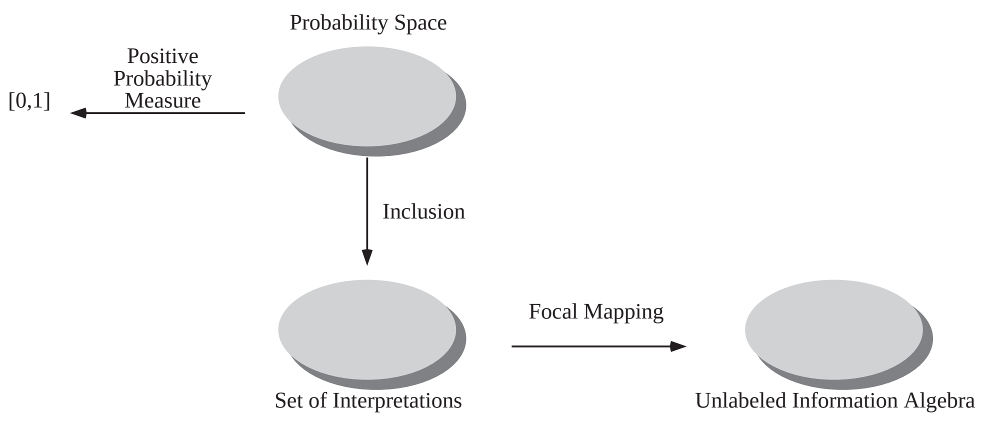


Figure 3.1: A generalized hint.

normalized ones by eliminating contradictory interpretations and conditioning the probability measure on the remaining interpretations; this process is called **normalization** and is described in the following definition:

Definition 3.3 (Normalization of a Hint) *Let $\mathcal{H} = (\Omega, \Gamma, \Phi, D, \mathcal{A}, P)$ be a hint. Then the normalized hint is defined by $\mathcal{H}' := (\Omega', \Gamma', \Phi, D, \mathcal{A}', P')$ where*

$$\begin{aligned} \Omega' &:= \{\omega \in \Omega : \Gamma(\omega) \neq z\} && (3.1)\\ \Gamma'(\omega') &:= \Gamma(\omega') \qquad \text{for } \omega' \in \Omega' && (3.2)\\ \mathcal{A}' &:= \{A \cap \Omega' : A \in \mathcal{A}\} && (3.3) \end{aligned}$$

We consider the probability space $(\Omega', \mathcal{A}', P')$ where the measure[1] P' is defined if Ω' is measurable with respect to $\mathcal{A}$ by

$$P'(A') := \frac{P(A')}{P(\Omega')} \quad \text{for } A' \in \mathcal{A}', \tag{3.4}$$

otherwise, if $P^*(\Omega') > 0$,by

$$P'(A') := \frac{P^*(A')}{P^*(\Omega')} \quad \text{for } A' \in \mathcal{A}', \tag{3.5}$$

where P^* denotes the outer measure with respect to measure P.

The case where $P^*(\Omega) = 0$ is not treated here, see (Monney, 2000) for a discussion of a special case.

Some special kinds of hints will be of interest, see also (Kohlas & Monney, 1995) for further details:

[1]See (Kohlas, 1995) for a proof that P' is a measure

- **Vacuous Hint $\mathcal{H}_e$:** Every focal information $\Gamma(\omega)$ consists of the empty information, that is $\Gamma(\omega) = e$. This means that the hint does not contain any information relative to the question.
- **Simple Hint:** There is at most one focal information different from the empty information e.
- **Consonant Hint:** The focal information can be numbered $F_1, F_2, \ldots$ and is nested $F_1 \geq F_2 \geq \cdots$. This means that the focal information points into the same direction.

Using the mapping Γ, the partial ordering "$\leq$" on the information algebra (Φ, D) can be transported to Ω. For $\omega_1, \omega_2 \in \Omega$, define

$$\omega_1 \preceq \omega_2 \quad \text{if} \quad \Gamma(\omega_1) \leq \Gamma(\omega_2). \tag{3.6}$$

$\omega_1 \preceq \omega_2$ reflects the fact that the information $\Gamma(\omega_2)$ implied by the interpretation ω_2 is "stronger" than $\Gamma(\omega_1)$ implied by ω_1.

Lemma 3.4 *"$\preceq$" is a partial order on the set of interpretations Ω.*

Proof of lemma 3.4 This follows from the definition (3.6) and the fact that "$\leq$" is a partial order on the information algebra (Φ, D). □

3.3 Combining Generalized Hints

The available information may define different hints on the same information algebra. These hints can emerge from different sources, for example one hint may contain the system description, another one some environment dependencies, a third one actual measurements, and so on. Several hints can be combined into a new hint containing all information. For every hint $\mathcal{H}_i$, $i = 1, \ldots, n$, there is an (unknown) correct interpretation $\omega_i \in \Omega_i$, so the correct interpretation of the combined hint $\mathcal{H}$ is the vector $(\omega_1, \ldots, \omega_n)$. The information which is true under this interpretation, that is the element in the information algebra (Φ, D), is then the combination of the corresponding focal information $\Gamma_1(\omega_1) \oplus \cdots \oplus \Gamma_n(\omega_n)$. So we have to look at all possible vectors in $\Omega_1 \times \cdots \times \Omega_n$. Yet some of the vectors may lead to an image which is the null element z in the information algebra, that is they are contradictory interpretations of the combined hint, and the resulting hint is therefore in general not normalized. Here we will usually work with not necessarily normalized hints. This has advantages especially for computations. The contradictory vectors can always be removed at any stage of the combinations using normalization (definition 3.3).

A vacuous hint $\mathcal{H}_e$ is clearly a neutral element of the combination of hints, that is for every hint $\mathcal{H}$ relative to the same information algebra as $\mathcal{H}_e$ we have $\mathcal{H} \oplus \mathcal{H}_e \sim \mathcal{H}$.

It can be shown that normalization and combination of hints are commutative in the sense that $(\mathcal{H}'_1 \oplus \mathcal{H}'_2)' = (\mathcal{H}_1 \oplus \mathcal{H}_2)'$ for two hints $\mathcal{H}_1$, $\mathcal{H}_2$ and normalization denoted by a prime.

3.3.1 Hints on the Same Probability Space and Information Algebra

A special case occurs when the hints, say $\mathcal{H}_i = (\Omega, \Gamma_i, \Phi, D, \mathcal{A}, P)$, $i = 1, 2$, to be combined share the same probability space $(\Omega, \mathcal{A}, P)$ and the same information algebra (Φ, D) and differ only in the focal mapping Γ_1 and Γ_2. In this case, the combined hint is defined with respect to the same probability space and information algebra, that is $\mathcal{H} = (\Omega, \Gamma, \Phi, D, \mathcal{A}, P)$ and Γ is the combination of the two focal mappings, that is $\Gamma(\omega) := \Gamma_1(\omega) \oplus \Gamma_2(\omega)$.

3.3.2 Hints on Different Probability Spaces

Let $\mathcal{H}_i = (\Omega_i, \Gamma_i, \Phi, D, \mathcal{A}_i, P_i)$ $i = 1, 2$ be two hints defined on the same information algebra but on different probability spaces. Let P be a measure on $\mathcal{A}_1 \times \mathcal{A}_2$ which reflects the common likelihood of combined interpretations and which has P_1 and P_2 as marginals. The two hints are called **independent** if the interpretations of the two hints are stochastically independent and therefore the probability measure P defined by the product of the measures P_1 and P_2. This case will be treated in the sequel:

Definition 3.5 (Combination of Independent Hints) *Consider two hints $\mathcal{H}_i = (\Omega_i, \Gamma_i, \Phi, D, \mathcal{A}_i, P_i)$, $i = 1, 2$. If they are independent, then their* ***combination****, $\mathcal{H} = \mathcal{H}_1 \oplus \mathcal{H}_2$, is a hint*

$$\mathcal{H} := (\Omega_1 \times \Omega_1, \Gamma, \Phi, D, \mathcal{A}_1 \times \mathcal{A}_2, P_1 P_2) \tag{3.7}$$

where

$$\Gamma((\omega_1, \omega_2)) \quad := \quad \Gamma_1(\omega_1) \oplus \Gamma_2(\omega_2) \quad \textit{for } \omega_j \in \Omega_j,\ j = 1, 2, \tag{3.8}$$

$P_1 P_2$ denotes the product measure of P_1 and P_2 (the hints are independent), therefore $(\Omega_1 \times \Omega_2, \mathcal{A}_1 \times \mathcal{A}_2, P_1 P_2)$ is a probability space.

The combined hint will not necessarily be normalized even if both hints it is combined from are normalized. Usually in the theory of hints (Kohlas & Monney, 1995), the combination operation is defined as first applying definition 3.5 and then normalizing its result by means of definition 3.3, in order to get a normalized hint as the result of the combination of two hints. This combination is called *Dempster's rule of combination*, because it was introduced by A. Dempster (1967) for multivalued mappings.

The above definition can also be extended to combine dependent hints, the only change being that the product of the measures $P_1 P_2$ on $\Omega_1 \times \Omega_2$ has to be replaced by a measure which reflects the dependencies between the interpretations of the hints being combined.

3.4 Transport or Focussing of Hints

The concepts of transport or focussing of hints as defined in (Kohlas & Monney, 1995) can be generalized to our framework as well. Consider a generalized hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D)$. The information contained in the focal mapping of this hint can be focussed or transported to a domain $x \in D$. So define the **focussing of hint** $\mathcal{H}$ by

$$\mathcal{H}^{\Rightarrow x} := (\Omega, \Gamma^{\Rightarrow x}, \Phi, D) \tag{3.9}$$

where $\Gamma^{\Rightarrow x}(\omega) := (\Gamma(\omega))^{\Rightarrow x}$.

From the definition, it follows that focussing is transitive, i.e. $(\mathcal{H}^{\Rightarrow x})^{\Rightarrow y} = \mathcal{H}^{\Rightarrow x \wedge y}$ and that combination is transitive over focussing, i.e. $(\mathcal{H}_1{}^{\Rightarrow x} \oplus \mathcal{H}_2)^{\Rightarrow x} = \mathcal{H}_1{}^{\Rightarrow x} \oplus \mathcal{H}_2{}^{\Rightarrow x}$.

3.5 Allocations of Support induced by Hints

Consider a hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D, \mathcal{A}, P)$ and a piece of information ϕ in the information algebra (Φ, D), also called a hypothesis. Some interpretations ω in Ω support the hypothesis, while some do not. In this section we define the concept of an allocation of support based on a given hint. For the hypothesis ϕ, such an allocation of support singles out a subset of all interpretations in Ω such that every member of this subset, if it holds, allows to deduce the hypothesis from the actual knowledge, and this hypothesis has therefore to be true.

In the sequel, we follow mainly the notation introduced in (Kohlas *et al.*, 2000).

3.5.1 Different Kinds of Interpretations

Interpretations, also called arguments, in favor of or against some hypotheses are a key point in diagnostics. Consider for example a digital circuit: one could be interested in the interpretations in favor or against the functioning of different sets of components in order to decide which components should be replaced. In this section, the definitions of several kinds of interpretations are introduced; in chapter 8 we will also focus on the computation of interpretations.

Consider a hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D, \mathcal{A}, P)$ and a hypothesis ϕ of the information algebra (Φ, D). An element $\omega \in \Omega$, that is an interpretation, is called an **quasi-supporting interpretation** for the hypothesis ϕ, if the information ω allows to "deduce" ϕ, that is if $\Gamma(\omega) \geq \phi$.

Note that if $\Gamma(\omega) = z$, then ω is a quasi-supporting interpretation for every information $\phi \in \Phi$ because z is the top element of the partial order "$\leq$", and such an ω is then especially a quasi-supporting interpretation for the information $z \in \Phi$; it is therefore called an **inconsistent** interpretation, because it is an interpretation which cannot be true. In a normalized hint there are clearly no inconsistent interpretations. An interpretation ω which is not inconsistent, that is $\Gamma(\omega) \neq z$, is called **consistent**.

In some cases, the user is interested in the "real" interpretations, that is those which are not inconsistent. An element $\omega \in \Omega$ is called called **supporting interpretation** for the hypothesis $\phi \in \Phi$, if it is a consistent quasi-supporting interpretation for ϕ, that is if

$$\Gamma(\omega) \geq \phi \quad \text{and} \quad \Gamma(\omega) \neq z. \tag{3.10}$$

Note that in a normalized hint, every quasi-supporting interpretation is also a supporting one.

Two other kinds of interpretations can be of interest. Consider those elements of Ω which are not quasi-supporting interpretations, but whose image under Γ is also not in contradiction with the hypothesis. Formally, an $\omega \in \Omega$ is called **possibly supporting interpretation** for the information $\phi \in \Phi$, if $\Gamma(\omega)$ is not in conflict with ϕ, that is if

$$\Gamma(\omega) \oplus \phi \neq z. \tag{3.11}$$

The opposite concept, i.e. elements of Ω whose images under Γ are contradicting with the hypothesis, are also said to refute the hypothesis. Such an interpretation represents a doubt one has in the hypothesis. Formally, an $\omega \in \Omega$ is called **refuting interpretation** of the information $\phi \in \Phi$, if $\Gamma(\omega)$ is in conflict with ϕ, that is if

$$\Gamma(\omega) \oplus \phi = z. \tag{3.12}$$

In fact, a refuting interpretation should really be called *quasi*-refuting, because it can also be an inconsistent interpretation. Based on this, we can define refuting and possibly refuting interpretations analogously to quasi-supporting interpretations and therefore get a duality between supporting and refuting. For more details on these concepts and the special case where a negation is available see (Kohlas *et al.*, 2000). In the sequel, we will usually focus on the "positive" interpretations, that is the three types of supporting interpretations.

Clearly, the union of the possibly supporting and the refuting interpretations is the whole set Ω, and the two sets of interpretations are disjoint.

3.5.2 Allocation of Support, Quasi-Support Sets

Let $\mathcal{H} = (\Omega, \Gamma, \Phi, D, \mathcal{A}, P)$ a hint and ϕ be an element of the unmarked information algebra (Φ, D). The **quasi-support set** $QSS(\phi)$ is then the set of all quasi-supporting interpretations for the current hypothesis ϕ. Formally the function QSS is defined by

$$\begin{array}{rrcl} QSS: & \Phi & \rightarrow & 2^{\Omega} \\ & \phi & \mapsto & QSS(\phi) := \{\omega \in \Omega : \; \Gamma(\omega) \geq \phi\}. \end{array} \tag{3.13}$$

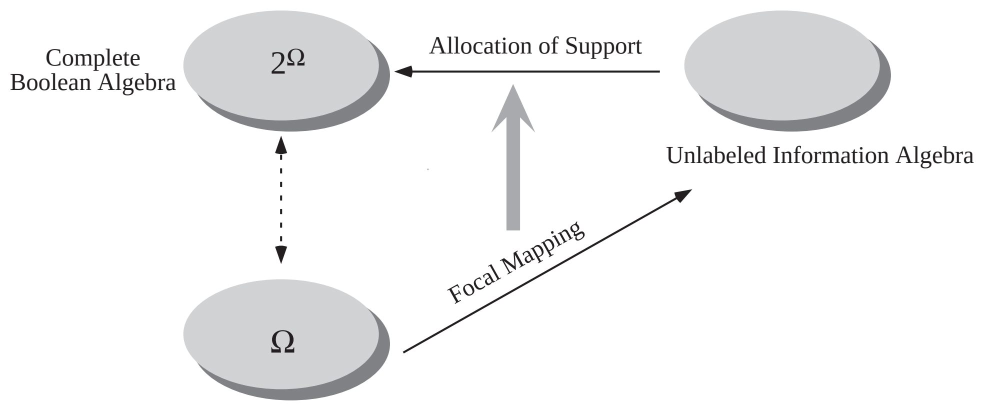


Figure 3.2: Allocation of support induced by a focal mapping.

In order to clarify the dependencies, the hint $\mathcal{H}$ which defines the allocation of quasi-support will sometimes be noted as a subscript, like $QSS_{\mathcal{H}}$.

This type of mapping is well known:

Theorem 3.6 *The mapping QSS is an **allocation of (quasi-) support**[2] (Besnard & Kohlas, 1995; Kohlas, 1997a) from the information algebra (Φ, D) into the Boolean algebra 2^Ω, i.e. for $\phi_1, \phi_2 \in \Phi$:*

$$
\begin{aligned}
QSS(e) &= \Omega, && (3.14)\\
QSS(\phi_1 \oplus \phi_2) &= QSS(\phi_1) \cap QSS(\phi_2). && (3.15)
\end{aligned}
$$

Proof of theorem 3.6 By definition, $QSS(e) = \{\omega \in \Omega : \Gamma(\omega) \geq e\}$. But, for every $\omega \in \Omega$, we have $\Gamma(\omega) \in \Phi$ and by corollary 2.3 $\Gamma(\omega) \geq e$. This proves (3.14).

$$
\begin{aligned}
QSS(\phi_1 \oplus \phi_2) &= \{\omega \in \Omega : \Gamma(\omega) \geq \phi_1 \oplus \phi_2\}\\
&= \{\omega \in \Omega : \Gamma(\omega) \geq \phi_1,\ \Gamma(\omega) \geq \phi_2\}\\
&= \{\omega \in \Omega : \Gamma(\omega) \geq \phi_1\} \cap \{\omega \in \Omega : \Gamma(\omega) \geq \phi_2\}\\
&= QSS(\phi_1) \cap QSS(\phi_2)
\end{aligned}
$$

and this proves (3.15). □

The allocation of quasi-support QSS is in general not normalized, that is $QSS(z)$ is empty only if the hint is normalized. From the definition it follows that every quasi-supporting interpretation for the impossible information z is also a quasi-supporting interpretation for every information $\phi \in \Phi$. Therefore the set $QSS(\phi)$ is called only a *quasi*-support set of ϕ (and not a support

[2]Allocations of support are also called meet homomorphisms or intersection homomorphisms (Shafer, 1979).

set, see below), because some of its members also support the impossible information. For the same reason we call QSS a quasi-support and not a support function.

Corollary 3.7 *The mapping QSS is monotone, that is for $\phi_1, \phi_2 \in \Phi$,*

$$\phi_1 \leq \phi_2 \qquad \textit{implies} \qquad QSS(\phi_1) \supseteq QSS(\phi_2), \tag{3.16}$$

and

$$QSS(\phi_1) \supseteq QSS(z). \tag{3.17}$$

Proof of corollary 3.7 $\phi_1 \leq \phi_2$ means by definition $\phi_1 \oplus \phi_2 = \phi_2$ and together with (3.15) we have $QSS(\phi_2) = QSS(\phi_1 \oplus \phi_2) = QSS(\phi_1) \cap QSS(\phi_2)$. This implies $QSS(\phi_2) \subseteq QSS(\phi_1)$.

The second equation follows from the first one with $\phi_2 = z$ and corollary 2.3.

□

The two mappings QSS and Γ are in general not inverse mappings, but have the following property:

Lemma 3.8 $\omega \in QSS(\Gamma(\omega))$ *for every* $\omega \in \Omega$*.*

Proof of lemma 3.8 Follows from the definition of QSS (3.13). □

A system $(\Phi, D, 2^\Omega, QSS)$ is then called a **body of arguments** (cf. section 2.3). It is called normalized if QSS is normalized. If, additionally, $(\Omega, \mathcal{A}, P)$ is a probability space, then $(\Phi, D, 2^\Omega, QSS, \mathcal{A}, P)$ is called a **body of evidence** (Kohlas & Brachinger, 1995; Kohlas, 1997b).

3.5.3 Support Sets

We have already mentioned that for $\phi \in \Phi$ the set $QSS(\phi)$ includes, besides others, the supporting interpretations for the impossible information z. Sometimes, we are interested in "proper" supports, i.e. we want to exclude the inconsistent interpretations. So for a formula $\phi \in \Phi$ define the **support set** $SS(\phi)$ as the set of all supporting interpretations,

$$\begin{aligned} SS(\phi) &:= QSS(\phi) - QSS(z) \\ &= \{\omega \in \Omega : \Gamma(\omega) \geq \phi,\, \Gamma(\omega) \neq z\}. \end{aligned} \tag{3.18}$$

This definition makes sense only in the case where $QSS(z) \neq \Omega$, that is when there are some consistent interpretations.

SS does also satisfy the conditions for an allocation of support, it is even a normalized allocation of support:

Corollary 3.9 *The mapping SS is a normalized* ***allocation of support*** *from the information algebra (Φ, D) into the Boolean algebra $2^{\Omega'}$ with $\Omega' = \{\omega \in \Omega : \Gamma(\omega) \neq z\}$, that is for $\phi_1, \phi_2 \in \Phi$:*

$$\begin{aligned} SS(e) &= \Omega', \\ SS(\phi_1 \oplus \phi_2) &= SS(\phi_1) \cap SS(\phi_2), \\ SS(z) &= \emptyset \end{aligned}$$

Proof of corollary 3.9 Follows from theorem 3.6 and the definition of SS (3.18).

□

Corollary 3.10 *The mapping SS is monotone, that is for $\phi_1, \phi_2 \in \Phi$,*

$$\phi_1 \leq \phi_2 \qquad \textit{implies} \qquad SS(\phi_1) \supseteq SS(\phi_2). \tag{3.19}$$

Proof of corollary 3.10 Follows from the definition (3.18) of SS and corollary 3.7.

□

The functions QSS and SS defined with respect to a normalized hint are clearly identical, because there are no conflicting interpretations.

Consider now a body of evidence $(\Phi, D, 2^{\Omega}, QSS, \mathcal{A}, P)$. As described above, we can define a normalized allocation of support SS. Further, the measure P is considered as a prior measure reflecting the situation before we learn that the interpretations $QSS(z)$ are conflicting. So this additional information is used to condition the probability measure, i.e. the probability space $(\Omega, \mathcal{A}, P)$ is normalized to a new probability space $(\Omega', \mathcal{A}', P')$ using the same technique as described in definition 3.3 for normalization of generalized hints, that is

$$\Omega' := \Omega - QSS(z) \tag{3.20}$$
$$\mathcal{A}' := \{A \cap \Omega' : A \in \mathcal{A}\} \tag{3.21}$$

and the measure P' is defined if Ω' is measurable with respect to $\mathcal{A}$ by

$$P'(A') := \frac{P(A')}{P(\Omega')} \quad \text{for } A' \in \mathcal{A}', \tag{3.22}$$

otherwise, if $P^*(\Omega') > 0$,by

$$P'(A') := \frac{P^*(A')}{P^*(\Omega')} \quad \text{for } A' \in \mathcal{A}', \tag{3.23}$$

where P^* denotes the outer measure with respect to measure P.[3]

Therefore we get a normalized body of evidence $(\Phi, D, 2^{\Omega'}, SS, \mathcal{A}', P')$.

[3]For the case $P^*(\Omega') = 0$ see the remark after definition 3.3.

3.5.4 Possibly Supporting and Refuting Sets

For $\phi \in \Phi$ define the **possibly supporting set** $PSS(\phi)$ as the set of all possibly supporting interpretations,

$$PSS(\phi) \quad := \quad \{\omega \in \Omega : \phi \oplus \Gamma(\omega) \neq z\}, \tag{3.24}$$

and the **refuting set** $RS(\phi)$ as the set of all refuting interpretations,

$$RS(\phi) \quad := \quad \{\omega \in \Omega : \phi \oplus \Gamma(\omega) = z\}. \tag{3.25}$$

The possibly supporting set and the refuting set are disjoint, and their union is the whole set Ω. But in contrast to QSS and SS, the mappings PSS and RS are not additive, that is they do both not satisfy a condition analogous to (3.15). Therefore they are both clearly not allocations of support (cf. theorem 3.6). Further, they both are not allowments of possibility[4] (Besnard & Kohlas, 1995; Kohlas, 1997a), that is $PSS(\phi_1 \oplus \phi_2)$ is in general not equal to $PSS(\phi_1) \cup PSS(\phi_2)$ and analogously for RS. In general, only the following relations hold:

Lemma 3.11 *For $\phi_1, \phi_2 \in \Phi$,*

$$\begin{aligned} PSS(\phi_1 \oplus \phi_2) \quad &\subseteq \quad PSS(\phi_1) \cap PSS(\phi_1), & (3.26)\\ RS(\phi_1 \oplus \phi_2) \quad &\supseteq \quad RS(\phi_1) \cup RS(\phi_1). & (3.27)\end{aligned}$$

Proof of lemma 3.11 Follows from the definitions of PSS (3.24) and RS (3.25). □

3.5.5 Conflicts and Diagnoses

In subsection 3.5.1, we introduced the distinction between consistent and inconsistent interpretations. In chapter 6, this distinction will get a meaning in the context of model-based diagnostic. Here, we define the two fundamental sets which will be used there, that is the so-called **conflicts** CS and **diagnoses** DS:

$$\begin{aligned} CS \quad &:= \quad \{\omega \in \Omega : \omega \text{ is an inconsistent interpretation}\} & (3.28)\\ &= \quad \{\omega \in \Omega : \Gamma(\omega) = z\} \\ DS \quad &:= \quad \{\omega \in \Omega : \omega \text{ is a consistent interpretation}\} & (3.29)\\ &= \quad \{\omega \in \Omega : \Gamma(\omega) \neq z\}\end{aligned}$$

Clearly, these two sets are complementary with respect to the state space, that is $\Omega - CS = DS$.

The following lemma shows the relation between quasi-support sets, the conflicts, and the diagnoses:

[4]Allowments of possibility are also called union homomorphisms (Shafer, 1979).

Lemma 3.12 *$CS = QSS(z)$ and $DS = SS(e)$.*

Proof of lemma 3.12 Follows from the definition of QSS (3.13) and of SS (3.18).
□

Clearly, the representation of these two concepts as subsets of Ω is not very handy, or often not even possible due to the size of Ω. In chapter 5, other representations of the conflicts and diagnoses are studied, in chapter 8 we address the computation of conflicts and diagnoses.

Every interpretation in the possibly supporting set does not contradict the hypothesis, so especially it is not a member of the conflicts:

Lemma 3.13 *$PSS(\phi) \subseteq DS$ for every $\phi \in \Phi$.*

Proof of lemma 3.13 Follows from the definitions of DS (3.29) and PSS (3.24).
□

Every member of the conflicts is clearly contained in the refuting set for every hypothesis $\phi \in \Phi$, therefore:

Lemma 3.14 *$CS \subseteq RS(\phi)$ for every $\phi \in \Phi$.*

Proof of lemma 3.14 Follows from the definitions of CS (3.28) and RS (3.25).
□

3.6 Information Algebra of Allocations of Quasi-Support

In this section, we will first consider the allocations of support from the information algebra (Φ, D) into the power set 2^{Ω}, and we show that this indeed is an information algebra. We denote the set of all these allocations of support by S_{Φ}. Note that all these allocations are defined on the same information algebra (Φ, D) and with respect to the same probability space $(\Omega, \mathcal{A}, P)$.

The combination of allocations defined on the same set of interpretations is presented in subsection 3.6.1. In subsection 3.6.2 the focusing of allocations is applied to the computation of quasi-supporting interpretations.

In subsection 3.6.3 a generalization of the combination operation is presented, namely the combination of allocations on different sets of interpretations.

Note that every allocation of support is also an allocation of quasi-support; therefore the definition and result presented hereafter apply also to allocations of support.

3.6.1 Combining and Focussing Allocations of Quasi-Support

Consider two allocations of quasi-support, that is they both satisfy the two conditions from theorem 3.6. The combination of these allocations follows from the combination of the underlying hints:

Theorem 3.15 *For $i = 1, 2$, let $\mathcal{H}_i = (\Omega, \Gamma_{\mathcal{H}_i}, \Phi, D)$ be a generalized hint and $QSS_{\mathcal{H}_i}$ the corresponding allocation of quasi-support. Then, the allocation of quasi-support $QSS_{\mathcal{H}_1 \oplus \mathcal{H}_2}$ generated by the combined hint $\mathcal{H}_1 \oplus \mathcal{H}_2$ can be computed from the allocations $QSS_{\mathcal{H}_1}$ and $QSS_{\mathcal{H}_2}$, that is for $\phi \in \Phi$,*

$$QSS_{\mathcal{H}_1 \oplus \mathcal{H}_2}(\phi) = \bigcup_{\phi \le \phi_1 \oplus \phi_2} (QSS_{\mathcal{H}_1}(\phi_1) \cap QSS_{\mathcal{H}_2}(\phi_2)). \tag{3.30}$$

We denote the combination of two allocation by $QSS_{\mathcal{H}_1} \oplus QSS_{\mathcal{H}_2} := QSS_{\mathcal{H}_1 \oplus \mathcal{H}_2}$.

Proof of theorem 3.15

$$\begin{aligned}
QSS_{\mathcal{H}_1 \oplus \mathcal{H}_2}(\phi) &= \{\omega \in \Omega : \Gamma_{\mathcal{H}_1 \oplus \mathcal{H}_2}(\omega) \ge \phi\} \\
&= \{\omega \in \Omega : \Gamma_{\mathcal{H}_1}(\omega) \oplus \Gamma_{\mathcal{H}_2}(\omega) \ge \phi\} \\
&= \{\omega \in \Omega : \phi \le \phi_1 \oplus \phi_2, \Gamma_{\mathcal{H}_1}(\omega) \ge \phi_1, \Gamma_{\mathcal{H}_2}(\omega) \ge \phi_2\} \\
&= \{\omega \in \Omega : \phi \le \phi_1 \oplus \phi_2, \omega \in QSS_{\mathcal{H}_1}(\phi_1) \cap QSS_{\mathcal{H}_2}(\phi_2)\} \\
&= \bigcup_{\phi \le \phi_1 \oplus \phi_2} (QSS_{\mathcal{H}_1}(\phi_1) \cap QSS_{\mathcal{H}_2}(\phi_2)).
\end{aligned}$$

□

Analogously, the focusing operation for hints carries over to the allocations of quasi-support:

Theorem 3.16 *Let $\mathcal{H} = (\Omega, \Gamma, \Phi, D)$ be a hint and $QSS_{\mathcal{H}}$ the corresponding allocation of quasi-support. For $x \in D$, the allocation $QSS_{\mathcal{H}^{\Rightarrow x}}$ corresponding to the focussed hint $\mathcal{H}^{\Rightarrow x}$ can be computed from the allocation $QSS_{\mathcal{H}}$, that is for $\phi \in \Phi$*

$$QSS_{\mathcal{H}^{\Rightarrow x}}(\phi) = \bigcup_{\psi = \psi^{\Rightarrow x} \ge \phi} QSS(\psi). \tag{3.31}$$

We denote the **focusing of an allocation** by ${QSS_{\mathcal{H}}}^{\Rightarrow x} := QSS_{\mathcal{H}^{\Rightarrow x}}$.

Proof of theorem 3.16

$$\begin{aligned}
QSS_{\mathcal{H}^{\Rightarrow x}}(\phi) &= \{\omega \in \Omega : {\Gamma_{\mathcal{H}}}^{\Rightarrow x}(\omega) \ge \phi\} = \{\omega \in \Omega : (\Gamma(\omega))^{\Rightarrow x} \ge \phi\} \\
&= \{\omega \in \Omega : \Gamma(\omega) \ge \psi = \psi^{\Rightarrow x} \ge \phi\} = \bigcup_{\psi = \psi^{\Rightarrow x} \ge \phi} QSS(\psi)
\end{aligned}$$

and this proves the theorem. □

The **vacuous allocation** s_ν, induced by the vacuous hint, satisfies $s_\nu(e) = \Omega$ and $s_\nu(\phi) = \emptyset$ for $\phi \neq e$, that is it allocates only arguments to the vacuous information e. It is clearly the neutral element of the combination operation.

Together with these two operations of combination and focussing, the set of allocations of quasi-support S_Φ is an unmarked information algebra. This can be shown just as in the case of allocations of probability in section 2.4. The only difference is that the reference set 2^Ω of the allocations of quasi-support is not a probability algebra, but only a complete Boolean algebra of subsets. Yet the proof from section 2.4 nevertheless carries over to the present case, because the probability measure itself is unimportant in the proof.

3.6.2 Focusing of Allocations for Computing Quasi-Supports

The operation of focusing an allocation of quasi-support allows to represent conflicts and quasi-support sets possibly within smaller domains in D:

Lemma 3.17 *Let $\phi \in \Phi$, then for $x \in D$*

$$SS^{\Rightarrow x}(e) = DS \qquad \text{for any } x, \tag{3.32}$$

$$QSS^{\Rightarrow x}(z) = CS \qquad \text{if } x \text{ is a support for } z, \tag{3.33}$$

$$QSS^{\Rightarrow x}(\phi) = QSS(\phi) \qquad \text{if } x \text{ is a support for } \phi. \tag{3.34}$$

Proof of lemma 3.17 The third equation follows from a property of the information algebra (S_Φ, D) which can be proved analogously to lemma 2.22 for (P_Φ, D), that is

$$QSS^{\Rightarrow x}(\phi) = \bigcup_{\psi^{\Rightarrow x} = \psi \geq \phi} QSS(\psi) = QSS(\phi),$$

because x is a support for ϕ and QSS is monotone.

For a proof of (3.33), we can apply (3.34) to $\phi = z$, therefore $QSS^{\Rightarrow x}(z) = QSS(z)$. (3.33) follows then by lemma 3.12.

Apply (3.34) to $\phi = e$. Every $x \in D$ is a support for e (cf. lemma 2.4) and SS is monotone, therefore $SS^{\Rightarrow x}(e) = SS(e)$. (3.32) follows then by lemma 3.12. □

3.6.3 Allocations of Quasi-Support on Different Sets of Interpretations

Let $\mathcal{H}_i = (\Omega_i, \Gamma_i, \Phi, D)$, $i = 1, 2$ be two hints with respect to the same information algebra (Φ, D). Denote by $QSS_{\mathcal{H}_1}$ and $QSS_{\mathcal{H}_2}$ the respectively induced allocations of quasi-support. To combine the information contained in the hints, we can either combine both hints to a new hint $\mathcal{H}$ according to section 3.3 and then derive the combined allocation of quasi-support $QSS_{\mathcal{H}}$ from this new hint; or the two allocations of quasi-support $QSS_{\mathcal{H}_1}$ and $QSS_{\mathcal{H}_2}$ can be combined directly. The following theorem states that the results are essentially the same:

Theorem 3.18 *Let $\mathcal{H}_i = (\Omega_i, \Gamma_i, \Phi, D)$ for $i = 1, 2$ be two hints and $\mathcal{H}_1 \oplus \mathcal{H}_2 = (\Omega, \Gamma, \Phi, D)$ their combination. For $\phi \in \Phi$,*

$$QSS_{\mathcal{H}_1 \oplus \mathcal{H}_2}(\phi) = \bigcup_{\phi \leq \phi_1 \oplus \phi_2} \Big((QSS_{\mathcal{H}_1}(\phi_1) \times \Omega_2) \cap (\Omega_1 \times QSS_{\mathcal{H}_2}(\phi_2)) \Big). \tag{3.35}$$

Proof of theorem 3.18 Adapted from (Kohlas, 1995). Denote the right-hand side of (3.35) by R. Let $\phi \in \Phi$, then by definition,

$$QSS_{\mathcal{H}_1 \oplus \mathcal{H}_2}(\phi) = \{\omega \in \Omega : \Gamma_{\mathcal{H}_1 \oplus \mathcal{H}_2}(\omega) \geq \phi\} \tag{3.36}$$

and denote this set by L.

Now let ω be in L. Then $\omega = (\omega_1, \omega_2)$ with $\omega_1 \in \Omega_1$ and $\omega_2 \in \Omega_2$. By definition of $\mathcal{H}_1 \oplus \mathcal{H}_2$, $\Gamma_{\mathcal{H}_1 \oplus \mathcal{H}_2}(\omega) = \Gamma_{\mathcal{H}_1}(\omega_1) \oplus \Gamma_{\mathcal{H}_2}(\omega_2)$. Further, for $i = 1, 2$ define $\phi_i := \Gamma_{\mathcal{H}_i}(\omega_i)$, then $\omega_i \in QSS_{\mathcal{H}_i}(\phi_i)$. Using $\phi_1 \oplus \phi_2 \geq \phi$, we have then that $(\omega_1, \omega_2) = (\omega_1 \times \Omega_2) \cap (\Omega_1 \times \omega_2)$ is contained in R.

Now let (ω_1, ω_2) in R. Then there are $\phi_1, \phi_2 \in \Phi$ with $\phi_1 \oplus \phi_2 \geq \phi$ and $\omega_i \in QSS_{\mathcal{H}_i}(\phi_i)$ and therefore $\Gamma_{\mathcal{H}_i}(\omega_i) \geq \phi_i$ for $i = 1, 2$. This shows that $\Gamma_{\mathcal{H}_1}(\omega_1) \oplus \Gamma_{\mathcal{H}_1}(\omega_2) = \Gamma_{\mathcal{H}_1 \oplus \mathcal{H}_2}(\omega) \geq \phi$, such that (ω_1, ω_2) is in L.

This shows that $L = R$ and completes the proof. □

A conclusion of this theorem is that if several allocations of support

$$QSS_i : \Phi \to 2^{\Omega_i}, \qquad i = 1, \ldots, n \tag{3.37}$$

have to be combined, then we can first extend each of them to $\Omega := \Omega_1 \times \cdots \times \Omega_n$ by defining

$$\begin{array}{rccl} QSS'_i : & \Phi & \to & 2^{\Omega} \\ & \phi & \mapsto & QSS'_i(\phi) := \left(\prod_{j=1}^{i-1} \Omega_j \right) \times QSS_i(\phi) \times \left(\prod_{j=i+1}^{n} \Omega_j \right) \end{array}$$

for $\phi \in \Phi$, and then combine the allocations of support $QSS'_1, \ldots, QSS'_n$ using the method described in section 3.6.1. Yet from a computational point of view, this is not very efficient.

3.7 Constructing Allocations of Probability and Belief Functions

Given a hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D, \mathcal{A}, P)$ and the corresponding body of evidence $(\Phi, D, 2^{\Omega}, QSS_{\mathcal{H}}, \mathcal{A}, P)$ (cf. section 3.5), we will now define an allocation of probability along similar lines as described in (Kohlas, 1995) for classical hints. Usually, we consider only normalized bodies of evidence at this point, but the definitions below are not restricted to them. By definition, the structure $(\Omega, \mathcal{A}, P)$ is a probability space, therefore in general not every subset of Ω can be measured. But we can compute $\xi(\phi) := P(QSS_{\mathcal{H}}(\phi))$ for $\phi \in \Phi$ if $QSS_{\mathcal{H}}(\phi)$ is P-measurable. The set of pieces of information ϕ which are P-measurable is closed under meets and contains the element e with $P(QSS_{\mathcal{H}}(e)) = P(\Omega) = 1$. $P(QSS_{\mathcal{H}}(\phi))$ is the probability that the interpretations which are quasi-supporting ϕ hold. We will show below how to extend this probability to the whole set 2^{Ω}.

It is reasonable to measure the not P-measurable elements in 2^{Ω} by least upper bounds, that is for $S \subseteq \Omega$ define

$$\xi_e(S) := \sup_{A \leq S,\, A \in \mathcal{A}} P(A). \tag{3.38}$$

This function is an extension of ξ because if S is P-measurable, then $\xi(S) = \xi_e(S)$.

Following (Kohlas, 1995), this definition can in fact be motivated.

Let I_0 be the σ-ideal of all elements $A \in \mathcal{A}$ with measure zero, that is $P(A) = 0$. This ideal permits now to define an equivalence relation on the algebra $\mathcal{A}$: For $A, A' \in \mathcal{A}$,

$$A \sim A' \quad \text{if and only if} \quad (A - A') \in I_0 \text{ and } (A' - A) \in I_0.$$

Lemma 3.19 *(Kohlas, 1995) $\sim$ is an equivalence relation on $\mathcal{A}$.*

Denote by $[A]$ the equivalence class of $A \in \mathcal{A}$, and by $\mathcal{B}$ the set of equivalence classes, that is the quotient algebra $\mathcal{B} = \mathcal{A}/\!\sim$.

The mapping $A \mapsto [A]$ is a σ-homomorphism from $\mathcal{A}$ into $\mathcal{B}$ and we can define a probability measure on $\mathcal{B}$ by

$$\mu([A]) := \mu(A) \quad \text{for } [A] \in \mathcal{B}. \tag{3.39}$$

Lemma 3.20 *(Halmos, 1963) $(\mathcal{B}, \mu)$ is a probability algebra.*

(For a proof see for example (Kohlas, 1995).)

For any $S \in 2^{\Omega}$ define

$$\rho(S) := \bigvee_{A \leq S,\, A \in \mathcal{A}} [A] \tag{3.40}$$

and it can be shown (Kohlas, 1995) that this mapping is a normalized, σ-complete allocation of support, that is

$$\begin{aligned}
\rho(\top) &= [\top], \\
\rho(\bot) &= [\bot], \\
\rho\left(\bigwedge_{i \in I} A_i\right) &= \bigwedge_{i \in I} \rho(A_i) \qquad \text{for any countable set } I.
\end{aligned}$$

The mapping $\pi_{\mathcal{H}} := \rho \circ QSS_{\mathcal{H}}$ is a chain of allocation of supports, therefore also an allocation of support. Further, $\pi_{\mathcal{H}}$ is also an allocation of probability from (Φ, D) into the probability algebra $(\mathcal{B}, \mu)$.

The allocation of probability $\pi_{\mathcal{H}}$ together with the measure μ defines then a belief function on (Φ, D),

$$\begin{array}{rccl} bel_{\mathcal{H}}: & \Phi & \rightarrow & [0,1] \\ & \phi & \mapsto & bel_{\mathcal{H}}(\phi) := \mu(\pi_{\mathcal{H}}(\phi)), \end{array} \tag{3.41}$$

which satisfies the axioms (B1) and (B2) of section 2.6. And this belief function $bel_{\mathcal{H}}$ is in fact equal to the extension ξ_e (Kohlas, 1995), therefore this links the former definition of ξ_e to this analysis. The situation is depicted in fig. 3.3.

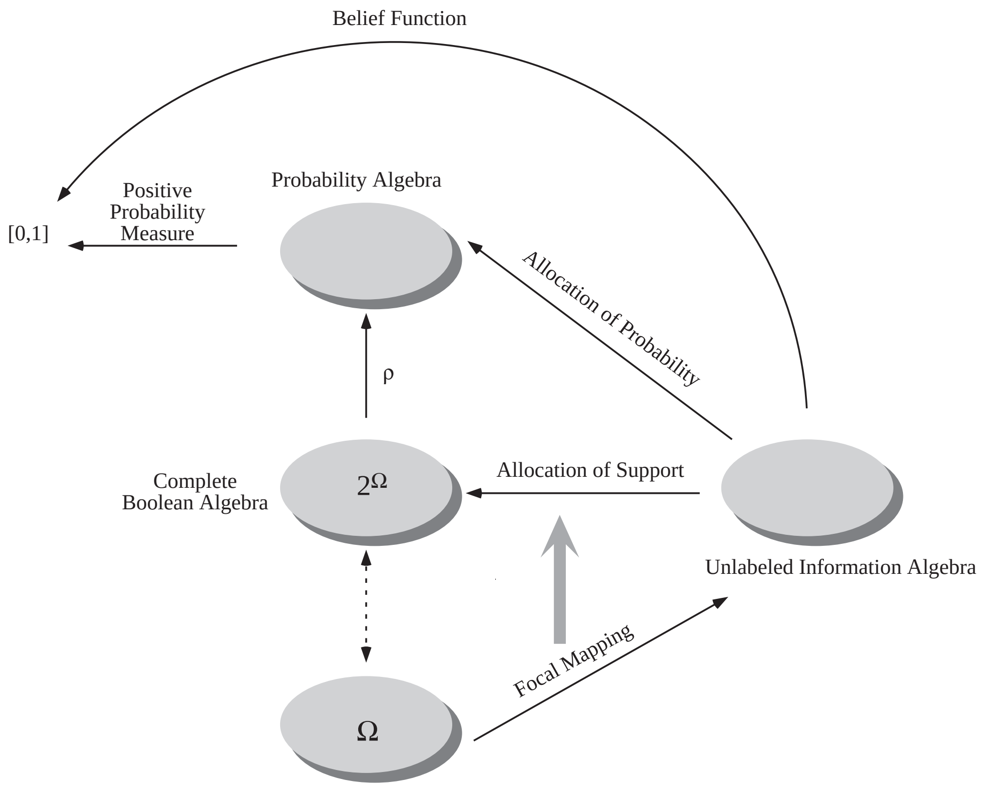


Figure 3.3: Allocation of probability, belief function and extended belief function.

3.8 Equality and Equivalence of Hints

The developments of the previous section allows now to define concepts of equality and equivalence of generalized hints based on the allocations of probability induced by them.

Definition 3.21 *Two hints $\mathcal{H}_1$ and $\mathcal{H}_2$ are called*

- ***equal****,* $\mathcal{H}_1 = \mathcal{H}_2$*, if the components they are made of are identical.*
- ***equivalent****,* $\mathcal{H}_1 \sim \mathcal{H}_2$*, if the induced allocations of probability* $\pi_{\mathcal{H}_1}$ *and* $\pi_{\mathcal{H}_2}$ *are equal.*

Both relations, "=" as well as "∼" are equivalence relations.

4 Information Systems

In the previous chapters, we worked with the concept of information algebras, marked and unmarked ones. Yet in practice, elements of this information algebra are in general infinite and therefore not tractable by computers. Linear manifolds, for example, consist in general of an infinite set of points, yet they can elegantly be described by finite sets of linear equations.

So in this chapter, we present the well-known concept of information systems and show its relation to information algebras. In chapter 5, the concept of an information system is used for a representation of sets of interpretations. In chapter 6, information systems will be used to represent information algebras in order to build argumentation systems.

In the first section, we introduce the concept of information systems, a theory developed by Scott (1982). In section 4.2, the connection between information systems and unlabeled information algebras is presented following (Kohlas, 1997b). This connection allows to use the computational theory for information algebras (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b) for information systems; this computational theory will be presented in a general context in chapter 8. In section 4.3, the special and very important case of information systems with variables is discussed and finally used for the concept of variable elimination.

4.1 The Concept of an Information System

Consider a language $\mathcal{L}$ which is just a set of sentences; we are not interested here in syntactic details of this language. Available information is expressed as a subset X of $\mathcal{L}$. Different sets of sentences may express the same information, and some sentences can be derived from others. Therefore we need a consequence relation (Tarski, 1957) or entailment relation $\vdash$ between sets of sentences X and a sentence c, where $X \vdash c$ means that c can be derived from X. We assume that this entailment relation $\vdash$ satisfies the axioms (E1) and (E2), for $X, Y \subseteq \mathcal{L}$ and $d \in \mathcal{L}$:

(E1) $X \vdash a$ for each $a \in X$.

(E2) If $X \vdash b$ for each $b \in Y$ and $Y \vdash d$, then $X \vdash d$.

A special element $\perp$ is contained in the language $\mathcal{L}$ which denotes the **falsity**. Every element in the language $\mathcal{L}$ follows from this elements by the consequence relation $\vdash$, that is $\{\perp\} \vdash a$ for every $a \in \mathcal{L}$. This element $\perp$ represents the information which "cannot be true", which implies therefore anything else.

Further, the entailment relation is called **finitary** or **compact** if it satisfies the following condition:

(E3) If $X \vdash a$ then there exists a finite subset $Y \subseteq X$ such that $Y \vdash a$.

In the sequel, we will always work with finitary entailment relations, because in practice, computers cannot work with infinite representations of information. As an abbreviation we will often omit the set boundaries on the left hand side of the entailment relation, i.e. we write $f_1, \ldots, f_m \vdash g$ instead of $\{f_1, \ldots, f_m\} \vdash g$.

Definition 4.1 *An **information system** is a tuple $(\mathcal{L}, \vdash)$ consisting of a language $\mathcal{L}$, together with a **compact** entailment relation $\vdash$ defined on the language $\mathcal{L}$ which satisfies the axioms (E1) to (E3).*

Scott introduced a closely related concept in (Scott, 1982). The main difference is that Scott singles out "consistent" subsets of $\mathcal{L}$, while in this paper we will not do that.

There are a variety of models of the theory of information systems, as has been mentioned in (Kohlas, 1997b), for example

- Systems of linear equations (Hertelendy, 1997)
- Systems of linear inequalities (Kalt, 1997)
- Propositional logic builds an information system; this is discussed in (Kohlas *et al.*, 1999b).
- Finite Set Constrains: in chapter 5 we will present this generalization of propositional logic and show that it also is an information system.
- Predicate logic

4.2 Information Systems and their Information Algebras

In this section we will show how an information algebra can be built on top of a given information system.

Suppose that there is a set S of sublanguages of $\mathcal{L}$, which represents all sublanguages we are interested in, $S \subseteq 2^{\mathcal{L}}$. This models the fact that we are not interested in every possible question with respect to the information system, but only in some of them. Further we suppose that $\mathcal{L}$ itself is a member of S and that S forms a $\cap$-system, that is the intersection of any family of sublanguages of S is also a sublanguage contained in S. If the top element, the meet and join are defined by

Top Element:	$\mathcal{L}$,
Meet:	$L_1 \wedge L_2 := L_1 \cap L_2 \in S$,
Join:	$L_1 \vee L_2 := \bigcap \{L' \in S : L_1 \cup L_2 \subseteq L'\}$,

for $L_1, L_2 \in S$, then S forms a complete lattice (Davey & Priestley, 1990; Grätzer, 1978).

In an information system, the same information can possibly be expressed by several different subsets of sentences. The entailment relation $\vdash$ allows to define an equivalence relation on subsets of $\mathcal{L}$ with the meaning that two sets of formulas are equivalent if the same information can be deduced from them. More formally, we define the operator C from $2^{\mathcal{L}}$ to itself, i.e. for a set $X \in \mathcal{L}$, we define the set $C(X)$ as the set of all sentences in $\mathcal{L}$ which can be derived from X by the entailment relation $\vdash$,

$$C(X) := \{a \in \mathcal{L} : X \vdash a\}. \tag{4.1}$$

Given that $\vdash$ satisfies (E1) to (E3), it is easy to see that the operator C satisfies the following properties and is therefore a **compact consequence operator** (Tarski, 1957):

(C1) $X \subseteq C(X)$ for every $X \subseteq \mathcal{L}$.

(C2) $C(C(X)) = C(X)$ for every $X \subseteq \mathcal{L}$.

(C3) If $X \subseteq Y \subseteq \mathcal{L}$, then $C(X) \subseteq C(Y)$.

(C4) $C(X) = \bigcup \{C(Y) : Y \subseteq X,\ Y \text{ finite}\}$.

Definition 4.2 *Two sets of formulas X and Y in $\mathcal{L}$ are called* ***equivalent****, $X \sim Y$, if $C(X) = C(Y)$. X is called* ***closed*** *if $C(X) = X$. The formulas contained in $C(\emptyset)$, i.e. the sentences following from the empty set of formulas $\emptyset$, are called* ***tautologies****.*

Note that the set of formulas $\mathcal{L}$ is closed itself because of (C1). An important property of C is stated in the following lemma:

Lemma 4.3 *(Kohlas & Stärk, 1996b) For $X, Y \in \mathcal{L}$,*

$$C(X \cup Y) \quad = \quad C(X \cup C(Y)). \tag{4.2}$$

Often, we are not interested in consequences with respect to the whole language $\mathcal{L}$, but only to some specific smaller subsets of formulas of $\mathcal{L}$. We define therefore for every subset M of $\mathcal{L}$

$$C_M(X) := C(X) \cap M \quad \text{for } X \subseteq \mathcal{L}. \tag{4.3}$$

C_M as operator from $2^{\mathcal{L}}$ to itself has similar properties as the operator C (Kohlas & Stärk, 1996b):

(M1) If $X \subseteq M$, then $X \subseteq C_M(X)$.

(M2) $C_M(C_M(X)) = C_M(X)$.

(M3) If $X \subseteq Y \subseteq \mathcal{L}$, then $C_M(X) \subseteq C_M(Y)$.

(M4) If $X \subseteq M$, then $C(C_M(X)) = C(X)$.

Usually, we are interested in sublanguages which belong to the lattice S.

Impose now two further restrictions on the consequence operator C:

(C5) $C(C_M(C_L(X))) = C(C_{M\cap L}(X))$ for $X \subseteq \mathcal{L}$ and $L, M \in S$.

(C6) $C_L(C_L(X) \cup Y) = C_L(C_L(X) \cup C_L(Y))$ for $X, Y \subseteq \mathcal{L}$ and $L \in S$.

Kohlas & Stärk (1996b) show that property (C5) is equivalent to C having the interpolation property with respect to S, and also, that the deduction property implies condition (C6).

Definition 4.4 *Let $L \in S$ and $X \subseteq L$. Then $\phi = \langle X, L\rangle$ is called a **marked statement**. $d(\phi) = L$ is called its **label**. The set of all marked statements is denoted by $\mathcal{MS}$.*

*Two marked statements $\phi_1 = \langle X_1, L_1\rangle$ and $\phi_2 = \langle X_2, L_2\rangle$ are called **equivalent** $\phi_1 \sim \phi_2$ if and only if $X_1 \sim X_2$ and $L_1 = L_2$.*

Two operations, marginalization and combination, are defined on marked statements:

Definition 4.5 *A marked statement $\phi_1 = \langle X_1, L_1\rangle$ is called a **marginal** of $\phi_2 = \langle X_2, L_2\rangle$ if and only if*

$$L_1 \subseteq L_2 \quad \textit{and} \quad C_{L_1}(X_1) = C_{L_1}(X_2) \tag{4.4}$$

In the sequel, we define

$${\phi_2}^{\downarrow L_1} := \langle C_{L_1}(X_2), L_1\rangle. \tag{4.5}$$

${\phi_2}^{\downarrow L_1}$ is clearly a marginal and for every other marginal ϕ' of ϕ_2 we have $\phi' \sim {\phi_2}^{\downarrow L_1}$.

Lemma 4.6 *(Kohlas, 1997b) The marginalization is transitive, i.e. for $\phi_i = \langle X_i, L_i\rangle \in \mathcal{MS}$, $i = 1,2,3$,*

$$\text{if } \phi_2 \sim {\phi_1}^{\downarrow L_2} \text{ and } \phi_3 \sim {\phi_2}^{\downarrow L_3} \quad \text{then} \quad \phi_3 \sim {\phi_1}^{\downarrow L_3}. \tag{4.6}$$

Definition 4.7 *A marked statement $\phi = \langle X, L\rangle$ is called a **combination** of the marked statements $\phi_1 = \langle X_1, L_1\rangle$ and $\phi_2 = \langle X_2, L_2\rangle$, if and only if*

$$L = L_1 \vee L_2 \quad \text{and} \quad C(X) = C(X_1 \cup X_2). \tag{4.7}$$

In the sequel, we define the product of ϕ_1 and ϕ_2 as

$$\phi_1 \oplus \phi_2 := \langle C(X_1 \cup X_2), L_1 \vee L_2\rangle. \tag{4.8}$$

Clearly, $\phi_1 \oplus \phi_2$ is a combination and every other combination ϕ' of ϕ_1 and ϕ_2 satisfies $\phi' \sim \phi_1 \oplus \phi_2$.

Lemma 4.8 *(Kohlas, 1997b) The combination is commutative and associative, i.e. for $\phi_i = \langle X_i, L_i\rangle \in \mathcal{MS}$, $i = 1,2,3$,*

$$\phi_1 \oplus \phi_2 = \phi_2 \oplus \phi_1 \qquad \phi_1 \oplus (\phi_2 \oplus \phi_3) = (\phi_1 \oplus \phi_2) \oplus \phi_3. \tag{4.9}$$

This lemma shows that marked statements form a commutative semigroup under combination. The unity element for the label L is marked statement $e_L = \langle \emptyset, L\rangle$ which satisfies

$$e_L \oplus \langle X, L\rangle = \langle X, L\rangle \tag{4.10}$$

for every marked statement $\langle X, L\rangle$. Furthers, the combination of marked statement is idempotent:

Lemma 4.9 *(Kohlas, 1997b) For a marked statement $\phi = \langle X, L\rangle$ and a sublanguage $L' \subseteq L$, we have $\phi \oplus \phi^{\downarrow L'} \sim \phi$.*

The following lemma is an essential result for computations with marked statements, especially in distributed computations:

Lemma 4.10 *(Kohlas, 1997a) If the consequence operator C satisfies (C6) and $\phi_i = \langle X_i, L_i\rangle$ for $i = 1,2$ are two marked statements, then*

$$(\phi_1 \oplus \phi_2)^{\downarrow L_1} = \phi_1 \oplus {\phi_2}^{\downarrow L_1 \cap L_2}. \tag{4.11}$$

Equivalent statements express the same information "in different words". Thus it makes sense to consider the classes of equivalent marked statements: For $\phi \in \mathcal{MS}$ denote the equivalence class which it belongs to by $[\phi]$; a representant of this class is for example $C_L(X)$ if $\phi = \langle X, L\rangle$. Denote by $[\mathcal{MS}]$ the family of all these equivalence classes. Define combination and marginalization in $[\mathcal{MS}]$ by

$$[\phi_1] \oplus [\phi_2] \quad := \quad [\phi_1 \oplus \phi_2] \tag{4.12}$$
$$[\phi]^{\downarrow L} \quad := \quad \left[\phi^{\downarrow L}\right], \tag{4.13}$$

and the labeling function d by $d([\phi]) := d(\phi)$.

The set $\mathcal{MS}$ and the lattice S together with the operations of combination and marginalization as defined above satisfy the axioms (M1) to (M10) of definition 2.5 if the additional properties (C5) and (C6) on the consequence operator C are satisfied, therefore $\langle \mathcal{MS}, S \rangle$ is a marked information algebra (for a proof see (Kohlas, 1997b)). Using the results from subsection 2.1.3, we can also construct the corresponding unmarked information algebra $(\mathcal{MS}, S)$.

Consider now an information algebra constructed from an information system as shown above. Then the computation of a marginal in the information algebra can be carried over to the computation of a marked statement representing the corresponding marginal in the information algebra. Therefore, we can work with representants of equivalence classes (that is marked statements) rather that with the equivalence classes (that is elements of the information algebra) themselves. Usually, a representant is a small set of formulas whereas the equivalence class can be infinite and therefore — from the computational aspect — not tractable.

The concept of local computations on hypertrees, as it will be introduced in chapter 8, can be applied to marked statements; this is in detail explained in (Kohlas, 1997b).

We have shown how to construct an information algebra from an information system. The other direction is only possible in some cases: the construction of an information system from a so-called finitary information algebra is presented in (Kohlas & Stärk, 1996b).

4.3 Elimination of Variables in Information Systems with Variables

A special case arises if the language $\mathcal{L}$ of the information system is formed over a finite or countable set $\mathcal{V}$ of symbols or variables; the information system is then called an **information system with variables** (Kohlas, 1997b). This is for example the case in propositional logic.

The lattice S contains then usually all sublanguages formed over subsets of $\mathcal{V}$. A sublanguage over a set of variables $V \subseteq \mathcal{V}$ is denoted by L_V. Usually, the following relations hold for $V', V'' \subseteq \mathcal{V}$:

$$L_{V' \cap V''} \quad = \quad L_{V'} \cap L_{V''} \tag{4.14}$$
$$L_{V' \cup V''} \quad = \quad \bigcap\{L_V : L_{V'} \cup L_{V''} \subseteq L_V\} \quad = \quad L_{V'} \vee L_{V''} \tag{4.15}$$

Let X be a set of formulas from $\mathcal{L}$. The set of variables which appear in X is called its **signature**, denoted by $\Sigma(X)$. Then $L_{\Sigma(X)}$ is the smallest sublanguage of S which contains X, and this sublanguage is completely characterized by $\Sigma(X)$. Therefore we write $\langle X, V\rangle$ for the marked statement with set of formulas X in the language V for a $V \in \mathcal{V}$.

For an information system with variables the concept of marginalization can be seen as the elimination of variables in a set of formulas. For a marked statement $\phi = \langle X, V\rangle$, we are interested in a marginal $\phi^{\downarrow V-\{v\}}$ for a $v \in V$. We suppose therefore that there is a fixed way to obtain a set X^{-v} such that

$$\langle X^{-v}, V - \{v\}\rangle \quad \sim \quad \langle X, V\rangle^{\downarrow V-\{v\}}. \tag{4.16}$$

The computation of X^{-v} from X is called the **elimination of the variable** v, and this elimination therefore corresponds to the marginalization. Let now X_{+v} be the subset of all formulas of X which contain the variable v which is to be deleted, and $X_{-v} := X - X_{+v}$. Then, for $\phi_{+v} := \langle X_{+v}, V\rangle$ and $\phi_{-v} := \langle X_{-v}, V - \{v\}\rangle$, we have $\phi \sim \phi_{+v} \oplus \phi_{-v}$. The consequence operator C satisfies the condition (C6), therefore using lemma 4.10, we have

$$\begin{aligned} \phi^{\downarrow V-\{v\}} \quad &\sim \quad (\phi_{+v} \oplus \phi_{-v})^{\downarrow V-\{v\}} && (4.17)\\ &\sim \quad {\phi_{+v}}^{\downarrow V-\{v\}} \oplus \phi_{-v} && (4.18)\\ &\sim \quad \langle {X_{+v}}^{-v} \cup X_{-v}, V - \{v\}\rangle. && (4.19) \end{aligned}$$

This means that X^{-v} and ${X_{+v}}^{-v} \cup X_{-v}$ are equivalent. Yet without loss of generality, we define in the sequel $X^{-v} := {X_{+v}}^{-v} \cup X_{-v}$ in order to simplify notations.

Furthers, for a sequence of variables $v_1, \ldots, v_p$ in $\mathcal{V}$ define inductively

$$X^{-(v_1,\ldots,v_p)} \quad := \quad \left(X^{-(v_1,\ldots,v_{p-1})}\right)^{-v_p}. \tag{4.20}$$

Then, using lemma 4.6, we get

$$\langle X^{-(v_1,\ldots,v_p)}, V - \{v_1, \ldots, v_p\}\rangle \quad \sim \quad \langle X, V\rangle^{\downarrow W-\{v_1,\ldots,v_p\}}. \tag{4.21}$$

Marginals of marked statements in the information system can therefore be computed by sequentially eliminating variables. Different orderings of those variables, which have to be eliminated, result usually in different but nevertheless equivalent marginals.[1] But clearly, the computational effort may considerably depend on a "good" ordering of the variables, see for example (Kohlas, 1997b). Together with the concept of computations in a hypertree (see chapter 8), the elimination of variables is useful for big computations; for an implementation of marginalization for different calculi see also chapter 12.

The special case of propositional logic with the usual consequence operator is clearly an information system with variables, as it has been considered in

[1]Kohlas *et al.* (2000) show that in the special case of propositional logic, the results are not only equivalent, but equal.

(Kohlas *et al.*, 1999b); see also chapter 5 for a generalization to set constraints. Combination means essentially union of sets of propositional formulas, computing marginals, that is elimination of variables in a set of formulas, means resolutions based on concepts developed by Davis & Putnam (1962; 1983).

Systems of linear equation are considered in the context of marginalization in (Kohlas, 1997b; Hertelendy, 1997), systems of linear inequalities can be treated similarly using Fourier-Motzkin elimination techniques (Kalt, 1997; Imbert, 1995; Williams, 1976).

5 Logical Representation of Arguments

The representations of conflicts and diagnoses as subsets of Ω defined in section 6 are not very easy to handle. The state space Ω normally is very huge, its size usually grows exponentially with the number of variables. In this chapter, a logical description for subsets of Ω is presented, and this description will also be used to formulate the computations in chapter 8. In addition, several useful properties of the logical description are stated and proved.

We consider the special case where the space Ω is a finite cartesian product of the possible values of a set of variables. This restriction is motivated by the use of Ω in the framework of an argumentation system (cf. chapter 6 for examples). This approach is very much motivated by the work with examples (see also chapter 12): Usually, there is a notion of components which are in a precise but unknown working mode out of a set of possible modes, and these sets of modes are then used to build the set of interpretations Ω; this means that an interpretation $\omega \in \Omega$ defines the working mode of all components. In the sequel, we will not restrict the formalism to systems with components, but present it in a more general context.

In the first section we introduce the concept of finite set constraints and show that they define an information system and algebra. In section 5.2, we show how arguments can be represented as formulas in the language of finite set constraints. Implicates and implicants are introduced in section 5.3 and used to define canonical representations for any formula. In section 5.4, the concepts of conflicts and diagnoses, in section 5.5 those of support and quasi-support sets, and in section 5.6 those of possibly supporting and refuting sets are carried over from Ω to the language of finite set constraints.

5.1 Finite Set Constraints

In this section, we introduce the concept of finite set constraints following (Anrig *et al.*, 1997c). Further literature on this concept can be found in (Haenni &

Lehmann, 1998a) and, together with a short discussion of the difference between finite set constraints and singed formulas in many-valued logics, also in (Haenni & Lehmann, 1998b). More general set constraint languages, also called signed logics, are discussed in (Hähnle, 2000; Beckert *et al.*, 1999; Hähnle & Escalada-Imaz, 1997).

Note that here, we use the term "set constraint" for a concept which differs substantially from the one used in the literature, for example in (Kozen, 1994; Aiken *et al.*, 1994; Gilleron *et al.*, 1993).

Set constraints are a generalization of ordinary variables from propositional logic which can only take two distinct values; so here, we consider variables with several possible values. This is especially handy for the formulation of different operating modes for components in complex systems.

5.1.1 Definition of the Language

Consider the finite set of variables $\{c_1, \ldots, c_n\}$ and for every variable c_i there is a fixed finite set V_i of possible values, also called the **frame** of the variable. A **finite set constraint (FSC)** over a variable c_i and a frame V with $V \subseteq V_i$ is denoted by $M(c_i, V)$. It is interpreted as a predicate which is true if and only if the variable c_i takes a value out of V. This is a simple constraint over one variable.

More general constraints can be built using several variables and a subspace of the cartesian product of their possible values. The predicate is true if and only if the vector of variables takes a value out of the subspace. For example, consider three variables c_i, c_j, and c_k together with a subspace $V \subseteq V_i \times V_j \times V_k$, then the predicate is true if and only if the value of (c_i, c_j, c_k) is in V. This is a generalization of FSC, but note that generalized restrictions can always be formulated using FSCs.[1] Efficient algorithms have been developed for reasoning with FSC (Anrig *et al.*, 1997c); these algorithm can be generalized too, but they become rather complex. Therefore in the sequel we focus on FSC as defined above.

The language $\mathcal{FSC}$ is built using the following rules:

(1) $\bot$, $\top$ are formulas in $\mathcal{FSC}$.

(2) $M(c_i, V)$ is a formula in $\mathcal{FSC}$ for every $i = 1, \ldots, n$ and every $V \subseteq V_i$.

(3) If f is a formula in $\mathcal{FSC}$, then so is $(\neg f)$.

(4) If f_1 and f_2 are formulas in $\mathcal{FSC}$, then so are $(f_1 \wedge f_2)$, $(f_1 \vee f_2)$, $(f_1 \rightarrow f_2)$ and $(f_1 \leftrightarrow f_2)$.

(5) All formulas in $\mathcal{FSC}$ are constructed by application of the rules above.

[1] Every generalized restriction can be formulated using an FSC DNF (see below for the definition of DNF).

Parentheses are omitted if they are not necessary, and for that purpose we define the usual increasing priority of the connectives, that is $\leftrightarrow$, $\rightarrow$, $\wedge$, $\vee$, $\neg$.

An **interpretation** $\omega = (\omega_1, \dots, \omega_n)$ is defined as being a vector containing the value of every variable, that is the variable c_i takes the value ω_i for $i = 1, \dots, n$. The possible values for ω are the elements of the cartesian product

$$\Omega \quad := \quad \prod_{i=1}^{n} V_i \tag{5.1}$$

and Ω is called the set of interpretations. This space will also be used to define hints on argumentation systems; this is described in chapter 6.

Given an interpretation $\omega = (\omega_1, \dots, \omega_n) \in \Omega$, the predicate $M(c_i, V)$ evaluates to true if and only if $\omega_i \in V$. Evaluating a formula $f \in \mathcal{FSC}$ under $\omega \in \Omega$ means then to replace the predicates by their respective evaluation $\top$ or $\bot$ under ω and then evaluate the formula using the usual logical truth tables for the connectives $\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\leftrightarrow$. Formally, we define the evaluation $\omega \models f$ of a formula $f \in \mathcal{FSC}$ under an interpretation $\omega = (\omega_1, \dots, \omega_n) \in \Omega$ by the following rules:

- $\omega \models \top$.
- Not $\omega \models \bot$.
- $\omega \models M(c_i, V)$ iff $\omega_i \in V$.
- $\omega \models \neg f$ iff not $\omega \models f$.
- $\omega \models f_1 \wedge f_2$ iff $\omega \models f_1$ and $\omega \models f_2$.
- $\omega \models f_1 \vee f_2$ iff $\omega \models f_1$ or $\omega \models f_2$.
- $\omega \models f_1 \rightarrow f_2$ iff $\omega \models \neg f_1$ or $\omega \models f_2$.
- $\omega \models f_1 \leftrightarrow f_2$ iff $\omega \models f_1 \rightarrow f_2$ and $\omega \models f_2 \rightarrow f_1$.

Note that the "or" in the previous definition are inclusive-or. As an abbreviation, we write $\omega \models f_1, \dots, f_m$ if $\omega \models f_1$, $\omega \models f_2$, $\dots$, $\omega \models f_m$.

Often we are interested in the set of all elements of Ω under which a given formula evaluates to true. Therefore we define the function N which maps a formula from $\mathcal{FSC}$ to a subset of Ω,

$$N(f) := \{\omega \in \Omega : \ \omega \models f\}. \tag{5.2}$$

We will say that a formula $f \in \mathcal{FSC}$ **describes** a subset $F \subseteq \Omega$ if $N(f) = F$. It is easy to show that the following properties hold for the mapping N and $f, g \in \mathcal{FSC}$:

$$N(\neg f) \quad = \quad N(f)^c \tag{5.3}$$

$$N(f \wedge g) \quad = \quad N(f) \cap N(g) \tag{5.4}$$

$$\begin{aligned} N(f \vee g) &= N(f) \cup N(g) && (5.5)\\ N(f_1 \rightarrow f_2) &= N(\neg f_1) \cup N(f_2) && (5.6)\\ N(f_1 \leftrightarrow f_2) &= N(f_1 \rightarrow f_2) \cap N(f_2 \rightarrow f_1) && (5.7) \end{aligned}$$

Two formulas f_1, f_2 in $\mathcal{FSC}$ are **logically equivalent**[2], written $f_1 \cong f_2$, if $N(f_1) = N(f_2)$. f_1 **induces** f_2, written $f_1 \models f_2$, if $N(f_1) \subseteq N(f_2)$. We abbreviate $f_1 \wedge \cdots \wedge f_n \models f$ by $f_1, \cdots, f_n \models f$ or sometimes by $F \models f$ if $F = \{f_1, \ldots, f_n\}$, that means sets of formulas are interpreted conjunctively.

A system state $\omega \in \Omega$ can be represented by a formula $c(\omega)$ in the language $\mathcal{FSC}$,

$$c(\omega) := \bigwedge_{i=1}^{n} M(c_i, \{\omega_i\}). \tag{5.8}$$

Clearly, the combination $N \circ c$ is the identity on Ω. For a formula $f \in \mathcal{FSC}$ we call

$$\bigvee_{\omega \in N(f)} c(\omega) \tag{5.9}$$

its **canonical representation**.

The usual equivalence relations known from propositional logic hold also for the language of finite set constraints: For $f, g, h \in \mathcal{FSC}$,

Idempotency: $f \wedge f \cong f,\ f \vee f \cong f.$

Commutativity: $f \wedge g \cong g \wedge f,\ f \vee g \cong g \vee f.$

Associativity: $f \wedge (g \wedge h) \cong (f \wedge g) \wedge h,\ f \vee (g \vee h) \cong (f \vee g) \vee h.$

Absorption: $f \wedge (f \vee g) \cong f,\ f \vee (f \wedge g) \cong f.$

Modularity: $f \wedge (g \vee (f \wedge h)) \cong (f \wedge g) \vee (f \wedge h),$
$f \vee (g \wedge (f \vee h)) \cong (f \vee g) \wedge (f \vee h).$

Distributivity: $f \wedge (g \vee h) \cong (f \wedge g) \vee (f \wedge h),\ f \vee (g \wedge h) \cong (f \vee g) \wedge (f \vee h).$

Universal bounds: $f \wedge \bot \cong \bot,\ f \vee \bot \cong f,\ f \wedge \top \cong f,\ f \vee \top \cong \top.$

Complements: $f \wedge \neg f \cong \bot,\ f \vee \neg f = \top.$

Involution: $\neg(\neg f) \cong f.$

De Morgan: $\neg(f \wedge g) \cong \neg f \vee \neg g,\ \neg(f \vee g) \cong \neg f \wedge \neg g.$

[2]We write "logically" instead of "FSC-logically" in order to simplify notations. This simplification will also be used for future definitions.

Further, there are several equivalence relations in the language $\mathcal{FSC}$ which are induced from set theory (Gries & Schneider, 1993) and follow from the definitions above. These relations allow to construct simpler but logically equivalent formulas by successively replacing formulas from the left hand-side by their respective equivalent formulas on the right-hand side. For $i = 1, \dots, n$ and $V, V' \in V_i$:

Lower bound: $M(c_i, \emptyset) \cong \bot$.

Upper bound: $M(c_i, V_i) \cong \top$

Negation: $\neg M(c_i, V) \cong M(c_i, V_i - V)$.

Disjunction: $M(c_i, V) \vee M(c_i, V') \cong M(c_i, V \cup V')$.

Conjunction: $M(c_i, V) \wedge M(c_i, V') \cong M(c_i, V \cap V')$.

An FSC $M(c_i, V)$ is called **proper** if $V \neq \emptyset$ and $V \neq V_i$. An **FSC clause** is a disjunction of proper FSC's such that every variable c_i occurs at most once. Analogously, we define a **FSC conjunction** as a conjunction of proper FSC's such that every variable c_i occurs at most once. Note that neither an FSC clause nor an FSC conjunction contains any negated literal. Any formula in $\mathcal{FSC}$ can be transformed into a logically equivalent disjunction of FSC conjunctions (DNF) or a conjunction of FSC disjunctions (CNF) using the two sets of equivalence relations defined above.

5.1.2 Information System and Algebra of Set Constraints

The tuple $(\mathcal{FSC}, \models)$ is an information system as described in chapter 4 because $\models$ is a compact entailment relation.

Lemma 5.1 *$\models$ is a compact entailment relation in $\mathcal{FSC}$.*

Proof of lemma 5.1 We have to show that (E1) to (E3) are satisfied for $\models$.

Let $X \subseteq \mathcal{FSC}$ and $f \in X$. Then

$$N(X) = N\left(\bigwedge_{g \in X} g\right) = \bigcap_{g \in X} N(g) \subseteq N(f)$$

and therefore $X \models f$ which proves (E1).

For (E2), let $X, Y \subseteq \mathcal{FSC}$ and $d \in \mathcal{FSC}$ such that $X \models f$ for every $f \in Y$ and $Y \models d$. This implies that $N(X) \subseteq N(f)$ for every $f \in Y$, hence

$$N(X) \subseteq \bigcap_{f \in Y} N(f) = N(Y).$$

Further $N(Y) \subseteq N(d)$. Together, this gives $N(X) \subseteq N(d)$ which by definition is $X \models d$ and therefore proves (E2).

For (E3), let $X \subseteq \mathcal{FSC}$ and $d \in \mathcal{FSC}$ such that $X \models d$. The set of interpretation Ω is a finite product of finite spaces, therefore there is a finite $Y \subseteq X$ such that $N(Y) = N(X)$. So $N(Y) \subseteq N(d)$, which by definition is $Y \models d$ and this proves (E3). $\square$

Using the techniques presented in subsection 2.1.3, we can build the corresponding marked and unmarked information algebra as follows.

Define $\mathcal{FSC}_L$ for $L \subseteq \{c_1, \ldots, c_n\}$ as the language of finite set constraints built with predicates relative to the variables in L. Therefore, $\mathcal{FSC}_{\{c_1,\ldots,c_n\}} = \mathcal{FSC}$ and $\mathcal{FSC}_\emptyset$ is the language constructed with the only terms $\perp$ and $\top$. Define the set

$$S := \{\mathcal{FSC}_L : L \subseteq \{c_1, \ldots, c_n\}\}. \tag{5.10}$$

S is clearly a complete lattice with top element $\mathcal{FSC}$, meet $\mathcal{FSC}_L \wedge \mathcal{FSC}_{L'} := \mathcal{FSC}_{L\cap L'}$, and join $\mathcal{FSC}_L \vee \mathcal{FSC}_{L'} := \bigcap\{FSC_{L''} \in S : \mathcal{FSC}_L \cup \mathcal{FSC}_{L'} \subseteq \mathcal{FSC}_{L''}\} = \mathcal{FSC}_{L'\cup L''}$. The consequence operator $C : \mathcal{FSC} \to \mathcal{FSC}$ is defined by

$$C(X) \quad := \quad \{f \in \mathcal{FSC} : X \models f\} \qquad \text{for } X \subseteq \mathcal{FSC}. \tag{5.11}$$

For finite set constraints, $\models$ is a compact entailment relation, and we have proved in section 4.2 that this implies that C is a compact consequence operator, i.e. it satisfies the conditions (C1) to (C4) of section 4.2. Further, define $C_L(X) := C(X) \cap \mathcal{FSC}_L$. In the lemma below we show that C satisfies also the condition (C5) and (C6) from the same section.

For $L \subseteq \{c_1, \ldots, c_n\}$ define the subspace

$$\Omega_L := \prod_{c_i \in L} V_i. \tag{5.12}$$

For $A \subseteq \Omega$ denote by $A\big|_{\Omega_L}$ the projection of A to the subspace Ω_L and by N_L the restriction of N to Ω_L , i.e. $N_L(X) := (N(X))\big|_{\Omega_L}$. Further, for $L' \subseteq L$, the canonical embedding of $B \subseteq \Omega_{L'}$ into Ω_L is denoted by $B^{\uparrow L}$.

Lemma 5.2 *C satisfies conditions (C5) and (C6) of section 4.2.*

Proof of lemma 5.2 The lemma can be proved by considering the subsets $N(f)$ for formulas $f \in \mathcal{FSC}$. First, note that for every $X \subseteq \mathcal{FSC}$ and $M \subseteq \{c_1, \ldots, c_n\}$ we have

$$\begin{aligned} N(C_M(X)) &= \bigcap_{\substack{f \in \mathcal{FSC}_M \\ N(X) \subseteq N(f)}} N(f) &&= \left(\bigcap_{N_M(X) \subseteq A \subseteq \Omega_M} A \right) \times \Omega_{\{c_1,\ldots,c_n\}-M} \\ &= N_M(X)^{\uparrow\{c_1,\ldots,c_n\}} &&= \left(N(X)\big|_{\Omega_M} \right)^{\uparrow\{c_1,\ldots,c_n\}}. \end{aligned}$$

Now we prove (C5). Let $X \subseteq \mathcal{FSC}$ and $L, M \subseteq \{c_1, \ldots, c_n\}$. Then

$$\begin{aligned}
N(C_M(C_L(X))) &= \left(N(C_L(X))\Big|_{\Omega_M}\right)^{\uparrow\{c_1,\ldots,c_n\}} \\
&= \left(\left(\left(N(X)\Big|_{\Omega_L}\right)^{\uparrow\{c_1,\ldots,c_n\}}\right)\Big|_{\Omega_M}\right)^{\uparrow\{c_1,\ldots,c_n\}} \\
&= \left(N(X)\Big|_{\Omega_{M\cap L}}\right)^{\uparrow\{c_1,\ldots,c_n\}} = N(C_{M\cap L}(X)).
\end{aligned}$$

This implies (C5) because

$$\begin{aligned}
C(C_M(C_L(X))) &= \{f \in \mathcal{FSC} : C_M(C_L(X)) \models f\} \\
&= \{f \in \mathcal{FSC} : N(C_M(C_L(X)) \subseteq N(f)\} \\
&= \{f \in \mathcal{FSC} : N(C_{M\cap L}(X)) \subseteq N(f)\} \\
&= \{f \in \mathcal{FSC} : C_{M\cap L}(X) \models f\} \\
&= C(C_{M\cap L}(X)).
\end{aligned}$$

Now we prove (C6). Let $X, Y \subseteq \mathcal{FSC}$ and $L \subseteq \{c_1, \ldots, c_n\}$. Then

$$\begin{aligned}
N_L(C_L(X) \cup Y) &= N_L(C_L(X)) \cap N_L(Y) = N_L(C_L(X)) \cap N_L(C_L(Y)) \\
&= N_L(C_L(X) \cup C_L(Y))
\end{aligned}$$

and this implies (C6) because

$$\begin{aligned}
C_L(C_L(X) \cup Y) &= \{f \in \mathcal{FSC}_L : N(C_L(X) \cup Y) \subseteq N(f)\} \\
&= \{f \in \mathcal{FSC}_L : N_L(C_L(X) \cup Y) \subseteq N_L(f)\} \\
&= \{f \in \mathcal{FSC}_L : N_L(C_L(X) \cup C_L(Y)) \subseteq N_L(f)\} \\
&= \{f \in \mathcal{FSC}_L : N(C_L(X) \cup C_L(Y)) \subseteq N(f)\} \\
&= C_L(C_L(X) \cup C_L(Y)).
\end{aligned}$$

□

Consider $L \subseteq \{c_1, \ldots, c_n\}$ and $X \subseteq \mathcal{FSC}_L$. Then $\langle X, L\rangle$ is a marked statement, and we define combination and marginalization like in section 4.2: For $L' \subseteq L$,

$$\begin{aligned}
\langle X, L\rangle^{\downarrow L'} &:= \langle C_{L'}(X), L'\rangle \\
\langle X, L\rangle \oplus \langle X', L'\rangle &:= \langle C_{L\cup L'}(X \cup X'), L \cup L'\rangle
\end{aligned}$$

In the present context, these operations can also be represented more intuitively as follows: For $L' \subseteq L$,

$$\begin{aligned}
\langle X, L\rangle^{\downarrow L'} &= \langle \{f \in \mathcal{FSC}_{L'} : N(X) \subseteq N(f)\}, L'\rangle \\
\langle X, L\rangle \oplus \langle X', L'\rangle &= \langle \{f \in \mathcal{FSC}_{L\cup L'} : N(X \cup X') \subseteq N(f)\}, L \cup L'\rangle
\end{aligned}$$

where $N(Y) := \bigcap_{g\in Y} N(g)$. The definition of equivalence of marked statements reads now

$$\langle X, L\rangle \sim \langle X', L'\rangle \quad \text{iff} \quad L' = L \text{ and } N(X) = N(X'). \tag{5.13}$$

This implies that the class of marked statements equivalent to $\langle X, L\rangle$ can be represented by $[N_L(X), L]$, and because $X \subseteq \mathcal{FSC}_L$ we have

$$(N_L(X))^{\uparrow\{c_1,\ldots,c_n\}} \quad = \quad N(X). \tag{5.14}$$

This reflects the fact that the class $[N_L(X), L]$ contains only information with respect to L, but no information outside L.

Combination and marginalization of these classes are defined following (4.12) and (4.13) respectively, that is for $L'' \subseteq L'$ in terms of interpretations

$$[\Omega', L'] \oplus [\Omega, L] \quad := \quad \left[\left(\Omega'^{\uparrow L' \cup L}\right) \cap \left(\Omega^{\uparrow L' \cup L}\right), L' \cup L\right] \tag{5.15}$$

$$[\Omega', L']^{\downarrow L''} \quad := \quad \left[\left\{\omega \in \Omega_{L''} : \{\omega\}^{\uparrow L'} \subseteq \Omega'\right\}, L''\right]. \tag{5.16}$$

The marked information algebra is then $\langle 2^\Omega, 2^{\{c_1,\ldots,c_n\}}\rangle$.

Using the results from subsection 2.1.3, we can construct the corresponding unmarked information algebra. For an element $[\Omega', L']$, this means that we have to identify it with any marked statement $[\Omega'', L'']$ satisfying

$$\Omega'^{\uparrow L' \cup L''} \quad = \quad \Omega''^{\uparrow L' \cup L''}, \tag{5.17}$$

therefore we identify all marked statements having the same set of interpretation. This identification results in equivalence classes which, for an element $[\Omega', L']$ in the class, can be represented by its canonical embedding into Ω, that is $\Omega'^{\uparrow\{c_1,\ldots,c_n\}}$, because for every $[\Omega', L']$ there is an equivalent element $[\Omega'^{\uparrow\{c_1,\ldots,c_n\}}, \{c_1, \ldots, c_n\}]$. This indeed gives the well known unmarked information algebra $(2^\Omega, 2^{\{c_1,\ldots,c_n\}})$. Given an element ϕ in this unmarked information algebra, a support of it is any subset L of $\{c_1, \ldots, c_n\}$ such that ϕ contains no information about variables outside L, that is

$$\phi \quad = \quad \left(N_L\left(\bigvee_{\omega\in\phi} c(\omega)\right)\right)^{\uparrow\{c_1,\ldots,c_n\}}. \tag{5.18}$$

5.2 Arguments

Consider a generalized hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D)$ where Ω is the set of interpretations of the language $\mathcal{FSC}$. The concept of arguments, defined in subsection 3.5.1 for elements of Ω, can be carried over to the language $\mathcal{FSC}$. For $f \in \mathcal{FSC}$ we define:

- f is a **quasi-supporting argument** for $\phi \in \Phi$, if

$$\Gamma(\omega) \geq \phi \quad \text{for every } \omega \in N(f). \tag{5.19}$$

- f is a **conflicting argument**, if it is a quasi-supporting argument for z, that is
$$\Gamma(\omega) = z \quad \text{for every } \omega \in N(f). \tag{5.20}$$

- f is a **supporting argument** for $\phi \in \Phi$, if it is a quasi-supporting argument for ϕ but not a quasi-supporting argument for z, that is
$$\Gamma(\omega) \geq \phi \text{ and } \Gamma(\omega) \neq z \quad \text{for every } \omega \in N(f). \tag{5.21}$$

- f is a **possibly supporting argument** for $\phi \in \Phi$, if
$$\Gamma(\omega) \oplus \phi \neq z \quad \text{for every } \omega \in N(f). \tag{5.22}$$

- f is a **refuting argument** for $\phi \in \Phi$ if
$$\Gamma(\omega) \oplus \phi = z \quad \text{for every } \omega \in N(f). \tag{5.23}$$

5.3 Implicates and Implicants

We are interested in representing general formulas in $\mathcal{FSC}$ by logically equivalent ones which have a simpler form. Optimally, a formula should be represented by a logically equivalent formula with minimal size, but the computations of such a representation is in general not feasible and the resulting formula not very handy to work with. In this subsection, generalizing the ideas from propositional logic, we will introduce the concepts of implicants and implicates, which allow to generate reasonable representations.

In order to simplify notations, we first define a partial ordering on FSC conjunctions. Let $co = \bigwedge_{(i,V)\in I} M(c_i, V)$ be an FSC conjunction. Then we define the partial relation $\succ$ by

$$\bigwedge_{(i,V)\in I} M(c_i, V) \quad \succ \quad \bigwedge_{(i,V')\in I'} M(c_i, V') \tag{5.24}$$

for every I' satisfying

$$I \neq I' = \{(i, V') : (i, V) \in I,\, V \subseteq V'\}.$$

The right-hand side of (5.24) must also be an FSC conjunction, that is every FSC satisfying $M(c, V) \cong \top$ is removed (the empty conjunction is $\top$).

A dual definition is possible for FSC clauses: Let $cl = \bigvee_{(i,V)\in I} M(c_i, V)$ be an FSC clauses. Then we define the partial relation $\prec$ by

$$\bigvee_{(i,V)\in I} M(c_i, V) \quad \prec \quad \bigvee_{(i,V')\in I'} M(c_i, V') \tag{5.25}$$

for every I' satisfying

$$I \neq I' = \{(i, V') : (i, V) \in I,\, V \supseteq V'\}.$$

The right hand side of (5.25) must also be a FSC clause, that is every FSC satisfying $M(c, \emptyset) \cong \perp$ is removed (the empty clause is $\perp$).

Clearly, $\succ$ defined by (5.24) is a **strict partial order** on the set of FSC conjunctions and $\prec$ defined by (5.25) is a **strict partial order** on the set of FSC clauses.

Lemma 5.3 *$co \succ co'$ implies $co \models co'$ and $cl \prec cl'$ implies $cl' \models cl$.*

Proof of lemma 5.3 Follows from the respective definitions (5.24) and (5.25). □

An FSC conjunction co is called an **implicant** of a formula $f \in \mathcal{FSC}$ if $co \models f$. It is called a **prime implicant** of f if it is an implicant of f and if any FSC conjunction co' satisfying $co \succ co'$ is not an implicant of f.

The definitions for FSC clauses are analogous: An FSC clause cl is called an **implicate** of a formula $f \in \mathcal{FSC}$ if $f \models cl$. It is called a **prime implicate** of f if it is an implicate of f and if any FSC clause cl' satisfying $cl \prec cl'$ is not an implicate of f.

Generalizing results from propositional logic, the set of prime implicants $PI(f)$ or the set of prime implicates $PC(f)$ of a formula $f \in \mathcal{FSC}$ can be used to represent the formula itself:

Lemma 5.4 *For $f \in \mathcal{FSC}$,*

$$f \cong \bigvee_{co \in PI(f)} co, \qquad f \cong \bigwedge_{cl \in PC(f)} cl. \tag{5.26}$$

Proof of lemma 5.4 We prove the first equality of the lemma. By definition, for an interpretation for which a prime implicant co is true also the formula f itself is true. On the other hand, consider an interpretation $\omega \in \Omega$ for which f is true. Consider the FSC conjunction $c(\omega)$. Clearly, ω is the only interpretation which makes $c(\omega)$ true, because $N(c(\omega)) = \{\omega\}$. Therefore $c(\omega)$ is clearly an implicant of f. But then, there must exist a prime implicant co satisfying $c(\omega) \succ co$. Lemma 5.3 implies that the interpretation makes co true and therefore the conjunction of the prime implicants is true. This proves the first equation of the lemma; the second one is proved analogously. □

The representations described in lemma 5.4 are also called the prime normal forms. Due to their canonical form, they are appropriate for implementations. Many methods for computing these representations in propositional logic have been presented in the field of minimizing Boolean circuit and applications for artificial intelligence; see (Birkhoff & Bartee, 1970; Haenni, 1996) for further literature and algorithms for computing the prime normal form, and (Ngair, 1992) for a complexity analysis of the problem. These algorithms can usually be generalized to FSC.

Yet the representation by prime implicants or implicates might be redundant, that is some implicates or implicates respectively may be eliminated in (5.26) without changing the logical equivalences. A subsets Q of $PI(f)$ with $f \cong \bigvee_{co \in Q} co$ is called **irredundant**, if the removal of any element from Q will destroy this equality. An analogous definition is possible for implicants.

Lemma 5.5 *The minimal and the shortest FSC DNFs or FSC CNFs of a formula $f \in \mathcal{FSC}$ are irredundant with respect to f.*

Proof of lemma 5.5 Analogous to the proof of theorem 5.3 of (Kohlas & Monney, 1995) □

Denote by $\mathcal{FSC}_{CL} \subseteq \mathcal{FSC}$ the set of clauses, by $\mathcal{FSC}_{CO} \subseteq \mathcal{FSC}$ the set of conjunctions in the language $\mathcal{FSC}$ such that $PI(f) \subseteq \mathcal{FSC}_{CO}$ and $PC(f) \subseteq \mathcal{FSC}_{CL}$. Note that we consider only "proper" clauses and conjunctions, that is every atom of the language occurs at most once therein.

For a further discussion of representations, rather on the level of implementations, see also (Kohlas *et al.*, 2000).

5.4 Conflicts and Diagnoses

Consider again a generalized hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D)$ where Ω is the set of interpretations of the language $\mathcal{FSC}$. In (3.28), the conflicts of the $\mathcal{H}$ are represented as a subset CS of the set of interpretations Ω. Here, we will focus on a representation of the conflicts in logical form in the language $\mathcal{FSC}$; this will have several advantages especially for the computation. Let c be the function defined in (5.8), then define the logical representation $Conf$ of the conflicts CS by

$$Conf := \bigvee_{\omega \in CS} c(\omega). \tag{5.27}$$

Clearly, $N(Conf) = CS$.

Following the ideas of (Kohlas *et al.*, 1998), it can now be shown that $Conf$ has a maximality property in the set of arguments:

Lemma 5.6 *The formula $Conf$ is the weakest conflicting argument (up to logical equivalence), that is*

(1) $Conf$ is a conflicting argument.

(2) If $f \in \mathcal{FSC}$ is a conflicting argument, then $f \models Conf$.

Proof of lemma 5.6 (1) Let $\omega \in N(Conf)$. Then by definition of $Conf$, $\omega \in CS$ and therefore, by the definition of CS (3.28), $\Gamma(\omega) = z$ which shows that $Conf$ is a conflicting argument.

(2) Let $f \in \mathcal{FSC}$ be a conflicting argument and $\omega \in N(f)$. Then $\Gamma(\omega) = z$ because f is a conflicting argument. $\Gamma(\omega) = z$ implies that $\omega \in CS$. By $CS = N(Conf)$ we have $N(f) \subseteq N(Conf)$, thus $f \models Conf$, and this proves the lemma. □

The diagnoses DS are those interpretations which are not contained in the conflicts CS and therefore the set can be represented by a logical formula $Diag$ as

$$Diag := \bigvee_{\omega \in DS} c(\omega). \tag{5.28}$$

Then it follows that

$$N(Diag) = DS = \Omega - CS = \Omega - N(Conf) = N(\neg Conf). \tag{5.29}$$

such that

$$Diag \cong \neg Conf. \tag{5.30}$$

This shows that the logical formulation of the diagnoses can be obtained from the logical formulation of the conflicts and vice versa. By definition, $Conf$ and $Diag$ are considered as disjoint normal forms (DNF), that is a disjunction of conjunctions, and therefore the computation of $Diag$ from $Conf$ requires a transformation of a negated DNF $\neg Conf$ to a DNF, which in the worst case is an exponential task. However, it is usually sufficient to deal with simpler formulas which are equivalent to $Diag$ or $Conf$.

Lemma 5.7 *If the information algebra (Φ, D) is consistent*[3], *then the formula $Diag$ is the weakest supporting argument (up to logical equivalence) for the neutral element e of (Φ, D), that is*

(1) $Diag$ is a supporting argument for e.

(2) If f is a supporting argument for e, then $f \models Diag$.

Proof of lemma 5.7 (1) Let $\omega \in N(Diag)$. Then by definition of $Diag$, $\omega \in DS$ and by corollary 2.3, $\Gamma(\omega) \geq e$. By the definition of DS (3.29) $\Gamma(\omega) \neq z$ which shows that $Diag$ is a supporting argument for e.

(2) Let $f \in \mathcal{FSC}$ be a supporting argument for e and $\omega \in N(f)$. Then $\Gamma(\omega) \neq z$ because f is a supporting argument. $\Gamma(\omega) \neq z$ implies that $\omega \in DS$. By $DS = N(Diag)$ we have $N(f) \subseteq N(Diag)$, thus $f \models Diag$, and this proves the lemma. □

[3]The information algebra is consistent if $e \neq z$, see subsection 2.1.1.

An FSC conjunction d is called a **diagnosis** if $d \models Diag$. It is called a **minimal diagnosis** if it is a diagnosis and no FSC conjunction $f' \in \mathcal{FSC}$ satisfying $f \succ f'$ is a diagnosis. The set of minimal diagnoses is denoted by $\mu Diag$. Analogously, an FSC conjunction c is called a **minimal conflicting argument** or simply a **minimal conflict** if it is a conflicting argument and no FSC conjunction $c' \in \mathcal{FSC}$ satisfying $c \succ c'$ is a conflicting argument. The set of minimal conflicts is denoted by $\mu Conf$. These are representations of $Diag$ and $Conf$ because clearly

$$Diag = \bigvee_{d \in \mu Diag} d \qquad \text{and} \qquad Conf = \bigvee_{c \in \mu Conf} c, \tag{5.31}$$

and they will be of interest in chapter 7. The set of minimal conflicts $\mu Conf$ can be obtained on the node by computing the prime implicates of the conflict $Conf$ (cf. for example (Birkhoff, 1948; Haenni, 1996)), and applying the theorem of Reiter & De Kleer (1987). A discussion of an efficient method for computing diagnoses from minimal conflicts as well as further literature on this topic can be found in (Haenni, 1998).

5.5 Quasi-Support and Support

In section 3.5.1 quasi-support and support sets have been introduced. Here, these concepts will be reformulated in the logical framework of finite set constraints $\mathcal{FSC}$. The formulas defined in this section are not very convenient representations of the respective sets. But in the sequel, we will in general only work with simpler formulas in $\mathcal{FSC}$ which are logically equivalent to them as well as, in some sense, easy to handle with, for example one of the respective prime normal forms.

The **quasi-support** $qs(\phi)$ of an information $\phi \in \Phi$ is defined as a logical formula representing $QSS(\phi)$,

$$qs(\phi) \quad := \quad \bigvee_{\omega \in QSS(\phi)} c(\omega), \tag{5.32}$$

therefore $N(qs(\phi)) = QSS(\phi)$.

The mapping qs is an intersection homomorphism like QSS (cf. theorem 3.6):

Lemma 5.8 *For $\phi_1, \phi_2 \in \Phi$,*

$$qs(\phi_1 \oplus \phi_2) \quad \cong \quad qs(\phi_1) \wedge qs(\phi_2), \tag{5.33}$$
$$qs(e) \quad \cong \quad \top. \tag{5.34}$$

Proof of lemma 5.8 This follows from theorem 3.6 and the definition of the quasi-support above. □

Similar, the **support** $sp(\phi)$ of an information $\phi \in \Phi$ is defined as a logical formula representing $SS(\phi)$,

$$sp(\phi) \quad := \quad \bigvee_{\omega \in SS(\phi)} c(\omega), \tag{5.35}$$

therefore $N(sp(\phi)) = SS(\phi)$.

The relation between the logical formulation of quasi-support and support of any information in Φ follows from the definition of the support set (3.18).

Lemma 5.9 *For an information $\phi \in \Phi$,*

$$sp(\phi) \cong qs(\phi) \wedge \neg qs(z). \tag{5.36}$$

Proof of lemma 5.9 By definition we have $QSS(\phi) \cap (\Omega - QSS(z)) = SS(\phi)$, which means that $N(qs(\phi)) \cap N(qs(z))^c = N(qs(\phi) \wedge \neg qs(z)) = N(sp(\phi))$, and this implies the lemma. $\square$

The quasi-support is a maximal element in the set of quasi-supporting arguments for a formula:

Lemma 5.10 *$qs(\phi)$ the weakest quasi-supporting argument (up to logical equivalence) for the hypothesis $\phi \in \Phi$, that is*

(1) $qs(\phi)$ is a quasi-supporting argument for ϕ, and

(2) if f is a quasi-supporting argument for ϕ, then $f \models qs(\phi)$.

Proof of lemma 5.10 (1) Let $\omega \in N(qs(\phi))$. Then we have $\omega \in QSS(\phi)$ and, by definition of QSS (3.13), $\Gamma(\omega) \geq \phi$. This shows (1).

(2) Let f be a quasi-supporting argument for ϕ and $\omega \in N(f)$. This implies $\Gamma(\omega) \geq \phi$. The definition of QSS (3.13) implies then that $\omega \in QSS(\phi)$. Therefore $N(f) \subseteq QSS(\phi) = N(qs(\phi))$ such that finally $f \models qs(\phi)$, and this proves the lemma. $\square$

The support sp defines a supporting argument if $sp(\phi) \ncong \bot$. If $sp(\phi) \cong \bot$ then $e = z$, and this means that the information algebra is inconsistent (cf. subsection 2.1.1).

Lemma 5.11 *If $sp(\phi) \ncong \bot$ then $sp(\phi)$ is a supporting argument of the information $\phi \in \Phi$.*

Proof of lemma 5.11 Suppose that $sp(\phi) \ncong \bot$ such that there is an $\omega \in N(sp(\phi))$, therefore $\omega \in SS(\phi)$. By the definition of SS (3.18), this implies that $\Gamma(\omega) \neq z$ and $\Gamma(\omega) \geq \phi$ such that $sp(\phi)$ is a supporting argument. $\square$

5.6 Possible Support and Refutation

Two other sets of interpretations have been defined in section 3.5: the possibly supporting set and the refuting set. Like in the previous section, we define logical representations of such sets in the language $\mathcal{FSC}$.

The **possible support** $pss(\phi)$ of a formula $\phi \in \Phi$ is defined as a logical formula representing $PSS(\phi)$,

$$pss(\phi) \quad := \quad \bigvee_{x \in PSS(\phi)} c(x). \tag{5.37}$$

Similarly, the **refutation** $rs(\phi)$ of a formula $\phi \in \Phi$ is defined as a logical formula representing $RS(\phi)$,

$$rs(\phi) \quad := \quad \bigvee_{x \in RS(\phi)} c(x). \tag{5.38}$$

Lemma 5.12 *For $\phi \in \Phi$,*

$$pss(\phi) \models Diag, \qquad Conf \models rs(\phi), \qquad pss(\phi) \cong \neg rs(\phi).$$

Proof of lemma 5.12 Follows from lemmas 3.12, 3.13 and 3.14 respectively. □

5.7 Marginalization and Combination

The definitions of marginalization and combination for allocations of quasi-support can be carried over to the quasi-support functions: for $\phi \in \Phi$,

$$qs^{\Rightarrow x}(\phi) \quad := \quad \bigvee_{\psi = \psi^{\Rightarrow x} \geq \phi} qs(\psi), \tag{5.39}$$

$$(qs_1 \oplus qs_2)(\phi) \quad := \quad \bigvee_{\phi \leq \phi_1 \oplus \phi_2} (qs_1(\phi_1) \wedge qs_2(\phi_2)). \tag{5.40}$$

An allocation of support sp is clearly also an intersection homomorphism (like the allocation of quasi-support, see lemma 5.8), and the definitions and results in this section can also be applied to sp.

Lemma 5.13 *For $x \in D$, $\phi \in \Phi$ and qs_i defined by (5.32),*

$$\begin{aligned} N({qs_1}^{\Rightarrow x}(\phi)) &= {QSS_1}^{\Rightarrow x}(\phi), \\ N((qs_1 \oplus qs_2)(\phi)) &= (QSS_1 \oplus QSS_2)(\phi). \end{aligned}$$

Proof of lemma 5.13 Follows from the definitions together with (3.30) and (3.31). □

The following results will be useful for computational purposes.

Lemma 5.14 *Let $\phi \in \Phi$ and $x \in D$, then*

$$\begin{array}{rcll} sp^{\Rightarrow x}(e) & \cong & Diag & \textit{for any } x, \\ qs^{\Rightarrow x}(z) & \cong & Conf & \textit{if } x \textit{ is a support for } z, \\ qs^{\Rightarrow x}(\phi) & \cong & qs(\phi) & \textit{if } x \textit{ is a support for } \phi. \end{array}$$

Proof of lemma 5.14 Follows from the definition of qs (5.32) and sp (5.35) together with lemma 3.17. □

6 Argumentation Systems

In chapters 2 to 5 the main ingredients for the present chapter have been introduced: allocations of arguments and probability, generalized hints, information systems, and FSC languages. Now these ingredients will be placed in a common framework, in order to build so-called argumentation systems. An argumentation system which satisfies some additional properties allows to construct a generalized hint, which can then be used to compute arguments in favor of or against a hypothesis by using the corresponding allocation of arguments.

In the first section we define the notion of an argumentation system, and in section 6.2 such a system is used to construct a generalized hint. Section 6.3 shows how the argumentation system can be interpreted as an information system and how an information algebra can be built on top of it. In chapter 7 we will show that argumentation systems are a convenient representation for model-based diagnostics, that they are in fact a sort of minimal requirement for doing model-based diagnostics.

6.1 Definition

Consider an information system $(\mathcal{L}, \vdash)$ as defined in chapter 4. Further assume that the language $\mathcal{FSC}$ describes subsets of Ω as in chapter 5. A partial mapping χ from $\mathcal{FSC}$ to $2^{\mathcal{L}}$ has the meaning that for a formula $f \in \mathcal{FSC}$, every interpretation $\omega \in \Omega$ with $\omega \models f$ "implies" the information modeled by the set of formulas $X = \chi(f)$, if it exists. This partial mapping describes for example the functioning of a component described by $\chi(f)$ under the assumption that one of the interpretations described by f is true. A mapping $f : A \to B$ is called partial if $f(a)$ is not necessarily defined for every $a \in A$. We use a partial mapping here because often information is specified in this way. For example, we often know how a component behaves if it works correctly and represent this information using some sort of implication, but we don't know anything about its behavior otherwise and do not represent this information at all.

We call the tuple $(\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$ an **argumentation system**.[1] If, additionally, a probability distribution pr is given for every variable c_i occurring in $\mathcal{FSC}$, that means if

$$pr(c_i = v_{ik}) = pr_{ik} \geq 0 \qquad \text{for every possible value } v_{ik}, \tag{6.1}$$

with

$$\sum_{k=1}^{r_i} pr_{ik} = 1 \qquad \text{for } i = 1, \ldots, n, \tag{6.2}$$

then we call the tuple $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$ a **probabilistic argumentation system**. Usually, for every pair of variables $c_{i_1} \neq c_{i_2}$, their probability distributions p_{i_1} and p_{i_2} are assumed to be stochastically independent, where $p_i(v_{ik}) := pr(x_i = v_{ik})$, $i = 1, \ldots, n$ and $k = 1, \ldots, r_i$; but the formalism is not restricted to this situation.

Argumentation systems are a convenient formalism to express information for model-based diagnostics as we will show in chapter 7. For every information expressed in the language $\mathcal{L}$ the argumentation system allows to compute the "arguments" for the information specified by the partial mapping χ.

Consider two argumentation systems $(\mathcal{FSC}, \chi_i, \mathcal{L}, \vdash)$, $i = 1, 2$, on the same language $\mathcal{FSC}$ and information system $(\mathcal{L}, \vdash)$. These argumentation systems can be combined into a new argumentation system $(\mathcal{FSC}, \chi_1 \oplus \chi_2, \mathcal{L}, \vdash)$ by defining

$$(\chi_1 \oplus \chi_2)(f) \quad := \quad \left\{ \begin{array}{ll} \chi_1(f) \cup \chi_2(f) & \text{if } \chi_1(f) \text{ and } \chi_2(f) \text{ are defined,} \\ \chi_1(f) & \text{if only } \chi_1(f) \text{ is defined,} \\ \chi_2(f) & \text{if only } \chi_2(f) \text{ is defined,} \\ \text{undefined} & \text{otherwise.} \end{array} \right\}$$

This combination operation is clearly associative, commutative and idempotent. The neutral element is $(\mathcal{FSC}, \chi_\nu, \mathcal{L}, \vdash)$ where the mapping χ_ν is undefined for every $f \in \mathcal{FSC}$.

A well known example of an argumentation system is the so called **propositional argumentation system** (Kohlas *et al.*, 2000; Haenni *et al.*, 1999b). It is usually represented as a triple (ξ, A, P) where A and P are two disjoint sets of proposition and ξ a sentence in the propositional language $L(A \cup P)$ over the atoms $A \cup P$. In our terms, the propositional language $L(A)$ is equivalent to a language $\mathcal{FSC}$ where the domain V_i of every variable c_i consist of only two values (true and false). The language $L(P)$ describes a propositional information system $(\mathcal{L}, \vdash)$, and $\xi \in L(A \cup P)$ can be represented by a set of formulas $X = \{\alpha_i \to \pi_i : \alpha_i \in L(A),\ \pi_i \in L(P),\ i \in I\}$ such that

$$\xi \quad \text{is logically equivalent to} \quad \bigwedge_{(\alpha \to \pi) \in X} (\alpha \to \pi), \tag{6.3}$$

[1]In fact, this can even be generalized to situations, where at the place of $\mathcal{FSC}$ there is only a lattice, see (Monney, 1994).

and we define

$$\chi(\alpha) \quad := \quad \{\pi : (\alpha \to \pi) \in X\}. \tag{6.4}$$

Clearly, the representation X is not unique, but all representations are logically equivalent and the corresponding deduced generalized hints are then equal. A propositional argumentation system can also be extended to a language built of finite set constraints (Haenni & Lehmann, 1998a). More general examples have also been presented in the context of algebraic assumption based systems (Haenni, 1997). Classical or generalized hints can also be interpreted as argumentation systems.

6.2 From Argumentation Systems to Generalized Hints

Let $(\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$ be a probabilistic argumentation system. Using techniques presented in the previous chapters, we can build a generalized hint based on this system.

In the sequel, we will usually represent the partial mapping χ by a set of formulas, i.e. for every existing $\chi(f) = X$, $f \in \mathcal{FSC}$ and $X \subseteq \mathcal{L}$, we write

$$f \rightarrowtail X, \tag{6.5}$$

where the arrow "$\rightarrowtail$" reflects the partial mapping χ, and we denote the set of all these formulas by Ξ,

$$\Xi \quad := \quad \{f \rightarrowtail X : \chi(f) = X,\ f \in \mathcal{FSC},\ X \subseteq \mathcal{L}\}, \tag{6.6}$$

called the **representation of** χ.

If the information system $(\mathcal{L}, \vdash)$ satisfies the conditions (C1) to (C6) stated in chapter 4, then we can construct the corresponding unmarked information algebra (Φ, S). For every set of formulas $X \subseteq \mathcal{L}$, the mapping $C : 2^{\mathcal{L}} \to \Phi$ selects the corresponding element of the information algebra (Φ, S) as induced by the construction in section 4.2. For two sets of formulas $X, Y \subseteq \mathcal{L}$, lemma 4.3 implies that $C(X) \oplus C(Y) = C(X \cup Y)$. For every interpretation $\omega \in \Omega$ we now collect all information available in Ξ which corresponds to ω, that means we select the set of formulas

$$F_\omega := \{(f \rightarrowtail X) \in \Xi : \omega \in N(f)\}$$

and define the mapping Γ which associates the interpretation ω with the information defined by the mapping χ by means of its representation Ξ,

$$\bigoplus_{(f \rightarrowtail X) \in F_\omega} C(X) \quad = \quad C\left(\bigcup_{(f \rightarrowtail X) \in F_\omega} X\right),$$

that is

$$\begin{aligned} \Gamma: \quad \Omega &\rightarrow \Phi \\ \omega &\mapsto \Gamma(\omega) := \bigoplus_{\substack{(f \rightarrowtail X) \in \Xi \\ \omega \in N(f)}} C(X) = C\left(\bigcup_{\substack{(f \rightarrowtail X) \in \Xi \\ \omega \in N(f)}} X \right). \end{aligned} \tag{6.7}$$

Note that the combination over an empty set is the neutral element e of the information algebra (Φ, S). Γ is a total mapping and reflects the information expressed by χ. In general, there is no one-to-one mapping from χ to Γ, but nevertheless, the construction presented is surjective. The image of some elements of Ω might be the null element z of the information algebra (Φ, S).

A probability measure P on 2^Ω is defined by

$$P(A) := \sum_{\substack{\omega \in D \\ \omega = (v_1, \ldots, v_n)}} \left(\prod_{i=1}^{n} pr(c_i = v_i) \right) \tag{6.8}$$

for every subset $A \subseteq \Omega$.

The tuple $\mathcal{H} = (\Omega, \Gamma, \Phi, S, 2^\Omega, P)$ is then a generalized hint. Based on this hint, an allocation of support and probability $QSS_{\mathcal{H}}$ and a belief function $bel_{\mathcal{H}} := P \circ QSS_{\mathcal{H}}$ can be defined using the concepts presented in chapter 3. The respective unmarked information algebra of allocations of support is denoted by (H_Φ, S) where (Φ, S) is the information algebra constructed from the information system $(\mathcal{L}, \vdash)$. These structures can then be used to compute conflicts, diagnoses, quasi-support, etc. as defined in chapter 3. Examples of argumentation systems are presented in chapter 7 in the context of model-based diagnostics.

An argumentation system together with the induced concepts of generalized hints, allocations of support and probability, and belief function is depicted in figure 6.1.

Consider two probabilistic argumentation systems $(\mathcal{FSC}, \chi_i, \mathcal{L}, \vdash, pr)$, $i = 1, 2$. Then the hint $\mathcal{H} = (\Omega, \Gamma, \Phi, S, 2^\Omega, P)$ defined on the combined argumentation system $(\mathcal{FSC}, \chi_1 \oplus \chi_2, \mathcal{L}, \vdash, pr)$ is clearly equal to the combination $\mathcal{H}_1 \oplus \mathcal{H}_2$ of the two hints $\mathcal{H}_i = (\Omega, \Gamma_i, \Phi, S, 2^\Omega, P)$, $i = 1, 2$ defined on the respective argumentation system, because

$$\begin{aligned} \Gamma(\phi) &= \bigoplus_{\substack{(f \rightarrowtail X) \in \Xi \\ \omega \in N(f)}} C(X) = \left(\bigoplus_{\substack{(f \rightarrowtail X) \in \Xi_1 \\ \omega \in N(f)}} C(X) \right) \oplus \left(\bigoplus_{\substack{(f \rightarrowtail X) \in \Xi_2 \\ \omega \in N(f)}} C(X) \right) \\ &= \Gamma_1(\phi) \oplus \Gamma_2(\phi), \end{aligned}$$

where Ξ_i is the representation of χ_i, $i = 1, 2$, and Ξ the representation of $\chi := \chi_1 \oplus \chi_2$. Further, we have already shown in chapter 3 that the allocation of support deduced from a combined hint is equal to the combination of the two allocations of support respectively induced by the two hints. This shows that the combination can be made in any step of the process.

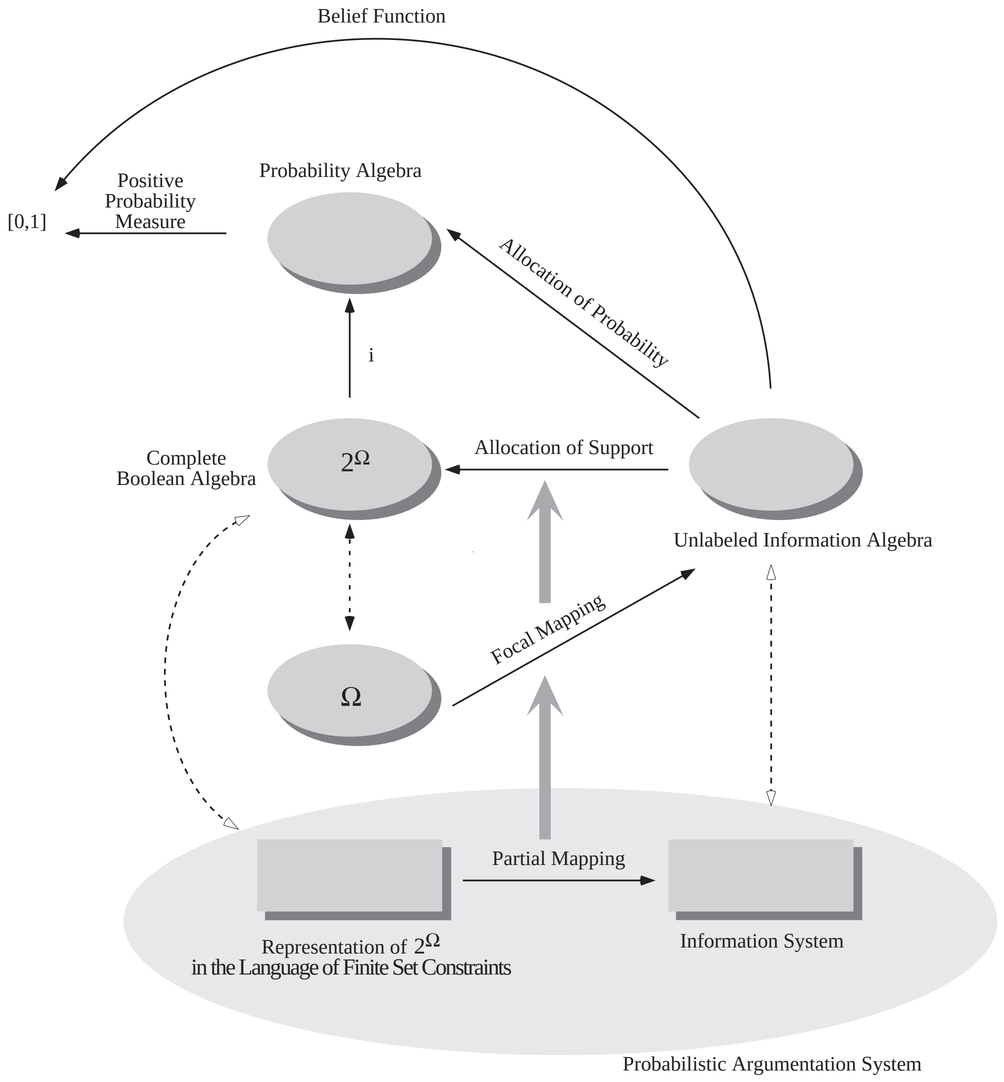

Belief Function
Positive Probability Measure
[0,1]
Probability Algebra
Allocation of Probability
i
Complete Boolean Algebra
2^Ω
Allocation of Support
Unlabeled Information Algebra
Focal Mapping
Ω
Partial Mapping
Representation of 2^Ω in the Language of Finite Set Constraints
Information System
Probabilistic Argumentation System

Figure 6.1: An argumentation system together with the induced concepts of generalized hint, allocation of support and of probability, and belief function.

For an argumentation system, the conflict CS and the diagnoses DS as well as their logical representations *Conf* and *Diag* are defined as the corresponding concepts for the induced allocations. Further, the quasi-support (set) of a set of formulas $X \subseteq \mathcal{L}$ is defined by extending the definition for formulas in the information algebra (Φ, S) as

$$\begin{aligned} QSS(X) &:= QSS(C(X)), && (6.9)\\ qs(X) &:= qs(C(X)), && (6.10)\end{aligned}$$

and similarly also other concepts like support, degree of support, etc. Analogous results as in chapter 5 can be proved, e.g. $QSS(X \cup Y) = QSS(X) \cup QSS(Y)$ for $X, Y \subseteq \mathcal{L}$.

6.3 Argumentation Systems as Information Systems

Let $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$ be an argumentation system and S the lattice of interesting sublanguages of $\mathcal{L}$ as introduced in section 4.2. We will show that an argumentation system is also an information system and an information algebra can be constructed on top of it without passing through generalized hints. Yet the resulting information algebra is not isomorphic to the one constructed via generalized hints; in section 6.3.3 we discuss the link between them.

6.3.1 The Corresponding Information System

The language $\mathcal{L}^+$ describes all possible formulas which can occur in a representation Ξ of the mapping χ of an argumentation system. We restrict the occurring FSC formulas to be FSC conjunctions. This can always be fulfilled by transforming a general FSC f into a logically equivalent FSC DNF $f_1 \vee \cdots \vee f_n$ and splitting the corresponding formula $f \rightarrowtail X$ into a set of formulas $\{(f_i \rightarrowtail X) : i = 1, \ldots, n\}$. Define

$$\mathcal{L}^+ \quad := \quad \{(f \rightarrowtail X) : f \in \mathcal{FSC}_{CO},\ X \subseteq \mathcal{L}\}. \tag{6.11}$$

The lattice S from the information system $(\mathcal{L}, \vdash)$ is extended to a lattice S^+ by

$$S^+ := \{ex(L) : L \in S\} \tag{6.12}$$

where

$$ex(L) := \{(f \rightarrowtail X) : f \in \mathcal{FSC}_{CO},\ X \in L\}. \tag{6.13}$$

This means in fact that S^+ does not restrict the $\mathcal{FSC}$ part. Using the compact entailment relation $\vdash$, a relation $\vdash^+$ is defined for formulas in $\mathcal{L}^+$:

$$\{(f_i \rightarrowtail X_i) : i \in I\} \vdash^+ (f \rightarrowtail X) \tag{6.14}$$

$$\text{if there is an } I' \subseteq I \text{ such that } X \subseteq C\left(\bigcup_{i \in I'} X_i\right) \text{ and } f = \bigwedge_{i \in I'} f_i.$$

Lemma 6.1 *The relation $\vdash^+$ is an entailment relation, that is it satisfies the conditions (E1) and (E2) from chapter 4.*

Proof of lemma 6.1 (E1) means $\{(f_i \rightarrowtail X_i) : i \in I\} \vdash^+ (f_j \rightarrowtail X_j)$ for every $j \in I$. This is fulfilled by $I' := \{j\}$.

(E2): Let $\{(f_i \rightarrowtail X_i) : i \in I\} \vdash^+ (g_j \rightarrowtail Y_j)$ for every $j \in J$ and $\{(g_j \rightarrowtail Y_j) : j \in J\} \vdash^+ (h \rightarrowtail Z)$. We have to prove that $\{(f_i \rightarrowtail X_i) : i \in I\} \vdash^+ (h \rightarrowtail Z)$.

For every $j \in J$ there is a subset $I'_j \subseteq I$ such that

$$Y_j \subseteq C\left(\bigcup_{i \in I'_j} X_i\right) \qquad \text{and} \qquad g_j = \bigwedge_{i \in I'_j} f_i.$$

There is a subset $J' \subseteq J$ such that

$$Z \subseteq C\left(\bigcup_{j \in J'} Y_j\right) \qquad \text{and} \qquad h = \bigwedge_{j \in J'} g_j.$$

This implies that

$$h = \bigwedge_{j \in J'} \left(\bigwedge_{i \in I'_j} f_i\right) = \bigwedge_{i \in I^*} f_i \qquad \text{where} \qquad I^* := \bigcup_{j \in J'} I'_j. \tag{6.15}$$

Further,

$$Z \subseteq C\left(\bigcup_{j \in J'} Y_j\right) \subseteq C\left(\bigcup_{j \in J'} C\left(\bigcup_{i \in I'_j} X_i\right)\right).$$

Yet for the right-hand side, we have

$$C\left(\bigcup_{j \in J'} C\left(\bigcup_{i \in I'_j} X_i\right)\right) \subseteq C\left(\bigcup_{j \in J'} C\left(\bigcup_{i \in I^*} X_i\right)\right) = C\left(\bigcup_{i \in I^*} X_i\right).$$

This implies that

$$Z \subseteq C\left(\bigcup_{i \in I^*} X_i\right),$$

and together with (6.15) we have $\{(f_i \rightarrowtail X_i) : i \in I\} \vdash^+ (f \rightarrowtail Z)$. Hence (E2) is proved. □

Note that $\vdash^+$ is in general not a *compact* entailment relation, that is (E3) is not fulfilled, see the following counterexample:

Example 6.2: $\vdash^+$ is not necessarily compact

Consider sets $X_i \in \mathcal{L}$ and $f_i \in \mathcal{FSC}$ for $i \in \mathbb{N}$. Define now

$$X := C\left(\bigcup_{i \in \mathbb{N}} X_i\right) \qquad \text{and} \qquad f := \bigwedge_{i \in \mathbb{N}} f_i.$$

Assume that C is compact and therefore using (E3),

$$X = \bigcup \left\{ C(Y) : Y \subseteq \bigcup_{i \in \mathbb{N}} X_i,\ Y \text{finite} \right\} = \bigcup_{i \in \mathbb{N}} C\left(\bigcup_{j \leq i} X_i \right).$$

In a general situation, we have for every finite $I \subset \mathbb{N}$

$$X \not\subseteq C\left(\bigcup_{i \in I} X_i \right)$$

such that

$$\{(f_i \rightarrowtail X_i), i \in I\} \not\vdash^{+} (f \rightarrowtail X).$$

Thus here, $\vdash^{+}$ is not compact. $\ominus$

6.3.2 Construction of the Information Algebra

Define the relation C^{+} by means of the entailment relation $\vdash^{+}$ as

$$C^{+}(X) := \{\ell \in \mathcal{L}^{+} : X \vdash^{+} \ell\} \tag{6.16}$$

for $X \subseteq \mathcal{L}^{+}$ and its restriction to a sublanguage $L \in S^{+}$ by

$$C_L^{+}(X) := C^{+}(X) \cap L \tag{6.17}$$

for $X \subseteq \mathcal{L}^{+}$. Then C^{+} satisfies the conditions (C1), (C2) and (C3) from chapter 4 and is therefore a consequence relation, but not necessarily compact, that is (C4) is not necessarily fulfilled. Yet additionally, C^{+} satisfies the conditions (C5) and (C6):

Lemma 6.3 *The consequence relation C^{+} satisfies the conditions (C5) and (C6) from chapter 4.*

Proof of lemma 6.3 First we prove that C^{+} satisfies condition (C5). Let $F = \{f_i \rightarrowtail X_i : i \in I\} \subseteq \mathcal{L}^{+}$ and $L, M \in S^{+}$.

$$C_L^{+}(F) \quad = \quad \left\{ (f \rightarrowtail X) : X \subseteq C_L\left(\bigcup_{i \in I'} X_i \right),\ f = \bigwedge_{i \in I'} f_i,\ I' \subseteq I \right\}$$

such that

$$\begin{aligned} & C_M^{+}(C_L^{+}(F)) \\ & \quad = \left\{ (g \rightarrowtail Y) : Y \subseteq C_M\left(\bigcup_{(h \rightarrowtail Z) \in Q} Z \right),\ g = \bigwedge_{(h \rightarrowtail Z) \in Q} h,\ Q \subseteq C_L^{+}(F) \right\} \end{aligned}$$

and denote this set by A. Further,

$$C^+_{M\cap L}(F) \quad = \quad \left\{(f \rightarrowtail X) : X \subseteq C_{M\cap L}\left(\bigcup_{i\in I'} X_i\right), \, f = \bigwedge_{i\in I'} f_i, \, I' \subseteq I\right\}$$

and denote this set by B. We have to show that $C^+(A) = C^+(B)$, or equivalently, $A \subseteq C^+(B)$ and $B \subseteq C^+(A)$. Let $(f \rightarrowtail X) \in B$. So there is a $I' \subseteq I$ such that $X \subseteq C_{M\cap L}(\bigcup_{i\in I'} X_i)$ and $f = \bigwedge_{i\in I'} f_i$. This implies that $X \subseteq C_L(\bigcup_{i\in I'} X_i)$ and therefore $(f \rightarrowtail X) \in C^+_L(F)$. Now, $X \subseteq M$ and thus by (M1) of section 4.2, $X \subseteq C_M(X)$. By setting $Q := \{(f \rightarrowtail X)\}$ we have then $(f \rightarrowtail X) \in A$.

Let now $(g \rightarrowtail Y) \in A$. Then there is a $Q \subseteq C^+_L(F)$ such that

$$Y \subseteq C_M\left(\bigcup_{(h \rightarrowtail Z)\in Q} Z\right) \qquad \text{and} \qquad g = \bigwedge_{(h \rightarrowtail Z)\in Q} h \; .$$

Further, for every $(h \rightarrowtail Z) \in Q$ there is an $I_Z \subseteq I$ with $Z \subseteq C_L(\bigcup_{i\in I_Z} X_i)$ and $h = \bigwedge_{i\in I_Z} f_i$. Set

$$I^* \quad = \quad \bigcup_{(h \rightarrowtail Z)\in Q} I_Z \; ,$$

then, using (C5) for C, we have

$$\begin{aligned} Y \quad &\subseteq \quad C_M\left(\bigcup_{(h \rightarrowtail Z)\in Q} Z\right) \qquad \subseteq \quad C_M\left(\bigcup_{(h \rightarrowtail Z)\in Q} C_L\left(\bigcup_{i\in I_Z} X_i\right)\right) \\ &\subseteq \quad C_M\left(C_L\left(\bigcup_{i\in I^*} X_i\right)\right) \quad \subseteq \quad C\left(C_M\left(C_L\left(\bigcup_{i\in I^*} X_i\right)\right)\right) \\ &= \quad C\left(C_{M\cap L}\left(\bigcup_{i\in I^*} X_i\right)\right) \end{aligned}$$

and

$$g \quad = \quad \bigwedge_{(h \rightarrowtail Z)\in Q} h \quad = \quad \bigwedge_{(h \rightarrowtail Z)\in Q}\left(\bigwedge_{i\in I_Z} f_i\right) \quad = \quad \bigwedge_{i\in I^*} f_i,$$

therefore $(g \rightarrowtail Y) \in C^+(B)$. This proves (C5) for C^+. The proof of (C6) is straightforward. □

C^+ is not compact, but nevertheless we can use it to construct a marked information algebra $\langle L_\Phi, S^+\rangle$ and from that an unmarked one, (L_Φ, S^+), using the techniques presented in section 4.2.

An information algebra constructed from a compact consequence operator yields a so-called **finitary information algebra** (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b), and from a finitary information algebra a compact consequence operator can be defined which is isomorphic to the starting one. Therefore this is a strong link between finitary information algebras and compact consequence operators. A non-compact consequence operator defines a non-finitary information algebra. The concept of finitary information algebra defines a way to approximate infinite information by finite information. Here we will not focus on this aspect, see (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b) for a discussion of finitary information algebras.

6.3.3 Comparing the Information Algebra Constructed Using Hints with the One using Information System

Given an argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$, section 6.2 shows how to construct the induced generalized hint and infer the algebra of allocations (H_Φ, S). In the preceding subsection, we have shown how the argumentation system is interpreted as an information system and the corresponding information algebra (L_Φ, S) constructed. The two unmarked information algebras (L_Φ, S^+) and (H_Φ, S) are in general not isomorphic, but there is a homomorphism from (L_Φ, S^+) to (H_Φ, S). We will show this using the corresponding marked information algebras $\langle L_\Phi, S^+\rangle$ and $\langle H_\Phi, S\rangle$, because the notations are simpler.

Denoting an equivalence class of $\langle L_\Phi, S^+\rangle$ by its representant $\langle C^+(X), L\rangle$, we define a mapping ζ by

$$\begin{array}{rrcl} \zeta: & \langle L_\Phi, S^+\rangle & \to & \langle H_\Phi, S\rangle \\ & \langle C^+(X), L\rangle & \mapsto & \zeta(\langle C^+(X), L\rangle) := \langle \rho_{\langle C^+(X),L\rangle}, L\rangle \end{array} \tag{6.18}$$

where

$$\rho_{\langle C^+(X),L\rangle}(\phi) \quad := \quad \bigcup_{\substack{(f \rightarrowtail Y)\in C^+(X) \\ C(Y)\geq\phi}} N(f).$$

An overview of the situation is depicted in fig. 6.2.

Theorem 6.4 *The mapping ζ (6.18) is a homomorphism from $\langle H_\Phi, S\rangle$ to $\langle L_\Phi, S^+\rangle$, that is for $L' \subseteq L$ (the labels of the marked allocations have been omitted),*

$$\rho_{\langle C^+(X_1),L_1\rangle} \oplus \rho_{\langle C^+(X_2),L_2\rangle} \quad = \quad \rho_{\langle C^+(X_1),L_1\rangle \oplus \langle C^+(X_2),L_2\rangle} \tag{6.19}$$

$$(\rho_{\langle C^+(X),L\rangle})^{\downarrow L'} \quad = \quad \rho_{\langle C^+(X),L\rangle^{\downarrow L'}} \tag{6.20}$$

Proof of theorem 6.4

$$(\rho_{\langle C^+(X_1),L_1\rangle} \oplus \rho_{\langle C^+(X_2),L_2\rangle})(\phi) = \bigvee_{\phi\leq\phi_1\oplus\phi_2} (\rho_{\langle C^+(X_1),L_1\rangle}(\phi_1) \wedge \rho_{\langle C^+(X_2,L_2\rangle}(\phi_2))$$

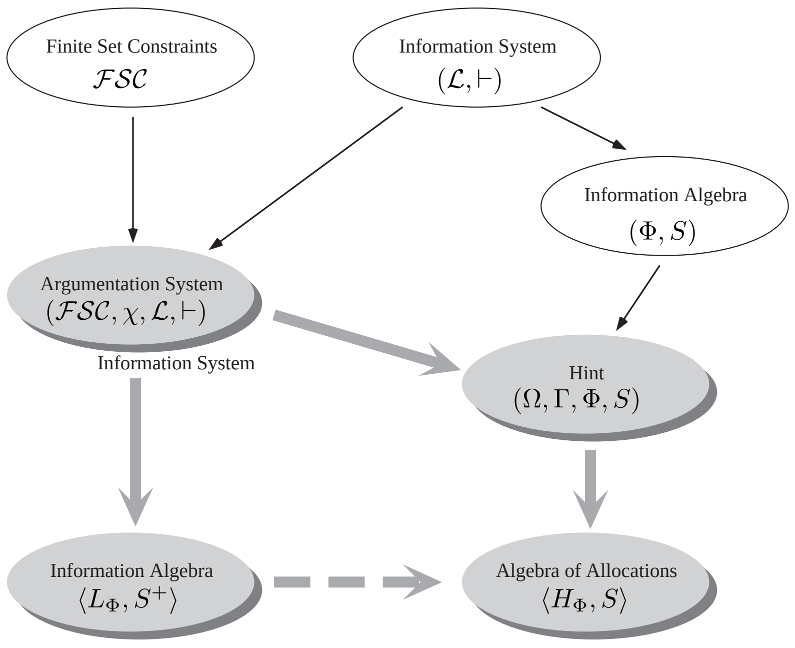


Figure 6.2: Argumentation system and related concepts.

$$
\begin{aligned}
&= \bigcup_{\phi \leq \phi_1 \oplus \phi_1} \left(\left(\bigcup_{\substack{(f \rightarrowtail Y) \in C^+(X_1) \\ C(Y) \geq \phi_1}} N(f) \right) \cap \left(\bigcup_{\substack{(g \rightarrowtail Z) \in C^+(X_2) \\ C(Z) \geq \phi_2}} N(g) \right) \right) \\
&= \bigcup_{\substack{(f \rightarrowtail Y) \in C^+(X_1) \\ (g \rightarrowtail Z) \in C^+(X_2) \\ \phi \leq C(Y) \oplus C(Z)}} (N(f) \cap N(g))
\end{aligned}
$$

and denote the last set by A. Further,

$$
\begin{aligned}
\rho_{\langle C^+(X_1), L_1 \rangle \oplus \langle C^+(X_2), L_2 \rangle}(\phi) &= \rho_{\langle C^+(C^+(X_1) \cup C^+(X_2)), L_1 \vee L_2 \rangle}(\phi) \\
&= \rho_{\langle C^+(X_1 \cup X_2), L_1 \vee L_2 \rangle}(\phi) \\
&= \bigcup_{\substack{(h \rightarrowtail H) \in C^+(X_1 \cup X_2) \\ C(H) \geq \phi}} N(h)
\end{aligned}
$$

and denote the last set by B. Now we show that $A = B$. First let $\omega \in A$. Then there exist $(f \rightarrowtail Y) \in C^+(X_1)$ and $(g \rightarrowtail Z) \in C^+(X_2)$ satisfying $C(Y) \oplus C(Z) \geq \phi$ and $\omega \in N(f) \cap N(g)$. Now, $C(Y \cup Z) = C(Y) \oplus C(Z) \geq \phi$ and $C^+(X_1) \cup C^+(X_2) \subseteq C^+(X_1 \cup X_2)$ imply $(f \wedge g \rightarrowtail Y \cup Z) \in C^+(X_1 \cup X_2)$ and therefore $\omega \in B$. Now let $\omega \in B$. Then there is a $(h \rightarrowtail H) \in C^+(X_1 \cup X_2)$

satisfying $C(H) \geq \phi$ and $\omega \in N(h)$. Now, let $X_j = \{(h_i \rightarrowtail Z_i) : i \in I_j\}$ for $j = 1, 2$. Then there is a $I' \in I_1 \cup I_2$ such that

$$h = \bigwedge_{i \in I'} h_i \qquad \text{and} \qquad H \subseteq C\left(\bigcup_{i \in I'} Z_i\right).$$

Define now $I'_j := I' \cap I_j$ and

$$h'_j := \bigwedge_{i \in I'_j} h_i \qquad \text{and} \qquad H'_j := \bigcup_{i \in I'_j} Z_i \tag{6.21}$$

for $j = 1, 2$, then we have

$$h = h'_1 \wedge h'_2 \qquad \text{and} \qquad H \subseteq C(H'_1 \cup H'_2).$$

This implies that $(h'_1 \rightarrowtail H'_1) \in C^+(X_1)$ and $(h'_2 \rightarrowtail H'_2) \in C^+(X_2)$. Furthers, $C(H'_1) \oplus C(H'_2) = C(H'_1 \cup H'_2) = C(C(H)) = C(H) \geq \phi$ and $N(h'_1) \cap N(h'_2) = N(h)$ such that $\omega \in N(h'_1) \cap N(h'_2)$. This shows that $\omega \in A$ and proves the first equation of the theorem.

For the second one consider

$$\begin{aligned}(\rho_{\langle C^+(X),L\rangle})^{\downarrow L'}(\phi) &= \bigvee_{\psi = \psi^{\Rightarrow L'} \geq \phi} \rho_{\langle C^+(X),L\rangle}(\psi) \\ &= \bigvee_{\psi = \psi^{\Rightarrow L'} \geq \phi} \left(\bigcup_{\substack{(f \rightarrowtail Y) \in C^+(X) \\ C(Y) \geq \psi}} N(f) \right) = \bigcup_{\substack{(f \rightarrowtail Y) \in C^+(X) \\ (C(Y))^{\Rightarrow L'} \geq \phi}} N(f)\end{aligned}$$

and denote the last set by A. Further, by definition

$$\rho_{\langle C^+(X),L\rangle^{\downarrow L'}}(\phi) = \rho_{\langle C^+(C^+_{L'}(X)),L'\rangle}(\phi) = \bigcup_{\substack{(g \rightarrowtail Z) \in C^+(C^+_{L'}(X)) \\ C(Z) \geq \phi}} N(g)$$

and denote the last set by B. We show now that $A = B$. First, let $\omega \in A$. Then there is a $(f \rightarrowtail Y) \in C^+(X)$ satisfying $(C(Y))^{\Rightarrow L'} \geq \phi$ and $\omega \in N(f)$. Yet

$$(C(Y))^{\Rightarrow L'} = C(C_{L'}(Y)) \geq \phi \tag{6.22}$$

such that $(f \rightarrowtail C_{L'}(Y)) \in C^+(C^+_{L'}(X))$ and therefore $\omega \in B$. Now let $\omega \in B$. Then there is $(g \rightarrowtail Z) \in C^+(C^+_{L'}(X))$ satisfying $C(Z) \geq \phi$ and $\omega \in N(g)$. Now

$$(C(Z))^{\Rightarrow L'} = C(C_{L'}(Z)) = C(C(C_{L'}(Z))) \geq C(Z) \geq \phi, \tag{6.23}$$

because $Z \subseteq C(C_{L'}(\{H : (h \rightarrowtail H) \in X\}))$ and $C \circ C_{L'}$ is also a consequence operator. Hence $(g \rightarrowtail Z) \in C^+(X)$ and finally $\omega \in A$. This proves the second equation of the theorem. □

The theorem implies that the same is true for the unmarked information algebras:

Corollary 6.5 *There is a homomorphism from the unmarked information algebra (H_Φ, S) to (L_Φ, S^+) respecting the focussing.*

These mappings allow to define the diagram in fig. 6.3, and the following theorem shows that this diagram is indeed commutative.

$$\begin{tikzcd}
\text{Information Algebra} & & \begin{array}{c}\text{Information Algebra}\\ \text{of Marked Allocations}\end{array} \\
\langle L_\Phi, S^+\rangle \arrow[rr, "\zeta"] & & \langle H_\Phi, S\rangle \\
\left\{\begin{array}{c}(\mathcal{FSC}, \chi, \mathcal{L}, \vdash) :\\ \chi : \mathcal{FSC} \to 2^{\mathcal{L}} \text{ partial map.}\end{array}\right\} \arrow[u, "\chi\mapsto\langle C^+(\Xi), L\rangle"'] \arrow[rr, "\chi\mapsto\Gamma"] & & \left\{\begin{array}{c}\mathcal{H} = (\Omega, \Gamma, \Phi, S) :\\ \Gamma : \Omega \to \Phi\end{array}\right\} \arrow[u, "\mathcal{H}\mapsto\langle\rho, L\rangle"'] \\
\text{Argumentation Systems} & & \text{Generalized Hints}
\end{tikzcd}$$

Figure 6.3: The connections between the argumentation system and the algebra of allocations.

Theorem 6.6 *The diagram of fig. 6.3 is commutative, i.e. for an argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$ and a representation Ξ of χ with $\Xi \subseteq L_\Xi \in S^+$, we have that*

$$\zeta(\langle C^+(\Xi), L_\Xi\rangle) \qquad \textit{and} \qquad \langle \phi \mapsto \{\omega \in \Omega : \Gamma(\omega) \geq \phi\}, L'_\Xi\rangle \tag{6.24}$$

are members of the same equivalence class of (H_Φ, S).

Proof of theorem 6.6 Given the argumentation system as specified in the theorem, the path from the argumentation system to a marked allocation via the information algebra $\langle L_\Phi, S^+\rangle$ constructed directly from the argumentation system results in an element

$$\zeta(\langle C^+(\Xi), L_\Xi\rangle) = \langle \phi \mapsto \underbrace{\bigcup_{\substack{(f \rightarrowtail Y) \in C^+(\Xi)\\ C(Y) \geq \phi}} N(f)}_{A:=}, L_\Xi\rangle.$$

The other path, via generalized hints, results in

$$\langle \phi \mapsto \{\omega \in \Omega : \Gamma(\omega) \geq \phi\}, L_\Xi\rangle = \langle \phi \mapsto \underbrace{\{\omega \in \Omega : \bigoplus_{\substack{(f \rightarrowtail Y) \in \Xi\\ \omega \in N(f)}} C(Y) \geq \phi\}}_{B:=}, L'_\Xi\rangle.$$

For a fixed ϕ, we first prove that $A = B$. So let $\omega \in A$. Then there exists $(g \rightarrowtail Z) \in C^+(\Xi)$ satisfying $C(Z) \geq \phi$ and $\omega \in N(g)$. By definition of C^+ there exists a set $\Xi' := \{(f_i \rightarrowtail Y_i) : i \in I\} \subseteq \Xi$ such that $\Xi' \vdash^+ (g \rightarrowtail Z)$ and $\omega \in \bigcap_{i \in I} N(g_i)$. This implies

$$\bigoplus_{\substack{(f \rightarrowtail Y) \in \Xi \\ \omega \in N(f)}} C(Y) \;\geq\; \bigoplus_{\{(f_i \rightarrowtail Y_i) : i \in I\}} C(Y_i) \;\geq\; C(Z) \;\geq \phi$$

and therefore $\omega \in B$. Now let $\omega \in B$. This implies that there exists $(g \rightarrowtail Z) \in C^+(\{(f \rightarrowtail Y) \in \Xi : \omega \in N(f)\})$. Therefore $\omega \in N(g)$ and $C(Z) \geq \phi$, and this implies $\omega \in A$. This proves that $A = B$.

Note that the labels L_Ξ and L'_Ξ are not necessarily equal, but both are labels of the allocation of support such that the two marked allocations are members of the same equivalence class of the induced unmarked information algebra (H_Φ, S). □

6.4 Computational Theory for Argumentation Systems

Given an argumentation system with the representation Ξ, we want to compute the conflict *Conf*. Using the definition of the mapping ζ above, we have

$$CS \;=\; \bigcup_{\substack{(f \rightarrowtail Y) \in C^+(\Xi) \\ C(Y) \geq C(\{\perp\})}} N(f) \;=\; \bigcup_{\substack{(f \rightarrowtail Y) \in C^+(\Xi) \\ \perp \in C(Y)}} N(f) \;=\; \bigcup_{(f \rightarrowtail \perp) \in C^+(\Xi)} N(f), \tag{6.25}$$

and for its representation

$$Conf \cong \bigvee_{(f \rightarrowtail \perp) \in C^+(\Xi)} f. \tag{6.26}$$

We will show in chapter 8 how this can be computed without constructing $C^+(\Xi)$ explicitly in most cases.

Further, we can compute the quasi-support $QSS(Z)$ for any set of formulas $Z \subseteq \mathcal{L}$ by using the definition of the mapping ζ above:

$$QSS(Z) \;=\; \bigcup_{\substack{(f \rightarrowtail Y) \in C^+(\Xi) \\ C(Y) \geq C(Z)}} N(f) \;=\; \bigcup_{\substack{(f \rightarrowtail Y) \in C^+(\Xi) \\ C(Z) \subseteq C(Y)}} N(f) \;=\; \bigcup_{\substack{(f \rightarrowtail Y) \in C^+(\Xi) \\ Z \subseteq C(Y)}} N(f). \tag{6.27}$$

and its representation

$$qs(Z) \;\cong\; \bigvee_{\substack{(f \rightarrowtail Y) \in C^+(\Xi) \\ Z \subseteq C(Y)}} f. \tag{6.28}$$

Again, it is usually not necessary to compute $C^+(\Xi)$ explicitly. C^+ is a consequence relation, therefore, as already mentioned in chapter 4, the general computational theory for information systems, i.e. the concept of local computations on hypertrees, can be applied to this problem (Kohlas, 1997a); this theory will be discussed in chapter 8.

7 Model-Based Diagnostics

In this chapter we finally approach the central point of this paper: model-based diagnostics.

Technical systems are in general not guaranteed to be working correctly, but they are more or less reliable, i.e. they are functioning more or less often as expected. One main problem for technical systems is the computation of the reliability of a system. This is studied in reliability theory (see for example (Kohlas, 1987; Beichelt, 1993; Barlow & Proschan, 1975)). The reliability depends on various factors like quality or age of components, complexity of the system, etc. The reliability of a system states some information about the behavior of the system in the future. The problem of reliability is not the main focus here, note however, that this problem can also be formulated using the following concepts (see also (Anrig *et al.*, 1999)). In the sequel, we will focus on the second main problem for technical system, namely the problem of diagnostics, which does not state information about the behavior in the future, but tries to explain the behavior of the system up to the present, usually based on measurements and observations of some parts of the system together with the system description in some framework.

Consider a system to be monitored together with observations of the system's behavior, for example input- and output-values of an electronic circuit. If these observations are in conflict with the behavior the system should have, then something must be wrong in the system. Therefore we have a diagnostic problem, i.e. we have to identify those system components which are functioning abnormally and therefore inducing the discrepancy between the predicted and the observed behavior of the system. The actual observations and the description of the system are the only ingredients for the computation of the diagnoses. Additionally, if probabilistic knowledge is available on the different operating modes of the components, then the weights of the system states can be computed and prior as well as posterior probabilities can be defined on the set of system states.

Many theories for diagnostic problems have been proposed. Especially, the mathematical foundations on diagnosis from first principles have been laid down by Reiter (1987). Reiter addresses the problem of computing the conflicts, and he (among many others) shows that the conflicts are the key to model-based diagnostics from which the other concepts, especially the diagnoses, can then be derived. Note that the notation used in (Reiter, 1987) does not always mean exactly the same thing as in this paper, see (Kohlas *et al.*, 1998) for a discussion of some major differences.

A variety of efficient algorithms for special cases have been developed for the computation of conflicts and, the opposite concept, diagnoses, for example for propositional logic (De Kleer *et al.*, 1992a; Inoue, 1992; Kohlas & Monney, 1993; Siegel, 1987), some of them for the special case of horn clauses.

In general, there are several possible diagnoses for a given diagnostic problem, and often there are even a very large number of diagnoses. Two problems then arise: first, the user usually wants more information about the different diagnoses in order to be able to make a decision and to choose one diagnosis which seems to be the best one; so techniques to differentiate between the diagnoses have to be developed. Such techniques are discussed in chapter 11, based on probabilistic information or on additional knowledge gathered about the system, i.e. for example additional measurements. Second, sometimes there are so many diagnoses that they cannot be computed explicitly and sometimes even their logical representation is too complex to be computed. So approximation techniques are needed which compute only the "best" diagnoses, and this selection is often based on either their size or probabilistic information in the system. But this is not a major problem addressed here, see for example (Kohlas *et al.*, 2000) for further work on what they call "focussing of computations".

In the sequel, we will follow the framework for model-based diagnostics presented in (Kohlas *et al.*, 1998), therefore also the one in (Reiter, 1987) and partially (De Kleer *et al.*, 1992a). For further literature and an introduction see also (Kohlas *et al.*, 1998). We will focus on the computation of conflicts and diagnoses as well as their probabilities, but we also show how to compute arguments for or against any hypothesis on the system (cf. chapter 8).

In the first section, we show how this work is influenced by (Kohlas *et al.*, 1998). In section 7.2, argumentation system are used for model-based diagnostics, and we show that argumentation systems are the right framework for this task, because all concepts needed for model-based diagnostics can clearly be defined therein. Several examples illustrate the use of argumentation systems for model-based diagnostics in section 7.3.

7.1 Precursors of the Concept of Argumentation Systems

Reiter's "theory of diagnosis from first principles" (Reiter, 1987) introduces a purely logical approach to model-based diagnostics. His goal is to find diagnoses, where a diagnosis is a collection of components whose malfunctioning is sufficient to explain the observations on a system. Usually, there are many diagnoses and Reiter proposes to make additional measurements to reduce their number. He shows that the diagnoses (and the conflicts) can be computed using theorem proving techniques. Using the additional measurements, he defines an incremental process for computing the diagnoses, but the number of diagnoses in this process is not monotone.

Our approach of argumentation system is very much inspired by the one considered by Kohlas *et al.* (1998), whose logical part is clearly influenced by Reiter (1987), but also by the approach restricted to propositional logic of Provan (1990) and Laskey & Lehner (1989). Kohlas *et al.* consider a very general language $\mathcal{L}_K$ [1] for modeling information about the system. This language contains the propositional language S_K built from the propositions $AB(c_i)$, $i = 1, \dots, n$, where c_i is a component of the system. The predicate $AB(c_i)$ is true if and only if the component c_i is in an abnormal mode, i.e. if it is not working correctly. The knowledge, i.e. the information about the system to be monitored, is expressed in the language $\mathcal{L}_K$. An instantiated formula $f \in \mathcal{L}_K$ is a formula in which every proposition $AB(c_i)$ has been replaced by its respective evaluation under a specified system state. A system state, as in our approach, defines the working mode of every component c_i, and the set of all system states is defined in (Kohlas *et al.*, 1998) by Ω_K, such that the formulae in S_K are representations of subsets of Ω_K.

Note that Kohlas *et al.* do not specify the language $\mathcal{L}_K$ any further, besides that there must be a test for consistency for sets of instantiated formulas. So they consider a system state s and a set of formulas, instantiate each formula by this system state s, and the test for consistency returns then either true if the instantiated formulas are consistent, or false if they are inconsistent. Using this test for consistency, they define the notion of conflicts and diagnoses, support and doubt, etc. Further they introduce probabilistic information about the system states which allows then to weight formulas in S_K. Note that this approach is different from the one described in (De Kleer & Williams, 1987) which is based on Bayes' rule for the computation of probabilities and therefore needs to make additional, somewhat unjustified assumptions on the probabilistic information (see (Kohlas *et al.*, 1998) for a comparison in details).

Generalizing the idea of components with only two possible working modes, we have constructed a language $\mathcal{FSC}$ of finite set constraints with atoms $M(c_i, V)$ (cf. chapter 5). An atom $M(c_i, V)$ is true if and only if the actual operating mode of component c_i is contained in the set V. The set of system states Ω is then defined as the cartesian product of the sets of all possible modes for every component (5.1). The language $\mathcal{FSC}$ is therefore a generalization of S_K. In the case where every component has only two modes, the resulting $\mathcal{FSC}$ language can be reduced to a propositional language where the mode of every component is denoted by a binary variable, and the resulting language is equivalent to S_K. Note that even more general ideas are possible: consider two components c_i and

[1]The subscript K is used to denote the concepts defined in (Kohlas *et al.*, 1998) in order to distinguish their notations from ours.

c_j and the corresponding predicate $M(c_i, c_j, V)$ which is true if and only if the pair (v_i, v_j) of actual states of the components c_i and c_j are contained in the set V. This can be generalized to n components. Reasoning with such predicates is possible, but the reasoning process gets rather complex. We do not include such predicates in our framework because in the examples modeled so far there was no need for such predicates, and because every predicate $M(c_i, c_j, V)$ can be replaced by a $\mathcal{FSC}$ formula containing only predicates of the form $M(c_i, V')$ and $M(c_j, V'')$. Nevertheless, note that if in a situation where such predicates appear to be quite handy, the concept of $\mathcal{FSC}$ language can be generalized to incorporate the generalized predicates over several components; the description of the reasoning process and the process itself just become more complex.

The language $\mathcal{L}_K$ as defined in (Kohlas *et al.*, 1998) is very general and does not have a lot of structure. Apart from the test for consistency, we have no information about this language. But there are two different concepts of information which are mixed in $\mathcal{L}_K$: the language S_K contained in $\mathcal{L}_K$ expresses a special kind of information which is different from the rest of the information contained in $\mathcal{L}_K$, namely information about the system states Ω_K. In our approach of argumentation systems, we explicitly separate the two different concepts of information occurring in $\mathcal{L}_K$ by means of the mapping χ in the argumentation system $(\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$. If $\chi(f) = X$, then $f \in \mathcal{FSC}$ describes the part of the information which is uncertain, unsure, but for which additional probabilistic information is usually available, whereas $X \subseteq \mathcal{L}$ describes the other part of information which often depends on a system state. The notation $\chi(f) = X$ is interpreted here as a conditional sentence: if f is true in the actual world, then the information described by the subset X must hold for sure. This gives a structure to the knowledge base which explicitly separates the probabilistic from the non-probabilistic data, an idea already mentioned in chapter 3.

The vaguely formulated concept of a test for consistency of instantiated formulas used in (Kohlas *et al.*, 1998) can therefore be replaced here by the clearly defined concept of a consequence relation C^+ (6.16) in the argumentation system $\mathcal{AS}$ based on the entailment relation $\models$ in the language $\mathcal{FSC}$ for the probabilistic data and the entailment relation $\vdash$ in the language $\mathcal{L}$ for the non-probabilistic data. The decomposition of the reasoning process lets us define the concepts for model-based diagnostics more clearly. Indeed, argumentation systems appear as the appropriate concept for doing model-based diagnostics because the basic structures used for doing model-based diagnostics can be defined within this concept.

7.2 Model-Based Diagnostics Using Argumentation Systems

The goal of this section is to present a theory of diagnosis which can be used to identify faulty components in a system. The identified components can then be replaced or repair strategies can be built using the diagnoses. Further, a probabilistic model allows to weight diagnoses and other formulas.

Consider a system to be monitored which consists of several interacting components. Assume that we can describe this system and the corresponding observation or measurements using the argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$ (chapter 6), i.e. we have a finite set constraint language $\mathcal{FSC}$ which models the modes of the components, an information system $(\mathcal{L}, \vdash)$ which models the information about the behavior of components, and a partial mapping χ between $\mathcal{FSC}$ and $(\mathcal{L}, \vdash)$. Additionally, we may have additional probabilistic information pr such that the argumentation is a probabilistic argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$. In the sequel, we will usually consider probabilistic argumentation systems, note however that those results which are not based on pr are also applicable to non-probabilistic argumentation systems. Examples of argumentation systems and their application to model-based diagnostics are presented in section 7.3.

In chapter 6 we have shown how a generalized hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D, \mathcal{A}, P)$ is constructed from an argumentation system. An allocation of support can be built from the hint using results from chapter 3. In section 3.5.5, the conflicts CS and the diagnoses DS have been defined, and in section 5.4, logical representations *Conf* and *Diag* of these sets have been presented.

In the framework of an argumentation system, these concepts now get a clear meaning. Consider an element $\omega \in CS$. By definition, this element induces the null element $z \in \Phi$ via the focal mapping Γ of the generalized hint $\mathcal{H}$ constructed on $\mathcal{AS}$, i.e. ω is an element which describes a system state which is not possible. That means that ω describes the working mode of every component in the system, and, considering this information and the knowledge modeled in χ, i.e. the knowledge about the behavior of the system together with the observations, this implies then the null element z of the information algebra (Φ, D). But this null element z is a representation of the null element $\perp$ of the language $\mathcal{L}$, an element which "cannot be true" and which implies therefore anything else in $\mathcal{L}$ (see chapter 4). This means that the system state ω, which implies the null element, is a system state which an impossible one for the present situation. Therefore by definition, the set CS contains all interpretations or system states which are not possible given the actual system description and, especially, the observations. In contrast, an element $\omega \in DS$ is a diagnosis of the system, i.e. an interpretation which explains the current behavior of the system and is consistent with all the knowledge available and especially with the observation of the system. Hence, such a system state ω specifies the working modes of all components in the system, and, together with the knowledge about the behavior of the system and the observations, it implies by the mapping Γ an element $\phi \neq \perp$ in the information algebra (Φ, D), which is consistent, i.e. which describes the possible behavior of the system given the modes of the components.

So the concept of an argumentation system allows to compute the sets of conflicts and diagnoses. Additionally, a probabilistic argumentation system also allows to weight the system states, for example the prior probability of every system state can be computed using the probability pr and (6.8).

If there is only one element in the set DS, then clearly this is the actual system state, and we know with certainty the precise working mode of each component, hence we have enough information to replace the ones which are specified as defective by the diagnosis, and we are sure that after the replacement of these components, the system will be working correctly. Yet normally, the set DS contains several elements, usually even a great number of elements. Often there exists quite a short logical representation of these elements by formulas in the language $\mathcal{FSC}$, for example the disjunction of the minimal diagnoses in $\mathcal{FSC}$ as defined in section 5.4. In the case where only one minimal diagnosis d exists, it describes the mode of some components in the system, and whatever the modes of the other components are, the resulting interpretation is a diagnosis for the system, i.e.

$$\Gamma(\omega) \neq z \qquad \text{for every } \omega \in N(d). \tag{7.1}$$

However, d does not describe the modes of all components. So even after replacing all components denoted as faulty by d, we are not sure that the system is working correctly, but the behavior of other faulty components may have been "hidden" by the replaced ones. In this case, additional measurements have to be made and a second diagnostic process has to be started.

Usually, there are several minimal diagnoses for a given situation, and the problem is to select one of them. One method consists in weighting these minimal diagnoses according to the posterior probability p' (this will be discussed in chapter 9) and selecting the most probable one. Another method is to gather additional information by a measurement of a part of the system (for example unobserved in- or output-values or some internal wires in an electronic circuit), and this new information will possibly reduce the number of diagnoses; this will be discussed in chapter 11. The selected diagnosis can then be used to fix a repair or replacement strategy for the system.

Yet even further, any formula f in $\mathcal{L}$ can be considered as a hypothesis about the state of the system to be monitored, and the arguments in favor, $qs(f)$ and $sp(f)$, or against, $rs(f)$ (cf. section 3.5), can be computed and weighted (see section 9.1 for the computation of the weights). A degree of support represents the strength with which the knowledge and the observations tend to "prove" the information represented by the formula. Arguments can be used as additional information for a repair strategy, e.g. for a component c, the formula $sp(M(c, \{ok\}))$ represents the arguments which support the hypothesis that c is in mode ok, i.e. that c is working correctly.

The probabilistic layer on top of the system states contains additional information, which is very useful for example for making decisions and computing strategies for further measurement (cf. chapter 11), for comparing diagnoses or arguments, etc. This layer was not considered in (Reiter, 1987); only De Kleer & Williams (1987) introduced probabilistic information, but their approach depends on additional assumptions on the probabilistic information; see (Kohlas *et al.*, 1998) for a discussion. Here, we do not need any additional assumption, but the probabilistic layer is well-defined using the theory developed in chapter 2, i.e. the degrees of quasi-support (support, ...) functions define belief functions in the sense of section 3.7. The computation of probabilistic information will be focused in chapter 9.

In the next section, we will explain these concepts using several examples.

7.3 Examples

The examples in this section are intended to clarify the ideas of model-based diagnostics using argumentation system presented in the section above. We use several small, well-known examples to show the strength of the concept. The examples are first presented as ABEL code (see chapter 12 for more information about the modeling language ABEL), because most of them are taken from (Anrig *et al.*, 1999).

Example 7.1: Three Serial Inverters

Consider the simple digital circuit introduced in example 1.1, which is built out of three serial inverters and connected as in fig. 7.1.

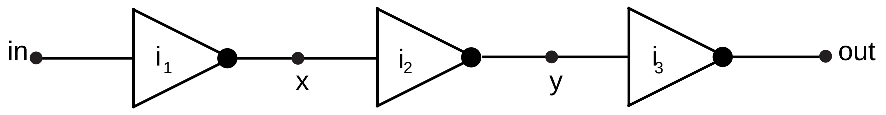


Figure 7.1: Three serial inverters.

Every component has two working modes. If an inverter i is in its correct working mode, then it inverts the incoming signal, i.e. its output signal is the negation of its input signal; this mode will be denoted by ok and the signals will be modeled by binary variables. If an inverter is in its faulty mode ($faulty = \neg ok$) then nothing is said about its behavior, so every combination of in- and output signals is possible, among which also the correct one. Assume that the initial probability of a failure of an inverter is 0.01. The set of components is $C = \{i_1, i_2, i_3\}$, and the variables for the connectors are in, x, y, and out. In ABEL (cf. chapter 12), this can be written as follows[2]:

```
(tell
   (type mode-type (ok faulty))
   (ass i1 i2 i3 mode-type (0.99 0.01))
   (var in x y out binary)

   (-> (= i1 ok) (<-> in (not x)))
```

[2]See section 11.5 for a better description of the example in ABEL using the modular structure of the situation.

```
        (-> (= i2 ok) (<-> x (not y)))
        (-> (= i3 ok) (<-> y (not out))))
```

In addition, the input- and output values are measured as 1, therefore

```
    (observe in out)
```

Apparently, these observations imply that the system is not functioning correctly, because the predicted output (0) of the system is in conflict with the observed one (1) if the input is 1.

The corresponding argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$ is then defined as follows: the language $\mathcal{FSC}$ consists of the finite set constraints over the three variables i_1, i_2 and i_3 with frames $V_1 = V_2 = V_3 = \{ok, faulty\}$. The language $\mathcal{L}$ is a propositional language over the propositional variables in, x, y, and out with the usual propositional connectives. Furthermore, $\vdash$ denotes the usual propositional entailment relation in $\mathcal{L}$. Then the representation Ξ of χ is

$$\Xi \quad = \quad \left\{ \begin{array}{ccc} M(i_1, \{ok\}) & \rightarrowtail & (x \leftrightarrow \neg in) \\ M(i_2, \{ok\}) & \rightarrowtail & (y \leftrightarrow \neg x) \\ M(i_3, \{ok\}) & \rightarrowtail & (out \leftrightarrow \neg y) \\ \top & \rightarrowtail & in \wedge out \end{array} \right\}$$

where the first three lines represent the information of the components i_1 to i_3, the fourth line represents the observations. The probability pr is given by

$$pr(i_j = ok) = 0.99 \quad \text{and} \quad pr(i_j = faulty) = 0.01$$

for $j = 1, 2, 3$.

The argumentation system allows then to construct the generalized hint and an allocation of support QSS. Define $\Omega := \{0,1\}^3$ where the j-th component of a vector $(\omega_1, \omega_2, \omega_3) \in \Omega$ means that component i_j is working ($\omega_j = 1$) or not ($\omega_j = 0$). From Ξ, the focal mapping Γ of the hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D, P)$ can be constructed, where the information algebra (Φ, D) is the algebra of subsets of the cartesian product $\{in, \overline{in}\} \times \{x, \overline{x}\} \times \{y, \overline{y}\} \times \{out, \overline{out}\}$ (the notation $\overline{x}$ means that the variable x is false):

(i_1, i_2, i_3)	$\overset{\Gamma}{\to}$	$\Gamma((i_1, i_2, i_3))$
$(0,0,0)$	$\longrightarrow$	$\left\{ \begin{array}{l} (in, out, x, y),\ (in, out, x, \overline{y}), \\ (in, out, \overline{x}, y),\ (in, out, \overline{x}, \overline{y}) \end{array} \right\}$
$(0,0,1)$	$\longrightarrow$	$\{(in, out, x, \overline{y}),\ (in, out, \overline{x}, \overline{y})\}$
$(0,1,0)$	$\longrightarrow$	$\{(in, out, x, \overline{y}),\ (in, out, \overline{x}, y)\}$
$(0,1,1)$	$\longrightarrow$	$\{(in, out, x, \overline{y})\}$
$(1,0,0)$	$\longrightarrow$	$\{(in, out, \overline{x}, y),\ (in, out, \overline{x}, \overline{y})\}$
$(1,0,1)$	$\longrightarrow$	$\{(in, out, \overline{x}, \overline{y})\}$
$(1,1,0)$	$\longrightarrow$	$\{(in, out, \overline{x}, y)\}$
$(1,1,1)$	$\longrightarrow$	$\emptyset$

The first line for example means that if all three inverters are not working correctly, i.e. $(i_1, i_2, i_3) = (0, 0, 0)$, then under the present knowledge one of the elements of the set on the right hand side is the actual state of the variables in, out, x, and y, that is one of (in, out, x, y), $(in, out, x, \overline{y})$, $(in, out, \overline{x}, y)$, $(in, out, \overline{x}, \overline{y})$ is the "right" one. The last line indicates the situation where all three inverters are working correctly, i.e. $(i_1, i_2, i_3) = (1, 1, 1)$, but the empty set on the right hand side means that under the actual knowledge there is no actual state of the variables in, out, x, and y which is consistent with that. Therefore $(1, 1, 1)$ is a system state which is inconsistent with the present knowledge together with the observations.

The conflict set and the quasi-support sets are then constructed from Γ using the definition from section 6.2, so for example

$$\begin{array}{rcl} CS & = & \{(1,1,1)\} \\ QSS(\neg x) & = & \{(1,0,0), (1,0,1), (1,1,0), (1,1,1)\} \\ QSS(in \wedge y) & = & \{(1,1,0), (1,1,1)\} \\ & \vdots & \end{array}$$

Further, the conflict and the quasi-support function qs can be computed, for example

$$\begin{array}{rcl} Conf & \cong & M(i_1, \{ok\}) \wedge M(i_2, \{ok\}) \wedge M(i_3, \{ok\}) \\ qs(\neg x) & \cong & M(i_1, \{ok\}) \\ qs(in \wedge y) & \cong & M(i_1, \{ok\}) \wedge M(i_2, \{ok\}) \\ & \vdots & \end{array}$$

using (6.10). Support, quasi-supports and refutation of other hypotheses are computed analogously.

The set of diagnoses DS is then $\Omega - CS$, which contains seven elements. So there are seven diagnoses which explain the behavior of the system. Using (5.30), the logical representation $Diag$ of the set of diagnoses DS can be computed, i.e.

$$\begin{array}{rcl} Diag & \cong & \neg Conf \\ & \cong & M(i_1, \{faulty\}) \vee M(i_2, \{faulty\}) \vee M(i_2, \{faulty\}), \end{array}$$

and there are three minimal diagnoses for this problem, namely $M(i_1, \{faulty\})$, $M(i_2, \{faulty\})$, and $M(i_3, \{faulty\})$, which, by the way, happen to have an equal posterior probability of 0.337 (see chapter 9 for details on how to compute this probability). Note however that in practice, the logical representation of the conflicts $Conf$ and of the diagnoses $Diag$ are computed first, and only afterwards, if at all necessary, the representations as sets CS and DS respectively (cf. chapter 8).

Assume now that a further measurement is taken at point x (cf. fig. 7.1) and that its result is positive, i.e. we denote it by x. This implies that either the argumentation system is "updated" or a second one is created and the respective hints or allocations of support are combined. We denote these new mappings by a prime $'$ in order to distinguish them from the ones defined above. The new information is represented by a second argumentation system $\mathcal{AS}' = (\mathcal{FSC}, \chi', \mathcal{L}, \vdash, pr)$ where χ' is represented by the set Ξ',

$$\Xi' \;=\; \{\top \rightarrowtail x\},$$

(the other parts of the argumentation system are identical to the ones from $\mathcal{AS}$) and this implies the hint $\mathcal{H}'$ with focal mapping Γ' whose image is constant, i.e.

(i_1, i_2, i_3)	$\overset{\Gamma'}{\to}$	$\Gamma'((i_1, i_2, i_3))$
$(0,0,0)$	$\longrightarrow$	$\{in, \overline{in}\} \times \{out, \overline{out}\} \times \{x\} \times \{y, \overline{y}\}$
$(0,0,1)$	$\longrightarrow$	$\{in, \overline{in}\} \times \{out, \overline{out}\} \times \{x\} \times \{y, \overline{y}\}$
$\vdots$		
$(1,1,1)$	$\longrightarrow$	$\{in, \overline{in}\} \times \{out, \overline{out}\} \times \{x\} \times \{y, \overline{y}\}.$

The induced support function is then

$$QSS'(f) \;=\; \{(0,0,0), \ldots, (1,1,1)\}$$

for any formula f with $N(f) \subseteq N(x)$, and

$$QSS'(f) \;=\; \{\}$$

for any formula f with $N(f) \not\subseteq N(x)$. We now combine the two hints $\mathcal{H}$ and $\mathcal{H}'$ by combining their focal mappings Γ and Γ', which results in a new hint with focal mapping $\Gamma \oplus \Gamma'$

(i_1, i_2, i_3)	$\overset{\Gamma\oplus\Gamma'}{\to}$	$(\Gamma \oplus \Gamma')((i_1, i_2, i_3))$
$(0,0,0)$	$\longrightarrow$	$\{(in, out, x, y),\ (in, out, x, \overline{y})\}$
$(0,0,1)$	$\longrightarrow$	$\{(in, out, x, \overline{y})\}$
$(0,1,0)$	$\longrightarrow$	$\{(in, out, x, \overline{y})\}$
$(0,1,1)$	$\longrightarrow$	$\{(in, out, x, \overline{y})\}$
$(1,0,0)$	$\longrightarrow$	$\emptyset$
$(1,0,1)$	$\longrightarrow$	$\emptyset$
$(1,1,0)$	$\longrightarrow$	$\emptyset$
$(1,1,1)$	$\longrightarrow$	$\emptyset$

The conflict set $CS_{\Gamma\oplus\Gamma'}$ and the quasi-support $QSS_{\Gamma\oplus\Gamma'}$ are then constructed from $\Gamma \oplus \Gamma'$, so

$$\begin{aligned}
CS_{\Gamma\oplus\Gamma'} &= \{(1,0,0),\ (1,0,1),\ (1,1,0),\ (1,1,1)\}\\
QSS_{\Gamma\oplus\Gamma'}(\neg x) &= \{(1,0,0),\ (1,0,1),\ (1,1,0),\ (1,1,1)\}\\
QSS_{\Gamma\oplus\Gamma'}(in \wedge y) &= \{(1,0,0),\ (1,0,1),\ (1,1,0),\ (1,1,1)\}\\
&\ \vdots
\end{aligned}$$

As above, the logical representation can be computed, i.e.

$$\begin{aligned} Conf_{\Gamma\oplus\Gamma'} &\cong M(i_1,\{ok\}) \\ qs_{\Gamma\oplus\Gamma'}(\neg x) &\cong M(i_1,\{ok\}) \\ qs_{\Gamma\oplus\Gamma'}(in \wedge y) &\cong M(i_1,\{ok\}) \\ &\vdots \end{aligned}$$

and, using again (5.30),

$$Diag_{\Gamma\oplus\Gamma'} \cong \neg Conf_{\Gamma\oplus\Gamma'} \cong M(i_1,\{faulty\}).$$

We see that now there is only one minimal diagnosis left, i.e. $M(i_1,\{faulty\})$, so we have found a minimal diagnosis explaining the behavior of the system, i.e. we can say with certainty that the component i_1 is defect. Note that we do not know if i_2 and i_3 are really working correctly because we have not (yet) measured the value of y. $\ominus$

Example 7.2: Arithmetical Network

This well-known example was introduced by (Davis, 1984). Subsequently, it was used in several papers on model-based diagnostics (Reiter, 1987; De Kleer & Williams, 1987; Kohlas *et al.*, 1998). The network consists of three multipliers m_1, m_2, m_3 and two adders a_1, a_2. These components are connected as shown in fig. 7.2. The values of the in- and outputs of the system are observed according to fig. 7.2.

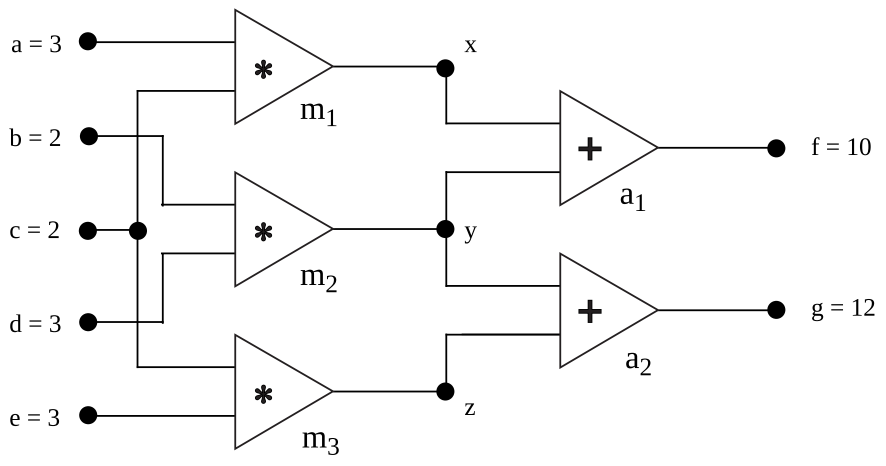


Figure 7.2: An arithmetical network with measured values.

The behavior of a component is expressed by a material implication, e.g. an adder is specified by

$$(adder = ok) \quad\rightarrow\quad (out = in_1 + in_2),$$

i.e. if the adder is working correctly then its output is the sum of its inputs, and nothing is known about the behavior of a faulty component. Assume that the probability of a component working correctly is 0.95 for adders and 0.97 for multipliers. Then the situation can be modeled in ABEL as follows:

```
(tell
   (type mode-type (ok faulty))
   (ass m1 m2 m3 mode-type (0.97 0.03))
   (ass a1 a2 mode-type (0.95 0.05))
   (var a b c d e f g x y z integer)

   (-> (= m1 ok) (= (* a c) x))
   (-> (= m2 ok) (= (* b d) y))
   (-> (= m3 ok) (= (* c e) z))

   (-> (= a1 ok) (= (+ x y) f))
   (-> (= a2 ok) (= (+ y z) g)))

(observe
    (= a 3) (= b 2) (= c 2) (= d 3) (= e 3) (= f 10) (= g 12))
```

Apparently, these observations imply that the system is not functioning correctly, because the predicted output (12) at point f is in conflict with the observed one (10).

We construct the corresponding argumentation system $(\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$: the language $\mathcal{FSC}$ consists of the finite set constraints over the variables m_1, m_2, m_3 and a_1, a_2 with equal frames $V = \{ok, faulty\}$. We call atoms the variables a to g and x, y, z, as well as the constant values 1, 2, ... An expression is either an atom or a formula ($expression * expression$) or ($expression + expression$) where parentheses are omitted using the normal precedence of operators. A constraint is a formula $expression = expression$ or the symbol $\bot$. A formula is then either a finite constraint or one of ($constraint \wedge constraint$) or ($constraint \vee constraint$). The language $\mathcal{L}$ consists of all (finite) formulas built using this scheme.

Denote by $\mathbb{Z}$ the integers. An element of

$$\mathbb{Z}^9 = d(a) \times \cdots \times d(g) \times d(x) \times d(y) \times d(z)$$

(where $d(a) = \mathbb{Z}$ is the domain of the variable a, ...), also called an assignment, can now be used to instantiate atoms, i.e. every element σ in $\mathbb{Z}^9$ defines the value of every atom of the language $\mathcal{L}$. For example the element $(2, 0, 0, 0, 0, 0, 0, 0, 0) \in \mathbb{Z}^9$ states that $a = 2$ and all other variables have the value 0, i.e. $0 = b = \cdots = g = x = y = z$.

The instantiation $\sigma(f)$ of a formula $f \in \mathcal{L}$ by an assignment σ is the formula f where all occurrences of variables have been replaced by the respective constants in σ. A formula $f \in \mathcal{L}$ evaluates to true under σ if the instantiation $\sigma(f)$ is valid, where the validity of an instantiation g is defined by the following rules:

a) If $g = \bot$ then g is not valid.

b) If $g = g_1 \vee g_2$ then g is valid if a least one of g_1 and g_2 is valid.

c) If $g = g_1 \wedge g_2$ then g is valid if both g_1 and g_2 are valid.

d) Otherwise g is a constraint and is valid if and only if g is a correct equation in the sense of usual integer arithmetic.

The concept of instantiation can be used to define an entailment relation on $\mathcal{L}$: For $X \subseteq \mathcal{L}$ and $f \in \mathcal{L}$ we define $X \vdash f$ if and only if for every assignment under which every formula in X evaluates to true, the formula f evaluates to true as well.

The representation Ξ of χ is then:

$$\Xi = \left\{ \begin{array}{ccl} M(m_1, \{ok\}) & \rightarrowtail & (x = (a * c)) \\ M(m_2, \{ok\}) & \rightarrowtail & (y = (b * d)) \\ M(m_3, \{ok\}) & \rightarrowtail & (z = (c * e)) \\ M(a_1, \{ok\}) & \rightarrowtail & (f = (x + y)) \\ M(a_2, \{ok\}) & \rightarrowtail & (g = (y + z)) \\ \top & \rightarrowtail & (a = 3) \wedge (b = 2) \wedge (c = 2) \wedge (d = 3) \\ & & \wedge (e = 3) \wedge (f = 10) \wedge (g = 12) \end{array} \right\}.$$

The probability pr is given by

$$pr(m_i = ok) = 0.97 \quad \text{and} \quad pr(m_i = faulty) = 0.03$$
$$pr(a_j = ok) = 0.95 \quad \text{and} \quad pr(a_j = faulty) = 0.05$$

for $i = 1, 2, 3,\ j = 1, 2$. This completes the specification of the argumentation system $\mathcal{AS}$.

Define $\Omega := \{0, 1\}^5$ where the j-th component of a vector $(\omega_1, \ldots, \omega_5) \in \Omega$ means that component i_j is working ($\omega_j = 1$) or not ($\omega_j = 0$), where $i_1 = m_1$, $i_2 = m_2$, $i_3 = m_3$, $i_4 = a_1$, $i_5 = a_2$. From Ξ, the mapping Γ of the hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D, P)$ can be constructed. The elements of the information algebra (Φ, D) are subsets of $\mathbb{Z}^9$:

$(m_1, m_2, m_3, a_1, a_2)$	$\xrightarrow{\Gamma}$	$\Gamma((m_1, m_2, m_3, a_1, a_2))$
$(0,0,0,0,0)$	$\longrightarrow$	$\{(3,2,2,3,3,10,12,i,j,k) : i,j,k \in \mathbb{Z}\}$
$(1,0,0,0,0)$	$\longrightarrow$	$\{(3,2,2,3,3,10,12,6,j,k) : j,k \in \mathbb{Z}\}$
	$\vdots$	
$(1,0,0,0,1)$	$\longrightarrow$	$\{(3,2,2,3,3,10,12,6,j,12-j) : j \in \mathbb{Z}\}$
	$\vdots$	
$(1,0,1,0,1)$	$\longrightarrow$	$\{(3,2,2,3,3,10,12,6,6,6)\}$
	$\vdots$	
$(1,1,1,1,1)$	$\longrightarrow$	$\emptyset$

Quasi-support sets and other concepts can then be defined on these hints as in the previous example. The logical representation of the conflict set is

$$\begin{aligned} Conf \cong & \Big(M(a_1, \{ok\}) \wedge M(m_1, \{ok\}) \wedge M(m_2, \{ok\}) \Big) \\ & \vee \Big(M(a_1, \{ok\}) \wedge M(a_2, \{ok\}) \wedge M(m_1, \{ok\}) \wedge M(m_3, \{ok\}) \Big) \end{aligned}$$

and, using (5.30), we get

$$\begin{aligned} \mathit{Diag} &\cong \neg \mathit{Conf} \\ &\cong M(a_1, \{\mathit{faulty}\}) \vee M(m_1, \{\mathit{faulty}\}) \\ &\quad \vee \Big(M(a_2, \{\mathit{faulty}\}) \wedge M(m_2, \{\mathit{faulty}\}) \Big) \\ &\quad \vee \Big(M(m_2, \{\mathit{faulty}\}) \wedge M(m_3, \{\mathit{faulty}\}) \Big). \end{aligned}$$

So there are four minimal diagnoses, namely

$$\begin{array}{l} M(a_1, \{\mathit{faulty}\}) \\ M(m_1, \{\mathit{faulty}\}) \\ M(a_2, \{\mathit{faulty}\}) \wedge M(m_2, \{\mathit{faulty}\}) \\ M(m_2, \{\mathit{faulty}\}) \wedge M(m_3, \{\mathit{faulty}\}). \end{array}$$

In order to discriminate between these diagnoses, either they are weighted by the posterior probability p' as will be explained in chapter 9, or additional knowledge has to be obtained by, for example, measurements of internal variables (see chapter 11). If no probabilistic information is available, often the size of the minimal diagnoses is considered and the shortest diagnosis is selected as information for the repair or replacement strategy. Here, this means that the adder a_1 as well as the multiplier m_1 are good candidates for replacement or repairing. ⊖

Example 7.3: Multiple Operating Modes

In a technical system, consider a component c having two input ports and one output port. Let in_1 and in_2 denote the value of the input ports and let out denote the value of the output port. Figure 7.3 shows this component graphically (cf. (Anrig & Monney, 1999)).

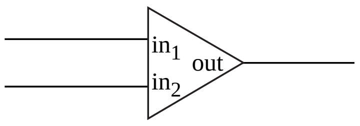


Figure 7.3: The component c

The value of the output port is a linear function of the values of the two input ports, i.e.

$$out = \alpha \cdot in_1 + \beta \cdot in_2.$$

The component can be in one of four possible operating modes, each characterized by some specific values of α and β:

- mode 1: $\alpha = 1, \beta = 1$
- mode 2: $\alpha = 1, \beta = -1$

- mode 3: $\alpha = -1, \beta = 1$
- mode 4: $\alpha = -1, \beta = -1$.

A priori it is assumed that the probability of the component being in mode 1 is 0.4, and the probability of its being in mode 2, resp. 3, resp. 4 is 0.3, 0.2, 0.1 respectively. Now consider three of these components c_1, c_2, and c_3 which interact according to the diagram of figure 7.4. Every component c_i has its own values α_i and β_i. The variables $y_1, \ldots, y_4$ in figure 7.4 represent the value of some ports in the system, the variables x_1 and x_2 internal variables.

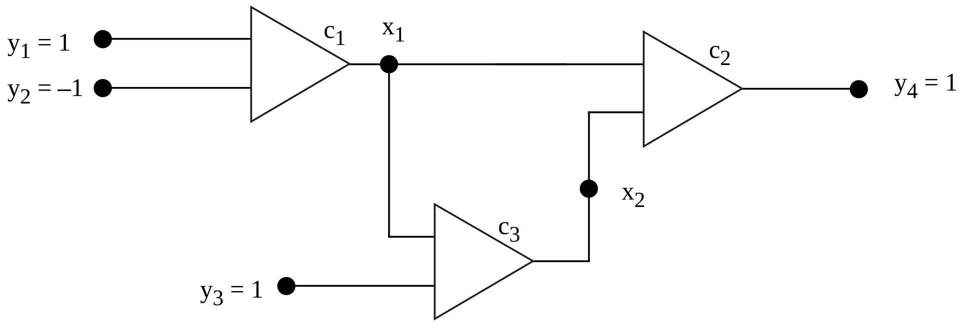


Figure 7.4: A system with components c_1, c_2 and c_3

Now suppose that we observe the values of the ports $y_1, \ldots, y_4$ being $1, -1, 1, 1$ respectively. Of course, this implies that some system states are no longer possible. The ABEL-code is then

```
(tell
  (type mode-type (1 2 3 4))

  (module COMPONENT ((var in1 in2 out integer))
    (ass mode m-type (0.4 0.3 0.2 0.1))
    (-> (= mode 1) (= out (+ in1 in2)))
    (-> (= mode 2) (= out (- in1 in2)))
    (-> (= mode 3) (= out (- in2 in1)))
    (-> (= mode 4) (= out (- 0 in1 in2))))

  (COMPONENT :c1 y1 y2 x1)
  (COMPONENT :c2 x1 y3 x2)
  (COMPONENT :c3 x1 x2 y4))

(observe (= y1 1) (= y2 -1) (= y3 1) (= y4 1))
```

Note that we used the concept of module in ABEL, so as to shorten the description of the model; see chapter 12 for this concept.

Let c_i denote the variable indicating the operating mode of component number i. The domain of every c_i is $V := \{1, 2, 3, 4\}$. The possible interpretations, i.e. the set Ω, is then defined by

$$\Omega \quad := \quad V(c_1) \times V(c_2) \times V(c_3) \quad = \quad \{1, 2, 3, 4\}^3.$$

We assume that the random variables c_i are stochastically independent, which implies that the prior probability of a vector $v = (k_1, k_2, k_3)$ in Ω, a so-called system state, is given by

$$P(v) = P(c_1 = k_1) \cdot P(c_2 = k_2) \cdot P(c_3 = k_3). \tag{7.2}$$

Equation (7.2) completely specifies a probability distribution on Ω. The language $\mathcal{L}$ and the entailment relation $\vdash$ are defined as in the previous example 7.2 but with subtraction "$-$" instead of multiplication "$*$". The mapping χ deduced from the ABEL-code above is represented by Ξ:

$$\Xi \quad = \quad \left\{ \begin{array}{lcl} M(c_1, 1) & \rightarrowtail & x_1 = y_1 + y_2 \\ M(c_1, 2) & \rightarrowtail & x_1 = y_1 - y_2 \\ M(c_1, 3) & \rightarrowtail & x_1 = -y_1 + y_2 \\ M(c_1, 4) & \rightarrowtail & x_1 = -y_1 - y_2 \\ M(c_2, 1) & \rightarrowtail & y_4 = x_1 + x_2 \\ M(c_2, 2) & \rightarrowtail & y_4 = x_1 - x_2 \\ M(c_2, 3) & \rightarrowtail & y_4 = -x_1 + x_2 \\ M(c_2, 4) & \rightarrowtail & y_4 = -x_1 - x_2 \\ M(c_3, 1) & \rightarrowtail & x_2 = x_1 + y_3 \\ M(c_3, 2) & \rightarrowtail & x_2 = x_1 - y_3 \\ M(c_3, 3) & \rightarrowtail & x_2 = -x_1 + y_3 \\ M(c_3, 4) & \rightarrowtail & x_2 = -x_1 - y_3 \\ \top & \rightarrowtail & (y_1 = 1) \wedge (y_2 = -1) \wedge (y_3 = 1) \wedge (y_4 = 1) \end{array} \right\}$$

This specifies the argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$. The hint $\mathcal{H}$ constructed from $\mathcal{AS}$ contains then the focal mapping Γ:

(c_1, c_2, c_3)	$\xrightarrow{\Gamma}$	$\{(y_1, y_2, y_3, y_4, x_1, x_2)\}$
$(1, 1, 1)$	$\longrightarrow$	$\{(1, -1, 1, 1, 0, 1)\}$
$(1, 1, 2)$	$\longrightarrow$	$\emptyset$
$(1, 1, 3)$	$\longrightarrow$	$\{(1, -1, 1, 1, 0, 1)\}$
$(1, 1, 4)$	$\longrightarrow$	$\emptyset$
$(1, 2, 1)$	$\longrightarrow$	$\emptyset$
$(1, 2, 2)$	$\longrightarrow$	$\{(1, -1, 1, 1, 0, -1)\}$
	$\vdots$	

Suppose that we are interested in the set of all diagnoses DS. This can be computed using the definitions from section 6.2, and we get

$$DS \quad = \quad \left\{ \begin{array}{llllll} (1,1,1), & (1,1,3), & (1,2,2), & (1,2,4), & (1,3,1), & (1,3,3), \\ (1,4,2), & (1,4,4), & (2,1,3), & (2,2,2), & (2,3,1), & (2,4,4), \\ (3,1,3), & (3,2,2), & (3,3,1), & (3,4,4), & (4,1,1), & (4,1,3), \\ (4,2,2), & (4,2,4), & (4,3,1), & (4,3,3), & (4,4,2), & (4,4,4) \end{array} \right\}.$$

Here, a diagnosis does not tell us which components are faulty because there is no faulty mode specified for the components; we only know that every component has exactly four possible working modes. But a diagnosis explicitly states the actual working mode of every component. If there are modes which are not allowed in specific situations, then the diagnosis tells us which components are in such a mode. If those modes are specified only afterwards, the diagnosis can always be reinterpreted to match this criterion.

The following formula represents the diagnoses in a logical form

$$\begin{aligned} \mathit{Diag} \;\cong\; & (M(c_2,\{2\}) \wedge M(c_3,\{2\})) \vee (M(c_2,\{1\}) \wedge M(c_3,\{3\})) \\ & \vee (M(c_2,\{3\}) \wedge M(c_3,\{1\})) \vee (M(c_2,\{4\}) \wedge M(c_3,\{4\})) \\ & \vee (M(c_1,\{1,4\}) \wedge M(c_2,\{2,4\}) \wedge M(c_3,\{2,4\})) \\ & \vee (M(c_1,\{1,4\}) \wedge M(c_2,\{1,3\}) \wedge M(c_3,\{1,3\})) \end{aligned}$$

and we see that there are six minimal diagnoses in this example. ⊖

Example 7.4: Binary Adder

Digital circuits are a nice example for doing model-based diagnostics, first because they are quite easy to model (cf. also example 7.1), and second because larger composite devices can be built up quite easily from small ones. This will be presented here using binary adders of different sizes.

Consider the digital circuit for a binary adder as shown in fig. 7.5. The system consists of five components: the logical gates and_1, and_2, xor_1, xor_2, and or. The input and output values of the components are the logical values *true* and *false* (interpreted as 1 and 0).

If every component works correctly, then the actual input and output values (cf. fig. 7.5) of the binary adder are in conflict with the expected behavior (according to the definition of a binary adder). Therefore, one or several components must be faulty, and the question is which ones.

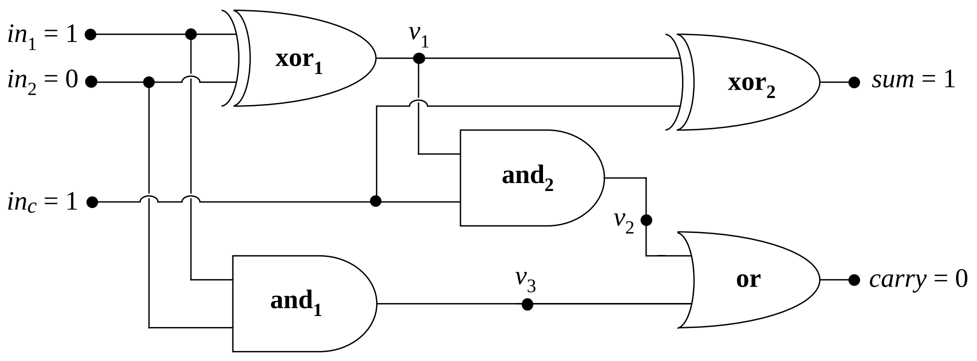


Figure 7.5: A binary adder built out of logical gates.

Suppose that each component has exactly two different operating modes, which are characterized by a binary predicate *ok*. A correctly functioning component means that *ok* is true, whereas for a faulty component *ok* is false (respectively $\neg ok$ is true). Figure 7.6 shows an *and*-gate with two inputs in_1 and in_2 and one output *out*; the *xor*- and *or*-gates are constructed similarly.

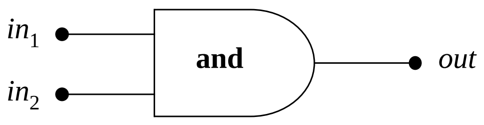


Figure 7.6: An *and*-gate.

Suppose that only the behavior of a correctly functioning component is known, whereas the error mode is not further specified. An *and*-gate can then be described as follows:

$$ok \rightarrow (out \leftrightarrow in_1 \wedge in_2).$$

The other components, i.e. the *or*- and the *xor* gate, are treated analogously. Further, we have some probabilistic information about the functioning of the components: the probability for a correct functioning of a *and*-gate is 0.99, for a *or*-gate it is 0.98, and for a *xor*-gate it is 0.95. For each type of component, a corresponding module is defined in ABEL (cf. chapter 12 for details about modules in ABEL):

```
(tell
  (module AND-GATE ((var in1 in2 out binary))
    (ass ok binary 0.99)
    (-> ok (<-> out (and in1 in2))))

  (module OR-GATE ((var in1 in2 out binary))
    (ass ok binary 0.98)
    (-> ok (<-> out (or in1 in2))))

  (module XOR-GATE ((var in1 in2 out binary))
    (ass ok binary 0.95)
    (-> ok (<-> out (xor in1 in2)))))
```

According to the circuit topology of fig. 7.5, a module for an entire adder is defined as follows:

```
(tell
  (module ADDER ((var in1 in2 inc sum carry binary))
    (var ok binary)
    (var v1 v2 v3 binary)

    (AND-GATE :AND1 in1 in2 v3)
    (AND-GATE :AND2 inc v1 v2)
    (XOR-GATE :XOR1 in1 in2 v1)
```

```
         (XOR-GATE :XOR2 inc v1 sum)
         (OR-GATE :OR v2 v3 carry)

         (<-> ok (and AND1.ok AND2.ok XOR1.ok XOR2.ok OR.ok))))
```

This module is instantiated with the actual variables, i.e. the input and the output variables of the adder according to fig. 7.5. Further, the instance of the module is give the name `:ADDER`:

```
     (tell (ADDER :ADDER in1 in2 inc sum carry))
```

Note that in ABEL it is not necessary to explicitly declare all the variables used in the model. The types of the variables are determined according to their first occurrence in a module instantiation (cf. chapter 12). For example, in_1 (in the instantiation of the adder `:ADDER`) is a binary variable as defined within the module `ADDER`.

Finally, the ABEL model is completed by the input and output values of the system as shown in fig. 7.5:

```
     (observe in1 (not in2) inc sum (not carry))
```

The corresponding argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$ is then defined as follows: The language $\mathcal{FSC}$ consists of the finite set constraints over the variables $and1.ok$, $and2.ok$, $xor1.ok$, $xor2.ok$, and $or.ok$ with equal frame $V = \{ok, faulty\}$. We abbreviate the finite set constraint $M(xor1.ok, \{ok\})$ by $xor1.ok$ and $M(xor1.ok, \{faulty\})$ by $\neg xor1.ok$, and similarly for the other variables. The language $\mathcal{L}$ is built up from the propositional variables $in1$, $in2$, inc, sum, $carry$, $v1$, $v2$, and $v3$ with the usual propositional connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$. Further, we use the usual propositional entailment relation denoted here by $\vdash$. Then the representation Ξ of χ is given by first the formulas representing the information of the components of the adder, i.e. the *and*-, the *xor*-, and the *or*-gates,

$$\begin{aligned} and1.ok &\rightarrowtail v3 \leftrightarrow (in1 \wedge in2) \\ and2.ok &\rightarrowtail v2 \leftrightarrow (inc \wedge v1) \\ xor1.ok &\rightarrowtail v1 \leftrightarrow ((in1 \wedge \neg in2) \vee (\neg in1 \wedge in2)) \\ xor2.ok &\rightarrowtail sum \leftrightarrow ((inc \wedge \neg v1) \vee (\neg inc \wedge v1)) \\ or.ok &\rightarrowtail carry \leftrightarrow (v2 \vee v3), \end{aligned} \tag{7.3}$$

then a formula which represents the connection between the variable $adder.ok$, which denotes the working mode of the whole adder, and the variables denoting the state of the individual components,

$$\neg(adder.ok \leftrightarrow (and1.ok \wedge xor1.ok \wedge xor2.ok \wedge or.ok)) \rightarrowtail \bot, \tag{7.4}$$

and finally the observations

$$\top \rightarrowtail in1 \wedge \neg in2 \wedge inc \wedge sum \wedge \neg carry,$$

The probability *pr* is given by

$$\begin{aligned} pr(and1.ok) &= pr(and2.ok) &= 0.99 \\ pr(xor1.ok) &= pr(xor2.ok) &= 0.95 \\ pr(or.ok) &= 0.98 \end{aligned}$$

Let $\Omega := \{0,1\}^8$ be the set of interpretations of the language $\mathcal{FSC}$ where the components of a vector ω in Ω correspond to the variables ok, $and1.ok$, $and2.ok$, $xor1.ok$, $xor2.ok$, $or.ok$ in that order. From the argumentation system, we can deduce a hint $\mathcal{H} = (\Omega, \Gamma, \Phi, D, P)$ where the elements of the information algebra (Φ, D) are subsets of the cartesian product

$$\{in1, \overline{in1}\} \times \{in2, \overline{in2}\} \times \{inc, \overline{inc}\} \times \{sum, \overline{sum}\} \\ \times \{carry, \overline{carry}\} \times \{v1, \overline{v1}\} \times \{v2, \overline{v2}\} \times \{v3, \overline{v3}\}.$$

The focal mapping Γ of the hint $\mathcal{H}$ is then deduced from Ξ, i.e. we get:

$$\begin{array}{ccc} \omega & \overset{\Gamma}{\to} & \Gamma(\omega) \\ \hline (0,0,0,0,0,0) & \longrightarrow & \left\{ \begin{array}{c} (in1, \overline{in2}, inc, sum, \overline{carry}, v1, v2, v3), \\ (in1, \overline{in2}, inc, sum, \overline{carry}, v1, v2, \overline{v3}), \\ \vdots \\ (in1, \overline{in2}, inc, sum, \overline{carry}, \overline{v1}, \overline{v2}, \overline{v3}) \end{array} \right\} \\ (0,0,0,0,0,1) & \longrightarrow & \left\{ \begin{array}{c} (in1, \overline{in2}, inc, sum, \overline{carry}, v1, \overline{v2}, \overline{v3}), \\ (in1, \overline{in2}, inc, sum, \overline{carry}, \overline{v1}, \overline{v2}, v3) \end{array} \right\} \\ & \vdots & \\ (0,1,1,0,1,1) & \longrightarrow & \{(in1, \overline{in2}, inc, sum, \overline{carry}, \overline{v1}, \overline{v2}, \overline{v3})\} \\ & \vdots & \\ (1,0,0,0,0,0) & \longrightarrow & \emptyset \\ & \vdots & \\ (1,1,1,1,1,1) & \longrightarrow & \emptyset \end{array}$$

Quasi-support set, diagnoses, conflicts, etc. can then be computed using this hint $\mathcal{H}$. Here, we are especially interested in the diagnoses, which, in a logical representation, are

$$Diag \;=\; \neg xor1.ok \vee \Big(\neg or.ok \wedge \neg xor2.ok\Big) \vee \Big(\neg and2.ok \wedge \neg xor2.ok\Big).$$

We have three minimal diagnoses: one which consist of only one faulty component, i.e. $\neg xor1.ok$, and two which consist of two faulty components, i.e. $\neg or.ok \wedge xor2.ok$ and $\neg and2.ok \wedge xor2.ok$. These diagnoses can be weighted by the probability *pr* and, if we consider the prior probability, we get

$$\begin{aligned} pr(\neg xor1.ok) &= 1 - 0.95 &= 0.05 \\ pr(\neg or.ok \wedge \neg xor2.ok) &= (1-0.98)(1-0.95) &= 0.001 \end{aligned}$$

$$pr(\neg and2.ok \wedge \neg xor2.ok) \quad = \quad (1-0.99)(1-0.95) \quad = \quad 0.0005,$$

and clearly in the present situation, the diagnosis $\neg xor1.ok$ is far more probable than the other two. Considering the huge difference in the probability of the diagnoses, usually we would replace or repair the component $xor1$ first.

A module for a simple 1-bit adder as constructed above can be used for building other, more complex systems. For example, two 1-bit adders can be used for a 2-bit adder. Similarly, two 2-bit adders can be used for a 4-bit adder, and so on. In this sense, 2-bit and 4-bit adders can be modeled as follows:

```
(tell
  (module ADDER2 ((var in1 in2 in3 in4 inc
                       sum1 sum2 carry binary))
    (var ok v binary)
    (ADDER :A1 in1 in2 inc sum1 v)
    (ADDER :A2 in3 in4 v sum2 carry)
    (<-> ok (and A1.ok A2.ok))))

(tell
  (module ADDER4 ((var in1 in2 in3 in4 in5 in6 in7 in8 inc
                       sum1 sum2 sum3 sum4 carry binary))
    (var ok v binary)
    (ADDER2 :A21 in1 in2 in3 in4 inc sum1 sum2 v)
    (ADDER2 :A22 in5 in6 in7 in8 v sum3 sum4 carry)
    (<-> ok (and A21.ok A22.ok))))
```

Further modules for 8-bit and 16-bit adders are defined similarly. Consider now the situation where a modular 16-bit adder is faulty. Suppose that all input and output values are 0, except the value of the last output $sum16$ is 1.

```
(tell
  (ADDER16 :ADDER16
    in01 in02 in03 in04 in05 in06 in07 in08 in09 in10
    in11 in12 in13 in14 in15 in16 in17 in18 in19 in20
    in21 in22 in23 in24 in25 in26 in27 in28 in29 in30
    in31 in32 inc
    sum01 sum02 sum03 sum04 sum05 sum06 sum07 sum08
    sum09 sum10 sum11 sum12 sum13 sum14 sum15 sum16 carry))

(observe
  (not in01) (not in02) (not in03) (not in04) (not in05)
  (not in06) (not in07) (not in08) (not in09) (not in10)
  (not in11) (not in12) (not in13) (not in14) (not in15)
  (not in16) (not in17) (not in18) (not in19) (not in20)
  (not in21) (not in22) (not in23) (not in24) (not in25)
  (not in26) (not in27) (not in28) (not in29) (not in30)
  (not in31) (not in32) (not inc)
  (not sum01) (not sum02) (not sum03) (not sum04) (not sum05)
  (not sum06) (not sum07) (not sum08) (not sum09) (not sum10)
  (not sum11) (not sum12) (not sum13) (not sum14) (not sum15)
  (not carry) sum16)
```

Clearly, such a system can also be modeled as an argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$ following the example of the 1-bit adder above. Yet the resulting spaces are too big to be represented explicitly here. Nevertheless, we show in chapter 12 that ABEL can compute diagnoses, conflicts, supports, etc. for such a system, for example ABEL computes the 47 minimal diagnoses of this problem:

```
? (ask (sp (not ADDER16.ok)))
QUERY: (SP (NOT ADDER16.OK))
   35.2% : (NOT ADDER16.A82.A42.A22.A2.XOR2.OK)
   35.2% : (NOT ADDER16.A82.A42.A22.A2.XOR1.OK)
   14.1% : (NOT ADDER16.A82.A42.A22.A1.OR.OK)
    7.0% : (NOT ADDER16.A82.A42.A22.A1.AND2.OK)
    7.0% : (NOT ADDER16.A82.A42.A22.A1.AND1.OK)
    ...
```

We show here only the most probable ones (according to the prior probability pr of the argumentation system). So there are two minimal diagnoses which are equiprobable (see chapter 9 for how to compute their probabilities). This means that in general, further information has to be gathered about the system, for example with an additional measurement (cf. chapter 11).

So far in this example, we considered components, i.e. gates, which have two operating modes: either they work correctly and we know their input-output relation, or they do not work correctly and we know nothing about their behavior. Now, suppose that in a more elaborated model we have explicit knowledge about one or several fault modes of a component. For example, consider an *and*-gate with three different possible fault modes:

st_0: the output is always 0 (stuck at 0),

st_1: the output is always 1 (stuck at 1),

inv: the output is always the negation of the correct output (inversion).

Assume that the probabilities for ok, st_0, st_1, and inv are 0.93, 0.01, 0.01, and 0.05, respectively. In an argumentation system, this can easily be modeled. The language $\mathcal{FSC}$ is now built over the finite set constraints with the variables $and1$, $and2$, ... and the respective domains $\{st_0, st_1, inv, ok\}$. The corresponding line from (7.3) for the *and*-gate $and1$ is replaced by the four lines

$$\begin{aligned}
M(and1, \{ok\}) &\rightarrowtail v3 \leftrightarrow (in1 \wedge in2)\\
M(and1, \{st0\}) &\rightarrowtail \neg v3\\
M(and1, \{st1\}) &\rightarrowtail v3\\
M(and1, \{inv\}) &\rightarrowtail v3 \leftrightarrow \neg(in1 \wedge in2),
\end{aligned}$$

and similar for other components. If all gates have possible fault modes according to this scheme, then the rule (7.4) of Ξ changes into

$$\neg(adder.ok \leftrightarrow \left(\begin{array}{l} M(and1, \{ok\}) \wedge M(xor1, \{ok\}) \\ \wedge M(xor2, \{ok\}) \wedge M(or, \{ok\}) \end{array} \right)) \quad \rightarrowtail \quad \perp.$$

Using such an extended model for the gates, it is possible to compute more precise diagnoses, but clearly with more computation effort.

Note that in ABEL, this situation can be modeled as well (cf. also chapter 12): a new variable type is created to deal with this kind of operating modes. The domain of the variable consists of the fault modes together with the correct operating mode *ok*:

```
(tell (type modes (ok st0 st1 inv)))
```

The corresponding module in ABEL can then be written as:

```
(tell
  (module AND-GATE ((var in1 in2 out binary))
    (ass mode modes (0.93 0.01 0.01 0.05))
    (-> (= mode ok) (<-> out (and in1 in2)))
    (-> (= mode st0) (not out))
    (-> (= mode st1) out)
    (-> (= mode inv) (<-> out (not (and in1 in2))))))
```

Similar modules can be defined for the other components. ⊖

Example 7.5: Failure Trees

The concept of failure trees (or event trees) can help to determine possible causes of a system failure, and it provides a useful structure for the computation of the probability of the system's working mode. In the case of an ordinary tree, the possible causes are rather easy to detect. Yet in the literature, the term *failure tree* is somewhat misleading because in general, the structure considered is a causal network and not a tree. In this more general case, the computation of the causes is more difficult.

The failure tree of the following medical example was originally modeled in (Simonaitus *et al.*, 1972), described in (Barlow & Proschan, 1975) and adapted to the context of model-based diagnostics in (Anrig *et al.*, 1999). The purpose is to analyze the electrical shock hazard to a patient using a certain heart assist device:

> "The intra-aortic balloon (IAB) circulatory assist device is intended to provide temporary circulatory assistance following obstruction of blood circulation from the heart. In its application, a balloon-catheter positioned in the thoracic aorta is synchronously inflated and deflated with the action of the heart, resulting in decreased heart work, increased coronary blood flow, and support of the general circulation.
>
> A synchronization of the balloon with the heart is obtained through monitoring of the electrocardiogram (ECG) of the patient by a control console. This console provides all of the control, monitoring display, and alarm functions required for operation of the balloon

device. In addition, provision is made for automatic deflation of the balloon in the event of an anomaly in the ECG signal or functioning of the balloon.

A major hazard to the patient is electrical shock, either macroshock or microshock. Macroshock can cause heart failure (ventricular fibrillation) by currents entering the body through connections at the skin. Microshock can cause heart failure by currents having a conductive path directly to the vicinity of the heart.

Other hazards to the patient are balloon inflation during the contraction period of heart function and balloon overpressurization."

A detailed structure of the example is shown in fig. 7.7. Note that the example is only treated partially here. In order to reduce the tree's complexity, some terminal nodes have been introduced to hide some subtrees. The abbreviations for the nodes introduced in the figure (e.g. HP0 for the top node) will be used in the ABEL model in order to keep the source code readable.

The structure of the tree of fig. 7.7 can easily be implemented in ABEL. The terminal nodes of the tree are modeled as binary assumptions, and the non-terminal nodes as binary variables.

```
(tell
  (var HP0 binary)

  (ass BI1 binary 0.01)
  (ass BO1 binary 0.01)
  (var ES1 binary)              (<-> HP0 (or BI1 BO1 ES1))

  (var MI2 MA2 binary)          (<-> ES1 (or MI2 MA2))

  (var PI3 PG3 binary)
  (ass PE3 binary 0.01)         (<-> MI2 (and PI3 PG3))
                                (<-> MA2 (and PE3 PG3))

  (ass TB4 binary 0.08)
  (ass TE4 binary 0.06)
  (ass TO4 binary 0.02)         (<-> PG3 (or TB4 TE4 TO4))

  (var PE5 PP5 binary)          (<-> PI3 (or PE5 PP5))

  (var CA6 BC6 BK6 binary)
  (ass AP6 binary 0.01)         (<-> PE5 (or CA6 BC6))
                                (<-> PP5 (or BK6 AP6))

  (var PL7 HL7 PE7 binary)
  (ass CP7 binary 0.01)
  (ass SV7 binary 0.05)         (<-> CA6 (and PL7 HL7))
                                (<-> BC6 (and HL7 PE7 CP7))
                                (<-> BK6 (and CP7 SV7))
```

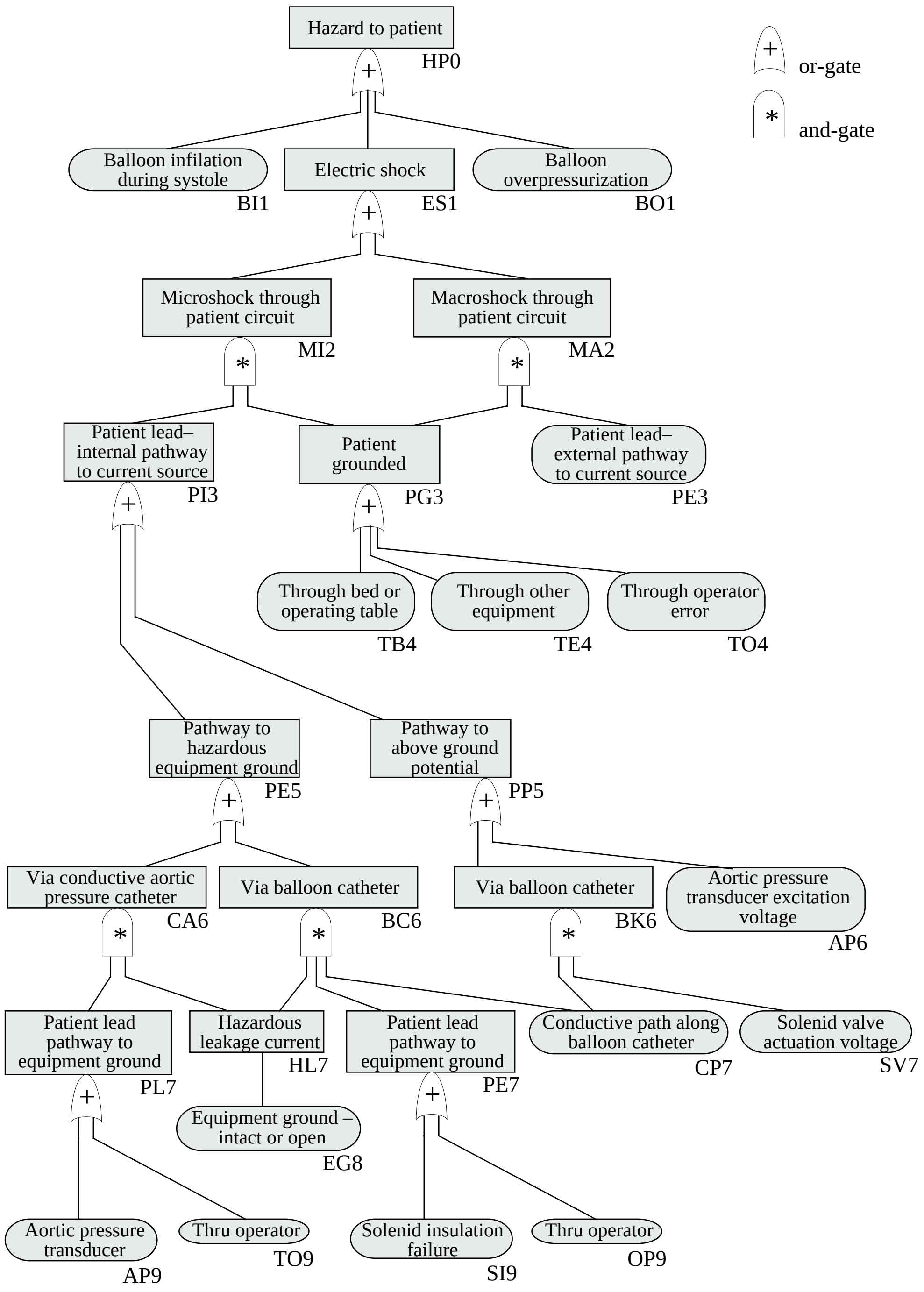

Hazard to patient
HP0
+
or-gate
*
and-gate
Balloon infilation during systole
BI1
Electric shock
ES1
Balloon overpressurization
BO1
Microshock through patient circuit
MI2
Macroshock through patient circuit
MA2
Patient lead–internal pathway to current source
PI3
Patient grounded
PG3
Patient lead–external pathway to current source
PE3
Through bed or operating table
TB4
Through other equipment
TE4
Through operator error
TO4
Pathway to hazardous equipment ground
PE5
Pathway to above ground potential
PP5
Via conductive aortic pressure catheter
CA6
Via balloon catheter
BC6
Via balloon catheter
BK6
Aortic pressure transducer excitation voltage
AP6
Patient lead pathway to equipment ground
PL7
Hazardous leakage current
HL7
Patient lead pathway to equipment ground
PE7
Conductive path along balloon catheter
CP7
Solenid valve actuation voltage
SV7
Equipment ground – intact or open
EG8
Aortic pressure transducer
AP9
Thru operator
TO9
Solenid insulation failure
SI9
Thru operator
OP9

Figure 7.7: The failure tree of the heart assist device.

```
(ass EG8 binary 0.05)         (<-> HL7 EG8)

(ass AP9 binary 0.15)
(ass TO9 binary 0.01)
(ass SI9 binary 0.01)
(ass OP9 binary 0.01)         (<-> PL7 (or AP9 TO9))
                              (<-> PE7 (or SI9 OP9)))
```

In reliability theory, minimal paths and reliabilities are calculated for such systems. Here, this means to compute the support for the top event `HP0`, whose results are the minimal paths, and the degree of support of `HP0`, whose result is the reliability of the system.

The corresponding argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash, pr)$ is then defined as follows: The language $\mathcal{FSC}$ consists of the finite set constraints over the variables

$$BI1,\ BO1,\ PE3,\ TB4,\ TE4,\ TO4,\ AP6,\ CP7,$$
$$SV7,\ EG8,\ AP9,\ TO9,\ SI9,\ OP9$$

with frames $\{on, off\}$. Again, because there are only two elements in the frames the finite set constraint $M(BI1, on\})$ is abbreviated by $BI1$ and $M(BI1, off\})$ by $\neg BI1$. The language $\mathcal{L}$ is the propositional language over all other variables, i.e. $HP0$, $ES1$, $MI2$, $MA2$, …, $PL7$, $HL7$, $PE7$, together with the usual entailment relation. The representation Ξ of χ consists of 44 rules. For example, the first ABEL rule `(<-> HP0 (or BI1 BO1 ES1))` translates into the set of rules

$$\begin{array}{rcl} BI1 & \rightarrowtail & HP0 \\ B01 & \rightarrowtail & HP0 \\ \top & \rightarrowtail & HP0 \vee ES1 \\ \neg BI1 \wedge \neg B1 & \rightarrowtail & ES1 \vee \neg HP0 \end{array}$$

This is done by using the special structure of the example, namely we can considering the rule as a formula in the propositional language built over all atoms, i.e. $HP0 \leftrightarrow (BI1 \vee BO1 \vee ES1))$, bringing the rule in a conjunctive normal form CNF, i.e.

$$(\neg BI1 \vee HP0) \wedge (\neg B01 \vee HP0) \wedge (HP0 \vee ES1)$$
$$\wedge (BI1 \vee B1 \vee ES1 \vee \neg HP0)$$

and transforming every disjunction into the desired form. The information algebra (Φ, D) is built as in the previous example, and the hint $\mathcal{H}$, which corresponds to this argumentation system, can be constructed. But the focal mapping cannot be shown here explicitly due to its size. Nevertheless, we can compute results using ABEL, for example the arguments for the hypothesis $HP0$, i.e.

```
? (ask (sp HP0))
QUERY: (SP HP0)
   40.7% : BI1
   40.7% : BO1
    3.3% : PE3 TB4
    3.3% : AP6 TB4
    2.4% : PE3 TE4
    2.4% : AP6 TE4
    2.4% : AP9 EG8 TB4
    1.8% : AP9 EG8 TE4
    0.8% : PE3 TO4
    0.8% : AP6 TO4
    0.6% : AP9 EG8 TO4
    0.2% : EG8 TB4 TO9
    0.2% : CP7 SV7 TB4
    0.1% : EG8 TE4 TO9
    0.1% : CP7 SV7 TE4
    0.0% : EG8 TO4 TO9
    0.0% : CP7 SV7 TO4
    0.0% : CP7 EG8 SI9 TB4
    0.0% : CP7 EG8 OP9 TB4
    0.0% : CP7 EG8 SI9 TE4
    0.0% : CP7 EG8 OP9 TE4
    0.0% : CP7 EG8 SI9 TO4
    0.0% : CP7 EG8 OP9 TO4
```

ordered by prior probability. These calculations and results are well-known in reliability theory, and a number of tools can be used for them. However, using the argumentation system, we can do more than that. Assume that there has been a hazard to the patient (HP0). Furthermore, suppose that neither balloon inflation during systole (BI1) nor balloon overpressurization (BO1) has been observed. So in terms of the argumentation system, we have the additional rules

$$\top \quad \rightarrowtail \quad HP0 \wedge \neg BI1 \wedge \neg B01.$$

Using ABEL, we compute the minimal diagnoses for this situation:

```
? (ask (sp tautology))
QUERY: (SP HP0)
   17.5% : PE3 TB4 (NOT BI1) (NOT BO1)
   17.5% : AP6 TB4 (NOT BI1) (NOT BO1)
   13.2% : PE3 TE4 (NOT BI1) (NOT BO1)
   13.2% : AP6 TE4 (NOT BI1) (NOT BO1)
   13.2% : AP9 EG8 TB4 (NOT BI1) (NOT BO1)
    9.9% : AP9 EG8 TE4 (NOT BI1) (NOT BO1)
    4.4% : PE3 TO4 (NOT BI1) (NOT BO1)
    4.4% : AP6 TO4 (NOT BI1) (NOT BO1)
    3.3% : AP9 EG8 TO4 (NOT BI1) (NOT BO1)
    0.9% : CP7 SV7 TB4 (NOT BI1) (NOT BO1)
    0.9% : EG8 TB4 TO9 (NOT BI1) (NOT BO1)
    0.7% : CP7 SV7 TE4 (NOT BI1) (NOT BO1)
```

```
0.7% : EG8 TE4 T09 (NOT BI1) (NOT BO1)
0.2% : CP7 SV7 T04 (NOT BI1) (NOT BO1)
0.2% : EG8 T04 T09 (NOT BI1) (NOT BO1)
. . .
ETC.
```

In total, there are 21 minimal diagnoses. Each diagnosis reflects the possible causes for the top level event, the hazard to the patient. Note that every diagnosis contains also the observed assumptions `BI1` and `BO1`. In the present case, there are several causes within a certain interval of probability, so some mechanisms have to be applied to discriminate between these diagnoses. For example, another variable (or assumption) of the system can be measured (see chapter 10) or numerical queries can be helpful for a better analysis of the situation (see chapter 9). ⊖

8 Computing Conflicts and Arguments

In this chapter, a general architecture for computing conflicts and symbolical arguments using local computations is presented. In general, for computing conflicts or arguments, the whole knowledge base has to be marginalized to some sublanguage or in a system with variables several variables have to be eliminated, but often such a global marginalization is infeasible due to the number of pieces of information which are generated during the marginalization. Generalizing concepts of Lauritzen & Spiegelhalter (1988), Shenoy & Shafer developed a concept called *Computation in Hypertrees* (or Valuation Networks) serving to “break down” this process into small pieces, which hopefully are small enough to be processed (Shenoy, 1989; Shenoy & Shafer, 1990; Shafer, 1991; Lauritzen & Shenoy, 1995). Others have used this concept for probabilistic assumption-based reasoning (Kohlas, 1993b; Haenni, 1995; Haenni, 1996; Kohlas *et al.*, 1999b), for information systems (Kohlas, 1997b), for information algebras (Kohlas & Stärk, 1996a; Kohlas & Stärk, 1996b), for valuation algebras (Shenoy & Kohlas, 2000), and for propositional information systems (Kohlas *et al.*, 1999b), and there are a lot of applications to other calculi. The advantage of this approach is that, once a hypertree is generated and the pieces of information distributed on the nodes, all computations for messages between the nodes take place within the respective sublanguages which are attached to the nodes. Clearly, one is interested in “small” sublanguages on the nodes in order to make the computations efficiently.

The results of the following sections are presented in terms of a general marked information algebra $\langle \Phi, D \rangle$. But in the present context, the results will specifically be applied to the marked algebra $\langle P_\Phi, D \rangle$, which corresponds to the algebra of allocations (P_Φ, D) defined in section 2.4, and to $\langle L_\Phi, S \rangle$ and $\langle H_\Phi, S \rangle$ which correspond to (L_Φ, S) and (H_Φ, S) respectively defined in section 6.3.

So in the sequel $\langle \Phi, D \rangle$ denotes a marked information algebra with combination operation $(\phi_1, \phi_2) \mapsto \phi_1 \oplus \phi_2$ and marginalization $(\phi_1, x) \mapsto {\phi_1}^{\downarrow x}$ for $\phi_1, \phi_2 \in \Phi$ and $x \in D$.

Note that this architecture can also be used for computations in information systems and hence also for computations in argumentation systems, because they are special case of information systems. This is presented in section 8.3 together with an example which shows concrete computations.

The first section introduces the concept of a hypertree and Markov tree. Then propagation of information on Markov trees is defined, in section 8.2 the inward phase, in section 8.4 the outward phase. Further, the problem of computing marginals on domains which are not contained in the tree is addressed in section 8.5. Finally, section 8.6 treats the concepts of compilation versus propagation "on demand".

The following sections form a short introduction into the main topics of computations on hypertrees within the framework of marked information algebras; for more details and further publications on hypertrees and related concepts see (Shafer, 1991; Haenni, 1996; Berge, 1989). In (Kohlas *et al.*, 2000), the hypertree is not constructed explicitly anymore but its structure is used only implicitly.

8.1 Hypertrees

Let ϕ be a marked information in $\langle \Phi, D \rangle$. For the computation of conflicts and quasi-supports, the problem is now to compute a marginal $\langle \psi, L' \rangle$ for some sublanguage $L' \in D$ such that $\langle \psi, L' \rangle = \langle \phi, L \rangle^{\downarrow L'}$. Assume that there is a decomposition of $\langle \phi, L \rangle$ consisting of marked information $\langle \phi_i, L_i \rangle$ for $i = 1, \ldots, m$ such that

$$\langle \phi, L \rangle \quad = \quad \langle \phi_1, L_1 \rangle \oplus \langle \phi_2, L_2 \rangle \oplus \cdots \oplus \langle \phi_m, L_m \rangle. \tag{8.1}$$

This implies that $L = L_1 \vee \cdots \vee L_m$ for $L_i = d(\langle \phi_i, L_i \rangle)$ and $L = d(\langle \phi, L \rangle)$. Assume further that this decomposition satisfies that for every $i = m, m-1, \ldots, 2$ there is a $b(i) < i$ such that

$$L_i \wedge L_{b(i)} \quad = \quad L_i \wedge \left(\bigvee_{j=1}^{i-1} L_j \right). \tag{8.2}$$

A family of languages $\{L_1, \ldots, L_m\}$ which satisfies the property (8.2) is called a **hypertree** with the hypertree construction sequence $L_1, \ldots, L_m$. The decomposition is called the hypertree decomposition. Starting from a marked statement, the determination of such a hypertree decomposition is not a easy task (Arnborg *et al.*, 1987), but very efficient heuristics have been developed. However, this task is not considered here, see (Haenni & Lehmann, 1999; Kohlas, 1997b; Kohlas & Monney, 1995) for algorithms and further literature and (Cano & Moral, 1995) for a comparison of different heuristics.

The hypertree can be represented as a normal tree (which is then called a **Markov tree** or a **join tree**) if the L_i's are considered as the nodes of the tree and the edges between them as $(L_i, L_{b(i)})$ (see for example fig. 8.1). Shenoy & Shafer (1990) show that the so constructed graph is indeed a tree with the additional property that for any node L_k on the (unique) path from L_i to L_j in the tree the condition $L_i \wedge L_j \subseteq L_k$ holds. In the sequel, if L_i is a node then often it will be addressed as node i, and the language L_i is called the **label** of node i, written $d(i) = L_i$. The Markov tree is then usually represented as a pair (E, N) where E is the set of nodes and N the set of edges, hence $N \subseteq E \times E$.

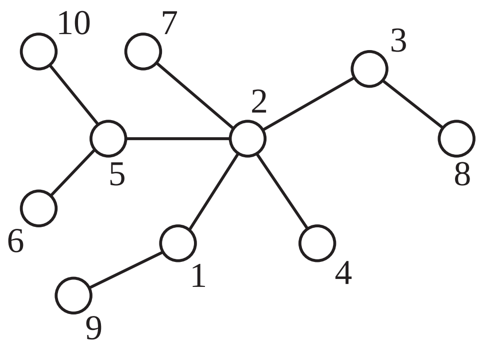


Figure 8.1: A Markov tree with nodes L_i labeled by i for $i = 1, \ldots, 10$.

Note that there is always a trivial hypertree decomposition $\phi = \phi$. In general, the mapping ϕ is a combination of different pieces of information in the information algebra, and this structure usually does not build a hypertree but can be used to construct a hypertree.

We assume in the sequel that there is a hypertree decomposition and use it for our computations.

8.2 Inward Propagation

The main result used for the so-called inward propagation is the following theorem. The idea behind it is a propagation process of pieces of information from the leafs towards a so-called root node in which the information is collected. In the sequel, we usually omit the label of the marked information in order to simplify notation.

Theorem 8.1 *(Shenoy & Shafer, 1990) Let (8.1) be a hypertree decomposition of ϕ and define*

$$H_i := \bigvee_{j=1}^{i} L_j \quad \text{for } i = m, m-1, \ldots, 1, \tag{8.3}$$

$$\phi_j^{(m)} := \phi_j \quad \text{for } j = 1, \ldots, m. \tag{8.4}$$

If, for $i = m, m-1, \ldots, 2$, we set

$$\phi_{b(i)}^{(i-1)} := \phi_i^{(i)\downarrow L_i \wedge L_{b(i)}} \oplus \phi_{b(i)}^{(i)}, \tag{8.5}$$

$$\phi_j^{(i-1)} := \phi_j^{(i)} \qquad \text{for } j \in \{1, \ldots, i-1\} - \{b(i)\}, \tag{8.6}$$

then, for $i = m, m-1, \ldots, 1$

$$\phi^{\downarrow H_i} = \phi_1^{(i)} \oplus \phi_2^{(i)} \oplus \cdots \oplus \phi_i^{(i)}, \qquad \text{and} \qquad d(\phi_j^{(i)}) = L_j. \tag{8.7}$$

For a proof see (Kohlas, 1997b).

Using this theorem, we can sequentially compute the marginal of ϕ with respect to $H_{m-1}, H_{m-2}, \ldots, H_1$. If $L_1 = L'$ then the problem is solved because $H_1 = L_1$.

Initially, on every node L_i, the information $\phi_i = \phi_i^{(m)}$ is stored. This is step $i = m$. For $i = m-1, m-2, \ldots, 1$ the step (according to theorem 8.1) can then be represented as the message passing scheme in fig. 8.2, focused on the nodes i and $b(i)$, because on all other nodes we have $\phi_j^{(i-1)} = \phi_j^{(i)}$.

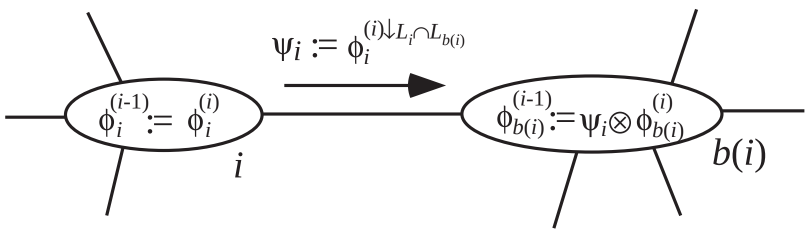

$\psi_i := \phi_i^{(i)\downarrow L_i \cap L_{b(i)}}$
$\phi_i^{(i-1)} := \phi_i^{(i)}$
i
$\phi_{b(i)}^{(i-1)} := \psi_i \otimes \phi_{b(i)}^{(i)}$
$b(i)$

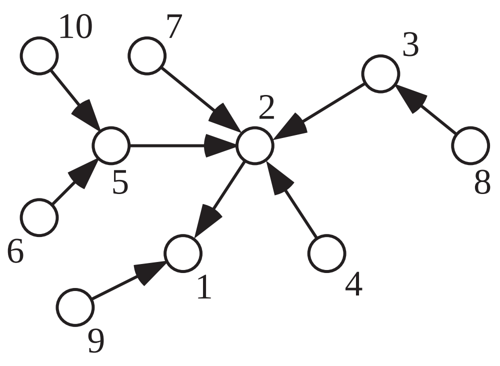
Figure 8.2: Message passing during inward propagation in step number i focused on nodes i and $b(i)$.

So a step i consists in fact of sending a message from node i to node $b(i)$ and combining the information thereon with the message. This graphical interpretation shows that parallel, distributed computing is possible. We call the node $b(i)$ the **inward neighbor** of node i. Therefore, every leaf node of the tree can send its message to its inward neighbor. Every node which is not a leaf waits until it has received the messages from all outward neighbors, combines those messages with the information contained in the node itself and sends a message ψ_i (see fig. 8.2) to its inward neighbor. The only node without any inward neighbor is called the root node, and the process stops if this node has received all messages from its outward neighbors and combined them with the information contained on itself.

This process is called the **inward propagation** or **collect phase**. An example with nodes 1 to 10 is depicted in fig. 8.3; the arrows show the direction of the messages.

After step i, the information contained in node j is $\phi_j^{(i)}$, and at the end of the inward phase we have therefore on node 1 the information

$$\phi_1^{(1)} = \phi^{\downarrow H_1} = \phi^{\downarrow L_1}. \tag{8.8}$$



(a) An arrow denotes the direction of the message which is passed on the (undirected) arc.

$i \rightarrow b(i)$	
10	5
9	1
8	3
7	2
6	5
5	2
4	2
3	2
2	1

(b) The corresponding function b.

Figure 8.3: General message passing scheme during inward propagation towards the root node 1.

If the hypertree construction sequence is given, this sequence can always be reordered such that any of the nodes is the first node of the sequence. Therefore in the hypertree every node can be chosen as the root node for the inward propagation, and the resulting information on that node is then ϕ marginalized to the domain of the node. Thus one hypertree can be used to compute different marginals.

There are two main advantages in this computational scheme: first, the computations can be made in parallel (as illustrated in fig. 8.3); second, all computations are done within a domain of some node of the tree. Therefore, if combination and marginalization on the domain of every node in the tree are feasible, then this ensures that all computations for the marginal of the whole information ϕ on any node of the tree are feasible. However, if several marginals have to be computed at once, then it is not efficient to start the process several times from scratch: in section 8.4, a solution for this problem is presented.

The concept of idempotency is not used in the inward phase, therefore this scheme is also useful for Bayesian or Belief networks (Shenoy & Shafer, 1990).

8.3 Computing Conflicts in Argumentation Systems

As already mentioned, this local computation concept can also be used for information systems (Kohlas, 1997c) and for argumentation systems as they are a special case of information systems.

Consider the argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$ as an information system. In chapter 4, we defined the concept of marginal and combination for information systems. Using these concepts, we can apply the concept of local computation. For a set of formulas Ξ and a language L we usually avoid to compute *the* marginal $\Xi^{\downarrow L} = \langle C_L^+(\Xi), L \rangle$, but instead only a marginal which is easy to compute and represent (cf. definition of a marginal in section 4.2); nevertheless in this section, we use the notation $\Xi^{\downarrow L}$ for denoting *any* marginal. Further, often it is only necessary to consider minimal sets of formulas, that is if $f \rightarrowtail \phi$ is a formula in X then X does not need to contain any formula $f \wedge g \rightarrowtail \phi$ for $g \in \mathcal{FSC}$. This is correct because at the end of the information processing we are either interested in conflict sets, support sets, etc., which are unions of system states (see (6.25) and (6.27)), or their logical representations, i.e. conflicts, supports, etc., which are disjunctions over $\mathcal{FSC}$ formulas (see (6.26) and (6.28)), but in both cases the results are equivalent.

We consider here especially the problem of computing conflicts (and diagnoses), therefore we construct the hypertree such that it contains a node with an "empty" label. "Empty" in this case means that the label is the language $\mathcal{L}_\emptyset := \{(f \rightarrowtail \bot) : f \in \mathcal{FSC}\}$, which is a sublanguage of the language $\mathcal{L}^+$ of the argumentation system. The node with label $\mathcal{L}_\emptyset$ is in the sequel usually denoted by $\emptyset$. Now, this node is considered as the root node of the inward propagation such that at the end of this process, the information available on the node $\emptyset$ is $\Xi_\emptyset \sim \Xi^{\downarrow \mathcal{L}_\emptyset}$ and it can be used to compute the conflicts, because

$$Conf \cong \bigvee_{(f \rightarrowtail \bot) \in C^+(\Xi)} f \cong \bigvee_{(f \rightarrowtail \bot) \in C^+_{\mathcal{L}_\emptyset}(\Xi)} f \cong \bigvee_{(f \rightarrowtail \bot) \in \Xi_\emptyset} f, \tag{8.9}$$

where the first equivalence is (6.26), the second follows from $\bot \in \mathcal{L}_\emptyset$ and the last one is true because $\Xi_\emptyset$ and $C^+_{\mathcal{L}_\emptyset}(\Xi)$ are equivalent.

Using the same techniques, we can also compute arguments (quasi-supports, supports, ...) in argumentation systems.

The following example illustrates these computations.

Example 8.2: Continuation of example 7.2

Consider the representation Ξ of the argumentation system of example 7.2. The interesting sublanguages are defined by languages over a part of the variables $a, b, \ldots, f, x, y, z$ together with any FSC of $\mathcal{FSC}$. A sublanguage over the variables x, y is denoted by $\{x, y\}$. The first five elements of Ξ are contained in the sublanguages $\{a, c, x\}$, $\{b, d, y\}$, $\{c, e, z\}$, $\{f, x, y\}$, and $\{g, y, z\}$ respectively. The element $\top \rightarrowtail (a = 3) \wedge \cdots \wedge (g = 12)$ is split into a set of equivalent elements $\top \rightarrowtail (a = 3)$, ..., $\top \rightarrowtail (g = 12)$, and each of these elements is contained in at least one of the sublanguages above. Therefore, using these sublanguages together with the sublanguage $\mathcal{L}_\emptyset$, a Markov tree is constructed (fig. 8.2), where each piece of information from Ξ is represented within the corresponding edge whose label (represented as a capital letter) represents a sublanguage containing the piece of information. If an information can be put in several edges, for example $(c = 3)$ can be put either in ACX, CXY, or CEZ, then one of the nodes

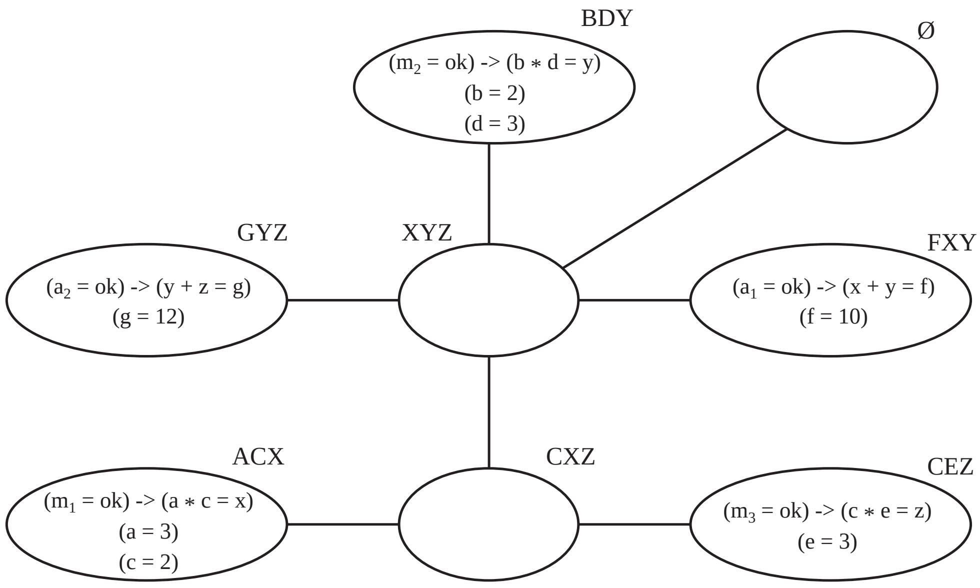


Figure 8.4: Markov tree of example 8.2

is selected arbitrarily. As an abbreviation, we write $(i = ok)$ for the finite set constraint $M(i, \{ok\})$.

We select the node with label $\emptyset$ as root node. The propagation algorithm starts now by sending messages $m_{n_1 \to n_2}$ from node n_1 to n_2, so for example consider the knowledge on node BDY, that is

$$\phi_{BDY} := \left\{ \begin{array}{l} \top \rightarrowtail b = 2 \\ \top \rightarrowtail d = 3 \\ (m_2 = ok) \rightarrowtail (b * d = y) \end{array} \right\}, \tag{8.10}$$

then the message towards its neighbor XYZ is

$$m_{BDY \to XYZ} \quad = \quad \phi_{BDY}{}^{\downarrow \{Y\}} \quad = \quad \{(m_2 = ok) \rightarrowtail (y = 6)\}.$$

How is this marginalization computed? In fact, we have to compute all formulas $(f \rightarrowtail X) \in \mathcal{L}^+{}_{\{Y\}}$ which are implied by the knowledge on node BDY, that is

$$\left\{ (f \rightarrowtail X) \in \mathcal{L}^+ : X \subseteq \mathcal{L}_{\{Y\}},\ \phi_{BDY} \vdash (f \rightarrowtail X) \right\}.$$

In this example, we are dealing with very simple equations, so we can use variable elimination for computing a minimal representation of the marginal, i.e. we isolate the variables to be eliminated on one side of the equation and replace their occurrences in the other formulas together with a simultaneous updating of the corresponding $\mathcal{FSC}$ part of the formula. Concretely, first we eliminate the variable b, so we isolate b on one side of the equation, which is already the case in $\top \rightarrowtail b = 2$ and replace the occurrences of b in the other formulas: there is not change for $\top \rightarrowtail d = 3$ and for the last formula, we replace the occurrence of b by the constant 2 and add the $\mathcal{FSC}$ formula $\top$ conjunctively to the left hand side of the equation, that is we get $(m_2 = ok) \wedge \top \rightarrowtail (2 * d = y)$. This formula is clearly equivalent to

$$(m_2 = ok) \quad \rightarrowtail \quad (2 * d = y).$$

We have now eliminated the variable b from ϕ_{BDY} and have two formulas left, namely $\top \rightarrowtail d = 3$ and $(m_2 = ok) \rightarrowtail (2 * d = y)$. From these formulas, we have to eliminate the variable d, and this is done just as the elimination of b before, so we get one formula $(m_2 = ok) \wedge \top \rightarrowtail (2 * 3 = y)$, which is equivalent to

$$(m_2 = ok) \quad \rightarrowtail \quad (6 = y).$$

The result of the elimination of d and b from ϕ_{BDY} is therefore the set $\{(m_2 = ok) \rightarrowtail (6 = y)\}$ consisting of just one formula in $\mathcal{L}^+{}_{\{Y\}}$. For this formula, we have

$$\phi_{BDY} \vdash (m_2 = ok) \rightarrowtail (6 = y)$$

and even $C^+_{\{Y\}}(\phi_{BDY}) = C^+_{\{Y\}}(\{(m_2 = ok) \rightarrowtail (6 = y)\})$ such that this formula is a representation of the marginal.

Analogously, the messages from the other leaves are computed:

$$\begin{aligned}
m_{GYZ \to XYZ} &= \{(a_2 = ok) \rightarrowtail (y + z = 12)\} \\
m_{FXY \to XYZ} &= \{(a_1 = ok) \rightarrowtail (x + y = 10)\} \\
m_{ACX \to CXZ} &= \{(m_1 = ok) \rightarrowtail (x = 6),\ \top \rightarrowtail (c = 2)\} \\
m_{CEZ \to CXZ} &= \{(m_3 = ok) \rightarrowtail (3 * c = z)\}.
\end{aligned}$$

Then the information on CXZ is the union of the two incoming messages from ACX and CEZ, that is $m_{ACX \to CXZ} \cup m_{CEZ \to CXZ} = \{(m_1 = ok) \rightarrowtail (x = 6), (m_3 = ok) \rightarrowtail (3 * c = z), \top \rightarrowtail (c = 2)\}$, and the next message is

$$\begin{aligned}
m_{CXZ \to XYZ} &= (m_{ACX \to CXZ} \cup m_{CEZ \to CXZ})^{\downarrow\{X,Z\}} \\
&= \{(m_1 = ok) \rightarrowtail (x = 6),\ (m_3 = ok) \rightarrowtail (z = 6)\},
\end{aligned}$$

such that on node XYZ we have the formulas

$$\begin{aligned}
\Xi_{XYZ} &:= m_{BDY \to XYZ} \cup m_{GYZ \to XYZ} \cup m_{FXY \to XYZ} \cup m_{CXZ \to XYZ} \\
&= \left\{ \begin{array}{rcl}
(m_2 = ok) & \rightarrowtail & (y = 6) \\
(a_2 = ok) & \rightarrowtail & (y + z = 12)\} \\
(a_1 = ok) & \rightarrowtail & (x + y = 10)\} \\
(m_1 = ok) & \rightarrowtail & (x = 6) \\
(m_3 = ok) & \rightarrowtail & (z = 6)
\end{array} \right\}.
\end{aligned}$$

Now, we continue the inward propagation towards the node $\emptyset$, that is, we compute the message $m_{XYZ \to \emptyset}$, that is

$$\begin{aligned}
m_{XYZ \to \emptyset} &= \Xi_{XYZ}{}^{\downarrow \mathcal{L}_\emptyset} \\
&= \left\{ \begin{array}{rcl}
(a_1 = ok) \wedge (m_1 = ok) \wedge (m_2 = ok) & \rightarrowtail & \bot, \\
(a_1 = ok) \wedge (a_2 = ok) \wedge (m_1 = ok) \wedge (m_3 = ok) & \rightarrowtail & \bot, \\
(a_1 = ok) \wedge (a_2 = ok) \wedge (m_1 = ok) \wedge (m_2 = ok) & \rightarrowtail & \bot.
\end{array} \right\}
\end{aligned}$$

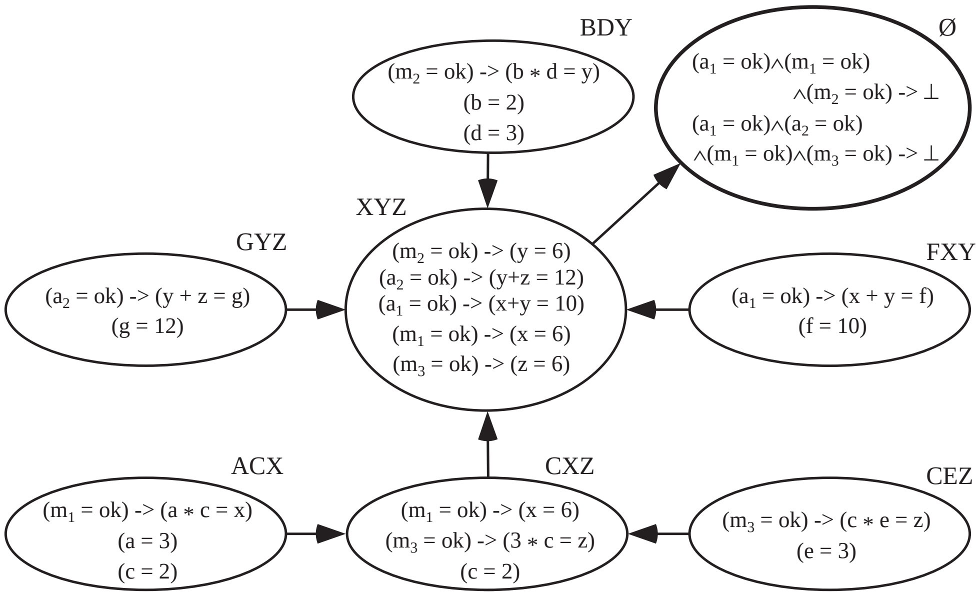


Figure 8.5: Markov tree with the node having an empty label.

The third element is subsumed by the first one and can be dropped. Therefore on node $\emptyset$, we now have the information $\Xi_\emptyset = m_{XYZ \to \emptyset}$ which is equivalent to $\Xi^{\downarrow \mathcal{L}_\emptyset}$. The actual situation with the information on the nodes after the propagation is depicted in fig. 8.5.

The conflicts can be computed on node $\emptyset$ using the formula above, that is

$$\begin{aligned} Conf &\cong \bigvee_{(f \rightarrowtail \perp) \in \Xi_0} f \\ &\cong \Big((a_1 = ok) \wedge (m_1 = ok) \wedge (m_2 = ok)\Big) \\ &\quad \vee \Big((a_1 = ok) \wedge (a_2 = ok) \wedge (m_1 = ok) \wedge (m_3 = ok)\Big). \end{aligned}$$

This result is equal to the one obtained in example 7.2. ⊖

8.4 Outward Propagation

If the inward phase has been terminated and one marginal on the root node has been computed, the following theorem can be used to subsequently compute the marginals on all other nodes in the tree.

Theorem 8.3 *(Kohlas, 1997b) Let $\phi_i^{(i)}$ denote the marked statement obtained at step $i = m, m-1, \ldots, 2$ of the inward phase. Then*

$$\phi^{\downarrow L_i} = \phi^{\downarrow L_i \wedge L_{b(i)}} \oplus \phi_i^{(i)}. \tag{8.11}$$

Doing sequential computations from $i = 2$ to m, this theorem allows to compute every marginal $\phi^{\downarrow L_i}$ on the corresponding node i. According to the theorem, every node $i \neq 1$ can do the computation and send messages toward its (eventual) outward neighbors only after its inward neighbor $b(i)$ has sent the message $\widehat{\psi}_i := (\phi^{\downarrow L_{b(i)}})^{\downarrow L_i \wedge L_{b(i)}}$ and this message has been combined to the actual information $\phi_i^{(i)}$ on node i (the node $i = 1$ can start immediately because it has no inward neighbor). But $\widehat{\psi}_i = \phi^{\downarrow L_i \wedge L_{b(i)}}$, therefore $\phi^{\downarrow L_i} = \phi^{\downarrow L_i \wedge L_{b(i)}} \oplus \phi_i^{(i)}$. This message passing scheme is called the **outward phase** or **compilation** and is illustrated in fig. 8.6.

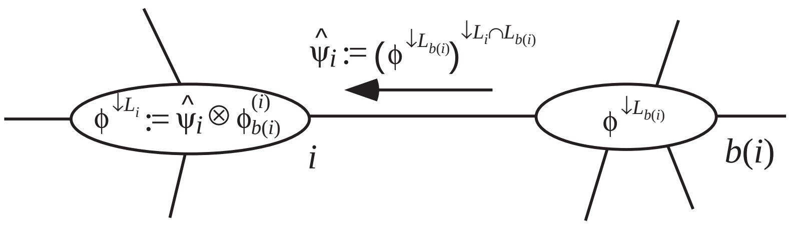
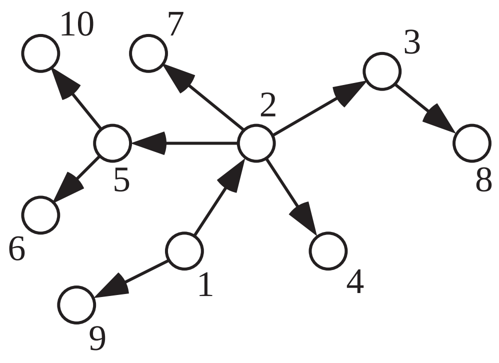

$\hat{\psi}_i := (\phi^{\downarrow L_{b(i)}})^{\downarrow L_i \cap L_{b(i)}}$
$\phi^{\downarrow L_i} := \hat{\psi}_i \otimes \phi_{b(i)}^{(i)}$
i
$\phi^{\downarrow L_{b(i)}}$
$b(i)$

Figure 8.6: Message passing during outward propagation in step number i focused on nodes i and $b(i)$.

The computations can again be represented in the Markov tree, fig. 8.7, but this time computations start at the root node and go towards the leaf nodes of the tree. If every leaf node has got the message from its inward neighbor and combined it with its actual information, then the process is stopped.



(a) An arrow denotes the direction of the message which is passed on the (undirected) arc.

i	$\leftarrow b(i)$
10	5
9	1
8	3
7	2
6	5
5	2
4	2
3	2
2	1

(b) The function b.

Figure 8.7: General message passing scheme during outward propagation towards the leafs.

Note that this theorem depends on the axiom (M9) of idempotency (see section 2.1.2). Other ideas have been developed for uncertainty calculi which do not respect this axiom, see for example (Bissig *et al.*, 1997; Pearl, 1988; Shenoy, 1995; Lepar & Shenoy, 1998; Stärk-Lepar, 1999).

8.5 Computing Marginals on Other Domains

The concept of local propagation described in the previous section allows to compute marginals on the domain of any node in the tree. Yet often this is not enough, because we are not only interested in these marginals, but in marginals with respect to other domains. So we go further. Assume that the marginal $\phi^{\downarrow L_i}$ on node i has been computed.

Depending on the domain L of the marginal we want to compute, we have to consider two cases:

- If there is a node i in the hypertree such that $L_i \subseteq L$, then, using the transitivity property (M9) from definition 2.5, we have

  $$\phi^{\downarrow L} = (\phi^{\downarrow L_i})^{\downarrow L}, \tag{8.12}$$

  and therefore the marginal can be computed on the node i. Note that the computations take place entirely within the domain of node i and are therefore feasible. If several nodes are available whose label is a superset of L, then one can select any of these nodes to compute the marginal $\phi^{\downarrow L}$. Yet in order to simplify the computations, a node with a small domain is usually a good choice.

- Otherwise $L \nsubseteq L_i$ for all nodes i in the hypertree. This means that there is no node in the actual hypertree on which the computations can take place. Therefore the actual hypertree has to be changed such that a new node is included whose domain is bigger than L. A similar problem appears in a more general context in chapter 10, where additions of new information to the hypertree are considered. The problem here can be reformulated as adding first an empty information with domain L to the hypertree (this problem is addressed in section 10.1.3), which adds a node i', and then computing $\phi^{\downarrow L} = (\phi^{\downarrow L_{i'}})^{\downarrow L}$ on the new generated node where $L \subseteq L_{i'}$.

8.6 Compilation of the Hypertree

The results above imply that if only one marginal has to be computed then the corresponding node is selected as the root node of the hypertree, and the whole inward propagation has to be done towards that node. If a second marginal is needed, then a partial outward propagation can be done from the root towards the node corresponding to the second query. This implies that usually, there is no need to do the whole inward and outward propagation, but propagation is only done “on demand”.

The other extreme, that is the computation of all marginals on all nodes, can be useful if there are a lot of queries on the same (unchanged!) knowledge base; this approach is also called the *compilation of the hypertree*.

For further information on this subject and especially a method for answering queries by negating the hypothesis (in cases where a negation is defined), that is doing only several inward propagations, see (Lehmann & Haenni, 1999; Lehmann, 2000).

9 Computing Probabilities

The structure of a probabilistic argumentation system as defined in chapter 6 can be used to weigh the symbolical argument computed for a hypothesis h. The straightforward way to do this is to consider the subspace in 2^Ω which represents the set of arguments, and to compute its weight using the probability measure P, for example $P(QSS(h))$. But usually, this is infeasible due to the size of 2^Ω. In chapter 5, we have shown how formulas in $\mathcal{FSC}$ can be used to represent subspaces of 2^Ω. So, instead of computing $QSS(h)$ explicitly, we computed a formula which is logically equivalent to its representation $qs(h)$ according to chapter 8, and the problem is then to weigh any formula in $\mathcal{FSC}$ using the measure P. In this chapter, we explain how the probability of such formulas can be computed directly, that is without computing the corresponding subspace in 2^Ω.

In the sequel, we suppose that the measure P on 2^Ω is defined as in (6.8), so we are in a case where the domains of the different set constraints are all finite. The generalization of the results of this chapter to non-finite set constraints together with a probability algebra, as for example in a generalized hint, has not yet been done.

In the first section, we show how arguments can be weighted by probabilities. In section 9.2 probabilities of FSC conjunctions are computed, and several approaches for computing the probability of FSC DNF's are presented.

Since most of the results of the reasoning process (computation of QSS,...) produces FSC DNF's, and all the other formulas in $\mathcal{FSC}$ can be transformed (often efficiently) into an equivalent FSC DNF (cf. section 5.1), we only have to specify an algorithm to compute $p(f)$ for FSC DNF's f.

9.1 Numerical Evaluation of Arguments

Consider a hypothesis ϕ in Φ. In section 3.5 and 5.5 we showed how to compute the arguments, as a subset of 2^Ω and in logical form respectively, which are supporting the hypothesis, that is the quasi-support and support of the hypothesis. Subsequently, these formulas can be weighted by the measure P. This can be done by considering the extended belief function bel_e defined in section 3.7. Note that the belief function is normalized, that is $bel_e(e) = 1$, therefore define the **degree of support** $dsp(\phi)$ of ϕ by

$$dsp(\phi) := bel_e(\phi) \tag{9.1}$$

as the strength of the proper arguments supporting the hypothesis. Sometimes, we denote by dqs the unnormalized belief function. Define the probability function

$$p(f) \quad := \quad P(N(f)) \tag{9.2}$$

for $f \in \mathcal{FSC}$. The following lemma shows how the degree of support dsp can be computed:

Lemma 9.1 *$dsp(\phi)$ can be computed by using the representation by subsets of 2^Ω as well as by using the representation by formulas in $\mathcal{FSC}$ as follows:*

$$dsp(\phi) \quad = \quad \frac{P(SS(\phi))}{P(DS)} \quad = \quad \frac{P(QSS(\phi)) - P(CS)}{1 - P(CS)} \tag{9.3}$$

$$dsp(\phi) \quad = \quad \frac{p(sp(\phi))}{p(Diag)} \quad = \quad \frac{p(qs(\phi)) - p(Conf)}{1 - p(Conf)} \tag{9.4}$$

Proof of lemma 9.1 Follows from the definition of bel_e, section 3.7 and the normalization of hints, section 3.2. □

The formulas $qs(\phi)$ and $Conf$ are usually given as FSC DNF's, therefore we have to compute probabilities of FSC DNF's. This problem is addressed in the next section.

Given the information $Conf$, we are sometimes interested in the posterior probability of an interpretation, a diagnosis, or even a more general formula in $\mathcal{FSC}$. Denote by $f \in \mathcal{FSC}$ such a formula resp. the representation of the interpretation(s). Then the posterior probability $p'(f)$ is defined by

$$\begin{aligned} p'(f) \quad &:= \quad p(f|\neg Conf) \qquad\qquad (9.5)\\ &= \quad \frac{p(f \wedge \neg Conf)}{1 - p(Conf)} \quad = \quad \frac{p(f) - p(f \wedge Conf)}{1 - p(Conf)}. \end{aligned}$$

Usually $Conf$ is an FSC DNF, then $f \wedge Conf$ can also be represented as an FSC DNF and we have to compute the probability of two FSC DNF's. Hence the computational problem is analogous to the one presented above.

Example 9.2: Computing Numerical Supports

For simplification of the notation, we abbreviate in this example $M(i, \{ok\})$ by i and $M(i, \{faulty\})$ by $\neg i$ for any component $i = a_1, a_2, m_1, m_2, m_3$, because the components have only two possible modes and can therefore be represented as binary variables. Consider the formula for *Conf* computed in example 7.2, that is

$$\mathit{Conf} \quad \cong \quad (a_1 \wedge m_1 \wedge m_2) \vee (a_1 \wedge a_2 \wedge m_1 \wedge m_3),$$

and assume the probability given there, that is

$$\begin{array}{rclcrcll} pr(m_i) & = & 0.97 & \text{and} & pr(\neg m_i) & = & 0.03 & \text{for } i = 1,2,3, \\ pr(a_j) & = & 0.95 & \text{and} & pr(\neg a_j) & = & 0.05 & \text{for } j = 1,2. \end{array}$$

Now, consider the hypothesis $h = m_2$ $(= M(m_2, \{ok\}))$ for which we want to compute $dsp(h)$. According to lemma 9.1, we have to determine $P(qs(h))$ and $P(\mathit{Conf})$. First we compute $qs(m_2)$ analogous to example 7.2:

$$qs(m_2) \quad \cong \quad m_2 \vee (a_1 \wedge a_2 \wedge m_1 \wedge m_3),$$

and for the computation of the probability, we have to split the disjunction into disjoint parts, so

$$\begin{aligned} P(qs(m_2)) &= P(m_2 \vee (a_1 \wedge a_2 \wedge m_1 \wedge m_3 \wedge \neg m_2)) \\ &= P(m_2) + P(a_1 \wedge a_2 \wedge m_1 \wedge m_3 \wedge \neg m_2) \\ &= 0.97 + 0.95^2 \cdot 0.97^2 \cdot 0.03 \\ &= 0.9955, \end{aligned}$$

and similar for $P(\mathit{Conf})$,

$$\begin{aligned} P(\mathit{Conf}) &= P((a_1 \wedge m_1 \wedge m_2) \vee (a_1 \wedge a_2 \wedge m_1 \wedge m_3 \wedge \neg m_2)) \\ &= P(a_1 \wedge m_1 \wedge m_2) + P(a_1 \wedge a_2 \wedge m_1 \wedge m_3 \wedge \neg m_2) \\ &= 0.95 \cdot 0.97^2 + 0.95^2 \cdot 0.97^2 \cdot 0.03 \\ &= 0.9193. \end{aligned}$$

The splitting of the disjunction into disjoint parts is made using an ad hoc method; in the next section we present several algorithms for this purpose.

So finally, using lemma 9.1, this leads to

$$dsp(m_2) \quad = \quad \frac{p(qs(h)) - p(\mathit{Conf})}{1 - p(\mathit{Conf})} \quad = \quad \frac{0.9955 - 0.9193}{1 - 0.9193} \quad = \quad 0.9442.$$

⊖

9.2 Probabilities of Disjunctive Normal Forms

First, we show how to compute the probability of an FSC conjunction. In the subsections 9.2.2 to 9.2.4, we present generalizations of three algorithms which have been developed for computing probabilities of formulas in propositional logic. This section follows the ideas presented in (Monney & Anrig, 1998).

9.2.1 Probabilities of Conjunctions of Set Constraints

Consider an FSC conjunction

$$co = M(c_1, X_1) \wedge \cdots \wedge M(c_p, X_p). \tag{9.6}$$

The advantage of the structure of a FSC conjunction is that its probability is very simple to compute, because, by definition, two literals in *co* do not contain information with respect to the same variable. Therefore we have

$$\begin{aligned} p(co) &:= P(N(co)) &&= P\left(N\left(\bigwedge_{i=1}^{p} M(c_i, X_i)\right)\right) \\ &= P\left(\bigcap_{i=1}^{p} N(M(c_i, X_i))\right) &&= \prod_{i=1}^{p} P\left(N(M(c_i, X_i))\right) \\ &= \prod_{i=1}^{p} P(X_i). \end{aligned} \tag{9.7}$$

However, in the case of argumentation systems (chapter 6), the sets V_i are finite, so (9.7) can be written as

$$p(co) = \prod_{i=1}^{p} \left(\sum_{x \in X_i} p(c_i = x) \right). \tag{9.8}$$

9.2.2 Inclusion-Exclusion

Consider an FSC DNF $d = co_1 \vee \cdots \vee co_m$. The well known inclusion-exclusion formula for computing probabilities of unions of events in probability theory can as well be applied to such a logical formula:

$$\begin{aligned} p(d) &= P(N(co_1 \vee \cdots \vee co_m)) = P\left(\bigcup_{i=1}^{m} N(co_i)\right) \\ &= \sum_{\emptyset \neq I \subseteq \{1,\ldots,m\}} (-1)^{|I|+1} P\left(\bigcap_{i \in I} N(co_i)\right) \\ &= \sum_{\emptyset \neq I \subseteq \{1,\ldots,m\}} (-1)^{|I|+1} P\left(N\left(\bigwedge_{i \in I} co_i\right)\right). \end{aligned}$$

A typical term of the sum in the last equation is then a conjunction c which has to be simplified to an FSC conjunction c' (see subsection 5.1.1), and its probability $p(c')$ is then easy to compute using (9.8). The big problem in this approach is that the number of terms in the sum grows exponentially with m and therefore cannot be computed for larger formulas.

9.2.3 A Generalization of the Algorithm of Abraham

Given a formula in DNF, the algorithm of Abraham (1979) is based on single variable inversion, such that the output of the algorithm is a disjoint DNF. The original algorithm is restricted to monotone formulas with binary variables. Kohlas & Monney (1995) present a generalization to non-monotone formulas with binary variables. A generalization to polytomic variables is presented in (Anrig *et al.*, 1997c) and (Monney & Anrig, 1998; Monney & Anrig, 2000). An alternative approach, using an explicit representation of the information that every component is in exactly one mode is presented in (Anrig & Monney, 1999). This second approach deals with explicit representation whereas we use here finite set constraints as implicit representation, so it will not be considered here.

In this subsection we follow (Monney & Anrig, 1998) and present a method for computing the probability $p(f)$ of a formula $f \in \mathcal{FSC}$, for example a logical representation *Conf* of *CS*. Two formulas f and g in $\mathcal{FSC}$ are called **disjoint** if $f \wedge g \cong \bot$, and in this case we write $f + g$ instead of $f \vee g$. The following lemma describes a necessary and sufficient condition for a test for disjointness in the special case of FSC conjunctions, which will be used later on:

Lemma 9.3 *(Monney & Anrig, 1998) Let* $f_1 = \bigwedge_{i \in I_1} M(c_i, X_i)$ *and* $f_2 = \bigwedge_{i \in I_2} M(c_i, Y_i)$ *be two FSC conjunctions different from* $\bot$. *Then* f_1 *and* f_2 *are disjoint if and only if there is an* $i \in I_1 \cap I_2$ *such that* $X_i \cap Y_i = \emptyset$.

The lemma inspires then the next result, which describes a method to split two not necessary disjoint FSC conjunctions into several mutually disjoint FSC conjunctions. This is the main ingredient for the generalized algorithm of Abraham:

Lemma 9.4 *(Monney & Anrig, 1998) Let* $f_1 = \bigwedge_{i \in I_1} M(c_i, X_i)$ *and* $f_2 = \bigwedge_{i \in I_2} M(c_i, Y_i)$ *be two FSC conjunctions different from* $\bot$. *For all* $i \in I_1 - I_2$ *define* $Y_i := V_i$, *and so*

$$f_2 \quad \cong \quad \bigwedge_{i \in I_1 + (I_2 - I_1)} M(c_i, Y_i),$$

and let $I := \{i \in I_1 : Y_i \not\subseteq X_i\}$. *Then,*

- *if* $I = \emptyset$, *then* $f_1 \vee f_2 \cong f_1$,
- *if* $I \neq \emptyset$, *then let* $\{i_1, \ldots, i_t\}$ *denote the set* I. *Then*

$$\begin{aligned} f_1 \vee f_2 \quad \cong \quad f_1 \quad & + \quad (f_2 \wedge M(c_{i_1}, V_{i_1} - X_{i_1})) \\ & + \quad (f_2 \wedge M(c_{i_1}, X_{i_1}) \wedge M(c_{i_2}, V_{i_2} - X_{i_2})) \\ & \vdots \\ & + \quad (f_2 \wedge M(c_{i_1}, X_{i_1}) \wedge \cdots \\ & \qquad \cdots \wedge M(c_{i_{t-1}}, X_{i_{t-1}}) \wedge M(c_{i_t}, V_{i_t} - X_{i_t})). \end{aligned}$$

Using the equivalence relations of section 5.1, the result of the lemma is an FSC DNF which is equivalent to $f_1 \vee f_2$. Now the generalized algorithm of Abraham called *Abraham** can be formalized:[1]

```
Algorithm Abraham*
(* input: an FSC DNF $f = \bigvee_{i=1}^{n} f_i$ *)
(* output: the sets $P_{i,j}$ with $j = 1, \ldots, r$ and $i = 1, \ldots, j-1$ *)
begin
  for $j = 1$ to $n$
     $P_{0,j} := \{f_j\}$
     for $i = 1$ to $j-1$
        $P_{i,j} := \emptyset$
        for all $d$ in $P_{i-1,j}$
           if $d$ and $f_i$ are disjoint then add $d$ to $P_{i,j}$
           else define $I := \{i_1, \ldots, i_t\}$ as in lemma 9.4
                                          with respect to $f_i$ and $d$
              if $I \neq \emptyset$ then add all the following formulas to $P_{i,j}$:
                 $d \wedge M(c_{i_1}, V_{i_1} - X_{i_1})$
                 $d \wedge M(c_{i_1}, X_{i_1}) \wedge M(c_{i_2}, V_{i_2} - X_{i_2})$
                 $\vdots$
                 $d \wedge M(c_{i_1}, X_{i-1}) \wedge \cdots$
                     $\cdots \wedge M(c_{i_{t-1}}, X_{i_{t-1}}) \wedge M(c_{i_t}, V_{i_t} - X_{i_t})$
end.
```

Note that lemma 9.3 can be used to test whether d and f_i are disjoint in the algorithm. The following theorem shows that the result is correct:

Theorem 9.5 *(Monney & Anrig, 1998) Let $P_{i,j}$ denote the sets generated by the algorithm Abraham* when it is applied to the FSC DNF $f = f_1 \vee \cdots \vee f_n$. Then the sets $P_{i,j}$ contain only FSC conjunctions and*

$$f \quad \cong \quad \sum_{j=1}^{n} \sum \{d \in P_{j-1,j}\}.$$

In order to compute the probability $p(f)$ of any formula f in $\mathcal{FSC}$, we have to apply the following steps:

1. Determine an FSC DNF g such that $f \cong g$.
2. Apply the algorithm *Abraham** to the FSC DNF g. This leads, by theorem 9.5, to an FSC DNF $h = d_1 + \cdots + d_r$ such that $g \cong h$.
3. Then

$$p(f) \quad = \quad \sum_{i=1}^{r} p(d_i), \tag{9.9}$$

[1]We use the same notation as in (Monney & Anrig, 1998): A star * at the end of a function or algorithm denotes the generalized version of the function or algorithm.

and the probability of every conjunction d_i, $i = 1, \ldots, r$, can be computed using (9.8).

The above is true because

$$\begin{aligned} p(f) &= P(N(f)) = P(N(h)) = P\left(N\left(\sum_{i=1}^{r} d_i\right)\right) \\ &= P\left(\bigcup_{i=1}^{r} N(d_i)\right) = \sum_{i=1}^{r} P(N(d_i)) = \sum_{i=1}^{r} p(d_i). \end{aligned}$$

Example 9.6: Application of the Algorithm

Assume that we want to compute $p(Diag)$ of the formula $Diag$ computed in example 7.3. This formula is already an FSC DNF, so we can directly apply the algorithm to the formulas

$$\begin{aligned} f_1 &= M(c_2, \{2\}) \wedge M(c_3, \{2\}), \\ f_2 &= M(c_2, \{1\}) \wedge M(c_3, \{3\}), \\ f_3 &= M(c_2, \{3\}) \wedge M(c_3, \{1\}), \\ f_4 &= M(c_2, \{4\}) \wedge M(c_3, \{4\}), \\ f_5 &= M(c_1, \{1,4\}) \wedge M(c_2, \{2,4\}) \wedge M(c_3, \{2,4\}), \\ f_6 &= M(c_1, \{1,4\}) \wedge M(c_2, \{1,3\}) \wedge M(c_3, \{1,3\}). \end{aligned}$$

So we start the algorithm:

$j = 1$: Set $P_{0,1} := \{f_1\}$.

$j = 2$: Set $P_{0,2} := \{f_2\}$ and, because f_2 is disjoint with f_1, set $P_{1,2} := P_{0,2} = \{f_2\}$.

$j = 3$: Analogous, $P_{2,3} := P_{1,3} := P_{0,3} := \{f_3\}$.

$j = 4$: Analogous, $P_{3,4} := P_{2,4} := P_{1,4} := P_{0,4} := \{f_4\}$.

$j = 5$: Set $P_{0,5} := \{f_5\}$ and $d := f_5$. d is not disjoint with f_1, so we have to define I according to lemma 9.4, that is $I = \{2, 3\}$. This means that $I \neq \emptyset$ and therefore

$$\begin{aligned} P_{1,5} &:= \left\{ \begin{array}{r} M(c_1, \{1,4\}) \wedge M(c_2, \{2,4\}) \wedge M(c_3, \{2,4\}) \\ \wedge M(c_2, \{1,3,4\}), \\ M(c_1, \{1,4\}) \wedge M(c_2, \{2,4\}) \wedge M(c_3, \{2,4\}) \\ \wedge M(c_2, \{2\}) \wedge M(c_3, \{1,3,4\}) \end{array} \right\} \\ &= \left\{ \begin{array}{l} M(c_1, \{1,4\}) \wedge M(c_2, \{4\}) \wedge M(c_3, \{2,4\}), \\ M(c_1, \{1,4\}) \wedge M(c_2, \{2\}) \wedge M(c_3, \{4\}) \end{array} \right\}. \end{aligned}$$

The second set contains the simplified conjunctions, that means FSC conjunctions. Both conjunctions in $P_{1,5}$ are disjoint with f_2 and f_3, therefore $P_{3,5} := P_{2,5} := P_{1,5}$. The second one is also disjoint with f_4, so it is put unchanged into $P_{4,5}$. The first one is not disjoint with f_4, hence $I = \{3\}$ and this leads to

$$\begin{aligned} P_{4,5} &:= \left\{ \begin{array}{l} M(c_1,\{1,4\}) \wedge M(c_2,\{2\}) \wedge M(c_3,\{4\}), \\ M(c_1,\{1,4\}) \wedge M(c_2,\{4\}) \wedge M(c_3,\{2,4\}) \wedge M(c_3,\{1,2,3\}) \end{array} \right\} \\ &= \left\{ \begin{array}{l} M(c_1,\{1,4\}) \wedge M(c_2,\{2\}) \wedge M(c_3,\{4\}), \\ M(c_1,\{1,4\}) \wedge M(c_2,\{4\}) \wedge M(c_3,\{2\}) \end{array} \right\}. \end{aligned}$$

$j = 6$: Set $P_{0,6} := \{f_6\}$. f_6 and f_1 are disjoint, therefore $P_{1,6} = P_{0,6}$. Set now $d = f_6$, then d is not disjoint with f_2, hence $I = \{2,3\}$ which leads to (already simplified)

$$P_{2,6} := \left\{ \begin{array}{l} M(c_1,\{1,4\}) \wedge M(c_2,\{3\}) \wedge M(c_3,\{1,3\}), \\ M(c_1,\{1,4\}) \wedge M(c_2,\{1\}) \wedge M(c_3,\{1\}) \end{array} \right\}.$$

The second disjunction in $P_{2,6}$ is disjoint with f_3, and therefore put unchanged into $P_{3,6}$. The first one is not disjoint, therefore $I = \{3\}$, such that

$$P_{3,6} := \left\{ \begin{array}{l} M(c_1,\{1,4\}) \wedge M(c_2,\{1\}) \wedge M(c_3,\{1\}), \\ M(c_1,\{1,4\}) \wedge M(c_2,\{3\}) \wedge M(c_3,\{3\}) \end{array} \right\}.$$

Both conjunctions in $P_{3,6}$ are disjoint with f_4 and f_5, therefore $P_{5,6} := P_{4,6} := P_{3,6}$.

Finally, the interesting sets are $P_{j-1,j}$, $j = 1,\ldots,6$, and according to theorem 9.5 we have the following disjoint FSC DNF representation of *Diag*:

$$\begin{aligned} \mathit{Diag} \;\cong\; & M(c_2,\{2\}) \wedge M(c_3,\{2\}) + M(c_2,\{1\}) \wedge M(c_3,\{3\}) \\ & + M(c_2,\{3\}) \wedge M(c_3,\{1\}) + M(c_2,\{4\}) \wedge M(c_3,\{4\}) \\ & + M(c_1,\{1,4\}) \wedge M(c_2,\{2\}) \wedge M(c_3,\{4\}) \\ & + M(c_1,\{1,4\}) \wedge M(c_2,\{4\}) \wedge M(c_3,\{2\}) \\ & + M(c_1,\{1,4\}) \wedge M(c_2,\{1\}) \wedge M(c_3,\{1\}) \\ & + M(c_1,\{1,4\}) \wedge M(c_2,\{3\}) \wedge M(c_3,\{3\}), \end{aligned}$$

and the prior probability of this formula is computed as the sum of the probabilities of the conjunctions, that is

$$\begin{aligned} p(\mathit{Diag}) &= p(M(c_2,\{2\}) \wedge M(c_3,\{2\})) + p(M(c_2,\{1\}) \wedge M(c_3,\{3\})) \\ &\quad \vdots \\ &\quad + p(M(c_1,\{1,4\}) \wedge M(c_2,\{3\}) \wedge M(c_3,\{3\})) \\ &= 0.3 \cdot 0.3 + 0.4 \cdot 0.2 + \cdots + (0.4 + 0.1) \cdot 0.2 \cdot 0.2 \\ &= 0.39. \end{aligned}$$

$\ominus$

9.2.4 A Generalization of the Algorithm of Bertschy-Monney

In contrast to the algorithm of Abraham, the algorithm of Heidtmann is based on multiple variable inversion techniques. Therefore the resulting formula is not a DNF anymore but, more general, a so-called mixed-product, a disjoint disjunction of formulas consisting of conjunctions and negated conjunctions. The original algorithm (Heidtmann, 1989) is restricted to monotone formulas with binary variables. Bertschy & Monney (1996) present a generalization to non-monotone formulas with binary variables, and a generalization to polytomic variables is presented in (Monney & Anrig, 1998).

In this subsection we follow (Monney & Anrig, 1998) and present the generalization of this algorithm. In the presentation of the algorithm, we use special formulas in $\mathcal{FSC}$ called *FSC mix-products*, which have the following form:

$$d \quad = \quad \left(\bigwedge_{k \in I_0} M(c_k, X_k) \right) \wedge \bigwedge_{s=1}^{r} \left(\neg \bigwedge_{k \in I_s} M(c_k, X_k) \right), \tag{9.10}$$

where $I_0, I_1, \ldots, I_r$ are subsets of $\{1, \ldots, n\}$ such that $I_u \cap I_v = \emptyset$ for all $u \neq v$, and $X_k \subseteq V_k$ for all $k \in \cup_{s=0}^{r} I_s$. For $s \in \{0, 1, \ldots, r\}$, the formula $\bigwedge_{k \in I_s} M(c_k, X_k)$ is an FSC conjunction in $\mathcal{FSC}$. The parameter r can be zero, in which case the FSC mix-product is simply an FSC conjunction. Given an FSC DNF f, the algorithm presented in the sequel will compute a collection of mutually disjoint FSC mix-products $d_1, \ldots, d_l$ such that

$$f \quad \cong \quad d_1 + \cdots + d_l. \tag{9.11}$$

Then it is easy to compute $p(f)$ because we have

$$p(f) \quad = \quad \sum_{i=1}^{l} p(d_i), \tag{9.12}$$

and $p(d_i)$ is easy to compute according to the following lemma.

Lemma 9.7 *(Monney & Anrig, 1998) Let d be the FSC mix-product in (9.10). Then*

$$p(d) \quad = \quad \prod_{k \in I_0} P(X_k) \cdot \prod_{s=1}^{r} \left(1 - \prod_{k \in I_s} P(X_k) \right),$$

or in the case where all domains are finite,

$$p(d) \quad = \quad \prod_{k \in I_0} \left(\sum_{x \in X_k} p(c_i = x) \right) \cdot \prod_{s=1}^{r} \left(1 - \prod_{k \in I_s} \left(\sum_{x \in X_k} p(c_i = x) \right) \right).$$

Let $f = \varphi_1 \vee \ldots \vee \varphi_r$ be an FSC DNF. First, we define the function *decompose** whose inputs are a family of disjoint FSC mix-products $m_1, \ldots, m_\ell$ and a family of FSC conjunctions $c_1, \ldots, c_s$, and its output is a collection of disjoint FSC mix-products $d_1, \ldots, d_k$ such that

$$\left(\sum_{i=1}^{\ell} m_i\right) \wedge \left(\neg \bigvee_{j=1}^{s} c_j\right) \;\cong\; \sum_{i=1}^{k} d_i. \tag{9.13}$$

This function can then be used to find the FSC mix-products $d_1 \ldots, d_l$ satisfying (9.11). Indeed, since

$$f \;\cong\; \sum_{i=1}^{r} \left(\varphi_i \wedge \left(\neg \bigvee_{k=1}^{i-1} \varphi_k\right)\right) \tag{9.14}$$

always holds, applying *decompose** with the arguments $m_1 = \varphi_i$, $\ell = 1$, and $c_1 = \varphi_1, \ldots, c_s = \varphi_{i-1}$ for every $i = 1, \ldots, r$ gives a collection R of disjoint FSC mix-products $d_1, \ldots, d_l$ such that

$$f \;\cong\; \sum_{t=1}^{l} d_t, \tag{9.15}$$

because *decompose** transforms each formula $\varphi_i \wedge \left(\neg \bigvee_{k=1}^{i-1} \varphi_k\right)$ into a sum of FSC mix-products. So the algorithm is the following:

```
Function bertschy-monney*($\varphi_1, \ldots, \varphi_r$);
(* The output is the set $R$ of disjoint FSC mix-products *)
begin
  $R := \emptyset$
  for $i = 1$ to $r$ do $R := R \cup$ decompose*($\varphi_i; \varphi_1, \ldots, \varphi_{i-1}$)
end.
```

In order to define the function *decompose** we are going to use a function called *find-disjoint**, to be defined below, whose arguments are an FSC mix-product m and an FSC conjunction c, and which returns a family of disjoint FSC mix-products $q_1, \ldots, q_n$ such that

$$m \wedge \neg c \;\cong\; \sum_{i=1}^{n} q_i. \tag{9.16}$$

To define *decompose**, note that

$$\left(\sum_{i=1}^{\ell} m_i\right) \wedge \left(\neg \bigvee_{j=1}^{s} c_j\right) \;\cong\; \sum_{i=1}^{\ell} \left(m_i \wedge \left(\neg \bigvee_{j=1}^{s} c_j\right)\right). \tag{9.17}$$

Since the terms $m_i \wedge \left(\neg \bigvee_{j=1}^{s} c_j\right)$ are disjoint, for the function *decompose** to find a representation of

$$\left(\sum_{i=1}^{\ell} m_i\right) \wedge \left(\neg \bigvee_{j=1}^{s} c_j\right) \tag{9.18}$$

as a sum of FSC mix-products, it is sufficient that it finds one for every term

$$m_i \wedge \left(\neg \bigvee_{j=1}^{s} c_j \right). \tag{9.19}$$

Yet

$$m_i \wedge \left(\neg \bigvee_{j=1}^{s} c_j \right) \cong m_i \wedge \left(\bigwedge_{j=1}^{s} \neg c_j \right) \cong (m_i \wedge \neg c_1) \wedge \left(\neg \bigvee_{j=2}^{s} c_j \right),$$

and the function *find-disjoint** can then be invoked to obtain a collection of disjoint FSC mix-products $q_{i_\nu}, \nu = 1, \ldots, n_i$ such that

$$m_i \wedge \neg c_1 \cong \sum_{\nu=1}^{n_i} q_{i_\nu}. \tag{9.20}$$

Then

$$m_i \wedge \left(\neg \bigvee_{j=1}^{s} c_j \right) \cong \left(\sum_{\nu=1}^{n_i} q_{i_\nu} \right) \wedge \left(\neg \bigvee_{j=2}^{s} c_j \right), \tag{9.21}$$

and a recursive call to the function *decompose** with arguments $q_{i_\nu}, \nu = 1, \ldots, n_i$ and $c_2, \ldots, c_s$ finally gives a representation of $m_i \wedge \left(\neg \bigvee_{j=1}^{s} c_j \right)$ as a sum of FSC mix-products. The function *decompose** is well defined, because the number of conjunctions in the recursive call is one less than in the original argument, and the basis of the recursion is reached when there is no conjunction at all as second argument, in which case *decompose** simply returns $m_1, \ldots, m_r$. Also note that if there is no mix-product at all as first argument, then *decompose** is defined to return the empty set. So we define the procedure *decompose** as follows:

```
Function decompose*($m_1, \ldots, m_\ell; c_1, \ldots, c_s$);
(* The set $R$ denotes the output of the function *)
begin
  if $\ell = 0$ then $R := \emptyset$
  else if $s = 0$ then $R := \{m_1, \ldots, m_\ell\}$
      else $R := \bigcup_{i=1}^{\ell}$ decompose*(find-disjoint*($m_i, c_1$); $c_2, \ldots, c_s$)
end.
```

Now what is left is the definition of the function *find-disjoint**. Its inputs are an FSC mix-product

$$m = \left(\bigwedge_{k \in I_0} M(c_k, X_k) \right) \wedge \bigwedge_{s=1}^{r} \left(\neg \bigwedge_{k \in I_s} M(c_k, X_k) \right) \tag{9.22}$$

and an FSC conjunction

$$c = \bigwedge_{k \in I_c} M(c_k, Y_k). \tag{9.23}$$

Note that the FSC's of the mix-product m are with respect to sets X_i and the ones from the conjunction c with respect to sets Y_i.

Its output is a family of disjoint FSC mix-products $q_1, \ldots, q_n$ such that $m \wedge \neg c = \sum_{i=1}^{n} q_i$. Let's introduce some notations. For the FSC mix-product m and the FSC conjunction c let $j(m,c)$ denote the number of sets I_s, $1 \le s \le r$, such that $I_s \cap I_c \ne \emptyset$. If $j(m,c) = 0$ then let $a = \bigwedge_{k \in I_c - I_0} M(c_k, Y_k)$. If $j(m,c) > 0$ then arbitrarily select $s_0 \in \{1, \ldots, r\}$ such that $I_c \cap I_{s_0} \ne \emptyset$ and define

$$\begin{aligned} m'' &= \left(\bigwedge_{k \in I_0} M(c_k, X_k) \right) \wedge \bigwedge_{s \ne s_0} \left(\neg \bigwedge_{k \in I_s} M(c_k, X_k) \right), \\ m_1 &= \bigwedge_{k \in I_c \cap I_{s_0}} M(c_k, X_k), \\ m_2 &= \bigwedge_{k \in I_{s_0} - I_c} M(c_k, X_k). \end{aligned}$$

Then the function *find-disjoint** is defined as follows:

```
Function find-disjoint*$(m, c)$;
(* The set $R$ denotes the output of the function *)
begin
  if already-disjoint*$(m, c)$ is true then $R := \{m\}$
  else if $j(m, c) = 0$
      then if $I_c \cap I_0 = \emptyset$ then $R := \{m \wedge \neg c\}$
           else let $\{i_1, \ldots, i_q\} = I_c \cap I_0$
                set $R := \left\{ \begin{array}{l} m \wedge \neg a, \\ m \wedge a \wedge \neg M(c_{i_1}, Y_{i_1}), \\ m \wedge a \wedge M(c_{i_1}, Y_{i_1}) \wedge \neg M(c_{i_2}, Y_{i_2}), \\ \vdots \\ m \wedge a \wedge M(c_{i_1}, Y_{i_1}) \wedge \cdots \\ \qquad \cdots \wedge M(c_{i_{q-1}}, Y_{i_{q-1}}) \wedge \neg M(c_{i_q}, Y_{i_q}) \end{array} \right\}$
      else select $s_0 \in \{1, \ldots, r\}$ such that $I_c \cap I_{s_0} \ne \emptyset$;
          $R :=$ (find-disjoint*$(m_1 \wedge \neg m_2 \wedge m'', c)) \cup \{\neg m_1 \wedge m''\}$
end.
```

The function *already-disjoint** returns true if m and c are disjoint, and false otherwise:

```
Function already-disjoint*$(m, c)$;
begin
   if $X_k \cap Y_k = \emptyset$ for every $k \in I_c \cap I_0$
   or there is $s_0 \in \{1, \ldots, r\}$ such that $I_{s_0} \subseteq I_c$
                 and $X_k \supseteq Y_k$ for every $k \in I_{s_0}$
   then return true
   else return false
end.
```

Considering the developments in this section, all that remains in order to prove that the algorithm *bertschy-monney** is correct is the proof that the function *find-disjoint** is correct.

Theorem 9.8 *(Monney & Anrig, 1998) The function find-disjoint* terminates and its output R is a set of disjoint FSC mix-products $\{q_1, \ldots, q_n\}$ such that*

$$m \wedge \neg c \quad = \quad \sum_{i=1}^{n} q_i.$$

10 Adding New Information

In some situations, it can occur that only a part of the knowledge is known at the very beginning of the decision process, but new information arrives sequentially and has to be added to the knowledge base. Suppose that between these additions of new knowledge, reasoning has to take place, and that we cannot wait with processing the data until the whole information has arrived; usually there is no criteria to tell us when the whole information has arrived. Therefore the results of the reasoning process have to be revised, updated, and are possibly not valid anymore. In the first part of this section, we suppose that the new information cannot be split into pieces, otherwise we just consider it as a sequence of new information, which will be treated strictly sequentially; in section 10.3 we will reconsider this approach.

Information can be added at several levels (see also fig. 6.1). Consider an argumentation system and the induced generalized hint, allocation of support, allocation of probability, and belief function. Then the problem of adding new information can always be interpreted as the combination of the already known information with the new one. We have already considered the combination of information in the form of

- argumentation systems: see chapter 6,
- generalized hints: see section 3.3 on the combination of generalized hints,
- (quasi-)support functions: see section 3.6 on the combination of support functions,
- allocations of probability: see section 2.4 on the combination of allocations of probability,
- belief functions: see section 2.6 on the combination of belief functions.

In this chapter we will focus on the situation where we have a knowledge base and, in addition, the corresponding minimal conflicts and minimal diagnoses are known, that is, have already been computed using the techniques presented so far. This means that some information processing has already been done, and the purpose is to re-use as much as possible of these computations. So for example consider the argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$, with the representation Ξ, and the corresponding conflicts $\mathit{Conf}(\Xi)$ and diagnoses $\mathit{Diag}(\Xi)$. New information $\mathcal{AS} = (\mathcal{FSC}, \chi', \mathcal{L}, \vdash)$ on the same language $\mathcal{FSC}$ and information system $(\mathcal{L}, \vdash)$ with representation Ξ' becomes available and has to be combined with the old one, and new conflicts and diagnoses have to be computed, denoted by $\mathit{Conf}(\Xi \cup \Xi')$ and $\mathit{Diag}(\Xi \cup \Xi')$. The idea is clearly that as much of the already computed solution, that is $\mathit{Conf}(\Xi)$ and $\mathit{Diag}(\Xi)$ should be used for the determination of $\mathit{Conf}(\Xi \cup \Xi')$ and $\mathit{Diag}(\Xi \cup \Xi')$ in order to keep the computational effort low.

In the context of a set of languages, where every language can be represented by a set of variables and the meet of two languages by the intersection of the corresponding sets of variables, we will in some situations consider a special type of additional information, a simple one called a measurement. A measurement restricts the actual values of one variable to a well specified subset (or interval) of its possible values, usually to a point.[1] For example in the argumentation system $\mathcal{AS}$ with variables, if the variable c_i is measured and has the value v_{ik}, then we assume that a simple restriction corresponding to this information is part of the language $\mathcal{L}$, and we write $\eta = \{c_i = v_{ik}\}$. By a simple restriction we mean that, for any language in S described by a set of variables V, the label of the information η is contained in V or has no intersection with V. Usually, in a system with variables, these pieces of information will be represented by finite set constraints, or more general, by set constraints. This special case occurs for example very often in an interactive procedure where new measurements are taken in order to reduce the number of diagnoses of a non-functioning system as far as possible; this will be discussed in chapter 11.

Some of the approaches presented in this chapter are restricted to specific knowledge representation, or additional knowledge (besides $\mathit{Conf}(\Xi)$ and $\mathit{Diag}(\Xi)$) is needed:

- Section 10.1 describes knowledge updating in the context of local propagation in Markov trees and is therefore an extension of chapter 8. Knowledge updating starts from a complete inward propagation in the tree, and therefore requires more information than just conflicts and diagnoses, that is, it requires that all messages having been sent during the inward propagation, as well as the resulting formulas on the nodes, are still known.

- Section 10.2 presents an approach restricted to propositional logic, using so called characteristic clauses. This approach is incremental, such that it starts from its own data structure already used for the computation of the conflicts and diagnoses so far; therefore, this approach also needs more information than just conflicts and diagnoses.

[1] We consider here only measurements with no errors involved. If errors *are* involved, the measurement is treated as a general new information.

- Another approach, presented in (Hou, 1994), is essentially based on just conflicts and diagnoses, together with the new information as input. This approach is restricted to first order logic. It computes the result mainly by means of a search strategy in a tree, together with a theorem prover.

The first approach has the advantage that also quasi-support and support, etc., can be computed at the end of the propagation algorithm.

10.1 Using the Structure of the Markov tree

Consider an argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$ and its representation Ξ. Suppose that a Markov tree for processing the information Ξ has been constructed and used to compute the minimal conflicts $\mathit{Conf}(\Xi)$ as has been described in chapter 8. So the whole inward propagation has been computed (additionally some outward propagation might have been done too), and on the root node n_r a marginal of the whole knowledge to the label of n_r is available, i.e. on n_r we have the information $\Xi^{\downarrow d(n_r)} = C^+(C^+_{d(n_r)}(\Xi))$.

Now, consider a new piece of information Ξ' arriving. This knowledge has to be added to the old one, and the conflicts and diagnoses with respect to $\Xi \cup \Xi'$ have to be computed.

Yet in the present section, we present this concept using the concept of a marked information algebra $\langle \Phi, D \rangle$ like in chapter 8. Therefore, what we are presenting can especially be applied to (L_Φ, S), (H_Φ, S), ..., $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$ as explained in chapter 8.

So consider a marked information algebra $\langle \Phi, D \rangle$ and an element $\phi \in \Phi$ which represents the actual information. Further, as defined in chapter 8, a Markov tree (E, N) has been built and a full inward propagation has been done in the tree towards a root node $n_r \in N$, that is the marginal of the combined information to the label of n_r has been computed. All messages and results on the different nodes are still available. In the context of an argumentation system, this means that the minimal conflicts and diagnoses with respect to ϕ have been computed following section 8.3.

Now a new knowledge $\eta \in \Phi$ is available. The goal is again the computation of the marginal on n_r of the knowledge. In the context of the argumentation system, we of are interested in the conflicts and diagnoses with respect to $\phi \oplus \eta$ which are denoted by $\mathit{Conf}(\phi \oplus \eta)$ and $\mathit{Diag}(\phi \oplus \eta)$.

The first and naïve idea is to throw away all structures and information processed so far, that is the Markov tree, all messages and marginals, and start from scratch with the new information $\phi \oplus \eta$ by using the same procedure as for the computation of the marginal (or minimal conflicts and diagnoses in the case of an argumentation system) with respect to ϕ (chapter 6 resp. 8). This approach is usually not very efficient. Here, we present an approach which reuses much of the information processed so far for the computations with respect to the updated knowledge base.

Consider the Markov tree (E, N) constructed for the information $\phi \in \Phi$. Depending on the label $d(\eta)$ of the new information η, we have to specify on which node η has to be put and which messages have to be sent in the tree in order to compute a new marginal of $\phi \oplus \eta$ on the root node n_r, and subsequently in the case of an argumentation system, conflicts and diagnoses can then be computed from this marginal as presented in chapter 8. Consider several different cases:

If the label of the new information η is contained in a label of one (or several) nodes of the Markov tree, then the Markov tree (E, N) does not have to be changed, and the algorithm for the computation of the new conflicts is presented in subsection 10.1.1; this case will also be treated in the special case of propositional logic in subsection 10.1.2.

If η can be decomposed (for example according to the structure of the tree), the pieces of the decomposition can be placed on several different nodes and we have the problem of adding several pieces of information at once, see subsection 10.3.

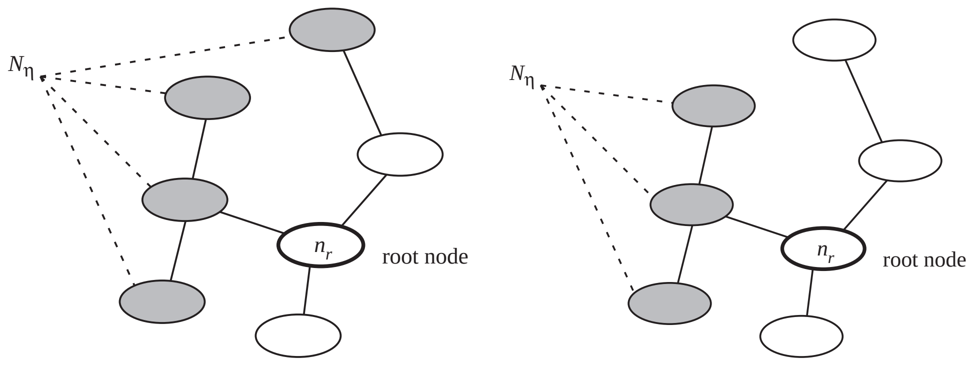
In the general case, the label of the new information η is not contained in any of the labels of the nodes in N, therefore η cannot be placed on any node of the Markov tree. The first idea is to add a new node to the Markov tree whose label is equal to the label of η, but in the general case the Markov property will not hold anymore for the resulting tree. So in this case the very structure of the tree has to be changed. Algorithms for this case are presented in subsection 10.1.3.

10.1.1 New Information Compatible with the Markov Tree

As mentioned above, in this subsection we consider new information η which is compatible with the Markov tree (E, N). This means that the label of η is contained in one or several nodes of (E, N). It then has to be incorporated in the structure of the present Markov tree. So assume that there is a set of nodes N_η, such that for every node n in N_η the label $d(\eta)$ is contained in $d(n)$. The question is now, in which of those nodes should we put the information η? The answer seems to be clear: We put the information into the node $n' \in N_\eta$ which is – in some sense – not far from the root node n_r of the previous inward propagation. Then we pursue the inward propagation from node n' to the root node n_r (using the previously computed information) and obtain finally a marginal on the new information on n_r, namely $(\phi \oplus \eta)^{\downarrow d(n_r)}$. Using this marginal, the minimal conflicts $Conf(\phi \oplus \eta)$ can then be obtained on the node n_r by computing a further marginal of the information on n_r, cf. lemma 5.14 or section 6.4. Let's formulate this more precisely. Two questions remain:

- What is the general way to choose the node $n' \in N$?
- How can we re-use as much information of the original inward propagation as possible?

Let's begin with the first question. Consider several cases:



(a) This situation cannot occur, because the tree is Markovian.

(b) This is a possible situation.

Figure 10.1: The problem of the nearest node in the set N_η.

1. If the old root n_r is contained in N_η then we clearly choose $n' = n_r$, and the new marginal $(\phi \oplus \eta)^{\downarrow d(n_r)}$ on this node is simply the old one $\phi^{\downarrow d(n_r)}$ combined with the new information η, that is
$$(\phi \oplus \eta)^{\downarrow d(n_r)} \quad = \quad \phi^{\downarrow d(n_r)} \oplus \eta, \tag{10.1}$$
because $d(\eta) \subseteq d(n_r)$.

2. If the set N_η consists of only one node, the selection problem is trivial.

3. Finally, if the set N_η consists of two or more elements, we have to choose one of them as the one on which we will put the information η. Assume that the root node n_r is not contained in N_η (otherwise we are in case 1), then, because the tree is actually a Markov tree, there is a unique node n' in N_η which is contained in every path from the old root node n_r to any node in the set $N_\eta - \{n'\}$. Consider a situation where there is no such node, that is, for example the situation depicted in fig. 10.1(a). However, this situation is not possible, because $d(\eta)$ is contained in the label of all nodes in N_η, and therefore, because the tree is Markovian, has also to be contained in the label of every node on the path from one node in the set N_η to any other node in this set; so for example $d(\eta)$ has to be contained in $d(n_r)$ and therefore $n_r \in N_\eta$, which was excluded in fig. 10.1(a). So clearly, there is a unique node n' which will be called the "nearest" node of N_η to n_r, i.e. only situations like the one in fig. 10.1(b) are possible.

Now we turn to the second question: How do we efficiently re-use the information of the inward propagation towards the old root n_r?[2] Again we have to

[2]See also p. 159 about using the information of the previous outward propagation for the special case of propositional logic.

look at the three different cases. For case 1, we have already shown how the marginal $(\phi \oplus \eta)^{\downarrow d(n_r)}$ has to be computed in (10.1), and from this marginal, the minimal conflicts and diagnoses can then be computed using techniques presented in chapter 6. The cases 3 and 4 are considered in the sequel.

Kohlas *et al.* (1999b) consider the problem of computing marginals on all nodes, therefore updating some information on one node n' induces an outward propagation on the whole Markov tree starting with the new root node n'. Yet the ideas of that article can as well be used in our case where we want to compute the new marginal on the old root node n_r and do not consider the outward propagation. New work in this area discusses the idea of replacing the outward propagation completely by inward propagation in the case where we are not interested in all marginals (Lehmann & Haenni, 1999; Kohlas *et al.*, 1999a).

Consider cases 3 and 4, i.e. assume that on the path from node n' to the root node n_r are the nodes $n_1, \ldots, n_s$ in this sequence, as shown in fig. 10.2. Denote the messages previously computed during the inward phase towards the root n_r by $\psi_{n' \to n_1}$, $\psi_{n_j \to n_{j+1}}$ for $j = 1, \ldots, s-1$, and $\psi_{n_s \to n_r}$. The inward messages towards the node n' do not have to be recomputed, because they are independent of the knowledge η which is to be put on node n'. The same is true for all the messages toward n_{j+1} besides the one from n_j, $j = 1, \ldots, s-1$, and also for all messages toward n_r besides the message from n_s. So the only information processing which has to be done is on the path from n' to n_r as shown in fig. 10.2. The new messages which will be computed in the sequel and sent along this path are denoted by an asterisk (e.g. $\psi^*_{n' \to n_1}$) in order to distinguish them from the ones already sent during the previous inward propagation. The old knowledge on node n' is denoted by $\varphi_{n'}$ and similarly on n_j by φ_{n_j}. By old we mean either the information available on this node after the inward phase, or if the outward phase has (maybe only partially) been computed, the updated information after this phase; the underlying calculus is idempotent, therefore incorporating the same information more than once does not influence the final result, but it can influence the speed of the computation, because there might be information coming from node n_1 which is sent back to it in the new message $\psi^*_{n' \to n_1}$, which is not necessary. It is therefore better to save the information present at the end of the previous inward phase on the nodes and use this information together with the new information η to compute the new message.

In the sequel, we first concentrate on the computation of the message $\psi^*_{n' \to n_1}$. The other messages $\psi^*_{n_j \to n_{j+1}}$, for $j = 2, \ldots, s$, and $\psi^*_{n_s \to n_r}$ are computed similarly. The idea is that the information previously sent to n_1 should not be sent again, therefore we would like to define the message $\psi^*_{n' \to n_1}$ as the division of the marginal by the message $\psi_{n' \to n_1}$, that is

$$\psi^*_{n' \to n_1} \overset{???}{:=} \frac{(\varphi_{n'} \oplus \eta)^{\downarrow d(n') \wedge d(n_1)}}{\psi_{n' \to n_1}},$$

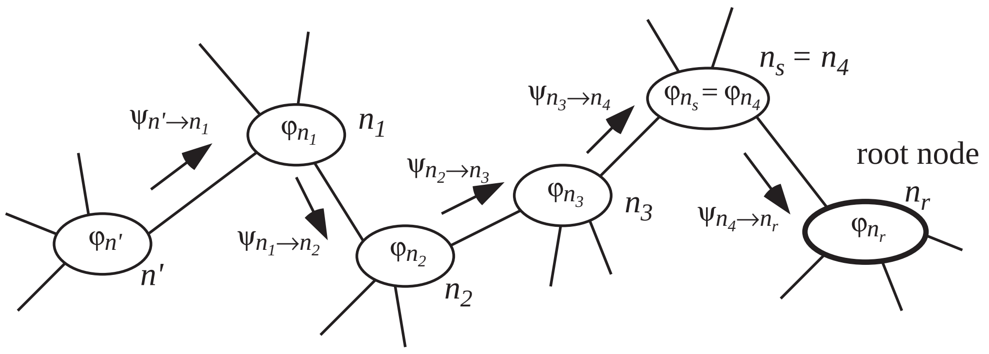


Figure 10.2: The part of the Markov tree containing the nodes between n' and n_r and the messages of the inward propagation towards n_r before the new information η has been added.

but clearly, there is no division operation in the general context. Yet written multiplicatively, the new message $\psi^*_{n'\to n_1}$ has to satisfy

$$(\varphi_{n'} \oplus \eta)^{\downarrow d(n')\wedge d(n_1)} \quad = \quad \psi^*_{n'\to n_1} \oplus \psi_{n'\to n_1}, \tag{10.2}$$

that is, the new message $\psi^*_{n'\to n_1}$ combined with the old message $\psi_{n'\to n_1}$ has to be equal to a message computed with respect to the knowledge $\varphi_{n'} \oplus \eta$ from n' to n_1.

The equation above has one trivial but computationally expensive solution, i.e. define the new message $\psi^*_{n'\to n_1}$ to be the equal to the marginal on the left-hand side, but this in fact would mean that the new messages contain all the information of the previous messages, and therefore the information concerning only $\varphi_{n'}$ is sent again through the tree and not only the updated part of it. For the general case, there is no better solution, but for special cases, better solutions exist. As an example, we consider the special case of propositional logic in the next subsection.

Example 10.1: Continuation of example 8.2

Consider the Markov tree from fig. 8.5, where the conflicts with respect to the actual knowledge Ξ have been computed. As further information, we are told that the variable x is measured to have the value 6, which is represented in the present framework as $\top \rightarrowtail (x = 6)$. This information must now be included into the Markov tree. There are several possible nodes, where this information fits, namely the four nodes XYZ, FXY, ACX, and CXZ. Following the considerations above, we put the information $\top \rightarrowtail (x = 6)$ on node XYZ because this is the nearest node from the root $\emptyset$ among the four. Due to the structure of the tree, the only message which has to be recomputed is the one from XYZ to $\emptyset$, $m^*_{XYZ\to\emptyset}$, which we will do in the present case by simply computing the marginal of the combined information, that is

$$m^*_{XYZ\to\emptyset} \quad = \quad (\Xi_{XYZ} \cup \{\top \rightarrowtail (x = 6)\})^{\downarrow\{\bot\}}$$

$$
\begin{aligned}
&= C^{+}_{\{\bot\}}(\Xi_{XYZ} \cup \{\top \rightarrowtail (x=6)\}) \\
&= \left\{ \begin{array}{rcc} (a_1 = ok) \wedge (m_2 = ok) & \rightarrowtail & \bot \\ (a_1 = ok) \wedge (a_2 = ok) \wedge (m_2 = ok) & \rightarrowtail & \bot \\ (a_1 = ok) \wedge (a_2 = ok) \wedge (m_3 = ok) & \rightarrowtail & \bot \end{array} \right\},
\end{aligned}
$$

where the marginal is computed as in example 8.2. On node $\emptyset$ we have now

$$
\begin{aligned}
\Xi_\emptyset &:= \Xi_\emptyset \cup m^*_{XYZ\to\emptyset} \\
&= \left\{ \begin{array}{rcc} (a_1 = ok) \wedge (m_2 = ok) & \rightarrowtail & \bot \\ (a_1 = ok) \wedge (a_2 = ok) \wedge (m_3 = ok) & \rightarrowtail & \bot \end{array} \right\}
\end{aligned}
$$

where the subsumed formulas have been omitted. The situation after the propagation is depicted in fig. 10.3.

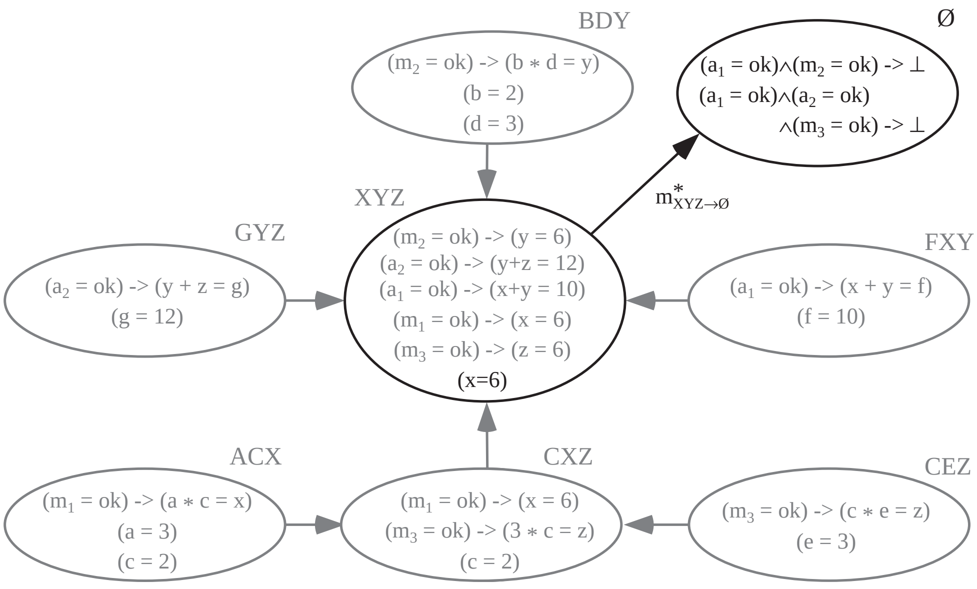


Figure 10.3: The updated Markov tree

According to (8.9), the representation of the conflict *Conf* is then

$$
\begin{aligned}
Conf &\cong \bigvee_{(f \rightarrowtail \bot)\in\Xi_\emptyset} f \\
&= \Big((a_1 = ok) \wedge (m_2 = ok)\Big) \vee \Big((a_1 = ok) \wedge (a_2 = ok) \wedge (m_3 = ok)\Big).
\end{aligned}
$$

The set of minimal conflicts $\mu Conf$ (cf. section 5.4) can be obtained on the node $\emptyset$ by first computing the prime implicates of the result (cf. for example (Birkhoff, 1948; Haenni, 1996)), and then applying the theorem of Reiter & De Kleer (1987), such that finally we get

$$\{(a_1 = ok) \wedge (m_2 = ok),\ (a_1 = ok) \wedge (a_2 = ok) \wedge (m_3 = ok)\}.$$

The set of minimal diagnoses $\mu Diag$ is therefore

$$\{(a_1 \neq ok),\ (a_2 \neq ok) \wedge (m_2 \neq ok),\ (m_2 \neq ok) \wedge (m_3 \neq ok)\}.$$

$\ominus$

10.1.2 Special Case: Propositional Logic

In this subsection, we present the ideas of (Kohlas *et al.*, 1999b) which allow to compute the new message $\psi^*_{n' \to n_1}$ more efficiently in the special case of propositional logic. This approach uses an incremental process for the computation of this message. Note that it can also be generalized to finite set constraints.

Consider the lattice S of propositional languages $\mathcal{L}_p$ over subsets of variables p of the set of variables. As an abbreviation, we denote the language $\mathcal{L}_p$ by p, such that $\mathcal{L}_p \wedge \mathcal{L}_{p'}$ is denoted by $p \cap p'$, etc. The notation from the previous subsection is used here in the special case of propositional logic.

Assume that the marginalization of information on n' to $d(n') \cap d(n_1)$ requires the deletion of a propositional symbol[3] p, and define

$$\begin{aligned}
\Sigma_+^{n'} &= \text{the set of clauses in } \varphi_{n'} \text{ containing } p \text{ positive,} \\
\Sigma_-^{n'} &= \text{the set of clauses in } \varphi_{n'} \text{ containing } p \text{ negative,} \\
\Sigma_0^{n'} &= \text{the set of clauses in } \varphi_{n'} \text{ not containing } p \text{ at all,} \\
\Sigma_+^{\eta} &= \text{the set of clauses in } \eta \text{ containing } p \text{ positive,} \\
\Sigma_-^{\eta} &= \text{the set of clauses in } \eta \text{ containing } p \text{ negative,} \\
\Sigma_0^{\eta} &= \text{the set of clauses in } \eta \text{ not containing } p \text{ at all.}
\end{aligned} \tag{10.3}$$

The new clauses are computed,

$$\begin{aligned}
\Sigma_+^{n' \to n_1} &= \mu(\Sigma_+^{\eta} \cup \Sigma_+^{n'} \cup \Sigma_0^{n'} \cup \Sigma_0^{\eta}) - (\Sigma_+^{n'} \cup \Sigma_0^{n'} \cup \Sigma_0^{\eta}), \\
\Sigma_-^{n' \to n_1} &= \mu(\Sigma_-^{\eta} \cup \Sigma_-^{n'} \cup \Sigma_0^{n'} \cup \Sigma_0^{\eta}) - (\Sigma_-^{n'} \cup \Sigma_0^{n'} \cup \Sigma_0^{\eta}),
\end{aligned}$$

and the new message can be defined as

$$\psi^*_{n' \to n_1} = \mu \left(\begin{array}{c} \Sigma_0^{\eta} \cup \{\rho(\xi, \xi') : \xi \in \Sigma_+^{n' \to n_1},\ \xi' \in \Sigma_-^{n' \to n_1}\} \\ \cup \{\rho(\xi, \xi') : \xi \in \Sigma_+^{n' \to n_1},\ \xi' \in \Sigma_-^{n'}\} \\ \cup \{\rho(\xi, \xi') : \xi \in \Sigma_+^{n'},\ \xi' \in \Sigma_-^{n' \to n_1}\} \end{array} \right). \tag{10.4}$$

A clause c subsumes a clause c' if $c = c' \vee c''$ for another clause c'' and the operator μ removes all subsumed clauses from a set of clauses. ρ denotes the so called resolution: for two clauses c and d with $p \in c$ and $\neg p \in d$, that is $c = p \vee c'$ and $d = \neg p \vee d'$ where c' and d' are clauses, the resolvent $\rho(c, c')$ is the concatenation of c' and d' where multiple occurrences of literals have been removed. $\rho(c, d) = \top$ if there is another literal $p' \neq p$ with $p' \in c$ and $\neg p' \in d$ (or $p' \in d$ and $\neg p' \in c$).

[3] See example 10.3 for how to delete more than one propositional symbol.

Consider the special case where the new information η has a very simple form, that is a literal, so either a positive or negative variable, therefore in the algorithm the expression "the set of formulas in the new information..." refers to either only $\{\eta\}$ or the empty set. So the formulas can be simplified, because only one of the sets Σ_+^η, Σ_-^η and Σ_0^η is non-empty, and the message $\psi^*_{n'\to n_1}$ usually can be computed rather easily despite the complicated formalism used above. Yet in the algorithm we stated it more generally, because the message $\psi^*_{n'\to n_1}$ is sent to the node n_1, on which we again have to compute a new message $\psi^*_{n_1\to n_2}$ to node n_2, depending on the new information on n_1. This new information on n_1, i.e. the message $\psi^*_{n'\to n_1}$, is in general a *set* of clauses, and therefore we do not have another special case on this node, but a more general updating of information. Yet the algorithm presented above is general enough to work with these structures, i.e. it can compute a new message $\psi^*_{n_1\to n_2}$ depending on the information φ_{n_1} and the new information $\psi^*_{n'\to n_1}$ (see also the example 10.3 at the end of this subsection). The other messages $\psi^*_{n_j\to n_{j+1}}$, for $j = 2, \dots, s-1$, and $\psi^*_{n_s\to n_r}$ are computed similarly.

Lemma 10.2 *The message $\psi^*_{n'\to n_1}$ defined by (10.4) fulfills equation (10.2).*

Proof of lemma 10.2 We have to proof that

$$(\varphi_{n'} \oplus \eta)^{\downarrow d(n')\cap d(n_1)} \quad = \quad \psi^*_{n'\to n_1} \oplus \psi_{n'\to n_1}.$$

Using theorem 8.1, the right-hand side develops into

$$\psi^*_{n'\to n_1} \oplus \psi_{n'\to n_1} \quad = \quad \psi^*_{n'\to n_1} \oplus \varphi_{n'}{}^{\downarrow d(n')\cap d(n_1)}.$$

Using (10.4) and the definition of marginalization, we have

$$\begin{aligned} \psi^*_{n'\to n_1} \quad &= \quad \mu \left(\begin{array}{c} \Sigma_0^\eta \cup \{\rho(\xi,\xi') : \xi \in \Sigma_+^{n'\to n_1},\ \xi' \in \Sigma_-^{n'\to n_1}\} \\ \cup \{\rho(\xi,\xi') : \xi \in \Sigma_+^{n'\to n_1},\ \xi' \in \Sigma_-^{n'}\} \\ \cup \{\rho(\xi,\xi') : \xi \in \Sigma_+^{n'},\ \xi' \in \Sigma_-^{n'\to n_1}\} \end{array} \right), \\ \varphi_{n'}{}^{\downarrow d(n')\cap d(n_1)} \quad &= \quad \Sigma_0^{n'} \cup \{\rho(\xi,\xi') : \xi \in \Sigma_+^{n'},\ \xi' \in \Sigma_-^{n'}\}; \end{aligned}$$

and, combining these equations,

$$\begin{aligned} &\psi^*_{n'\to n_1} \oplus \varphi_{n'}{}^{\downarrow d(n')\cap d(n_1)} \\ &= \mu\left(\Sigma_0^\eta \cup \Sigma_0^{n'} \cup \{\rho(\xi,\xi') : \xi \in \Sigma_+^{n'} \cup \Sigma_+^{n'\to n_+},\ \xi' \in \Sigma_-^{n'} \cup \Sigma_-^{n'\to n_1}\}\right), \end{aligned}$$

which, in fact, is the same as $(\varphi_{n'} \oplus \eta)^{\downarrow d(n')\cap d(n_1)}$ up to eliminated subsumed clauses. □

This lemma drastically reduces the computation on the Markov tree. After choosing the node $n' \in N$ where the new information η is added, we simply have to compute the messages $\psi^*_{\cdot\to\cdot}$ on the path from that node to the root node and update the information on the nodes in-between.

The same ideas can clearly also be applied if parts or the whole outward propagation have already been done before the new information η has been added. In this case, if we are only interested in the minimal contradictions and the minimal diagnoses, all we have to do is to look for the nearest node from n' on which the outward propagation has been done (this node clearly is unique) and to consider this node as new root node. The new "inward propagation" of the updated knowledge has then to be done only towards this new root node. If afterwards there is also a new "outward propagation" from this new root node, then the already computed messages on the respective paths can be re-used as well by applying the ideas presented above.

Example 10.3: Continuation of example 7.1

The information expressed in example 7.1 can very well be express in propositional logic. Assume that the variable m_i represents the state of component i, that is inverter i is working correctly if m_i is true. Further, assume that the mapping of the argumentation system ($\rightarrowtail$) is considered here as a logical implication. Then, the knowledge of the example is represented by the formulas

$$\begin{array}{rcl} m_1 & \rightarrow & (x \leftrightarrow \neg in) \\ m_2 & \rightarrow & (y \leftrightarrow \neg x) \\ m_3 & \rightarrow & (out \leftrightarrow \neg y) \\ in & & \\ out & & \end{array}$$

We usually use the corresponding conjunctive normal form to represent this information, that is

$$\Xi = \left\{ \begin{array}{l} \neg m_1 \vee x \vee in,\ \neg m_1 \vee \neg x \vee \neg in, \\ \neg m_2 \vee y \vee x,\ \neg m_2 \vee \neg y \vee \neg x, \\ \neg m_3 \vee out \vee y,\ \neg m_3 \vee \neg out \vee \neg y, \\ in,\ out. \end{array} \right\} \tag{10.5}$$

We consider the set of literal $\{x, y, in, out\}$ for the construction of the Markov tree, and therefore we get a Markov tree with three nodes, see fig. 10.4.

For the computation of the conflicts, we add a node with an empty label to one of the three nodes, like in the previous subsection. For the inward computation, first the message from A to B is computed using variable elimination according to (10.4):

$$\begin{array}{rcl} m_{A\rightarrow B} & = & \{\neg m_1 \vee x \vee in,\ \neg m_1 \vee \neg x \vee \neg in,\ in\}^{-in} \\ & = & \{\neg m_1 \vee \neg x\} \end{array}$$

and similar the ones from B to C and from C to $\emptyset$:

$$\begin{array}{rcl} m_{B\rightarrow C} & = & (\{\neg m_2 \vee y \vee x,\ \neg m_2 \vee \neg y \vee \neg x\} \cup m_{A\rightarrow B})^{-x} \\ & = & \{\neg m_1 \vee \neg m_2 \vee y\} \\ m_{C\rightarrow \emptyset} & = & (\{\neg m_3 \vee out \vee y,\ \neg m_3 \vee \neg out \vee \neg y,\ out\} \cup m_{B\rightarrow C})^{-y,out} \\ & = & \{\neg m_1 \vee \neg m_2 \vee \neg m_3\} \end{array}$$

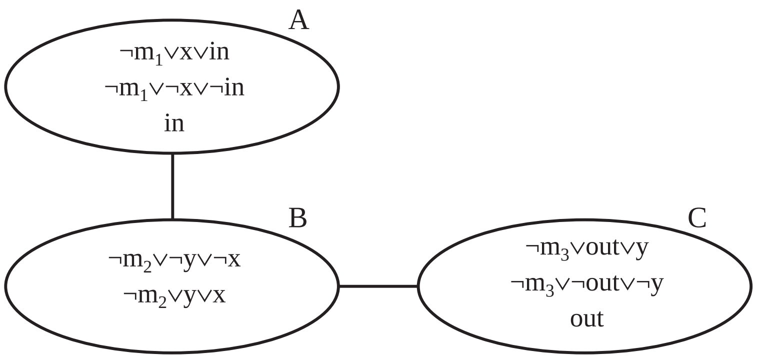


Figure 10.4: A Markov tree for the example 10.3.

So on the empty node, the information is $\{\neg m_1 \vee \neg m_2 \vee \neg m_3\}$. After the complete inward propagation on the Markov tree towards the additional empty node, the situation looks like in fig. 10.5.

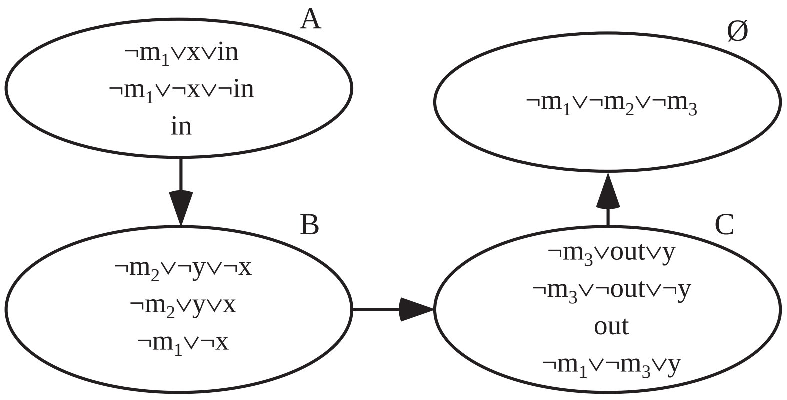


Figure 10.5: Inward propagation towards the node with empty label.

Now assume that further information about the system becomes available, namely that x is true, $\eta = \{x\}$. This information can be added to node A or B but because B is nearer to the root node $\emptyset$, we choose B. According to the algorithm described above and (10.4) we compute the new message with the literal x to be eliminated, $\psi^*_{B\to C}$ using the sets defined in (10.3)

$$\begin{aligned} \Sigma_+^B &= \{\neg m_2 \vee y \vee x\}, & \Sigma_+^\eta &= \{x\}, \\ \Sigma_-^B &= \{\neg m_2 \vee \neg y \vee \neg x, \neg m_1 \vee \neg x\}, & \Sigma_-^\eta &= \emptyset, \\ \Sigma_0^B &= \emptyset, & \Sigma_0^\eta &= \emptyset, \end{aligned}$$

and therefore

$$\Sigma_+^{B\to C} = \{x\}, \qquad \Sigma_-^{B\to C} = \emptyset.$$

The new message to be sent is finally

$$\psi^*_{B\to C} \quad = \quad \{\neg m_2 \vee \neg y,\, \neg m_1\},$$

and this information denoted by η_1 is considered as the new information on node C. A new message $\psi^*_{C\to\emptyset}$ depending on the "old" information on node C, i.e. the four clauses

$$\begin{array}{ll} \neg m_3 \vee \mathit{out} \vee y, & \mathit{out}, \\ \neg m_3 \vee \neg \mathit{out} \vee \neg y, & \neg m_1 \vee \neg m_2 \vee y, \end{array} \tag{10.6}$$

and the "new" information η_1 has to be sent now. Therefore two atoms y and *out* have to be eliminated; we first consider the elimination of y and compute therefore

$$\begin{array}{lclclcl} \Sigma^C_+ &=& \{\neg m_3 \vee \mathit{out} \vee y,\, \neg m_1 \vee \neg m_2 \vee y\}, & \quad & \Sigma^{\eta_1}_+ &=& \emptyset, \\ \Sigma^C_- &=& \{\neg m_3 \vee \neg \mathit{out} \vee \neg y\}, && \Sigma^{\eta_1}_- &=& \{\neg m_2 \vee \neg y\}, \\ \Sigma^C_0 &=& \{\mathit{out}\}, && \Sigma^{\eta_1}_0 &=& \{\neg m_1\}, \end{array}$$

and

$$\Sigma^{C\to\emptyset}_+ \;=\; \emptyset, \qquad \Sigma^{C\to\emptyset}_- \;=\; \{\neg m_2 \vee \neg y,\, \neg m_3 \vee \neg y\}.$$

This gives the intermediate result computed by (10.4):

$$\hat\psi \quad = \quad \{\neg m_1,\, \neg m_3 \vee \mathit{out}\}.$$

$\hat\psi$ contains still the atom *out* which has to be eliminated. Applying (and slightly generalizing) the ideas from the algorithm above, we compute the new set (marked with a hat ˆ),

$$\begin{array}{lcllcl} \hat\Sigma^C_+ &=& \text{the set of clauses in } C \text{ containing } \mathit{out} \text{ positive} && \\ && \text{and not containing } y \text{ at all} &=& \{\mathit{out}\} \\ \hat\Sigma^C_- &=& \text{the set of clauses in } C \text{ containing } \mathit{out} \text{ negative} && \\ && \text{and not containing } y \text{ at all} &=& \emptyset, \\ \hat\Sigma^C_0 &=& \text{the set of clauses in } C \text{ containing} && \\ && \text{neither } \mathit{out} \text{ nor } y &=& \{\neg m_1\}, \\ \hat\Sigma^{\eta_1}_+ &=& \text{the set of clauses in } \hat\psi \text{ containing } \mathit{out} \text{ positive} &=& \{\neg m_3 \vee \mathit{out}\}, \\ \hat\Sigma^{\eta_1}_- &=& \text{the set of clauses in } \hat\psi \text{ containing } \mathit{out} \text{ negative} &=& \emptyset, \\ \hat\Sigma^{\eta_1}_0 &=& \text{the set of clauses in } \hat\psi \text{ not containing } \mathit{out} \text{ at all} &=& \{\neg m_1\} \end{array}$$

and using the definitions of the algorithm,

$$\hat\Sigma^{C\to\emptyset}_+ = \emptyset, \qquad \hat\Sigma^{C\to\emptyset}_- = \emptyset.$$

This finally delivers the message computed by (10.4)

$$\psi^*_{C\to\emptyset} \quad = \quad \{\neg m_1\},$$

which is sent to the node $\emptyset$ and added to the actually present knowledge on it. Therefore on node $\emptyset$ the actual information is $\Xi_\emptyset = \{\neg m_1 \vee \neg m_2 \vee \neg m_3\} \cup \{\neg m_1\}$, where the first element is subsumed by the second one and can therefore be eliminated. Fig. 10.6 shows the result of the propagation on the Markov tree towards the root node $\emptyset$.

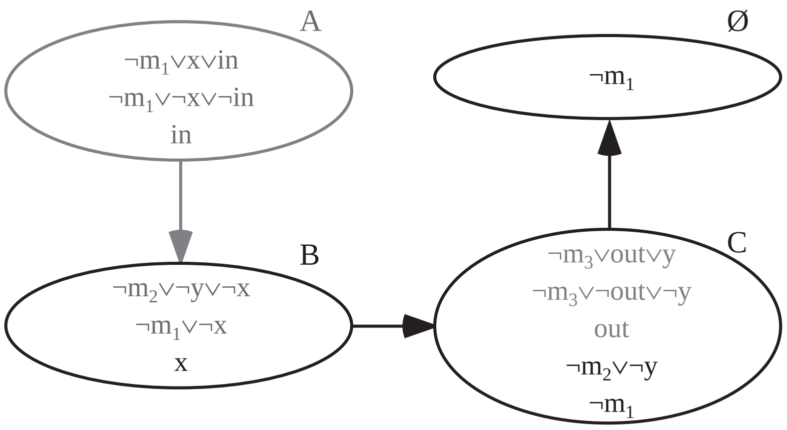


Figure 10.6: The information on the Markov tree after the updating.

So the conflict is

$$Conf(\Xi \cup \eta) = \bigvee_{f \in \Xi_\emptyset} \neg f = m_1.$$

The minimal conflicts $\mu Conf(\Xi \cup \eta)$ can be obtained on the node $\emptyset$ by computing the prime implicates of the result (cf. for example (Birkhoff, 1948; Haenni, 1996)), and applying the theorem of Reiter & De Kleer (1987), which in this simple is trivial, such that finally we get

$$\mu Conf(\Xi \cup \eta) \quad = \quad \{m_1\}.$$

The only minimal diagnosis (with respect to $\Xi \cup \eta$) is $\neg m_1$, i.e. the first multiplier is not working correctly. ⊖

10.1.3 Changing the Structure of the Markov Tree

Now we are going to consider the case where the label $d(\eta)$ of the new information η is not contained within the label of any node in the Markov tree (E, N). This means that η cannot be put on any node in the tree. Nevertheless, we have to take into account the new information η, therefore the previous structure of the information representation, i.e. the Markov tree, has to be changed. There are several possible strategies we can consider, e.g.

- Change only "small parts" of the original Markov tree, such that "a lot" of already computed messages and marginals can be re-used.

- Minimize the size of the labels of newly generated nodes in order to get a good Markov tree.

- Minimize the computations for the determination of the new Markov tree.

We present several approaches, which fulfill some of these optimization issues.

Rebuild the Whole Markov Tree

A first and straightforward approach is to throw away the present Markov tree and build a new one for the knowledge $\phi \oplus \eta$. This results in loosing all messages and all marginals on nodes computed so far, therefore wasting a lot of computation. On the other hand, this possibly generates a Markov tree which is strictly better than any other constructed by the methods presented hereafter.

In some special cases, this approach might be a good strategy, especially when the Markov tree is very small or when a lot of new information with various big labels arrives.

Xu's Modification Method

Xu (1995) presented a method for computing marginals whose labels are not contained within a node in the Markov tree. It is mainly a method to modify the tree, in such a way that finally the marginal can be computed on a newly generated node in the tree. Generalizing this approach, we can apply it to our problem.

The interesting part of the label of the new information, that is the part of the sublanguages already occurring in the Markov tree, is denoted by

$$d' = d(\eta) \wedge \left(\bigvee_{n \in N} d(n) \right). \tag{10.7}$$

Then the following pseudo-algorithm will generate a new Markov tree which contains a node whose label does include $d(\eta)$.

Pseudo-Algorithm *Xu*

```
(* Input: Markov tree $(E, N)$, new information $\eta$ *)
(* Output: Markov tree $(E', N')$ *)
1. Find a set $b = \{E_1^\eta, \dots, E_k^\eta\} \subseteq N$ such that $d' \subseteq \bigvee_{i=1}^{k} d(E_i^\eta)$.
2. Build a set $a$ which contains $b$ as well as all nodes on the paths
   between the nodes in $b$.
3. Let $N' := N \cup \{n_\eta\}$ where $n_\eta$ is a new node with
       $d(n_\eta) = d(\eta) \vee \bigvee_{X_i, X_j \in a, (X_i, X_j) \in E} (d(X_i) \wedge d(X_j))$.
4. Remove all edges among the nodes in $a$ and add the edges between
   $n_\eta$ and the nodes in $a$, i.e.
       $E' := (E - \{(X_i, X_j) : X_i, X_j \in a\}) \cup \{(n_\eta, X_i) : X_i \in a\}$.
```

Example 10.4: (Xu, 1995)

Consider a system with variables where the sublanguages are characterized by sets of variables (e.g. $\{a, b\}$) and the intersection of two languages (e.g. $\{a, b\}$ and $\{b, v\}$) is a language characterized by the intersection of the variables (e.g. $\{b\} = \{a, b\} \cap \{b, c\}$). Let (N, E) be a Markov tree with

$$N = \{\{a, b\}, \{b, c, d\}, \{d, g\}, \{c, d, e\}, \{d, e, f\}, \{c, e, h\}\}$$

and E as in fig. 10.7(a). Now, an new information η with label $\{a, g, i\}$ has to be added. According to the steps of the pseudo-algorithm of Xu,

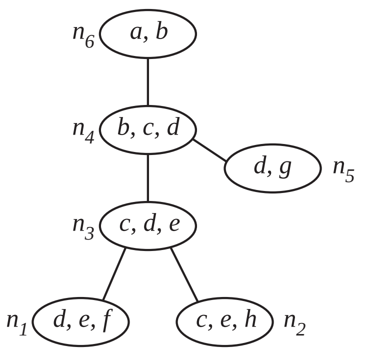
$$\begin{aligned} b &= \{\{a, b\}, \{d, g\}\}, \\ a &= \{\{a, b\}, \{b, c, d\}, \{d, g\}\}, \end{aligned}$$

and this gives us the label of the new node n_7

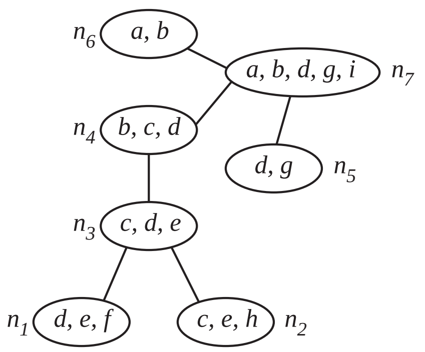
$$\begin{aligned} d(n_7) &= \{a, g, i\} \cup \Big(\{a, b\} \cap \{b, c, d\}\Big) \cup \Big(\{b, c, d\} \cap \{d, g\}\Big) \\ &= \{a, b, d, g, i\}. \end{aligned}$$

The updated Markov tree $(N \cup \{n_7\}, E')$ is shown in fig. 10.7(b). $\ominus$

Xu also shows that this pseudo-algorithm indeed yields a Markov tree and that there is always a smallest possible set a which can be chosen in step 2 of the algorithm so as to get the smallest possible label $d(n_\eta)$. Note that this does not imply that the generated Markov tree is optimal, i.e. there might be a strictly "better" tree for the combined knowledge $\phi \oplus \eta$.



(a) Original Markov tree (N, E).



(b) Modified Markov Tree $(N \cup \{n_7\}, E')$.

Figure 10.7: Example of Markov tree modification.

Lemma 10.5 *(Xu, 1995) For the algorithm of Xu applied to the Markov tree (N, E) and the new information η we have:*

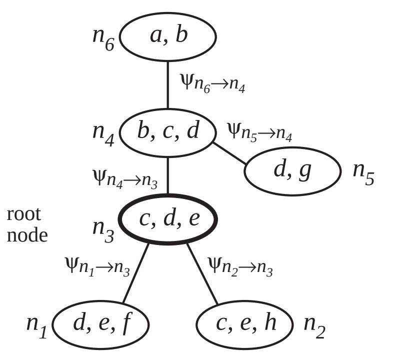
a) The resulting tree (N', E') is a Markov tree.

b) There is a node $n_\eta \in N'$ satisfying $d(\eta) \subseteq d(n_\eta)$.

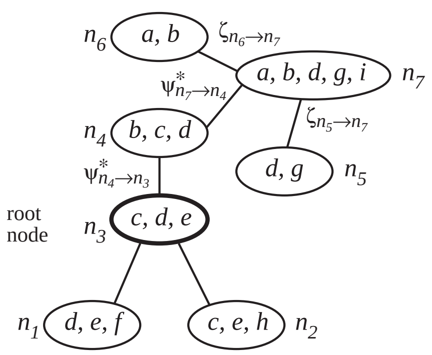
The marginals and the messages computed so far in the original Markov tree can partially be re-used in the resulting tree. If the root of the inward propagation is the same as before, n_η takes the newly computed messages of its outward neighbor nodes and sends the new inward message in direction of the root node. So the messages which have to be re-computed are the ones coming to the new node n_η from its neighbor and the ones on the path from n_η to the root node. The first ones are usually easy to compute if the intermediate results of the computations of the original messages are kept in memory, because the label of the old message is a subset of the new messages (which is clearly a subset of the label of the node itself). The messages on the path from n_η to the root node can be computed using the techniques presented in section 10.1. These computations are illustrated by an example.

Example 10.6: Continuing example 10.4

Fig. 10.8(a) shows the messages $\psi_{\cdot\to\cdot}$ computed on the original Markov tree (N, E) from fig. 10.7(a) during the inward phase towards the root node n_3. After the new information with label $\{a, g, i\}$ has been placed on the newly created node n_7 (see fig. 10.7(b)), fig. 10.8(b) shows where the updated messages have to be sent. The new messages to n_7 from its outward neighbors are denoted by $\zeta_{n_5 \to n_7}$ and $\zeta_{n_6 \to n_7}$, the new messages computed using the techniques presented in section 10.1 from n_7 towards the root node n_3 by $\psi^*_{n_7 \to n_4}$ and $\psi^*_{n_4 \to n_3}$. ⊖



(a) The messages $\psi_{\cdot \to \cdot}$ which have been sent on the original Markov tree (N, E).



(b) The new messages which have to be sent on the modified Markov tree $(N \cup \{n_7\}, E')$.

Figure 10.8: Example of where new messages have to be sent when the Markov tree has been changed according to the algorithm of Xu.

An improved method

Taking into account the ideas of Xu presented above, we define a pseudo-algorithm which computes a Markov tree which is better than (or a least equal to) the one generated by the algorithm of Xu. The idea is to make only local changes to the Markov Tree, but, in contrast to the algorithm of Xu, we also allow nodes of the actual Markov tree to be changed.

Define d' as in (10.7). The result of the following pseudo-algorithm is the Markov tree (E', N'):

Pseudo-Algorithm *Improved Version*

```
(* Input: Markov tree $(E, N)$, new information $\eta$ *)
(* Output: Markov tree $(E', N')$ *)

1. Find a set $b = \{E^\eta_1, \dots, E^\eta_k\} \subseteq N$ such that $d' \subseteq \bigvee_{i=1}^{k} d(E^\eta_i)$.
2. Build a set $a$ which contains $b$ as well as all nodes on the paths
   between the nodes in $b$.
3. Create a new node $n_\eta$ with $d(n_\eta) = d(\eta)$.
4. Build a set $c$ which contains $a$ as well as all neighbors of elements
   of $a$ and also $n_\eta$.
5. Compute a new Markov tree $(E^*, N^*)$ of the elements in $c$.
6. Let $N'$ be the union of the elements of $N - c$ and $N^*$ where subsumed
   neighbor nodes are (eventually) eliminated, i.e. "devoured" by the
   subsuming neighbor nodes, which also take over the corresponding
   edges, therefore let $E'$ be defined according to this.
```

A node a is subsumed by a node b if $d(a) \subseteq d(b)$. Formally, the last step of the pseudo-algorithm can be written as

```
set $done := false$
set $N' := (N - c) \cup N^*$
set $E' := E^* \cup \{(n, n') \in E : n, n' \notin c\}$
while $done = false$ do
begin
  if $n_1, n_2 \in N'$ with $d(n_1) \subseteq d(n_2)$ and $(n_1, n_2) \in E$ or $(n_2, n_1) \in E$
  then set $N' := N' - \{n_1\}$ and
              $E' := \{(n, n') \in E' : n \neq n_1 \neq n'\} \cup \{(n, n_2) : (n, n_1) \in E'\}$
                  $\cup \{(n_2, n') : (n_1, n') \in E'\}$
  else set $done := true$
end.
```

The resulting tree has the desired structure as mentioned in the corollary which follows from lemma 10.5 and (Xu, 1995):

Corollary 10.7 *Applying the improved pseudo-algorithm above to the Markov tree (N, E) and the label $d(\eta)$ of the new information η we have:*

a) The resulting tree (N', E') is a Markov tree.

b) For every node $n \in N - c$ there is a node $n' \in N'$ with $d(n) \subseteq d(n')$.

c) There is a node $n' \in N'$ satisfying $d(\eta) \subseteq d(n')$.

Example 10.8:

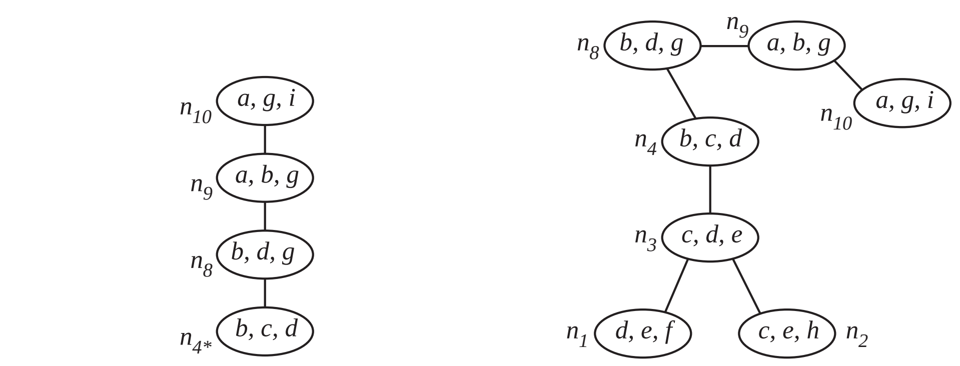
We re-use example 10.4. So consider the Markov tree in fig. 10.7(a), to which should be added the information η with label $\{a, g, i\}$. The sets computed in the algorithm are:

$$\begin{aligned} b &= \{\{a, b\}, \{d, g\}\} \\ a &= \{\{a, b\}, \{b, c, d\}, \{d, g\}\} \\ c &= \{\{a, b\}, \{b, c, d\}, \{d, g\}, \{c, d, e\}\}. \end{aligned}$$

The elements of c are then used to construct the Markov tree (N^*, E^*) shown in fig. 10.9(a). This tree is then "added" to the rest of the tree (N, E), and the nodes n_{4^*} and n_4, which both have the label $\{e, c, d\}$, are identified (they subsume each other, and one of them is therefore "devoured" by the other one). The resulting Markov tree is shown in fig. 10.9(b).

Comparing this result with the result of example 10.4 in fig. 10.7(b), we see that the maximal size of the labels of the nodes is two less in the improved algorithm than in the algorithm of Xu, which is a big advantage in view of the propagation on the Markov tree. ⊖

It is clear that the result of the improved algorithm is not necessarily globally optimal, but we call it locally optimal because the newly computed nodes



(a) Intermediate Markov tree (N^*, E^*). (b) Resulting Markov tree (N', E').

Figure 10.9: The improved algorithm applied to example 10.8.

form an optimal Markov tree (with respect to whatever optimality criteria are specified in the respective algorithm).

Note that, analogous to the algorithm of Xu, the marginals and messages computed so far in the original Markov tree can partially be re-used in the resulting Markov tree. There are not necessarily fewer messages to be sent than in the original algorithm of Xu, but the label on which they are computed are smaller, and clearly, the computation of several messages with small labels is usually more efficient than the computation of one message with a big label.

Step 5 of the improved method can further be improved such that it works not with the labels of the nodes, but with the labels of the information contained on them[4], together with the labels of the neighbor nodes:

5'. compute a new Markov tree (E^*, N^*) based on the *labels of the pieces of information* contained in the nodes a as well as the *labels of the nodes* in $c - a$.

It can be proved that the resulting tree (E', N') is also a Markov tree. The advantage of this approach is that in general the resulting Markov tree is better. The disadvantage is that there might exist one (or several) node in the original Markov tree whose label is not contained in the label of a node of the resulting tree anymore, this means that we cannot re-use the marginals (and the respective messages) on this node anymore. However, depending on the application, this algorithm might be interesting.

[4]This makes sense because the labels of these nodes are mainly joins of labels of pieces of information created during the construction of the Markov tree.

Comparing the Algorithms

Table 10.1 shows a short comparison of the three algorithms in general. The comparison includes first the quality of the final Markov tree (e.g. the maximal size of the labels of nodes, or another criteria), secondly how many of the old marginals and messages processed so far can be re-used, and thirdly the computations necessary to achieve the result (i.e. constructing/changing the Markov tree and the subsequent message passing).

	Rebuild Markov T.	Xu's Algo.	Improved Algo.
Markov Tree	good (optimal)	rather bad	locally optimal
Old Marginals	lost	re-usable	partly re-usable
Computations	expensive	fair	fair

Table 10.1: Comparing the algorithms.

10.2 Using Characteristic Clauses

Based on (Inoue, 1991; Inoue, 1992; Siegel, 1987), Kohlas & Monney (1993) presented algorithms for an incremental computation of the diagnoses and conflicts, or more generally quasi-supports and supports, in propositional systems. The algorithms have been implemented a.o. by Hänni (1997) in the framework of assumption-based reasoning. These ideas can also be used in the present case of updating a knowledge base by a new information, but they are restricted to propositional logic.

Consider an argumentation system $\mathcal{AS} = (\mathcal{L}_{\{a_1,\dots,a_p\}}, \chi, \mathcal{L}_B, \vdash)$ analogous to the one in example 10.3 where $\mathcal{L}_{\{a_1,\dots,a_p\}}$ denotes the propositional language over the atom $\{a_1, \dots, a_p\}$ and $\mathcal{L}_B$ is the propositional language over another set of atoms, say B. The entailment relation $\vdash$ is the usual one in propositional logic. This approach restricts the formulas in $\mathcal{L}_{\{a_1,\dots,a_p\}}$ being conjunctions and the formulas in $\mathcal{L}_B$ being clauses, but clearly, the argumentation system can be transformed as to fulfill this restriction.

The representation Ξ of χ is

$$\Xi \;=\; \left\{ \begin{array}{c} (\alpha_1^i \wedge \cdots \wedge \alpha_{j_i}^i) \rightarrowtail (\beta_1^i \vee \cdots \vee \beta_{k_i}^i) : i = 1, \dots, n, \\ \alpha_1^i \wedge \cdots \wedge \alpha_{j_i}^i \in \mathcal{L}_{\{a_1,\dots,a_p\}},\ \beta_1^i \vee \cdots \vee \beta_{k_i}^i \in \mathcal{L}_B \end{array} \right\}.$$

In the sequel, the mapping $\rightarrowtail$ will be interpreted as the usual connective "$\rightarrow$" in the language $\mathcal{L} := \mathcal{L}_{\{a_1,\dots,a_p\}} \vee \mathcal{L}_B$ such that the elements of Ξ can be represented as clauses in $\mathcal{L}$,

$$\Xi \;=\; \left\{ \begin{array}{c} \neg\alpha_1^i \vee \cdots \vee \neg\alpha_{j_i}^i \vee \beta_1^i \vee \cdots \vee \beta_{k_i}^i : i = 1, \dots, n \\ \alpha_1^i \wedge \cdots \wedge \alpha_{j_i}^i \in \mathcal{L}_{\{a_1,\dots,a_p\}},\ \beta_1^i \vee \cdots \vee \beta_{k_i}^i \in \mathcal{L}_B \end{array} \right\}.$$

A **production field** P is a nonempty set of clauses in the language $\mathcal{L}$. It is called **stable** if every clause c in $\mathcal{L}$ which is subsumed by a clause of P is also in P, i.e. if $c \in P$ and $c' \subseteq c$ then $c' \in P$. Define $\neg P := \{\neg p : p \in P\}$. An interesting production field is for example the set of all clauses in $\mathcal{L}_{\{a_1,\ldots,a_p\}} \subseteq \mathcal{L}$, which is denoted by $P_{\{a_1,\ldots,a_p\}}$.

Definition 10.9 *Let P be a stable production field. Then the **characteristic clauses of** Ξ **with respect to** P are*

$$Carc(\Xi, P) \quad = \mu(C^+(\Xi) \cap P). \tag{10.8}$$

The next theorem shows the relation between characteristic clauses, minimal conflicts and quasi-supports.

Theorem 10.10 *Let $f \in \mathcal{L}$, then*

$$\mu Conf(\Xi) \quad = \quad \neg Carc(\Xi, P_{\{a_1,\ldots,a_p\}}) - \{a_i \wedge \neg a_i : i = 1, \ldots, p\}$$

and therefore

$$Conf(\Xi) \quad \cong \quad \bigvee \left\{\neg Carc(\Xi, P_{\{a_1,\ldots,a_p\}})\right\}, \tag{10.9}$$

$$qs(f, \Xi) \quad \cong \quad \bigvee \left\{\neg Carc(\Xi \cup \{\neg f\}, P_{\{a_1,\ldots,a_p\}})\right\}. \tag{10.10}$$

Proof of theorem 10.10 Follows from theorem 5.17 of (Kohlas & Monney, 1993) by taking into consideration that the set of characteristic clauses contains the trivial clauses of the form $a_i \vee \neg a_i$ (if they are not subsumed by other clauses), whose negations $a_i \wedge \neg a_i$ are, by definition, not part of the minimal conflict and have therefore to be removed. □

The following interesting result shows how these characteristic clauses are computed incrementally based on the set $Prod(\Xi, c, P)$ using the concept of skipped ordered resolution on structured clauses. For a definition of $Prod(\Xi, c, P)$ see (Kohlas & Monney, 1993) based upon work of Inoue (1991; 1992) and Siegel (1987); for a discussion of an implementation see (Hänni, 1997).

Theorem 10.11 *(Inoue, 1991; Kohlas & Monney, 1993) If c is a clause in $\mathcal{L}$ and P a stable production field, then*

$$Carc(\emptyset, P) \quad = \quad \{p \vee \neg p \in P : p \in \{a_1, \ldots, a_p\} \cup B\}, \tag{10.11}$$

$$Carc(\Xi \cup \{c\}, P) \quad = \quad \mu(Carc(\Xi, P) \cup Prod(\Xi, c, P)). \tag{10.12}$$

These results, together with (10.9), can be used to compute the minimal conflict of the new knowledge base $\Xi \cup \{\eta\}$,

$$\begin{aligned} \mu Conf(\Xi \cup \{\eta\}) \quad &= \quad \neg Carc(\Xi \cup \{\eta\}, P_{\{a_1,\ldots,a_p\}}) \\ &= \quad \neg\mu\left(Carc(\Xi, P_{\{a_1,\ldots,a_p\}}) \cup Prod(\Xi, \eta, P_{\{a_1,\ldots,a_p\}})\right) \\ &= \quad \neg\mu\left(\neg Conf(\Xi) \cup Prod(\Xi, \eta, P_{\{a_1,\ldots,a_p\}})\right) \end{aligned} \tag{10.13}$$

and, similarly, the quasi-support for a clause $c \in \mathcal{L}_B$ with respect to the updated knowledge base can be computed using the minimal contradictions of $\Xi \cup \{\eta\}$, i.e.

$$qs(c, \Xi \cup \{\eta\}) \;\; \cong \;\; \bigvee \{\neg\mu\left(\neg \mathit{Conf}(\Xi \cup \{\eta\}) \cup \mathit{Prod}(\Xi \cup \{\eta\}, c, P_{\{a_1,\ldots,a_p\}})\right)\}$$

Example 10.12: Continuation of example 10.3

Consider again the information in (10.5). We use (10.8) or more efficiently (10.12) and (10.11) to compute its characteristic clauses with respect to the productions field $P := P_{\{a_1,a_2,m_1,m_2,m_3\}}$,

$$\mathit{Carc}(\Xi, P) \;\; = \;\; \left\{ \begin{array}{l} \neg m_1 \vee \neg m_2 \vee \neg m_3, \\ a_1 \vee \neg a_1,\; a_2 \vee \neg a_2, \\ m_1 \vee \neg m_1,\; m_2 \vee \neg m_2,\; m_3 \vee \neg m_3, \end{array} \right\}$$

where the elements of the second and third line are the trivial characteristic clauses.

Now again, new information becomes available, i.e. we have the new information $\eta = \{x\}$. First, let's compute the newly produced clauses with respect to the production field P, i.e.

$$\mathit{Prod}(\Xi, \eta, P) \;\; = \;\; \{\neg m_1\}$$

and, using (10.13), this allows to compute the new minimal conflicts,

$$\begin{array}{rcl} \mu\mathit{Conf}(\Xi \cup \{x\}) & = & \neg\mu\Big(\mathit{Carc}(\Xi, P) \cup \mathit{Prod}(\Xi, x, P)\Big) \\ & & -\, \{a_1 \wedge \neg a_1,\; a_2 \wedge \neg a_2,\; m_1 \wedge \neg m_1, \ldots,\; m_3 \wedge \neg m_3\} \\ & = & \neg\mu\left(\left\{ \begin{array}{l} \neg m_1 \vee \neg m_2 \vee \neg m_3, \\ a_1 \vee \neg a_1,\; a_2 \vee \neg a_2, \\ m_1 \vee \neg m_1,\; \ldots,\; m_3 \vee \neg m_3, \end{array} \right\} \cup \{\neg m_1\}\right) \\ & & -\, \{a_1 \wedge \neg a_1,\; a_2 \wedge \neg a_2,\; m_1 \wedge \neg m_1, \ldots,\; m_3 \wedge \neg m_3\} \\ & = & \{m_1\}. \end{array}$$

⊖

10.3 Adding Several Pieces of Information at Once

Sometimes, not only one new piece of information becomes available, but several pieces at the same time, say $\{\eta_1, \ldots, \eta_q\}$. Clearly, we can update the actual argumentation system incrementally by considering one piece of information after the other, and applying the concepts introduced so far incrementally. Yet often, this is not efficient.

In the approach of propagations on Markov trees (section 10.1), adding several pieces of information at once is no problem. Usually, every piece of information can be put onto several different nodes, so the only problem is to choose those nodes which allow to do the computations of the marginal on the root node efficiently. So the idea is to put the new information onto nodes "close to each other", and then to do the inward propagation only from these nodes.

The concept of characteristic clauses (section 10.2) allows an incremental approach, and theorem 10.11 shows how to compute the conflicts incrementally. The only problem here is to find a "good" ordering of the new information $\eta_1, \ldots, \eta_q$.

10.4 Removing Information

Consider now the inverse problem of the one presented in the previous section, i.e. the minimal conflicts and the minimal diagnoses of a knowledge base Ξ have been computed, and now an element $\zeta \in \Xi$ is considered not valid anymore and therefore has to be removed from the knowledge base. Typically ζ could be an observation which has changed in time and is not true anymore. The problem is the same as before: find an algorithm which computes the minimal conflicts and diagnoses with respect to the knowledge base $\Xi - \{\zeta\}$ and uses as many of the messages and marginals already computed as possible for this task.

In non-trivial cases, we do not know of an approach allowing to use characteristic clauses (presented in section 10.2) besides throwing all previously computed information away and re-starting from scratch. Nevertheless, the third approach (subsection 10.1) of computations on a Markov tree and the ideas developed there can partly be used for this problem as well.

Assume that a Markov tree for the information Ξ is present and every node knows the previous information it has contained, namely the information which was put on it initially and the information contained after the (last) inward propagation. The node on which the information ζ initially has been put is denoted by n'. Because the minimal conflicts are known, the inward propagation towards the root n_r has been completed. Possibly also some parts (or even the whole) of the outward propagation have been computed. These parts of the outward propagation have to be deleted, because the computed messages contain the knowledge ζ which has to be removed, and we do not know of an algorithm which "divides" the information ζ out of the messages.

Two cases are now possible: either the Markov tree cannot be simplified, because the information on n' besides ζ does not allow it, or the Markov tree can be changed.

First, look at the case when the Markov tree will not be changed after the removal of ζ, for example if there is still information on n' whose label is equal to the old label. So in fact, we are in the same situation as presented in fig. 10.2, i.e. we compute a new information on the node n' which is the original information minus ζ, combined with the incoming messages from its outward neighbors. Then we continue the new inward phase towards the root node

n_r by recomputing only the messages $\psi_{\cdot\to\cdot}$ as in fig. 10.2, and the respective information on the nodes. After this process, the information on n_r is the marginal $(\Xi - \{\zeta\})^{\downarrow d(n_r)}$, and the minimal conflicts and minimal diagnoses can be computed on n_r as described in section 10.1.

Secondly, consider the case where, due to the removal of ζ from n', it would now be possible to simplify the Markov tree. Here the same techniques as presented in subsection 10.1.3 can be applied and the same problems arise: if the tree is changed in a non-trivial way, then usually some messages and information on nodes are lost and have to be recomputed.

Yet as already mentioned, the information ζ, which has to be removed, will in general not have a very big label, usually ζ will be just an observation. In this case it does not make sense to change the structure of the tree, because this implies heavy computations and does not allow to build a much better tree, because after the elimination of the observation ζ, this variable will typically still be present in some pieces of information in the tree.

Eventually changes of the tree are possible and should be made, like removing leafs containing no information, or eliminating subsumed nodes by taking over the information contained in them and the connections by the subsuming node.

10.5 Mutations – Changing Information

In this section, we will consider a combined problem, namely changing some information, by which we mean that some information ζ has become obsolete and is removed from the knowledge base Ξ, therefore $\Xi' = \Xi - \{\zeta\}$, and some new information η is available which has to be added to the knowledge base Ξ'. A typical example is the change of a measurement in time, e.g. the variable x was known to have the value 3, but now its value changes to 5.

In a first approach, this can be done by first removing ζ, using the ideas of section 10.4, and then adding η by using section 10.1. In the approach of propagation on Markov trees, we can usually do better than this, especially if ζ can be placed on the same node as the one from which η has to be removed. Then, the ideas of sections 10.1 and 10.4 can be straightforwardly combined, and only one inward propagation towards the node n_r is needed.

11 The Decision Problem

So far, given a probabilistic argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash, p)$, techniques for computing explanations for the malfunctioning of the system, also called diagnoses of the system (Reiter, 1987) have been shown, but the problem is that generally there are a lot of different diagnoses (even minimal ones). The goal clearly is either to identify just one diagnosis or to compute a "good" ranking of the diagnoses and propose this to the user, or, possibly, tell the computer how to make a decision automatically; so in the sequel, we understand by a user also a computer. This means we have to give decision support to the user.

One way to rank the diagnoses is to weigh them according to the posterior probability as presented in section 9.1, and select the most probable one. This approach is useful if there is one diagnosis which is much more probable than the other ones, but problematic if there are a lot of diagnoses with almost identical posterior probability. Yet already a definition of "much more probable" is not easy and depends highly on the problem. Another ranking is the length of the diagnoses: a short diagnosis is supposed to be more probable than a long one. However, both approaches are problematic, because in general, there is always a worst case where the ranking is not optimal at all.

In this chapter we will focus on another method: we try to get more information about the actual system state by making additional measurements, thereby reducing the set of possible diagnoses and updating posterior probabilities; thus we aim to get a more informative picture of the actual situation. For this purpose, we have to choose points where additional measurements (observations) should be made (subsection 11.1). Given the resulting information after the measurement, the user can then choose to make either further measurements, to replace some components by functioning ones, or to stop the process. Different versions of this strategy are discussed in section 11.2. Every strategy leads then to a decision tree. Several data are used for the tree: probabilistic data (section 11.3) like the probability that a measurement at a given point results in a given value, and non-probabilistic data (section 11.4), like costs of measurements or of replacements of a component. So for every possible point of measurement, we have to compute the measurement's expected utility. The utility is mainly based on cost of additional measurements, of replacements and penalties for unnecessary actions. A strategy requires several sequential measurements, therefore a decision analysis process has to be executed and a global optimum for the utility has to be computed. This is presented in section 11.5. The computation of a global optimum is in general too complex due to the size of the decision tree as well as the computation of the probabilistic data. Therefore in section 11.6, an algorithm is presented which allows to choose a point, the so-called **best next measurement**, with a one step lookahead strategy. A main ingredient of such an algorithm is the computation of a kind of information content; several aspects of this notion are presented in section 11.7. Examples using one step lookahead strategies are discussed in section 11.8. Finally, section 11.9 will discuss some comparisons of these different approaches.

Note that we consider here only the situation where every component which has been faulty in the beginning of the diagnosis process stays faulty until it is detected and replaced, and similarly, a component which has been working correctly stays working for the whole diagnosis process. Also, measurements do not induce the change of any values.

A major restriction in this chapter is that we consider only argumentation systems with variables; therefore we can measure the values of variables at a specific point and formulate the result of such a measurement as an information in the information algebra. In the sequel we restrict ourselves to the case where the measurements are correct and do not include error terms in order to simplify the concepts.

In the sequel, we will consider questions with respect to different argumentation systems. Therefore, we specify explicitly the underlying argumentation system, e.g. for a hypothesis h and an actual argumentation system $\mathcal{AS}$, we write

$$\begin{array}{llll} qs(h, \mathcal{AS}), & sp(h, \mathcal{AS}), & dsp(h, \mathcal{AS}), & \dots \\ QSS(h, \mathcal{AS}), & SS(h, \mathcal{AS}), & CS(\mathcal{AS}), & \dots \end{array}$$

to emphasize $\mathcal{AS}$, or we replace the argumentation system $\mathcal{AS}$ by its representation Ξ.

11.1 Where Can Measurements Be Made?

How do we choose points, where is it possible to make a further measurement? The best way to do this is to make a choice during the modeling process and explicitly specify these points. For example in the case of a system built of modules, the connections between the modules and to the outside world are often good points for a measurement. Otherwise two general strategies are possible:

- Every variable is assumed to be a possible point for a measurement, or
- Every variable which does not describe the functioning of a component is a possible point for a measurement; this is motivated by the idea that

the determination of the actual mode of a component requires more than one measurement, therefore it is better to measure the connectors independently.

These restrictions allow to reduce the complexity of the computations; from a theoretical point of view, we can always compute a ranking for all points in the system and only afterwards select an optimal point within those where a measurement is really possible.

In the sequel, we always specify during the modeling process those variables where measurements can be taken.

In the strategies discussed in this chapter, for every point where a measurement could be made, the possible outcomes have to be known. This work was influenced by working with ABEL (see section 12) restricted to binary and FSC variables as well as integer-valued variables in special situations. For the first two types the interesting values of each point are always known already during the modeling process, but for variables with more than a finite number of possible values, techniques have to be developed to get information about the values which are really probable to occur in the system[1]; in the case where only a finite number of values are probable to occur, the presented methods can be applied as well.

11.2 Strategies for Decisionmaking

As mentioned in the introduction to this chapter, a lot of different strategies are available for the diagnosis process. The strategies may include

- replacement of components,
- measurement of some unobserved variables,
- measurement of variables which have been observed before but whose values may have changed due to the replacement of components,
- changing the input configuration of the system and re-measuring variables.

Many combinations are possible, but we focus on three versions presented in subsections 11.2.2 to 11.2.4. For representing and evaluating these versions, we will first give a short introduction to decision trees in the following subsection.

It is well known that decision trees are not an optimal representation for complex decision situations. Better possibilities for handling the complexity of the decision process exist, see for example (Xu, 2000) for an introduction into network based decision algorithms. One possibility are so called influence diagrams

[1]Consider the combined information with respect to the variable in question. This results in a belief function with respect to the variable, and possibly the focal elements can be used to characterize the "interesting" values.

introduced in (Howard & Matheson, 1981; Miller *et al.*, 1976). An influence diagram is a network representation of the decision problem. Originally, it was only a representation and afterwards transformed into a decision tree, which was then evaluated, but solutions for directly evaluating influence diagrams have later been presented in (Olmsted, 1983; Schachter, 1986a; Schachter, 1986b), and a lot of further work has been done for speeding up the computations and extending the approach. A second, more recent possibility is called valuation-based system, introduced by Shenoy (1992a; 1992b); this approach can also be used in more general contexts. Both approaches try to factorize the decision problem in order to reduce its complexity. But in the sequel, we will not discuss these approaches, because our decision problem also generates influence diagrams or valuation based systems which are too big, and therefore we will focus on a approximation strategy based on decision trees (section 11.6).

In order to simplify notations, we assume in the sequel that the components of the system are $c_1, \ldots, c_n$, and for every component c_i there are one or several fault modes, but only one correct working mode denoted by *ok*, so the component c_i is working correctly if and only if $M(c_i, ok)$ is true. More general cases can be treated similarly.

Assume that measurements can be made at some specified points $x_1, \ldots, x_m$ in order to get more information about the state of the system. The range of every measurement is known and finite, i.e. the possible values at point x_i are $v_{i1}, \ldots, v_{ir_i}$. The goal is to identify the faulty component(s) and to replace them with lowest possible overall costs in order to get a working system.

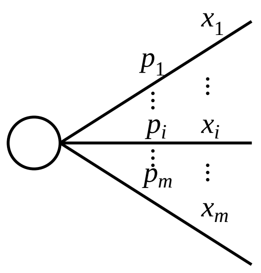
The language $\mathcal{L}$ of the argumentation system $\mathcal{AS} = (\mathcal{FSC}, \chi, \mathcal{L}, \vdash)$ must be general enough to contain the formulas $M(x_i, X_i)$, $X_i \subseteq \{v_{i1}, \ldots, v_{ir_i}\}$ for $i = 1, \ldots, m$. For simplification, we use also the abbreviation $(x_i = v')$ and $(x_i \neq v')$ and omit parentheses.

11.2.1 Decision Trees

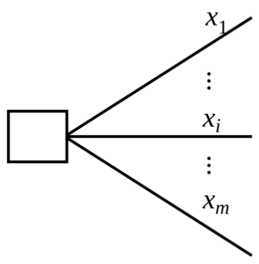
Following (Raiffa, 1968), we now offer a short introduction into the field of decision trees for representing the diagnosis process. The two main elements of a decision tree are presented in fig. 11.1: decision nodes and chance nodes.

A decision node is a place where, under the actual knowledge (sometimes explicitly specified within the node), the user can decide between a known set of alternatives x_1 to x_n. A chance node is a node where "nature" decides between a known set of alternatives x_1 to x_n with respect to the probability p given by $p(x_i) = p_i$ where $p_i \geq 0$ and $p_1 + \cdots + p_n = 1$.

The decision process is then a walk through the tree: Starting at the top level node, at the decision nodes, the user makes a decision, at the chance node, nature decides, and once the user is at a leaf node, he obtains the outcome marked at that leaf node (the leaf node is usually omitted and only the path to it is drawn).



(a) A chance node with possible outcomes x_1 to x_n and respective probabilities p_1 to p_n with $p_1 + \cdots + p_n = 1$.



(b) A decision node with possible decisions x_1 to x_n.

Figure 11.1: The two main elements of a decision tree.

In order to complete the decision tree defined in the next subsections, the payoffs for every leaf (subsection 11.4.2), the cost of the different actions to be taken at the decision nodes (subsection 11.4.1), and the path probabilities for every probabilistic node (section 11.3) have to be specified. The well known operation of roll-back analysis (Raiffa, 1968) can then be applied on this decision tree in order to solve the problem, that is, to find a sequence of decisions at the decision nodes with maximal expected outcome, see section 11.5.

11.2.2 Method 1

The first method consists of only two steps:

1. decide whether to take some measurements and which ones,
2. decide whether to replace some components and which ones.

Figure 11.2 shows a possible decision tree for this situation, the probabilities having been omitted. The measurements are considered as sequentially ordered, therefore after each measurement the user can choose to end the measuring process and start with the replacing process.

The actual knowledge is denoted by the representation Ξ of the argumentation system. Starting point is the node on the left-hand side labeled with D. The two possibilities to proceed are now either to

- replace component(s), that is, proceed to node R, or to
- make a measurement, that is, proceed to node M,

a choice which is denoted by the decision node D. In the first case, the user can then choose at node R to replace any set of components $\{c_i, i \in I\}$, which is

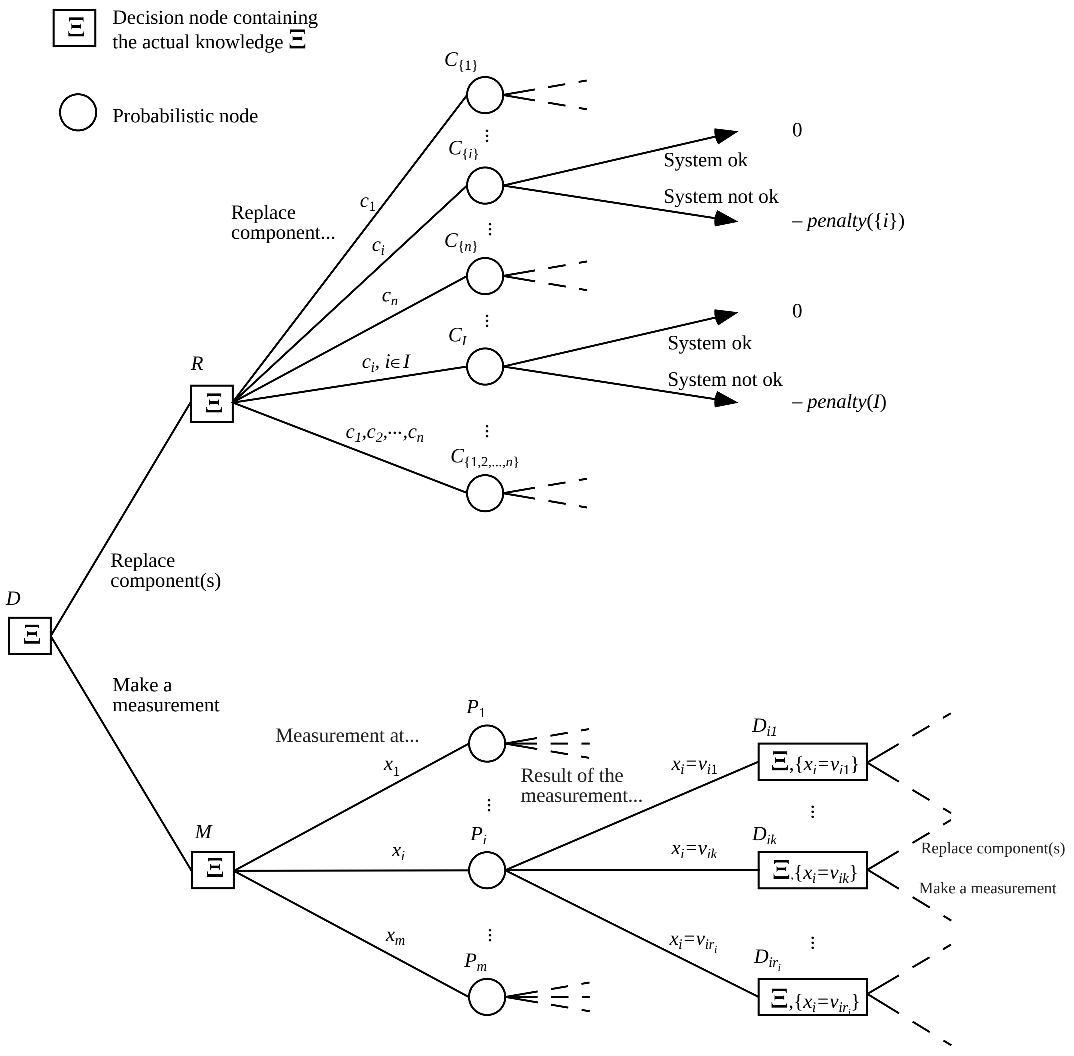


Figure 11.2: The decision tree for method 1.

denoted by the node C_I. After a replacement, the system either works correctly or not which is indicated by the two arrows starting at node C_I.

In the second case, i.e. at node M, the user can choose to measure at one point x_i. Depending on the measured value v_{ik} of x_i the process continues then at node D_{ik}. During this process, the result of every measurement $x_i = v_{ik}$ is added to the knowledge base (see node D_{ik}). The process then continues at node D_{ik} with the actual knowledge $\Xi \cup \{\top \rightarrowtail (x_i = v_{ik})\}$, abbreviated in fig. 11.2 as $\Xi, \{x_i = v_{ik}\}$.

In the sequel this method will be used for the examples.

11.2.3 Method 2

A generalization of the method above allows to do measurements and replacements of components in an arbitrary order. This means that the user can take some measurements, then replace some components, take some measurements again, then replace some components, and so on until he stops the process. This idea can be represented by a decision tree (fig. 11.3) similar to the one in fig. 11.2.

After the replacement of components $\{c_i, i \in I\}$, some of the already measured values are not valid anymore. If we can distinguish between measurements of input, output and internal variables, then clearly the measured values of the input variables are still valid, but some of the measurements of output and internal variables are not valid anymore. If the model permits to detect the ones which are still true (without explicitly remeasuring them), for example due to a modular structure, then the knowledge after the replacement is denoted by Ξ^* consisting of the knowledge Ξ minus the measurements which are not valid anymore, that is those where the value measured has bee changed. If the still valid measurements cannot be detected from the model, then Ξ^* represent the knowledge Ξ where all measurement of output and internal variables have been removed. (For a slightly different version see subsection 11.2.4.) Additionally, the replaced components are functioning, that is we have the information $M(c_i, \{ok\})$, $i \in I$, at node C_I, or written in the formalism of argumentation systems, this means $\neg M(c_i, \{ok\}) \rightarrowtail \perp$ for $i \in I$.

This method is clearly more general than method 1, but the loss of information after the replacement of one or several components is big, therefore in the case where measurements are expensive, method 1 is much more efficient, but if the measurements are cheap, then this method can give more accurate results if only few components are replaced in every step.

11.2.4 Method 3

The third method consists of first taking some measurements, then replacing some components and predicting the new output values, then taking some measurements again, replacing some components, and so on.

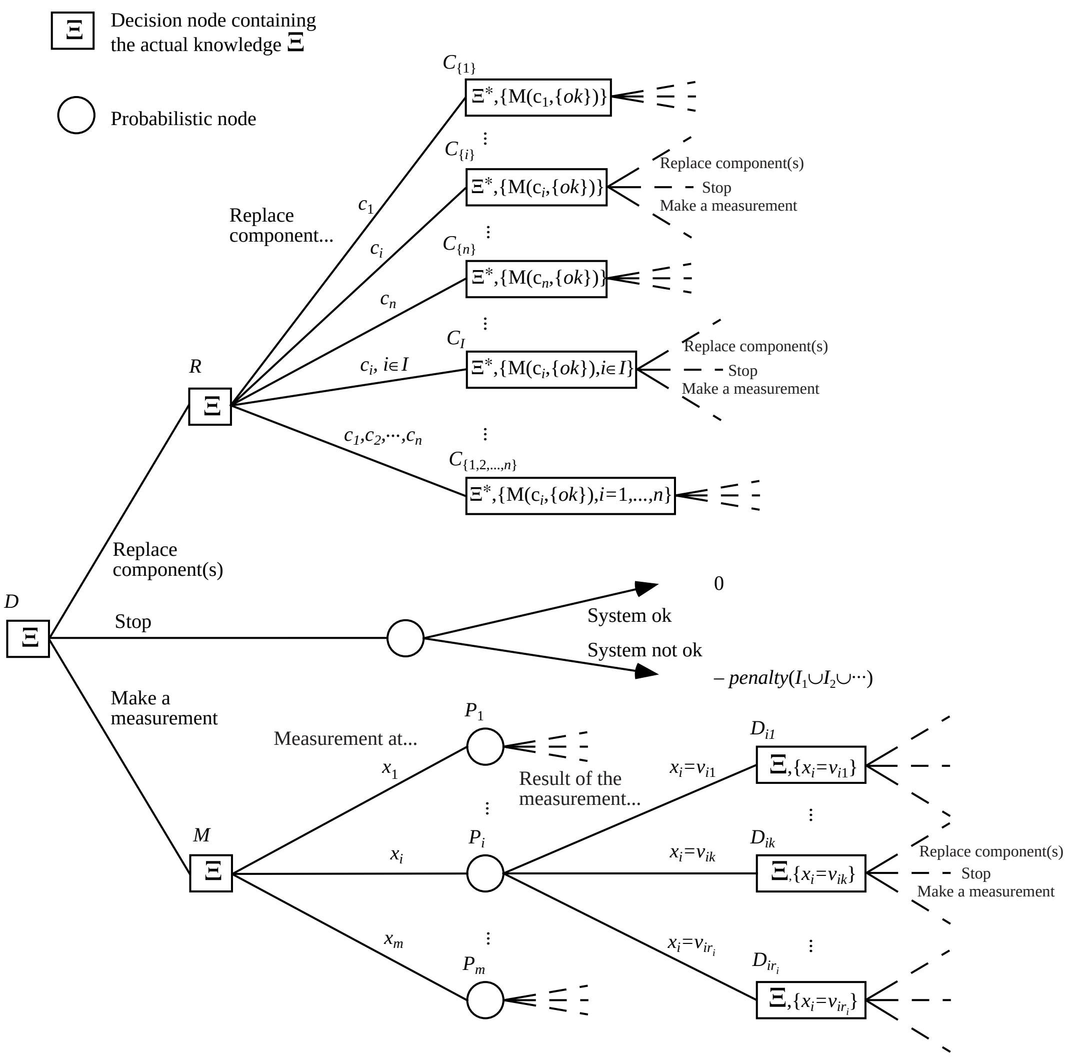


Figure 11.3: The decision tree for method 2.

This method is similar to method 2, but here the new output values are not measured but predicted from the available knowledge, that is Ξ and the replaced components $\{c_i, i \in I\}$. The knowledge clearly allows to compute a belief function on each output variable, and then either this belief function is considered as the new knowledge on the variable, or better, using the pignistic probability (defined in subsection 11.3.2), a concrete value of the output variable is chosen to be the actual one and this information is added to the knowledge.

A main problem in this approach is first of all the specification of input, output and inner variables, which in general is not possible. Furthermore, the computation of the belief functions on the output variables and the respective pignistic probabilities is computationally expensive, and often the modes of the components are not full specified, for example, there is a fault mode, and nothing is said about the behavior of the relations between the connectors of the component if it is in this fault mode. But in this case, nothing can be said about the output variables.

11.3 Probabilistic Data

There are two types of probabilistic data which have to be known in order to be able to do computations in the decision tree: the probability that the system works normally (subsection 11.3.1) and the probability of the outcome of a measurement (subsection 11.3.2).

11.3.1 Normal Working of the System

For $I \subseteq \{1, \ldots, n\}$, consider the path probability after the replacement of the set $\{c_i, i \in I\}$ of components, that is the path right upwards from the choice node C_I in fig. 11.2. The actual knowledge, before replacing components, is denoted by Ξ. Remember that we assume that a replaced component is always working correctly. So after replacing the components $\{c_i, i \in I\}$, we are interested in the event *system ok* and especially the

$$\textit{probability that the system is ok given } \Xi^* \cup \{M(c_k, \{ok\}), k \in I\} \tag{11.1}$$

where Ξ^* is, like in the previous section, the knowledge Ξ minus the old output and internal observations plus the new output observations, because after the replacement of the components $\{c_i, i \in I\}$, the output and internal values of the system might be different from before and therefore have to be re-measured (or possibly predicted). This means that all measurements which directly or indirectly depend on components which have been replaced or repaired, have to be re-made and the new results of these measurements replace the corresponding previous ones in the knowledge base.

The question is now: what does the event *system ok* mean in this situation? Our interpretation is that the observed output values of the system must be equal to the predicted ones. So we would like to compute:

$$\begin{aligned}&\textit{probability that the new observed output is equal to}\\&\textit{the predicted output given } \Xi^* \cup \{M(c_k, \{ok\}), k \in I\}\end{aligned} \tag{11.2}$$

The problem in (11.2) is that the new output values have to be measured or predicted in order to allow actually to describe Ξ^*, but we do not want to do this because of its complexity. So we need another way to do the computation.

Assuming that all the replaced components $\{c_i, i \in I\}$ do work correctly, we know that the new output values will correspond to the predicted ones when all the components c_i with $i \notin I$ (i.e. the ones which have not been replaced nor repaired) are working (but the contrary is not necessarily true). Furthermore, we assume that those components which worked before the replacement of other components are still working afterwards. So therefore we can use the previous measurements to get information about the working modes of the components $\{c_i, i \notin I\}$. The conjunction $\bigwedge_{i \notin I} M(c_i, \{ok\})$ represents the event that all component $\{c_i, i \notin I\}$ are working correctly, and the posterior probability of this conjunction, defined in (9.5) by $p'\left(\bigwedge_{i \notin I} M(c_i, \{ok\})\right)$, can be used as a value for the future functioning of the system.

In the sequel we will use the notation $\pi(\textit{System ok}, \mathcal{AS}, I)$ to denote the path probabilities of the path leaving node C_I right upwards in fig. 11.2, i.e. the path representing the system being ok after the replacement of the components c_i, $i \in I$, with previous knowledge $\mathcal{AS}$ (represented by Ξ), therefore we define

$$\pi(\textit{system ok}, \mathcal{AS}, I) := p'\left(\bigwedge_{i \notin I} M(c_i, \{ok\}), \mathcal{AS}\right), \tag{11.3}$$

and the probability of the other path, namely the event that the system is not working correctly, is just the difference to 1, i.e.

$$\pi(\textit{system not ok}, \mathcal{AS}, I) := 1 - \pi(\textit{system ok}, \mathcal{AS}, I), \tag{11.4}$$

and we therefore have a probability distribution on these paths.

11.3.2 Results of Measurements

Second, the path probabilities of the outcomes of a measurement have to be specified. In a first attempt one would like to compute the probability of the event $\{x_i = v_{ik}\}$ given the knowledge $\mathcal{AS}'$. But how should this be computed?

The function *dsp* is a belief functions, therefore we have the following lemma:

Lemma 11.1 *Let x_i be a point of a possible measurement with the domain $\{v_{i1}, \ldots, v_{ir_i}\}$. Then*

$$\sum_{k=1}^{r_i} dsp(\{x_i = v_{ik}\}, \mathcal{AS}) \quad \leq \quad 1. \tag{11.5}$$

This lemma implies that there is a problem in defining the path probability, because usually the underlying DS-belief function on x_i is non-Bayesian. So what can be done? In the sequel, we will present different solutions to this problem and mention some of their advantages and disadvantages. The formula $\pi(\{x_i = v_{ik}\}, \mathcal{AS})$ will be used to denote the probability of the path for which the measured value of x_i is v_{ik} and the knowledge before the measuring was $\mathcal{AS}$, for example in fig. 11.2 the path from node P_i to D_{ik}.

Note that we always assume that the domains of the respective variables are finite; if infinite or even continuous domains of variables occur, then other approaches have to be developed, but this problem will not be treated here.

Degrees of Support and Plausibility of the Singletons

The first idea is that we can work with the degrees of support of the singletons $dsp(\{x_i = v_{ik}\}, \mathcal{AS})$ and ignore the degrees of support of disjunctions, i.e. set

$$\pi_1(\{x_i = v_{ik}\}, \mathcal{AS}) := dsp(\{x_i = v_{ik}\}, \mathcal{AS}). \tag{11.6}$$

- Advantage: the computations are relatively simple.
- Disadvantage: there is a loss of information, and the result does not specify a proper decision tree, because π_1 is in general not a probability measure, i.e. $\sum_{k=1}^{r_i} \pi_1(\{x_i = v_{ik}\}, \mathcal{AS})$ does not necessarily equal 1 as shown in lemma 11.1.

Similarly, we can take into consideration only the degrees of plausibility of the singletons $dpl(\{x_i = v_{ik}\}, \mathcal{AS})$ and ignore the plausibilities of disjunctions, but an analogous problem arises, because $\sum_{k=1}^{r_i} dpl(\{x_i = v_{ik}\}, \mathcal{AS}) \geq 1$. In between these two extreme possibilities, there are a lot of possibilities using the weighted sum $\alpha dsp(\{x_i = v_{ik}\}, \mathcal{AS}) + (1-\alpha) dpl(\{x_i = v_{ik}\}, \mathcal{AS})$ for $0 \leq \alpha \leq 1$, but the problem is then the estimation of the parameter α.

Distributing the “Rest” Equally

Another approach to define a probability measure using the normalized belief function *dsp* was presented in (De Kleer & Williams, 1987; De Kleer *et al.*, 1992b). In the sequel we present a variant of this approach.

First, compute the rest of the probability not “used” by the degrees of support of the singletons, i.e.

$$rest_i := 1 - \sum_{k=1}^{r_i} dsp(\{x_i = v_{ik}\}, \mathcal{AS}), \tag{11.7}$$

and then distribute this rest equally on all singletons, that is, define

$$\pi_2(\{x_i = v_{ik}\}, \mathcal{AS}) := dsp(\{x_i = v_{ik}\}, \mathcal{AS}) + \frac{rest_i}{r_i}. \tag{11.8}$$

- Advantage: the computations are relatively simple and the result specifies a decision tree, i.e. π_2 is a probability measure,

$$\sum_{k=1}^{r_i} \pi_2(\{x_i = v_{ik}\}, \mathcal{AS}) = 1.$$

- Disadvantage: there is a loss of information.

Distributing the DS-belief (Pignistic Probability)

Another method to be considered is the concept of *pignistic probability* (Dubois & Prade, 1982; Smets & Kennes, 1994; Smets, 1990; Smets & Kennes, 1990). The function *dsp* defines a belief function on the information algebra of hypotheses in the sense of the Dempster-Shafer theory of evidence (Kohlas *et al.*, 1998; Kohlas, 1995; Shafer, 1976). Therefore it seems natural to use the concept of pignistic probability, a theory for decisionmaking with belief functions, sometimes also called the *betting probabilities* based on the belief function. This comes from the idea that it is a probability on how we would bet on the outcome of an event if we know "only" the belief function on the results of the events.

Every belief function can also be represented as a mass function. This concept will be of interest in the sequel. The **normalized mass function** associated with *dsp* with respect to the variable x_i will be denoted by $m(\cdot)$ or $m(\cdot, \mathcal{AS})$ to stress the argumentation system $\mathcal{AS}$, and it can be defined on the finite set constraints associated with the measurement point x_i as the posterior probability of every argument which implies the finite set constraint but does not imply any subset of the finite set constraint, i.e. for $\emptyset \neq K \subseteq \{v_{i1}, \ldots, v_{ir_i}\}$,

$$\begin{aligned} m(M(x_i, \emptyset)) &:= 0 \qquad (11.9)\\ m(M(x_i, K)) &:= \frac{1}{P(DS)} P\left(QSS(M(x_i, K)) - \bigcup_{K' \subsetneq K} QSS(M(x_i, K')) \right). \end{aligned}$$

Lemma 11.2 *The function m defined by (11.9) is a mass function, i.e. it is positive and sums up to one.*

Proof of lemma 11.2 By definition, $m(\cdot) \geq 0$. In addition to this we have to show that the sum over all subsets is equal to one:

$$\begin{aligned} &\sum_{K \subseteq \{v_{i1}, \ldots, v_{ir_i}\}} m(M(x_i, K)) \\ &= \frac{1}{P(DS)} \sum_{K \subseteq \{v_{i1}, \ldots, v_{ir_i}\}} P\left(QSS(M(x_i, K)) - \bigcup_{K' \subsetneq K} QSS(M(x_i, K')) \right) \\ &= \frac{1}{P(DS)} P\left(\bigcup_{K \subseteq \{v_{i1}, \ldots, v_{ir_i}\}} QSS(M(x_i, K)) - QSS(M(x_i, \emptyset)) \right) \end{aligned}$$

$$= \frac{1}{P(DS)} P(DS) \quad = \quad 1.$$

□

For other relations between belief functions, mass function, commonality functions, and q-functions see for example (Smets & Kennes, 1994; Smets, 1990). The definition of the mass function is rather complex, but in concrete situations, (hopefully) only few formulas have a non-zero mass (the so-called focal sets of the mass function).

Generalizing the idea of the previous approach, the belief that is accorded to a formula $M(x_i, \{v_{ik}, v_{i\ell}\})$ is divided in two and added to the belief of each "single formula" $\{x_i = v_{ik}\}$ and $\{x_i = v_{i\ell}\}$. The same idea is then used for longer disjunctions. Thus we define

$$\pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) \quad = \sum_{K: v_{ik} \in K \subseteq \{v_{i1},\ldots,v_{ir_i}\}} \frac{m(M(x_i, K))}{card(K)}. \tag{11.10}$$

The function $\pi_3(\cdot, \mathcal{AS})$ is called a **pignistic probability** to the belief function (Smets, 1990; Smets & Kennes, 1990; Smets & Kennes, 1994) and is really a probability measure, because $\pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) \geq 0$ and

$$\begin{aligned} \sum_{k=1}^{r_i} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) &= \sum_{k=1}^{r_i} \sum_{K: v_{ik} \in K \subseteq \{v_{i1},\ldots,v_{ir_i}\}} \frac{m(M(x_i, K))}{card(K)} \\ = \sum_{\emptyset \neq K \subseteq \{v_{i1},\ldots,v_{ir_i}\}} card(K) \frac{m(M(x_i, K))}{card(K)} &= \sum_{\emptyset \neq K \subseteq \{v_{i1},\ldots,v_{ir_i}\}} m(M(x_i, K)) \\ = \quad 1 - m(M(x_i, \emptyset)) &= 1. \end{aligned}$$

The definition of pignistic probability happens to reflect the generalized insufficient reason principle, but Smets & Kennes (1990) provide an axiomatic justification for the definition of pignistic probability and justify it. They also show that pignistic probability has nice properties when belief functions are combined and specialized.

- Advantage: there is no information lost and the result does specify a decision tree, i.e. $\sum_{k=1}^{r_i} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) = 1$.
- Disadvantage: the computations are more complex.

In the sequel, this definition will be used in the examples due to its advantages.

Generalized Decision Trees

Another, quite different approach to the problem are the so-called generalized decision trees for the modeling of decision analysis using belief functions proposed in section 3.2 of (Strat, 1990). Generalized decision trees, in contrast to usual decision trees introduced in subsection 11.2.1, allow

1. disjunctions of events on branches emanating from chance nodes,
2. intervals as payoffs for leaf nodes, and
3. branches emanating from chance nodes representing a mass function, i.e. the masses still sum up to one, but the events are no longer disjoint.

Strat proposes a method to evaluate generalized decision trees. Define

$$\begin{aligned} E_*(x) &= \sum_{A_i \in \Theta} \inf(A_i) \cdot m_\Theta(A_i), \\ E^*(x) &= \sum_{A_i \in \Theta} \sup(A_i) \cdot m_\Theta(A_i), \end{aligned}$$

where m_Θ is a mass function on the frame of discernment Θ. The operators inf and sup select the smallest respectively largest element out of a set $A \subseteq \Theta$, in fact Strat assumes that the frame of discernment Θ is a set of scalar values.

A chance node represents a belief function and its value, an interval, is computed as

$$E(x) = [E_*(x), E^*(x)].$$

The utility of every branch of a decision node is computed as

$$E(x) = E_*(x) + \zeta \cdot (E^*(x) - E_*(x)),$$

with a parameter $\xi \in [0,1]$. The maximal value of the branches is then the value of the decision node.

Note that a strategy is needed for estimating a concrete value of the parameter ζ in the interval $[0,1]$. Strat defines ζ to be the *"probability that ambiguity will be resolved as favorably as possible"*. This seems to be a rather clear definition of the parameter ζ, but we do not now how it could generally be used to estimate ζ in our framework in a computationally tractable way.

We believe that in the actual context, the approach of Strat is not that interesting. The different branches emanating from chance nodes do not represent actually possible measurements any more; so in fact, if we re-model our decision tree using generalized decision trees, we would eventually get branches labeled with the event $\{x_i = v_{ik}\} \vee \{x_i = v_{i\ell}\}$. This makes the decision tree to grow exponentially, which is in contrast to the goal we are focussing in section 11.6, namely to *approximate* (and therefore simplify) the computations in the decision tree. Therefore we will not use this approach in the sequel.

11.4 Non-Probabilistic Data

There is also some non-probabilistic data which has to be modeled in the decision tree: costs of action and payoffs at leaf nodes.

11.4.1 Costs of Measurements and Replacements

In the decision tree, two actions require costs, namely

- measuring at point x_i costs $\mathit{Measure}(x_i)$,
- replacing component c_i costs $\mathit{Replace}(c_i)$.

We assume that these costs are constant through time, i.e. the total cost of the measurements does not depend on the order of the measurements, and that the cost of several measurements or replacement which are done at once is simply the sum of the individual costs. These are slight simplifications which can be justified in most cases. The general case can be treated as well, but it only adds a lot of branches to the decision tree without including new concepts; so we will not consider it here.

Sometimes it make sense to simplify these functions such that they are independent of the actual point x_i or component c_i, i.e.

$$\begin{aligned} \mathit{Measure}(x_i) &\equiv \mathit{measure\text{-}c}, & (11.11)\\ \mathit{Replace}(c_i) &\equiv \mathit{replace\text{-}c}, & (11.12)\end{aligned}$$

where *measure-c* and *replace-c* denote constant values.

11.4.2 Payoffs

The payoff of a good replacement, i.e. a replacement after which the system appears to work correctly, is defined to be zero. In the case where the wrong components have been replaced, we introduce a so-called penalty, so: if the system does not work correctly after the components $\{c_i, i \in I\}$, have been replaced, then a penalty $\mathit{Penalty}(I)$ is due.

Usually, the penalty function can be simplified such that it depends only on the number of components which have been replaced,

$$\mathit{Penalty}(I) = card(I)\mathit{penalty\text{-}c}, \qquad (11.13)$$

or in some cases, it is even reasonable to define it constant,

$$\mathit{Penalty}(I) \equiv \mathit{penalty\text{-}c}. \qquad (11.14)$$

11.5 Exact Computation in the Decision Tree

Using the definitions from the previous sections, the decision tree introduced in subsection 11.2.2 is completely specified, and so we can compute the path with minimal expected cost using the well-known roll-back analysis scheme (Raiffa, 1968). We will present these computations using the decision tree in fig. 11.2;

the decision trees of other methods introduced in subsection 11.2.3 and 11.2.4 can be treated similarly. The first subsection shows how these computations are done generally, and in subsection 11.5.2 some examples illustrate the process.

In this section we always define the path probabilities by the pignistic probability function (11.10).

11.5.1 Roll-Back Analysis

For every node in the decision tree, its profit can be computed based on the respective values of its direct successors. Every type of nodes has to be treated differently:

Leaf Nodes: The profit is equal to the payoff.

Probabilistic Nodes C_I, $\emptyset \neq I \subseteq \{1, \ldots, n\}$: The profit of a node C_I is the sum of the payoffs weighted by the respective path probabilities:

$$\begin{aligned} \mathit{Profit}(C_I) &= 0 \cdot \pi(\mathit{system\ ok}, \mathcal{AS}, I) \\ &\quad - \mathit{Penalty}(I) \cdot \pi(\mathit{system\ not\ ok}, \mathcal{AS}, I). \end{aligned} \tag{11.15}$$

Decision Node R: The profit is computed as the maximum value of the profits of its successor nodes minus the costs of the replacement of the respective components,

$$\mathit{Profit}(R) = \max_{\emptyset \neq I \subseteq \{1,\ldots,n\}} \left\{ \mathit{Profit}(C_I) - \sum_{i \in I} \mathit{Replace}(c_i) \right\}. \tag{11.16}$$

Probabilistic Nodes $P_1, \ldots, P_m$: The profit of P_i is the sum of the possible profits of its successor nodes weighted by the respective path probabilities,

$$\mathit{Profit}(P_i) = \sum_{k=1}^{r_i} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) \cdot \mathit{Profit}(D_{ik}), \tag{11.17}$$

and the values $\mathit{Profit}(D_{ik})$ have to be computed recursively.

Decision Node M: The profit of M is the maximum value of the profits of its successors minus the costs of the respective measurements,

$$\mathit{Profit}(M) = \max_{i=1}^{m} \left(\mathit{Profit}(P_i) - \mathit{Measure}(x_i) \right). \tag{11.18}$$

Decision Node D: The profit of D is the maximum of the profits of R and M,

$$\mathit{Profit}(D) = \max\{R, M\}. \tag{11.19}$$

The difficulty is that in general, these computations are not possible for bigger problems as on one hand the complexity of the tree increases too fast and on the other hand the computations of the path probabilities are rather complex, even if we take into consideration the simpler definition of costs and penalties described in (11.11), (11.12) and (11.14). This problem will be addressed in section 11.6.

11.5.2 Examples

We restate some examples presented by de Kleer (1987; 1990) in order to show how to do the exact computations. The examples are presented in the language ABEL (see chapter 12).

Example 11.3: Three Serial Inverters

Consider the example 7.1 consisting of three serial inverters presented in section 7.3. The minimal diagnoses have been computed as $M(i_j, \{\mathit{faulty}\})$, $j = 1, 2, 3$. All three happen to have an equal posterior probability of 0.337. So we need to make further measurements to discriminate between the diagnoses.

The corresponding decision tree is (partly) shown in fig. 11.4, the bold arrow on the left denotes the starting point of the diagnosing and measuring process. The decision nodes are labeled by the letters Di, Mi and Ri with $i = 1, 2, \ldots$ by generalizing the notation from fig. 11.2; the probabilistic nodes will be referenced in the sequel by their name (e.g. "AB") and their preceding decision node (e.g. "R2") in brackets (e.g. "AB[R2]") . The actual knowledge at the beginning (D1) is the argumentation system $\mathcal{AS}$ described in ABEL, and A, B, and C denote the replacement of components i_1, i_2, and i_3 respectively.

The values beneath the "gates" (a line with two dots "●—●") denote costs which are due when the branch is passed: M is the cost of a measurement, R, $2R$ and $3R$ are the costs of replacing 1, 2 and 3 components. The values at the end of the arrows denote the final payoffs: either the system is fixed, i.e. "system ok" with a revenue of 0, or the system is still faulty, i.e. "system not ok" where a penalty P is due. The values on the branches are the corresponding probabilities. Let's see how we compute these values in the tree. For this example, we fix these values:

- $P = 100$, so a penalty is a hundred units,
- $R = 20$, replacing a component costs 20 units, and
- $M = 5$, taking a measurement costs 5 units.

As an example, we show the detailed computation at the subtree of node R6 in fig. 11.5.

The actual knowledge on node R6 is $\mathcal{AS}$ as well as a measurement x and a measurement $(\mathit{not}\ y)$; see the branches leading from the start node D1 to node R6 in fig. 11.4. So the probabilities of the arrows of node A[R6] are computed according to (11.3):

$$\begin{aligned}&\pi(\mathit{system\ ok}, \mathcal{AS} \cup \{x, (\mathit{not}\ y)\}, \{A\})\\&\quad = \mathit{dsp}\,(M(B, \{\mathit{ok}\}) \wedge M(C, \{\mathit{ok}\}), \mathcal{AS} \cup \{x, (\mathit{not}\ y)\}) = 0.81,\end{aligned}$$

$$\begin{aligned}&\pi(\mathit{system\ not\ ok}, \mathcal{AS} \cup \{x, (\mathit{not}\ y)\}, \{A\})\\&\quad = 1 - \pi(\mathit{system\ ok}, \mathcal{AS} \cup \{x, (\mathit{not}\ y)\}, \{A\}) = 0.19,\end{aligned}$$

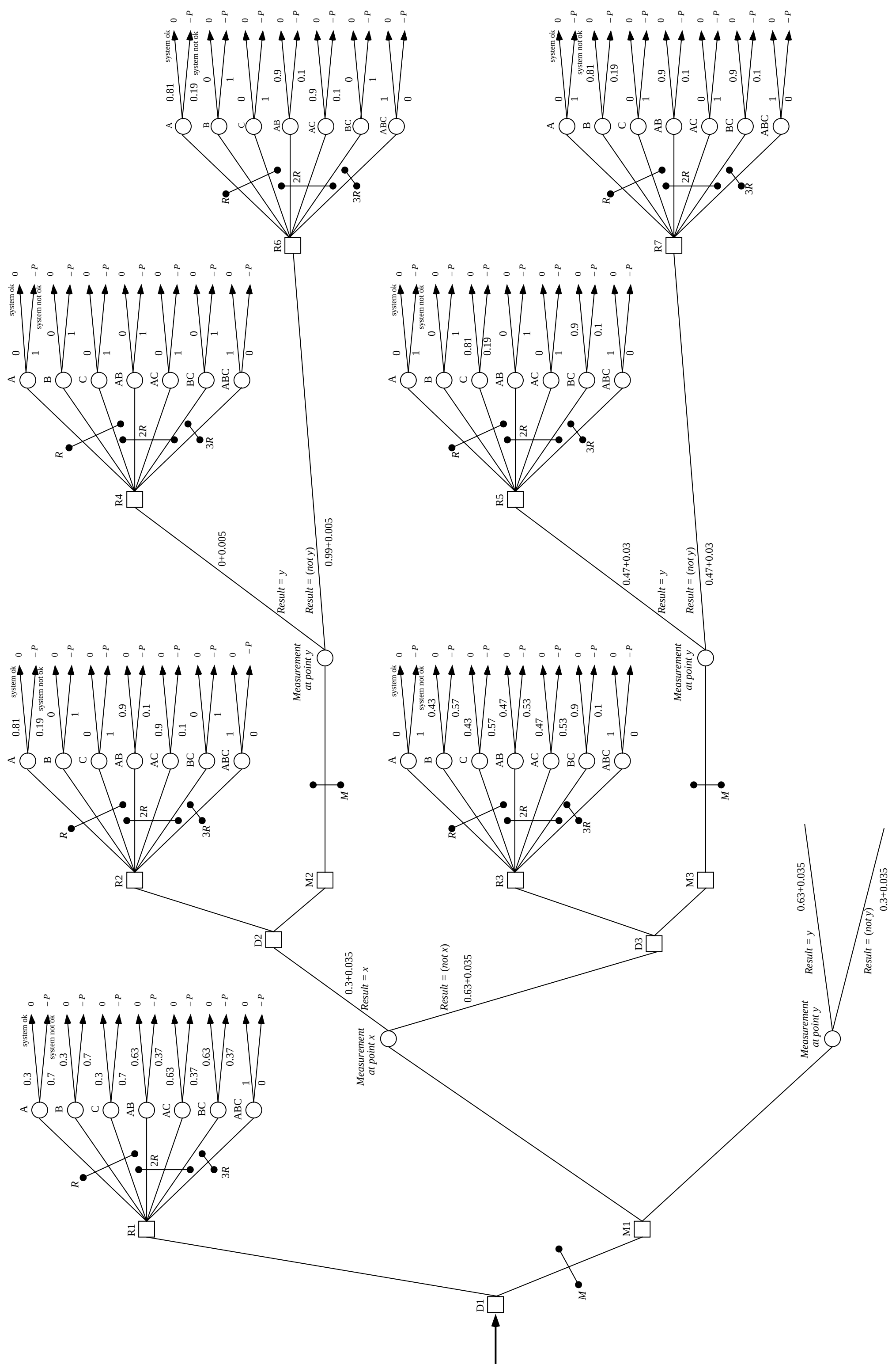


Figure 11.4: Decision tree for the tree inverters.

and similarly for the nodes B[R6], C[R6], etc. Again, $\mathcal{AS} \cup \{x, (not\ y)\}$ is an abbreviation for $\mathcal{AS} \cup \{\top \rightarrowtail x,\ \top \rightarrowtail (not\ y)\}$.

The profits of the nodes are marked on the left side of the respective node in fig. 11.5. The profit of the probabilistic node A[R6] is computed according to (11.15)

$$\begin{aligned} \mathit{Profit}(\text{A[R6]}) &= 0 \cdot \pi(\mathit{system\ ok}, \mathcal{AS} \cup \{x, (not\ y)\}, \{A\}) \\ &\quad - P \cdot \pi(\mathit{system\ not\ ok}, \mathcal{AS} \cup \{x, (not\ y)\}, \{A\}) \\ &= -0.19P \quad = \quad -19, \end{aligned}$$

and the profit of the decision node R6 according to (11.16)

$$\begin{aligned} \mathit{Profit}(\text{R6}) &= \max \left\{ \begin{array}{l} \mathit{Profit}(\text{A}) - R,\ \mathit{Profit}(\text{B}) - R, \\ \mathit{Profit}(\text{C}) - R,\ \mathit{Profit}(\text{AB}) - 2R, \\ \mathit{Profit}(\text{AC}) - 2R,\ \mathit{Profit}(\text{BC}) - 2R, \\ \mathit{Profit}(\text{ABC}) - 3R \end{array} \right\} \\ &= \max \left\{ \begin{array}{l} -0.19P - R,\ -P - R,\ -P - R, \\ -0.1P - 2R,\ -0.1P - 2R,\ -P - 2R, \\ 0 - 3R \end{array} \right\} \\ &= -39. \end{aligned}$$

Now everything is specified in the decision tree in fig. 11.5, and we see that the optimal decision at node R6 is to replace component i_1 (node A).

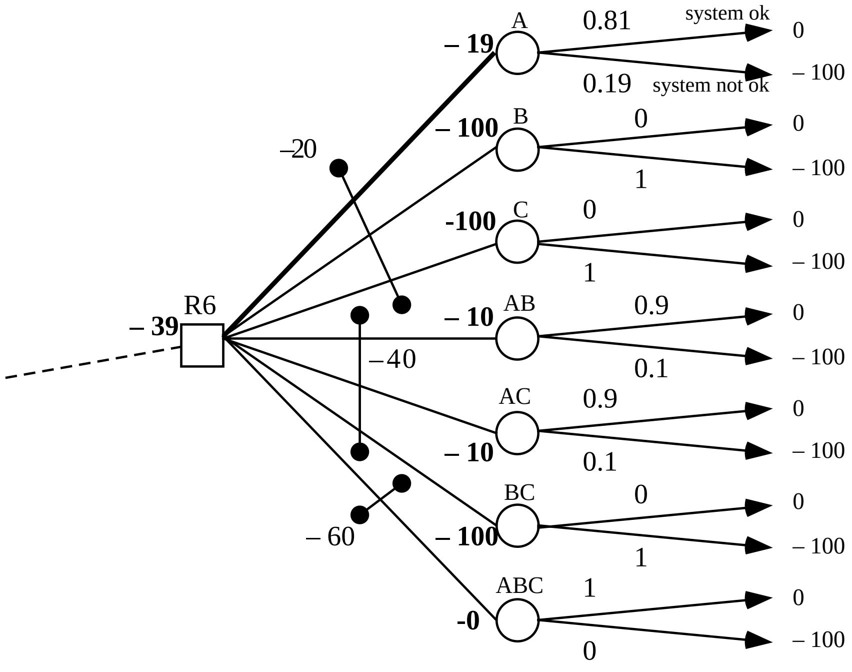


Figure 11.5: The subtree of node R6.

The other nodes in the tree are treated similarly, and the final situation is presented in fig. 11.6, where the not-optimal branches are “cut off” by double lines and most of them have been left out. Note that the branches starting at decision node M1 have the same profit, so the decision at this node is arbitrary and no branch is cut off; this observation is not a surprise, as the problem itself is clearly symmetric.

It is interesting to discuss the impact of the parameters P, R and M on the final result. In fact, the decision tree can be solved formally without specifying these values. In order to get the same optimal path as above, several restrictions have to be put on their values:

$$\begin{aligned} R &\leq 0.35P, & (11.20)\\ R &\geq 0.095P, & (11.21)\\ R &\geq M + 0.09P. & (11.22)\end{aligned}$$

The first inequality (11.20) means that if the cost of a replacement R is too big with respect to a penalty P, then it is not worth replacing a component, but the penalty is accepted as final profit.

Inequality (11.21) means that the cost of replacing a component R must not be lower than that of $0.095P$, because otherwise every component will always be replaced.

And finally the last inequality (11.22) means that the replacement of a component must be more expensive than a further measurement plus a small percentage of a penalty. $\ominus$

Example 11.4: The “Three Buffers”

Consider the example consisting of three buffers in figure 11.7 introduced in (De Kleer, 1990). If a buffer is working correctly, then its input values equals its output value; this mode is denoted by the propositional variable $i.OK$. Nothing is known about the behavior of a faulty buffer. Two measurements are specified in the figure. This specifies an argumentation system and is modeled in ABEL as follows:

```
(tell
   (module BUFFER ((var in out binary))
       (ass ok binary 0.9)
       (-> ok (<-> out in)))

   (BUFFER :A in x)
   (BUFFER :B x out1)
   (BUFFER :C x out2))

(observe (not in) out1)
```

The system is apparently not working correctly, as the measured output-value $out1$ is in contradiction with the predicted one. So ABEL can compute the two minimal diagnoses:

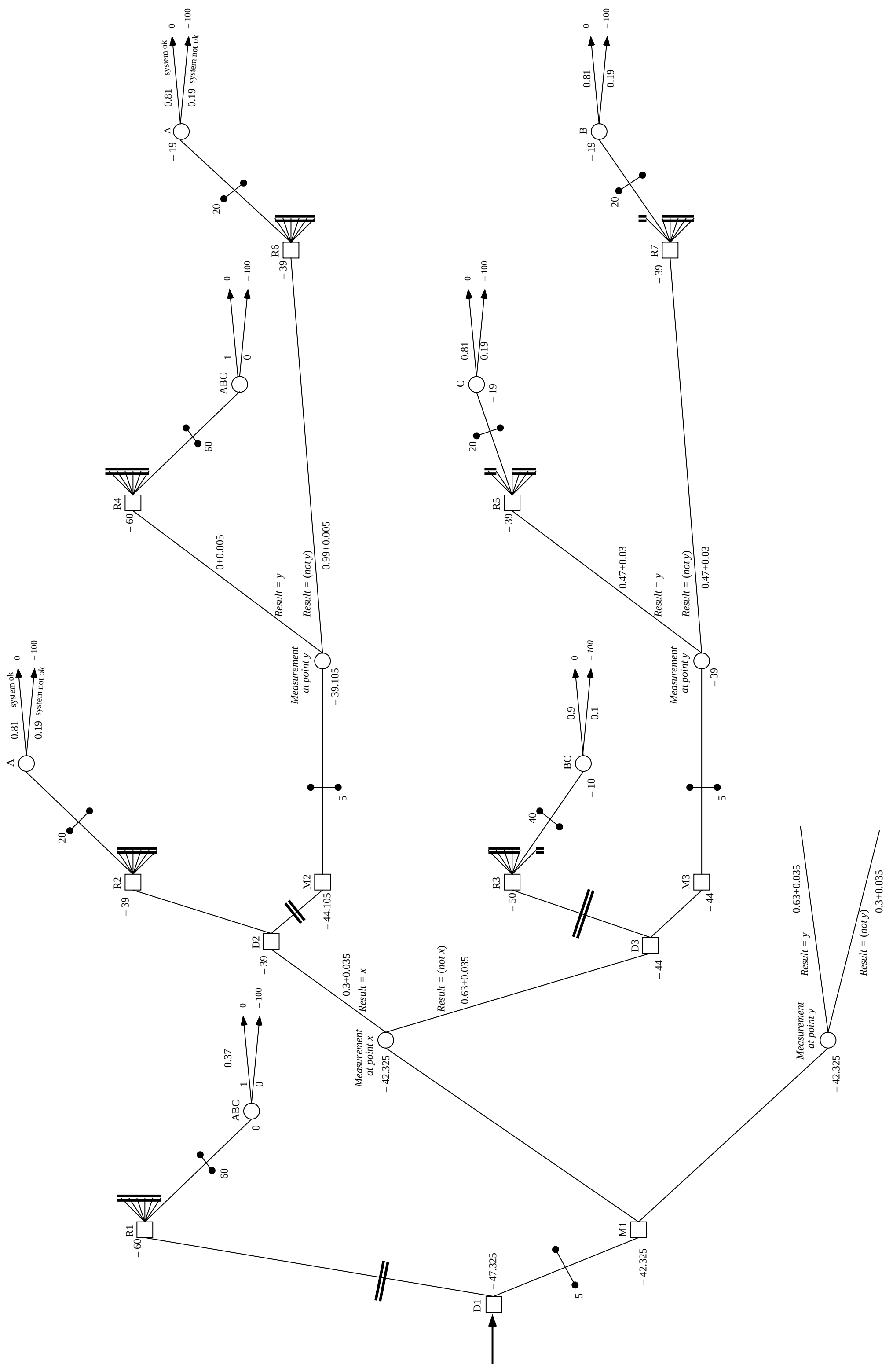


Figure 11.6: The optimal path for the tree inverters.

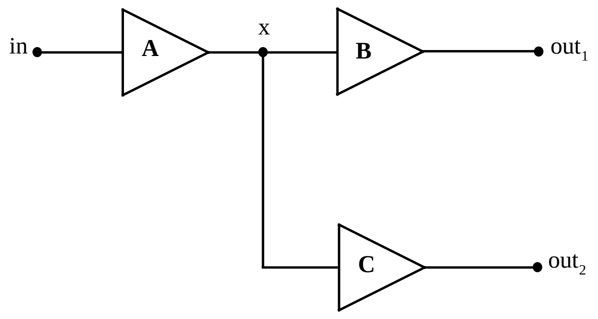


Figure 11.7: A network consisting of three buffers.

```
(NOT A.OK)
(NOT B.OK).
```

There are two possible points to measure at, namely x and $out2$. The different outcomes of a measurement then imply a division of the diagnoses according to the following results:

$$\begin{aligned} qs(\neg x) &= \{\texttt{NOT B.OK}\} \\ qs(x) &= \{\texttt{NOT A.OK}\} \\ qs(\neg out2) &= \{\texttt{NOT B.OK}, (\texttt{NOT A.OK}) \wedge (\texttt{NOT C.OK})\} \\ qs(out2) &= \{\texttt{NOT A.OK}, (\texttt{NOT B.OK}) \wedge (\texttt{NOT C.OK})\} \end{aligned}$$

Note that we work with minimal diagnoses, whereas in (De Kleer, 1990) only single fault diagnoses are considered.

The optimal point to be measured at is obviously x, as more information is gained and better discrimination of the diagnoses is obtained. Assume the same values of the parameters *replace-c*, *measure-c* and *penalty-c* as in the previous example. The corresponding decision tree is constructed in figs. 11.8 and 11.9 where the computed profits are shown on the left-hand side of the respective nodes and the not-optimal branches are "cut off" by with double lines. ⊖

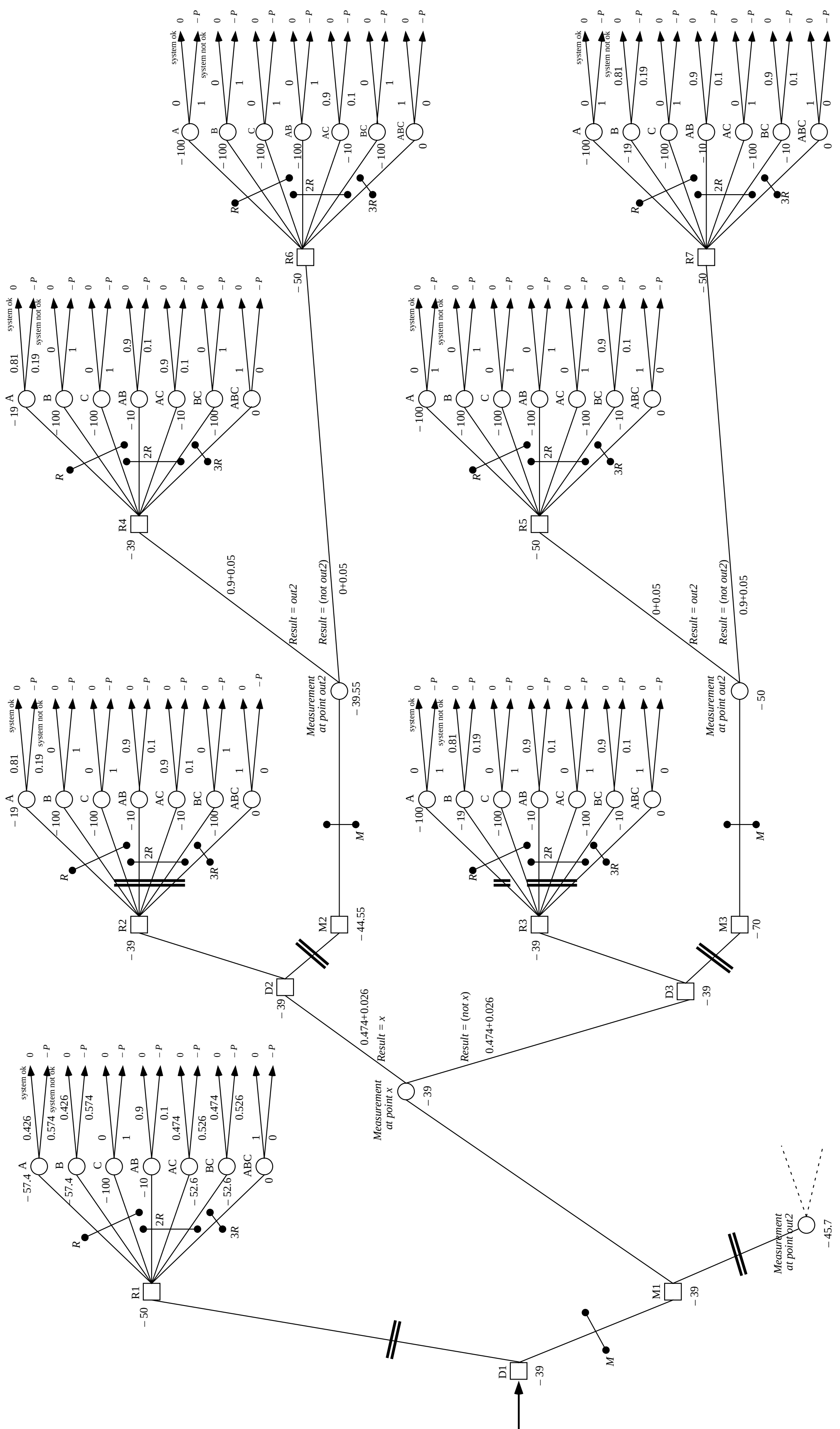


Figure 11.8: Optimal path for “three buffers”.

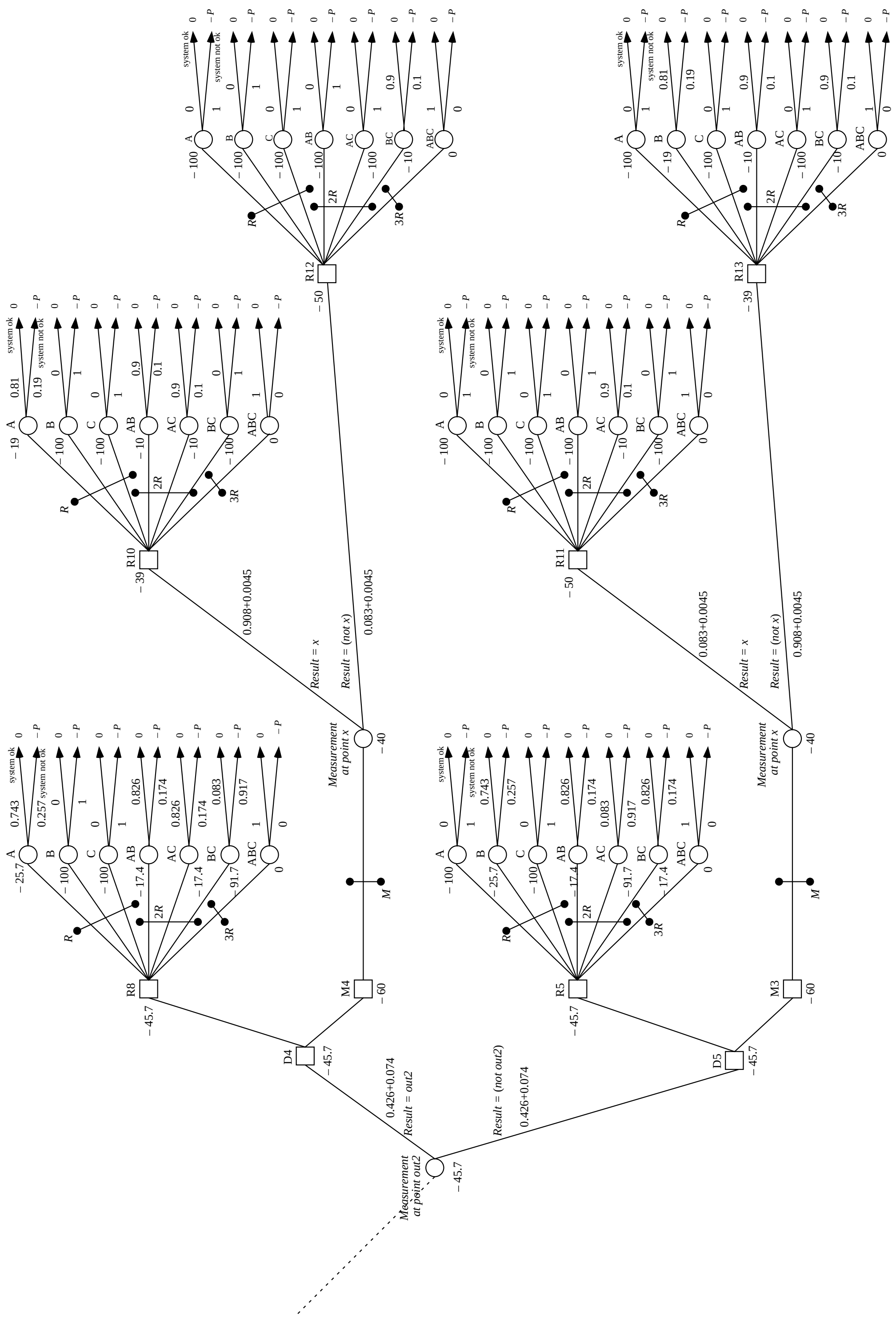


Figure 11.9: Continuation of fig. 11.8.

11.6 One Step Lookahead Methods

In this section we present the main concepts of one step lookahead methods. Together with the concepts of the previous section, this can easily be generalized to n-step lookahead methods. A major ingredient, a kind of measure of information[2] gained by a measurement, is only afterwards introduced and discussed in section 11.7.

11.6.1 The Method

Due to the complexity of the decision tree for bigger problems as presented in the previous section, we have to introduce techniques for approximating the computation on the tree, i.e. the roll-back analysis. Considering the decision tree in fig. 11.2, we need an approximation strategy for the profits $\mathit{Profit}(D_{ik})$ such that in (11.17) there is no more need to make recursive steps, but everything can be computed by just "looking one step ahead" from the actual situation (described by the knowledge $\mathcal{AS}$) in the decision tree. So for every possible point x_i for a measurement, we have to compute a kind of information $H(x_i)$ which is available after the measurement at point x_i has been done, and then select the point where a measurement has a maximal expected gain of information; such a point is called a **best next measurement**. Different possible concepts for the computation of this kind of information are presented in section 11.7. In the present section, we assume that we have selected one of them and denote it by H.

Note that $H(x_i)$ represents not an approximation for the value $\mathit{Profit}(P_i)$, but is only a value $\mathit{Profit}^*(P_i)$ which can be compared with the values $\mathit{Profit}^*(P_j)$, $j \neq i$. Therefore, in the decision tree (fig. 11.2) we can compute the value $\mathit{Profit}^*(M)$, but this value cannot be compared directly with $\mathit{Profit}(R)$ in order to compute the overall profit $\mathit{Profit}(D)$, because $\mathit{Profit}^*(M)$ is not an approximation. Other techniques are needed for the decision at node D, i.e. the decision when to stop the diagnosing and measuring process and start with replacing component(s). This problem is discussed in section 11.6.2.

The one step lookahead process in this form is inspired by (De Kleer *et al.*, 1992b). They also mention that these kinds of strategies are "pretty good", i.e. their experiments show that the number of measurements needed to isolate a single fault using their one step lookahead method is rarely more than 20% higher than the number of measurements needed in a complete decision tree, but because of the complexity of computations they do not consider multiple faults in their experiments. These results apply also to our case. The concepts can be generalized easily by looking several steps ahead in the decision tree, so called n-step lookahead methods, see also (De Kleer *et al.*, 1992b) for comparison of one step with n-step lookahead methods.

[2]The term "information" is used here with a very general meaning, and it should neither be confused with the concept of information algebras presented in chapter 2, nor with the sense in which Shannon (1948) uses it.

In the whole section we assume that the cost of all measurements are equal. Ideas on how to include varying measurement costs are presented in (De Kleer *et al.*, 1992b).

11.6.2 Stopping the Process

During the full computations on the decision tree the complete path is specified, so the whole process is known. But when we use one step lookahead methods, a criterion is needed which indicates when to stop the diagnosing process and to begin with the replacement of component(s), because, as noted above, we do not compute an approximation of the value $\mathit{Profit}(M)$ (cf. 11.2), but a value which cannot be compared directly with $\mathit{Profit}(R)$; so we need another algorithm to make the decision at node D.

The information available at node D is $\mathcal{AS}$, therefore an intuitive idea is that we compute all minimal diagnoses of $\mathcal{AS}$ and weight them according to the posterior probability defined by $\mathcal{AS}$. If the probability of the most probable diagnosis is far higher than the probability of the second most probable one, the diagnosing process is stopped and the components are replaced according to the most probable diagnosis, i.e. we continue the decision tree at node R, otherwise a new measurement is selected according to node M. Another method, proposed by de Kleer *et al.* (1992b), is to fix a threshold (e.g. 0.9) and to stop diagnosis if the probability of the most probable candidate approaches this threshold.

The only open question within both strategies is: how much higher should the probability of the most probable diagnosis be compared with the probabilities of the other diagnoses? There is no general answer, but this value has to be specified depending on the concrete case and on the seriousness of a misdiagnosis.

11.7 Computing Information

In this section we present several approaches for computing a value for the expected gain of "information" for a measurement at a given point with respect to the diagnoses. The first approach in subsection 11.7.1 deals with the "size" of the contradictions. The one presented in subsection 11.7.2 goes in the same direction as the idea of de Kleer. And finally the approach presented in subsection 11.7.3 consists of computing the information about variables.

11.7.1 Increasing the Contradictions

As discrimination of the different possible points of measure, we will use here the weight of the contradictions relative to the whole knowledge. The idea is that a measurement should enlarge the space of contradictions CS and so reduce the space of explanations of possible failures, the space of diagnoses DS which, hopefully, will then discriminate then better between minimal diagnoses. So we are looking for a point (or a set of points) where the expected enlargement of the contradiction implied by the outcome of the measurement is maximal.

Before the measurement is taken, the contradictions with respect to the available information $\mathcal{AS}$ are $CS(\mathcal{AS})$ and, after measuring at point x_i the value v_{ik}, are $CS(\mathcal{AS} \cup \{x_i = v_{ik}\})$. These formulas are then weighted by the probability measure P.

In other words, some sort of "size" of the added contradictions after measuring at point x_i can be computed. An expected value of this "size" (in the sequel called **score**) is thus the weighted sum

$$\begin{aligned} & H_1(x_i) \qquad (11.23) \\ = & \sum_{k=1}^{m} \pi_3(\{x_i = v_{ik}\}) \Big(P(CS(\mathcal{AS} \cup \{x_i = v_{ik}\})) - P(CS(\mathcal{AS})) \Big) \\ = & \sum_{k=1}^{m} \pi_3(\{x_i = v_{ik}\})\ P(CS(\mathcal{AS} \cup \{x_i = v_{ik}\})) - P(CS(\mathcal{AS})). \end{aligned}$$

Often, the argumentation system contains a negation operation for the set constraints describing the measurements: The formulas $x_i = v_{ik}$ are clearly disjoint and exhaustive, and if an operation "$\bigvee$" is defined, $\bigvee_{k=1}^{r_i} \{x_i = v_{ik}\} = [\top]$, then the negation of the hypothesis $x_i = v_{ik}$ in $\mathcal{L}$ is

$$\neg\{x_i = v_{ik}\} \quad = \quad \bigvee_{\substack{j=1,\ldots,r_i \\ j \neq k}} \{x_i = v_{ij}\}, \qquad (11.24)$$

and this implies that $CS(\mathcal{AS} \cup \{x_i = v_{ik}\}) = QSS(\neg\{x_i = v_{ik}\}, \mathcal{AS})$. Therefore, H_1 can also be written as

$$\begin{aligned} & H_1(x_i) \qquad (11.25) \\ = & \sum_{k=1}^{m} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) \Big(P(QSS(\neg\{x_i = v_{ik}\}, \mathcal{AS})) - P(CS(\mathcal{AS})) \Big) \\ = & P(DS(\mathcal{AS})) \\ & \cdot \sum_{k=1}^{m} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) \frac{P(QSS(\neg\{x_i = v_{ik}\}, \mathcal{AS})) - P(CS(\mathcal{AS}))}{1 - P(CS(\mathcal{AS}))} \\ = & P(DS(\mathcal{AS})) \sum_{k=1}^{m} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) dsp(\neg\{x_i = v_{ik}\}, \mathcal{AS}). \end{aligned}$$

The advantage of this second formulation is that the computations of the different degrees of support all take place on the same knowledge base $\mathcal{AS}$, while in the first formulation, conflicts with respect to a changed argumentation system have to be computed, and therefore, the second variant can be computed more efficiently.

The point at which the subsequent measurement should be taken is the one (or one of the set) that maximizes H_1. In order to simplify the computations, one could easily drop the first multiplicative factor $P(DS(\mathcal{AS}))$ in the definition of H_1.

The method presented above does not take into consideration the different diagnoses and their respective probabilities obtained after measuring a value at some point. This would lead to a different approach, which consists in computing for every possible value v_{ik} for a point x_i the most and the second most probable diagnoses with respect to $\mathcal{AS} \cup \{x_i = v_{ik}\}$. Then the optimal point for a measurement is the one which maximizes the expected difference of probabilities. But computations for concrete examples show that this approach, without considering the expected enlargement of the contradictions, does not give good results. Nevertheless, this idea can be combined with the method of expected enlargement of contradictions, in particular this will supply a criterion for choosing a point in the case where two (or more) different points x_i and x_j lead to the identical values $H_1(x_i) = H_1(x_j)$.

11.7.2 Entropy

If a set of n different possible events with respective probabilities $p_1, \ldots, p_n$ is given and nothing is known about which event will occur, then a measure of how much information is "produced" can be the Entropy (Shannon, 1948)

$$H = -K \sum_{i=1}^{n} p_i \log p_i, \tag{11.26}$$

where K is a positive constant and we will subsequently set it to 1. In the sequel, an event will be interpreted as "measuring a specified value at a specified point". For a discussion of the properties of entropy see (Guiasu, 1977; Shannon, 1948).

De Kleer (1990) shows that, assuming that any measurement does not influence the value measured, the expected entropy $H_e(x_i)$ of candidate probabilities after measuring a quantity x_i with possible values $v_{i1}, \ldots, v_{ir_i}$ is

$$H_e(x_i) = -\sum_{k=1}^{r_i} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) H(x_i = v_{ik}),$$

where $H(x_i = v_{ik})$ denotes the entropy resulting if x_i is measured to be v_{ik}. Minimal expected entropy results then in maximal security.

Depending on the eventually observed in- and output variables of the system, de Kleer first computes the set of all candidates whose size is less than some predefined constant. During the process of making new measurements, these candidates are then either eliminated if they are contradictory to the measurement or otherwise their probabilities are updated by dividing them by the probability of the measurement $\{x_i = v_{ik}\}$. But in the case where some candidates (say q of them) predict no value for the point of the measurement, their probabilities have also to be updated, and in this case de Kleer assumes that every possible value for x_i is equally likely and divides the probability of all such candidates by the probability of the measurement $p(\{x_i = v_{ik}\})$ and the number of such candidates q. At the end, just one candidate should be left which explains the misbehavior of the system.

Starting from the entropy (11.26), de Kleer (1990) considers the p_i as probabilities that a candidate C_i is the actual candidate given the hypothesized measurement outcome. De Kleer then defines a so-called score of measuring at point x_i by

$$\$(x_i) = \sum_{k=1}^{r_i} c_k \log c_k, \tag{11.27}$$

where c_k denotes the number of candidates of a predefined maximal size s which predict $\{x_i = v_{ik}\}$. This formula applies in the simplified case when every candidate predicts a value for the point of the measurement. The more general formula for the score computes first the probabilities $p(S_{ik})$ of the candidates S_{ik} supporting a specified possible value v_{ik} of the point x_i. The "rest" of the probability measure, $p(U_i)$, is then equally distributed onto all values. More formally:

$$\begin{aligned}\$'(x_i) = &\sum_{k=1}^{r_i} \left(p(S_{ik}) + \frac{p(U_i)}{r_i}\right) \log \left(p(S_{ik}) + \frac{p(U_i)}{r_i}\right) \\ &- p(U_i) \log \frac{1}{r_i},\end{aligned} \tag{11.28}$$

where S_{ik} denotes the set of diagnoses which predict that probing point x_i will obtain value v_{ik}, and U_i the set of candidates which predict no value for x_i.

De Kleer defines his score (or entropy) as a sum over candidates, and he restricts the size of the candidates to be considered to some fixed value in order to reduce computations. De Kleer also mentions that this approach is not optimal, as in some examples the optimal solution can only be computed if this restriction is relaxed (cf. page 388 of (De Kleer, 1990)). Another approach would therefore be a sum over all possible candidates, but the computation of such a sum is usually not possible due to the number of candidates.

Transferring these ideas to our framework, the idea is to replace, in the simplified approach (11.27), the number of candidates by the pignistic probabilities π_3 and so we define

$$H_2(x_i) = -\sum_{k=1}^{m} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) \log \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}). \tag{11.29}$$

In the more general approach (11.28) we replace the probabilities $p(S_{ik})$ of the set containing only the candidates of restricted size by the degrees of support:

$$H_3(x_i) = -\sum_{k=1}^{r_i} \Upsilon \left(dsp(\{x_i = v_{ik}\}, \mathcal{AS}) + \frac{rest_i}{r_i}\right) + rest_i \log \frac{1}{r_i},$$

where

$$\begin{aligned}\Upsilon(t) &:= t \log t \\ rest_i &:= 1 - \sum_{k=1}^{r_i} dsp(\{x_i = v_{ik}\}, \mathcal{AS}),\end{aligned}$$

or, using (11.8),

$$H_3(x_i) = -\sum_{k=1}^{r_i} \pi_2(\{x_i = v_{ik}\}, \mathcal{AS}) \log \pi_2(\{x_i = v_{ik}\}, \mathcal{AS}) + rest_i \log \frac{1}{r_i}.$$

As in the other methods, the point which maximizes H_2 resp. H_3 is the one which should be chosen for a further measurement.

11.7.3 Computing the Information about a Variable

Inspired by ideas in (Kohlas & Monney, 1994) and, again, the concept of pignistic probability (subsection 11.3.2), the information contained in an belief function relative to a variable which represents a point for a possible measurement can be computed. Using the mass function m defined in (11.9), we define the total information relative to the variable x_i as

$$H_4(x_i) = -\sum_{\emptyset \neq V \subseteq \{v_{i1},\dots,v_{ir_i}\}} m(M(x_i, V)) \log \frac{m(M(x_i, V))}{card(V)}. \tag{11.30}$$

The best next measurement is then the one with maximizes H_4.

11.8 Examples

We re-use the examples from sections 7.3 and 11.5 as well as others from (De Kleer & Williams, 1987; De Kleer, 1990) to illustrate the computations.

In the results shown below, the scores of those points where a measurement has already been taken are not mentioned, because they always have score 0 in every method. The best point, i.e. the point with the highest score, is marked in boldface.

Example 11.5: Three Serial Inverters

We continue the example 7.1. Every component is equally likely to fail and nothing is said about the faulty behavior of a component, so the best next measurement should (intuitively) be equidistant from the two measured points, i.e. the scores of the points x and y should be equal.

Let's for example compute the score of x using the first method H_1. The degree of support of x and of $\neg x$ can be computed using ABEL, $dsp(x, \mathcal{AS}) = 0.330$ and

$dsp(\neg x, \mathcal{AS}) = 0.663$, and the probability of the conflict set is $P(CS(\mathcal{AS})) = 0.99^3$, therefore

$$
\begin{aligned}
& H_1(x) \\
= \quad & \Big(1 - P(QSS(\perp, \mathcal{AS}))\Big) \sum_{k=1}^{2} \pi_3(\{x_i = v_{ik}\}, \mathcal{AS}) dsp(\neg\{x_i = v_{ik}\}, \mathcal{AS}) \\
= \quad & \Big(1 - P(QSS(\perp, \mathcal{AS}))\Big) \\
& \cdot \Big(\pi_3(x, \mathcal{AS}) dsp(\neg x, \mathcal{AS}) + \pi_3(\neg x, \mathcal{AS}) dsp(x, \mathcal{AS})\Big) \\
= \quad & (1 - 0.99^3)\Big((0.330 + 0.0035) \cdot 0.663 + (0.663 + 0.0035) \cdot 0.330\Big) \\
= \quad & 0.01310.
\end{aligned}
$$

The scores of the other points are computed similarly. The results of all methods are consistent with the idea above as shown in table 11.1. ⊖

Point	Utility			
	H_1	H_2	H_3	H_4
x	**0.01310**	**0.6366**	**0.6318**	**0.6763**
y	**0.01310**	**0.6366**	**0.6318**	**0.6763**

Table 11.1: Results for the three inverters example.

Example 11.6: Four Serial Inverters

Let's extend the example above by one inverter:

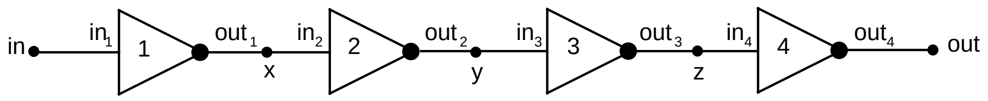


Figure 11.10: Four serial inverters.

```
(tell
   (module INVERTER ((var in out binary)
                     (ass ok binary 0.99))
       (-> ok (<-> out (not in))))

   (INVERTER in x  ok-1)
   (INVERTER x y   ok-2)
   (INVERTER y z   ok-3)
   (INVERTER z out ok-4))
```

Consider several different cases of measurements:

(a) No measurement has been taken. In this case the different methods do not predict a best point to measure at, but all scores are equal.

(b) Measurement *in*. Clearly *out* is the best point to measure at in this case.

(c) Measurements *in* and $\neg out$. These measurements conflict with the prediction (i.e. *out*) if all components are assumed to be working correctly. Because every component is equally likely to fail, this situation is symmetric, therefore the best point for a measurement is the one that is equidistant from the extremities, i.e. y.

(d) Measurements *in* and *out*. The measurements do not conflict with the prediction, but still there might be a double fault in the system, and again the best point to measure at is y.

(e) Measurements *in* and $\neg out$, but consider the special case where the components are not equally likely to fail: assume the component number 1 has probability 0.025 of a failure whereas the other ones still have 0.01. This implies that the point of the best next measurement is (intuitively) no more equidistant from the extreme points, but should be nearer to the component which is more likely to fail.

The results presented in table 11.2 show that all methods are consistent with the remarks above and propose the (intuitively) correct points. ⊖

Example 11.7: The "Three Buffers"

We continue the example 11.4. The values of the utility functions for measuring at a specified point are shown in table 11.3. Note that the results of the different methods are not the same.

As already shown in example 11.4, the optimal point to measure at is x, as more information is gained and better discrimination of the diagnoses is obtained. So in this case, the results of methods H_1 and H_3 appear to be better than the ones of method H_2 and H_4. In fact, if we define the path probabilities by (11.6) instead of (11.10) then the results of method H_3 are worse: the point `out2` is then computed as being the optimal point, whereas method 1 still computes the optimal solution; de Kleer already describes this problem in (De Kleer, 1990). ⊖

Example 11.8: Arithmetical Network

Consider the arithmetical network of example 7.2. The system has four minimal diagnoses:

$$\begin{array}{l} M(a_1, \{faulty\}) \\ M(m_1, \{faulty\}) \\ M(a_2, \{faulty\}) \wedge M(m_2, \{faulty\}) \\ M(m_2, \{faulty\}) \wedge M(m_3, \{faulty\}) \end{array}$$

Point	Utility			
	H_1	H_2	H_3	H_4
in	0.6931	0.0	0.6931	0.0
x	0.6931	0.0	0.6931	0.0
y	0.6931	0.0	0.6931	0.0
z	0.6931	0.0	0.6931	0.0
out	0.6931	0.0	0.6931	0.0

(a) No measurement.

Point	Utility			
	H_1	H_2	H_3	H_4
x	0.0050	0.0314	0.0245	0.0629
y	0.0098	0.0558	0.0420	0.1114
z	0.0144	0.0773	0.0567	0.1543
out	**0.0969**	**0.0696**	**0.1934**	**0.0189**

(b) Measurement *in*.

Point	Utility			
	H_1	H_2	H_3	H_4
x	0.0146	0.5635	0.5571	0.6056
y	**0.0195**	**0.6962**	**0.6862**	**0.7494**
z	0.0146	0.5635	0.5571	0.6056

(c) Measurements *in* and $\neg out$.

Point	Utility			
	H_1	H_2	H_3	H_4
x	0.00015	0.0015	0.0013	0.0029
y	**0.00020**	**0.0019**	**0.0016**	**0.0038**
z	0.00014	0.0015	0.0013	0.0029

(d) Measurements *in* and *out*.

Point	Utility			
	H_1	H_2	H_3	H_4
x	**0.02642**	**0.6894**	**0.6798**	**0.7619**
y	0.02459	0.6549	0.6459	0.7228
z	0.01578	0.4729	0.4675	0.5188

(e) Measurements *in* and $\neg out$ and $p(ok_1) = 0.975$.

Table 11.2: Results for the four inverters example.

Point	Utility			
	H_1	H_2	H_3	H_4
x	**0.09000**	**0.69315**	**0.6567**	0.8993
$out2$	0.08100	**0.69315**	0.5910	**1.1113**

Table 11.3: Results for the three buffers example.

The integer variables have been considered as infinite, but it makes sense to restrict them to a finite interval, therefore we restrict them to be finite, say maximally 15, that is

$$0 \leq a, b, c, d, e, f, g, x, y, z \leq 15. \tag{11.31}$$

The mass functions can be computed, and for example for the point x we have

$$m(M(x, K)) = \begin{cases} 0.58830 & K = \{4\} \\ 0.39111 & K = \{6\} \\ 0.00227 & K = \{0, \ldots, 10\} \\ 0.01832 & K = \{0, \ldots, 15\} \\ 0 & \text{otherwise} \end{cases}$$

and therefore the pignistic probability, needed for H_1 and H_2, is

$$\pi_3(\{x_i = v\}) = \begin{cases} 0.58965 & v = 4 \\ 0.39246 & v = 6 \\ 0.00135 & v = 1, 2, 3, 5, 7, \ldots, 10 \\ 0.00128 & v = 11, \ldots, 15. \end{cases}$$

The restriction of the value of y to the interval $[0..15]$ has created a focal set $K = \{0, \ldots, 10\}$ with positive mass. This is due to the fact that the observed output value $f = 10$ and the value of y being non-negative imply that x must be less than or equal to 10. If y was not restricted but just an integer, this focal set would be empty.

The values for the points y and z are computed similarly. The values of the different methods for computing best next measurements are presented in table 11.4, and again the results of the different methods are equivalent, even in this example where several different focal sets exist[3]. $\ominus$

11.9 Comparing the Approaches

So the big question now is: which of the approaches presented in section 11.7 should be chosen in a concrete situation?

[3]This is in contrast to the previous examples where all variables have been binary, therefore only four different focal sets could possibly be non-empty.

Point	Utility			
	H_1	H_2	H_3	H_4
x	**0.03939**	**0.8015**	**0.7414**	**0.8226**
y	0.00717	0.2099	0.2015	0.2179
z	0.00468	0.1625	0.1500	0.1679

Table 11.4: Results for the arithmetical network example.

The answer is not easy at all. In the special case of binary variables there are maximally four different focal sets for every variable x_i, therefore the approaches are quite similar to each other. Nevertheless, the examples in the previous subsection as well as other examples show that the algorithms H_1 and H_3 are usually better than the other approaches. In the general case, the approaches are quite different, and it is difficult to make general statements about their behavior.

In concrete situations, the limiting element usually is time. Therefore the decision trees presented in fig. 11.2 cannot be fully computed as in section 11.5, but techniques for computing only one step ahead in the tree, i.e. computing a best next measurement, have to be used as introduced in section 11.6. So the computation of a best next measurement has to be both fast and good, but not necessarily optimal. In view of this decision, we also can agree with the fact that the discrimination of the different diagnoses will not be obtained by a minimal set of measurements, but with some (only a few) more measurements, which — in the general case — is computationally easier than to look for an optimal solution. So in a concrete situation, the faster methods will be applied to compute a best next measurement.

12 An Implementation: ABEL

This chapter presents a brief introduction into the language ABEL, an Assumption-Based Evidential Language, following its main reference (Anrig *et al.*, 1997a). A main source for information about ABEL is its homepage

`http://www-iiuf.unifr.ch/tcs/abel`

(see (Haenni *et al.*, 1999a)). There some examples, an introduction and even the source code of the actual version as well as stand-alone applications for different platforms are available.

The development of the theory presented in this thesis and the language together with its solver have influenced each other strongly. The language is, in some sense, a way to formulate data for some specific types of argumentation systems.

The most popular numerical approaches to reasoning under uncertainty are the theory of Bayesian networks (Lauritzen & Spiegelhalter, 1988), the Dempster-Shafer theory of evidence (Shafer, 1976), and possibility theory (Dubois & Prade, 1990). Implementations for these systems are available. Further, there are various symbolical approaches, based on different non-monotonic logics. De Kleer proposes assumption-based truth maintenance systems (ATMS), a general architecture for problem solvers in the domain of uncertain reasoning (De Kleer, 1986a; De Kleer, 1986b), based essentially on propositional logic. As noted in (Laskey & Lehner, 1989; Provan, 1990), the ATMS can be combined with probability theory, which results in interesting additional dimensions for both parts.

ABEL provides a languages for describing argumentation systems with variables, based essentially on propositional and multi-valued variables. Therefore classical ATMS problems as well as Bayesian Networks can be formulated in ABEL. Numerous examples from different areas are presented in (Anrig *et al.*, 1997a). In ABEL, uncertainty is expressed by **assumptions**, a special type of variables. Assumptions represent unknown circumstances, risks, or interpretations. Often it is possible to assign probabilities to the values of an assumption.

Furthermore, assumptions are the basic elements to build arguments for hypotheses.

12.1 Implementation

Based on the idea of probabilistic assumption-based reasoning (Kohlas & Monney, 1993; Kohlas & Monney, 1995), a first prototypic language ABL was defined and implemented in a solver called EVIDENZIA (Lehmann, 1994) mainly restricted to propositional logic. After experiences made using this implementation and the approach implemented in (Hänni, 1997), discussion was started for a new, more general language, which was then called ABEL, Assumption-Based Evidential Language, see (Anrig *et al.*, 1997b; Anrig *et al.*, 1997a) and, in more detail, (Anrig *et al.*, 1999). As a basis for this language, many of the example applications discussed in these papers have been considered. The two major issues of this language are:

- ABEL defines a new and general language to express uncertain knowledge and corresponding queries.
- A general solver (also called ABEL) for this language has to be implemented.

Clearly, the language is quite general and also allows to describe unsolvable problems. First, reasoning with propositional variables was implemented and then successfully extended to finite set constraints and now also partially to discrete and continuos variables. Yet future steps will go further than that, and this will also require a further development of the language. The actual state of the solver is ABEL 2.2; its source code written in Common LISP (Steele, 1990), stand-alone applications for several platforms, and further up-to-date information can be found on the ABEL homepage (Haenni *et al.*, 1999a).

The implementation of this language led to a lot of problems in the field of computer science which had to be solved, efficient data structures, efficient hypertree generation, etc., see also (Haenni, 1996; Lehmann, 2000). The examples considered for the development of the language ABEL helped to see and solve problems which are not problematic in theory, but in practice, so for example different approximation strategies for numerical as well as symbolical computations are being developed for dealing with big problems, for example hierarchical, cost-bound, or modular focusing. However, these concepts are not yet implemented in ABEL 2.2. Also the concept of decision or best next measurements (chapter 11) are still subject to further research.

12.2 ABEL – the Language

In this section, a short introduction into the language ABEL is presented; for details, the grammar of ABEL, and further literature see (Anrig *et al.*, 1997b).

This section is mainly intended to help the reader to understand the examples presented in the previous chapters and in the following section.

The language is based on three other computer languages: from Common LISP (Steele, 1990) it adopts prefix notation, from Pulcinella (Saffiotti & Umkehrer, 1991) it uses the idea of the commands `tell`, `ask`, and `empty`, and from the existing ABEL prototype (Lehmann, 1994; Haenni, 1996) it inherits the concept of modules and the syntax of the queries.

For a given problem, there are usually two different types of data: stable data, or the problem formulation, which will usually be modeled in ABEL using the command `tell` (subsection 12.2.1). Then there are observations, or measurements, which will usually be modeled using the `observe` command (subsection 12.2.2); this type of data is, in contrast to the problem formulation, subject to change in time, etc. Queries about the available information, stable data and observations, are then formulated using the `ask` command (subsection 12.2.3). Other facilities which are useful for deleting parts of knowledge in the model or making comments are also part of the language, but will be skipped here.

12.2.1 Modeling Information

Stable information is modeled in ABEL by a sequence of **instructions** interpreted conjunctively and treated in parallel, that is

```
(tell <instr-1>
      <instr-2>
        ...
      <instr-n>)
```

An instruction is a definition of type, variables, assumptions, or modules, a statement, or an instance of a module (see below). Keywords may be added to name a `tell` command.

Domains, or **types**, of variables are declared using the identifier `type`, where `integer`, `real`, and `binary` denote pre-defined types which can also be constrained by limit values. Further, set types are defined by enumerating the elements. For example

```
(type test (passed failed))
(type colors (red green blue yellow))
(type month (1 2 3 4 5 6 7 8 9 10 11 12))
(type year integer)
(type month (integer 1 12))
(type size (real 0 250))
(type pos-integer (integer 0 *))
(type neg-real (real * 0))
```

Variables and **assumptions** are different identifiers. Assumptions represent variables which, in the context of an argumentation system, are within the language $\mathcal{FSC}$, whereas variables are in the language $\mathcal{L}$, therefore often a probability distribution is attached to assumptions. Variable and assumption definition is done by specifying the type of the variables and, in the case of assumptions, possibly probabilities. The type of a variable or an assumption is either a pre-defined type or defined by a `type` command. Examples:

```
(var c1 c2 colors)
(var a b n integer)
(var x y z real)
(var language (french german spanish english))
(var s (real 0 250))
(var pi 3.1416)
(var p q r binary)
(ass test1 test2 test (0.8 0.2))
(ass weather (sun clouds rain) (0.5 1/3 1/6))
(ass k1 k2 k3 colors)
(ass ok? binary 0.75)
```

So far, assumptions must have finite domains.

Numerical constants and variables allow to build compound algebraic **expressions** using the operators `+`, `-`, `*`, `/`, `sqr`, `sqrt`, `exp`, `expt`, `log`, `abs`, `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `mod`, `min`, and `max` in prefix notations. The semantics of these operators corresponds to Common LISP (Steele, 1990). Examples:

```
(+ x y z)
(/ (* x y) z)
(- (max x y z) (min x y z))
(abs (sin (sqrt x)))
```

Note that variables (e.g. `x`, `y`, `c1`, `language`), assumptions (e.g. `test1`, `test2`, `weather`, `ok?`), numbers (e.g. `17`, `1/3`, `-23.5`), symbols (e.g. `french`, `sun`), and sets (e.g. `(german spanish)`, `(1 2 4 5 10)`) are also considered as (atomic) expressions.

Expressions are then used to build **constraints** which are restrictions of the possible values of variables and assumptions. For this purpose, there are the operators `=`, `<>`, `<`, `<=`, `>`, `>=` with the usual meaning. Note that `=` and `<>` can also be applied to sets or variables with a set type, and `in` restricts a first parameter to be an element of the second parameter. Examples:

```
(= c1 blue)
(= n 17)
(<> (+ x y) z)
(in language (german spanish))
(< x 23.5)
(>= (/ (sin x) (cos x)) (tan x))
ok?
```

Variables and assumptions of type `binary` (e.g. `ok?`, `p`, `q`) are also considered as (atomic) constraints, and there are two predefined constraints `tautology` and `contradiction` which have the constant logical values true and false respectively.

Logical **statements** are built of constraints using the usual logical connectors `and`, `or`, `not`, `->`, `<->`, and `xor`, again using prefix notation. Every constraint is itself already an expression. Examples:

```
(and (= n 17) (= c1 blue))
(-> (= (+ x y) z) (in language (german spanish)))
(not (>= (/ (sin x) (cos x)) (tan x)))
ok?
```

A **module** is a step of abstraction in the modeling process. It has a unique name, a set of parameters and a body. Parameters can be variables or assumptions, the body is a sequence of ABEL instructions. Its syntax is:

```
(module <name> (<par-1> <par-2> ... <par-n>)
  <instr-1>
  <instr-2>
     ...
  <instr-n>)
```

Every set of parameters `<par-i>` is a specification of types of variables or assumptions just like for defining variables and assumptions above. So every parameter, variable, and assumption is typed.

An instance of a module with name `<name>` is created by

```
(<name> <p-1> <p-2> ... <p-n>)
```

where the `p-i`'s denote actual parameters, i.e. variables or assumptions. Note that the types of the actual parameters are implicitly given by the parameter specification of the module definition. It is therefore not necessary (but nevertheless possible) to specify the types of the actual parameters outside the module.

12.2.2 Modeling Observations

Observations are the part of the knowledge base which might change in time or context. So we distinguish them from the knowledge modeled by `tell` commands. This distinction is not reflected in the general theory of argumentation systems where both types are represented using the same structure, but for efficient handling of changing observations, clearly this information has to be modeled differently from the stable part of the system. ABEL provides a command `observe` to specify observations. Examples:

```
(observe (= n 17)
         (= c1 blue))
(observe (in language (german spanish)))
(observe ok?)
```

Keywords may be attached to observations like to `tell` commands.

12.2.3 Formulating Queries

The command `ask` is used for queries with respect to the actual knowledge base, that is the information modeled by `tell` and `observe` commands. A query is a general term for computing symbolical or numerical arguments in favor or against a hypothesis, that is minimal quasi-supports (`qs`), supports (`sp`), plausibilities (`pl`), doubts (`db`), and degrees of quasi-support (`dqs`), of support (`dsp`), of plausibility (`dpl`), and of doubt (`ddb`). Examples:

```
(ask (sp (and (= n 17) (< x 5))))
(ask (qs (in language (german spanish)))
    (sp (not ok?)))
(ask (dsp (and (= n 17) (< x 5))))
```

Arguments are conjunctions of normal or negated ABEL constraints over assumptions. The minimal support of a hypothesis, for example, is the set of all minimal conjunctions which allow to deduce the hypothesis from the given knowledge base.

A second type of a query is the marginalization of the knowledge base to one variable in order to get the information expressed using arguments with respect to this variable, e.g.

```
(ask language)
```

and the result is also a set of arguments.

12.3 Modeling using ABEL

Several examples have already been modeled in ABEL in the previous chapters, especially the examples in sections 7.3, 11.5.2 and 11.8. Many further examples in different domains can be found in (Anrig *et al.*, 1997a).

Here, we reconsider an example and show how modules can be used to add structural information to the model in ABEL.

Example 12.1: Continuation of example 7.1

Consider again the simple digital circuit built out of three serial inverters and connected as in fig. 7.1. The modeling presented in example 7.1 was:

```
(tell
   (type mode-type (ok faulty))
   (ass i1 i2 i3 mode-type (0.99 0.01))
   (var in x y out binary)

   (-> (= i1 ok) (<-> in (not x)))
   (-> (= i2 ok) (<-> x (not y)))
   (-> (= i3 ok) (<-> y (not out))))
```

The modular structure from the figure is not reflected in the modeling. However, the concept of module in ABEL allows to represent and name components as structures, i.e. a module `INVERTER` represents a general inverter component, which is then instantiated three times with the actual parameters:

```
(tell
   (type mode-type (ok faulty))

   (module INVERTER ((var in out binary))
      (ass i mode-type (0.99 0.01))
      (-> (= i ok) (<-> in (not out))))

   (INVERTER :inv1 in x)
   (INVERTER :inv2 x y)
   (INVERTER :inv3 y out))
```

From the view of the argumentation system, both models are equal, because the module structure is just a syntactic abbreviation. Yet from the computational point of view, the modules add information which can be used for more efficient construction of the hypertree as well as for modular approximations.

Input and output values are measured as 1, therefore, as in example 7.1, the observations are modeled using the `observe` command, to which the keywords `:input` and `:output` respectively have been added:

```
(observe :input in)
(observe :output out)
```

The keywords can later be used to delete or change the observations from the knowledge base independently of each other; for details see (Anrig *et al.*, 1997c).

Querying this knowledge base, the results are the same as in example 7.1, i.e.

```
? (ask (sp tautology) (qs (not x)) (qs (and x y)))
QUERY: (SP TAUTOLOGY)
  33.3% : (= INV2.I FAULTY)
  33.3% : (= INV1.I FAULTY)
  33.3% : (= INV3.I FAULTY)
QUERY: (QS (NOT X))
 100.0% : (= INV1.I OK)
QUERY: (QS (AND X Y))
 100.0% : (= INV2.I OK) (= INV3.I OK) (= INV1.I OK)
```

Note that a variable defined in a module is prefixed by the name of the instantiation of the module, for example the variable `i` of inverter `INV1` is denoted by `INV1.I`. ⊖

The examples presented in the previous chapters can also be reformulated more clearly using modules in the same spirit, for example 7.2. For other modeling issues see also (Anrig *et al.*, 1999).

Clearly, the modeling of a problem in ABEL (not only the use of modules) can significantly influence the computations, on one hand of the computation of the hypertree and on the other hand the message passing on the hypertree.

12.4 Model-Based Diagnostics using ABEL

In the previous section, we have shown how ABEL can be used for modeling problems, stating queries about variables in the model, and how the results are interpreted. Now, we go one step further and show how a diagnostic process looks in the framework of ABEL, how the results of the example of the chapters 7 and 11 are computed using ABEL. The main ingredients for this section have been discussed in the previous chapters, especially the concept of best next measurements in chapter 11.

We present model-based diagnostics in ABEL using three different examples which show the main capabilities of the software. A lot of other features cannot be presented in this limited context, see the homepage (Haenni *et al.*, 1999a) for plenty of further information.

Example 12.2: Continuation of example 12.1

In example 12.1, we have shown that the digital circuit of three inverters together with the actual observations leads to three equiprobable diagnoses,

```
? (ask (sp tautology))
QUERY: (SP TAUTOLOGY)
  33.3% : (= INV2.I FAULTY)
  33.3% : (= INV1.I FAULTY)
  33.3% : (= INV3.I FAULTY)
```

that is, one of the inverters is not working correctly.

Usually, this result is considered as not being informal enough, especially because the probabilities of the minimal diagnoses are equal, hence the user has no information on which of them he should choose. So additional information has to be gained from the circuit and introduced into the ABEL model. In the present situation, additional information means especially a measurement at a specific point in the circuit, that is either at point x or at point y, because these are the only points left where a measurement of a variable can take place; the other variables of the model denote unobservable entities like `INV1.I`. Now, techniques presented in section 11.6 are applied to compute a best next measurement point. We will use here the method H_1 (11.25) and follow section 11.8. So first, we have to compute the degrees of support of the values of the points, that is

```
? (ask (dsp x) (dsp (not x)) (dsp y) (dsp (not y)))
QUERY: (DSP X)
  0.330
```

```
QUERY: (DSP (NOT X))
  0.663
QUERY: (DSP Y)
  0.330
QUERY: (DSP (NOT Y))
  0.663
```

Using (11.25), the scores of the two points are computed:

$$
\begin{aligned}
H_1(x) &= \alpha\Big((0.330 + 0.0035) \cdot 0.663 + (0.663 + 0.0035) \cdot 0.330\Big) &= 0.441\alpha \\
H_1(y) &= \alpha\Big((0.330 + 0.0035) \cdot 0.663 + (0.663 + 0.0035) \cdot 0.330\Big) &= 0.441\alpha
\end{aligned}
$$

Note that the multiplicative factor α has not to be computed, because we are only interested in the proportion of the two values $H_1(x)$ and $H_1(y)$.

Due to the identical degrees of support computed above, both points have the same score, $H_1(x) = H_1(y)$, and we can arbitrary select one of them, say x, where the additional measurement has to be made. Suppose now that the measured value is x (that is the value is positive). This new information is modeled as an argumentation system as well (see example 7.1), and its representation can be described in ABEL. The combined argumentation system consists then of the ABEL code above together with the new information modeled using the `observe`-command:

```
(observe x)
```

Now the diagnoses have to be recomputed, because additional knowledge is available:

```
? (ask (sp tautology))
QUERY: (SP TAUTOLOGY)
 100.0% : (= INV1.I FAULTY)
```

Fortunately, there is only one diagnosis left indicating that the first inverter is not working correctly and the diagnosis process comes to an end after one additional measurement at x. This result corresponds to the one computed in example 7.1.

Yet in general, one additional measurement is not enough. Consider for example the case where the result of the measurement is not the value x as above but $\neg x$ instead. This means, that the additional information for ABEL is modeled by the `observe`-command:

```
(observe (not x))
```

and the new diagnoses are

```
? (ask (sp tautology))
QUERY: (SP TAUTOLOGY)
  50.0% : (= INV2.I FAULTY)
  50.0% : (= INV3.I FAULTY)
```

that is, there are still two equiprobable diagnoses. The process of next measurements is now iterated, but as there is only one point left for a measurement, namely y, the selection problem is trivial and the score of y is not computed. Suppose, that the value measured is $\neg y$, therefore the additional information modeled in ABEL is

```
(observe (not y))
```

and, computing the diagnoses

```
? (ask (sp tautology))
QUERY: (SP TAUTOLOGY)
 100.0% : (= INV2.I FAULTY)
```

we see that there is only one diagnosis left which means that the second component is not working correctly, and the diagnosis process comes to an end.

Now, consider the situation where the result of the second measurement is not $\neg y$ but y, that is its model in ABEL is

```
(observe y)
```

In this case, the resulting diagnosis is

```
? (ask (sp tautology))
QUERY: (SP TAUTOLOGY)
 100.0% : (= INV3.I FAULTY)
```

that is, the third component is not working correctly. ⊖

The previous example shows that several measurements have to be made in order to obtain a single minimal diagnosis. Yet in general, the user is not always interested in obtaining a single diagnosis, but nevertheless one which is quite probable and which is significantly more probable than the other ones. The difference required between the most probable and the other ones depends highly on the concrete situation and the user. Yet often it is not even possible to reduce the number of diagnoses to just one, even if any number of additional measurements have been made.

Example 12.3: Continuation of example 7.2

The arithmetical network together with the observations has four minimal diagnoses, as computed already in example 7.2, that is using ABEL, we have

```
? (ask (sp tautology))
QUERY: (SP (NOT NETWORK-OK))
  59.5% : (NOT M1)
  35.7% : (NOT A1)
   3.0% : (NOT M2) (NOT M3)
   1.8% : (NOT A2) (NOT M2)
```

Now, either this is enough information for the user and he decides to repair the circuit according to the most probable diagnosis, that is he replaces component m_1, or he wants a better discrimination between the diagnoses. In the latter case, additional information about the circuit has to be added, and again this means that a further measurement has to be taken. In example 11.8 we showed that the best next measurement hast to be taken at point x. So suppose that the value measured at point x is 6, that is we add the following to the ABEL model:

```
(observe (= x 6))
```

Now the minimal diagnoses with respect to the updated knowledge base are computed, and we get

```
? (ask (sp tautology))
QUERY: (SP (NOT NETWORK-OK))
  88.2% : (NOT A1)
   7.4% : (NOT M2) (NOT M3)
   4.4% : (NOT A2) (NOT M2)
```

that is, one very probable diagnosis (component a_1 is faulty) and two rather improbable diagnoses. Usually, the diagnosis process stops here, because the user is satisfied of the discrimination between the different diagnoses and chooses to replace component a_1. But in critical situations, this discrimination is not enough, and further information has to be gathered, that is further measurements have to be taken so as to reduce the number of diagnoses and improve the discrimination. Further measurements can take place at point y and z, so assume that the user decides to make another measurement at y whose result is 6, that is we add

```
(observe (= y 6))
```

to the ABEL model and get then

```
? (ask (sp tautology))
QUERY: (SP (NOT NETWORK-OK))
 100.0% : (NOT A1)
```

that is only one diagnosis is left and the diagnosis process is stopped. ⊖

Example 12.4: Communication Network

Consider a communication network which consists of nodes connected by communication wires, some of them one-directional, some of them bi-directional. If the wire is intact, then communication from its start node to its end node is possible, if it is not intact, then communication through this wire is not possible. But instead, communication may be possible through other paths in the network.

Assume that we have a communication network like in fig. 12.1. The nodes are labeled by a, b, u, v, w, x, and y, the wires by $w_0, \ldots, w_9$. We look now at the situation where a communication from node a to node b has to be guaranteed, that is at least all wires of one of the paths from a to b have to be working.

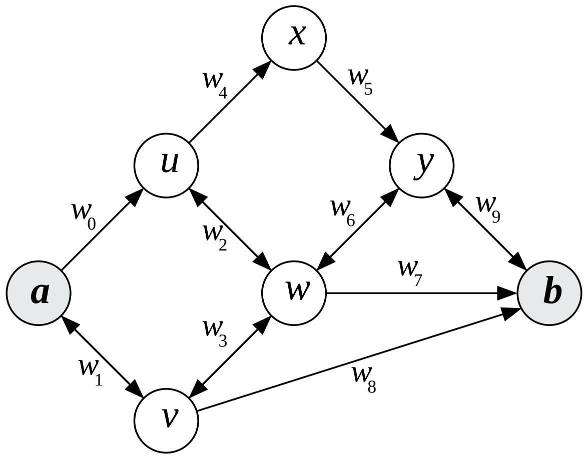


Figure 12.1: Example of a communication network.

The model in ABEL begins with the declaration of the variables and assumptions, which have additional probabilities attached to them representing the availability of the respective wire. As in previous examples, binary assumption `oki` denotes the working mode of wire w_i (with probability 0.6 for $i = 0$, etc.).

```
(tell
  (var a b u v w x y binary)
  (ass ok0 binary 0.6)
  (ass ok1 binary 0.9)
  (ass ok2 binary 0.9)
  (ass ok3 binary 0.3)
  (ass ok4 binary 0.5)
  (ass ok5 binary 0.8)
  (ass ok6 binary 0.9)
  (ass ok7 binary 0.4)
  (ass ok8 binary 0.2)
  (ass ok9 binary 0.7))
```

The network is then modeled by a set of implications (and equivalences for bi-directional communications), so for example the first implication describes that if the wire w_0 is ok (denoted by `ok0`) then the communication from a to u is possible, `(-> a u)`.

```
(tell
  (-> ok0 (-> a u))
  (-> ok1 (<-> a v))
  (-> ok2 (<-> u w))
  (-> ok3 (<-> v w))
  (-> ok4 (-> u x))
  (-> ok5 (-> x y))
  (-> ok6 (<-> w y))
  (-> ok7 (-> w b))
  (-> ok8 (-> v b))
  (-> ok9 (<-> y b)))
```

ABEL can now compute the reliability of the connection from a to b, the connectivity of the network, etc., but here we are interested in diagnostic questions. For example, consider the situation that someone at a wants to talk to someone at b, but the message from a does not arrive at b; this can be modeled in ABEL as:

```
(observe (not (-> a b)))
```

The possible diagnoses are then:

```
? (ask (sp tautology))
QUERY: (SP TAUTOLOGY)
   44.2% : (NOT OK0) (NOT OK3) (NOT OK8)
   28.4% : (NOT OK7) (NOT OK8) (NOT OK9)
    7.9% : (NOT OK0) (NOT OK1)
    5.5% : (NOT OK2) (NOT OK3) (NOT OK4) (NOT OK8)
    4.7% : (NOT OK4) (NOT OK6) (NOT OK7) (NOT OK8)
    2.5% : (NOT OK1) (NOT OK3) (NOT OK7) (NOT OK9)
    2.2% : (NOT OK2) (NOT OK3) (NOT OK5) (NOT OK8)
    1.9% : (NOT OK5) (NOT OK6) (NOT OK7) (NOT OK8)
    1.0% : (NOT OK1) (NOT OK2) (NOT OK4)
    0.4% : (NOT OK1) (NOT OK3) (NOT OK4) (NOT OK6) (NOT OK7)
    0.4% : (NOT OK1) (NOT OK2) (NOT OK5)
    0.4% : (NOT OK0) (NOT OK2) (NOT OK6) (NOT OK7) (NOT OK8)
    0.3% : (NOT OK2) (NOT OK3) (NOT OK6) (NOT OK8) (NOT OK9)
    0.2% : (NOT OK1) (NOT OK3) (NOT OK5) (NOT OK6) (NOT OK7)
    0.1% : (NOT OK1) (NOT OK2) (NOT OK6) (NOT OK9)
```

A diagnosis is a cut between the nodes a and b, that is a minimal set of wires which, if all are not working, explain the behavior of the system (Kohlas, 1987; Beichelt, 1993). The first diagnosis means that a communication between a and b fails when the wires w_0, w_3, and w_8 simultaneously fail. Assume now that the user is not satisfied by this result, i.e. he wants a better discrimination of the diagnoses. So an additional measurement has to be made. Suppose here that it is easy to check if the message has arrived at a node, but rather hard to check if the wire itself is functioning. So we consider only the nodes as possible points for a measurement. Again, we use method H_1 (11.25). First, we compute the degrees of support of the possible values of the nodes, that is

```
? (ask (dsp u) (dsp (not u)) (dsp v) (dsp (not v))
       (dsp w) (dsp (not w)) (dsp x) (dsp (not x))
       (dsp y) (dsp (not y)))
QUERY: (DSP U)
  0.354
QUERY: (DSP (NOT U))
  0.481
QUERY: (DSP V)
  0.863
QUERY: (DSP (NOT V))
  0.049
QUERY: (DSP W)
  0.300
QUERY: (DSP (NOT W))
  0.555
QUERY: (DSP X)
  0.149
QUERY: (DSP (NOT X))
  0.471
QUERY: (DSP Y)
  0.249
QUERY: (DSP (NOT Y))
  0.602
```

and then, using (11.25), we get

	u	v	w	x	y
$H_1(\cdot)$	**0.409**	0.125	0.395	0.258	0.362

Therefore, the next measurement should be made at node u. Assume that the value measured is u, thus in ABEL:

```
(observe u)
```

Again, the diagnoses are computed:

```
? (ask (sp tautology))
QUERY: (SP TAUTOLOGY)
  59.7% : (NOT OK7) (NOT OK8) (NOT OK9)
  11.6% : (NOT OK2) (NOT OK3) (NOT OK4) (NOT OK8)
   9.9% : (NOT OK4) (NOT OK6) (NOT OK7) (NOT OK8)
   5.2% : (NOT OK1) (NOT OK3) (NOT OK7) (NOT OK9)
   4.6% : (NOT OK2) (NOT OK3) (NOT OK5) (NOT OK8)
   4.0% : (NOT OK5) (NOT OK6) (NOT OK7) (NOT OK8)
   2.1% : (NOT OK1) (NOT OK2) (NOT OK4)
   0.9% : (NOT OK1) (NOT OK3) (NOT OK4) (NOT OK6) (NOT OK7)
   0.8% : (NOT OK1) (NOT OK2) (NOT OK5)
   0.7% : (NOT OK2) (NOT OK3) (NOT OK6) (NOT OK8) (NOT OK9)
   0.3% : (NOT OK1) (NOT OK3) (NOT OK5) (NOT OK6) (NOT OK7)
   0.1% : (NOT OK1) (NOT OK2) (NOT OK6) (NOT OK9)
```

Now, the most probable diagnosis is much more probable than the second one. Further information can be gained by computing the degrees of support of the individual wires, that is

```
? (ask (dsp (not ok0)) (dsp (not ok1)) (dsp (not ok2))
       (dsp (not ok3)) (dsp (not ok4)) (dsp (not ok5))
       (dsp (not ok6)) (dsp (not ok7)) (dsp (not ok8))
       (dsp (not ok9)))
QUERY: (DSP (NOT OK0))
  0.400
QUERY: (DSP (NOT OK1))
  0.123
QUERY: (DSP (NOT OK2))
  0.231
QUERY: (DSP (NOT OK3))
  0.741
QUERY: (DSP (NOT OK4))
  0.582
QUERY: (DSP (NOT OK5))
  0.233
QUERY: (DSP (NOT OK6))
  0.189
QUERY: (DSP (NOT OK7))
  0.924
QUERY: (DSP (NOT OK8))
  0.980
QUERY: (DSP (NOT OK9))
  0.773
```

The wires w_8, w_7, and w_9, which occur in the most probable diagnosis, but also w_3, have a rather high degree of support of not working. It is therefore better to replace these components or at least check them. Depending on the costs of the measurements of wires and the checking or replacing of components, the strategy will be different. $\ominus$

12.5 The Future

Several concepts are to be integrated in ABEL. First, some sort of meta-language has to be developed in order to allow the user to explicitly specify the type of the solver (numerical versus symbolical propagation), the approximation method (cost-bound, modular, or hierarchical), and the degree of approximation (usually a numerical value). Further work is being done in automatic determination of these parameters from the model and the actual query.

Further work is also being done in the framework of decisions. Actual decision algorithms have to be incorporated into ABEL for automatic and user aided decision making under uncertainty. The approach "Electre" described in (Roy, 1985) might be useful for this task.

A new architecture is being developed for testing and integrating this and other new approaches for the ABEL language. It is called the ABEL workbench. For further developments on this subject see the ABEL homepage (Haenni *et al.*, 1999a).

References

Abraham, J.A. 1979. An Improved Algorithm for Network Reliability. *IEEE Transactions on Reliability*, **28**, 58–61. 139

Aiken, A., Kozen, D., Vardi, M., & Wimmers, E. 1994. The Complexity of Set Constraints. *Lecture Notes in Computer Science*, **832**. 66

Anrig, B., & Monney, P.A. 1999. Using Propositional Logic to Compute the Probability of Diagnoses in Multistate Systems. *Int. J. of Approximate Reasoning*, **20**(2), 113–143. 108, 139

Anrig, B., Haenni, R., Kohlas, J., & Monney, P.A. 1996. Probabilistic Analysis of Model-Based Diagnosis. *Pages 123–128 of: IPMU'96, Proc. 6th int. conf., Granada, Spain.* 2

Anrig, B., Haenni, R., & Lehmann, N. 1997a. *ABEL — A New Language for Assumption-Based Evidential Reasoning under Uncertainty.* Tech. Rep. 97–01. University of Fribourg, Institute of Informatics. 211, 212, 216

Anrig, B., Haenni, R., Kohlas, J., & Lehmann, N. 1997b. Assumption-based Modeling using ABEL. *In:* Gabbay, D., Kruse, R., Nonnengart, A., & Ohlbach, H.J. (eds), *First Int. Joint Conf. on Qualitative and Quantitative Practical Reasoning, ECSQARU–FAPR'97*, Springer, for Lecture Notes in Artif. Intell. 212

Anrig, B., Lehmann, N., & Haenni, R. 1997c. *Reasoning with Finite Set Constraints.* Tech. Rep. 97–11. University of Fribourg, Institute of Informatics. 3, 65, 66, 139, 217

Anrig, B., Bissig, R., Haenni, R., Kohlas, J., & Lehmann, N. 1999. *Probabilistic Argumentation Systems: Introduction to Assumption-Based Modeling with ABEL.* Tech. Rep. 99-1. Institute of Informatics, University of Fribourg. 95, 101, 117, 212, 217

Arnborg, S., Corneil, D., & Proskurowski, A. 1987. Complexity of Finding Embedings in a k-Tree. *SIAM J. of Algebraic and Discrete Methods*, **8**, 277–284. 124

Barlow, R.E., & Proschan, R. 1975. *Statistical Theory of Reliability and Life Testing.* New York. 95, 117

Beckert, B., Hähnle, R., & Manyá, F. 1999. Transformations between Signed and Classical Clause Logic. *Pages 248–255 of: Proc. 29th Int. Symposium on Multiple-Valued Logics, Freiburg, Germany.* IEEE Press, Los Alamitos. 66

Beichelt, F. 1993. *Zuverlässigkeits- und Instandhaltungstheorie.* Teubner, Stuttgart. 95, 223

Berge, C. 1989. *Hypergraphs.* North Holland. 124

Bertschy, R., & Monney, P.A. 1996. A Generalization of the Algorithm of Heidtmann to Non-Monotone Formulas. *J. of Computational and Applied Mathematics*, **76**, 55–76. 143

Besnard, P., & Kohlas, J. 1995. Evidence Theory Based on General Consequence Relations. *Int. J. of Foundations of Computer Science*, **6**(2), 119–135. 46, 49

Birkhoff, G. 1948. *Lattice Theory.* New York: American Mathematical Society. 77, 156, 162

Birkhoff, G., & Bartee, T.C. 1970. *Modern Applied Algebra.* Mc-Graw Hill, New York. 74

Bissig, R., Kohlas, J., & Lehmann, N. 1997. Fast-Division Architecture for Dempster-Shafer Belief Functions. *In:* Gabbay, D., Kruse, R., Nonnengart, A., & Ohlbach, H.J. (eds), *First Int. Joint Conf. on Qualitative and Quantitative Practical Reasoning, ECSQARU–FAPR'97*, Springer, for Lecture Notes in Artif. Intell. 133

Cano, A., & Moral, S. 1995. Heuristic Algorithms for the Triangulation of Graphs. *Pages 166–171 of:* Bouchon-Meunier, B., Yager, R.R., & Zadeh, L.A. (eds), *IPMU'95, Proc. 5th int. conf.* Springer. 124

Davey, B.A., & Priestley, H.A. 1990. *Introduction to Lattices and Order.* Cambridge University Press. 59

Davis, M., & Putnam, H. 1962. A Computing Procedure for Quantification Theory. *J. of the ACM*, **5**, 394–397. 64

Davis, M., & Putnam, H. 1983. A Computing Procedure for Quantification Theory. *Pages 125–139 of:* Siekmann, J., & Wrightson, G. (eds), *Automation of Reasoning 1: Classical Papers on Computational Logic 1957–1966.* Berlin, Heidelberg: Springer. 64

Davis, R. 1984. Diagnostic Reasoning based on Structure and Behaviour. *Artif. Intell.*, **24**, 347–410. 1, 105

De Kleer, J. 1976. *Local Methods for Localizing Faults in Electronical Circuits.* MIT AI Memo 394, MIT Cambridge, MA. 1

De Kleer, J. 1986a. An Assumption-based TMS. *Artif. Intell.*, **28**, 127–162. 2, 211

De Kleer, J. 1986b. Extending the ATMS. *Artif. Intell.*, **28**, 163–196. 2, 211

De Kleer, J. 1990. Using Crude Probability Estimates to Guide Diagnosis. *Artif. Intell.*, **45**, 381–391. 191, 194, 196, 202, 203, 204, 206

De Kleer, J. 1993. A Perspective on Assumption-based Truth Maintenance. *Artif. Intell.*, **59**, 63–67. 2

De Kleer, J., & Williams, B.C. 1987. Diagnosing Multiple Faults. *Artif. Intell.*, **32**, 97–130. 1, 2, 97, 100, 105, 185, 191, 204

De Kleer, J., Mackworth, A. K., & Reiter, R. 1992a. Characterizing Diagnoses and Systems. *Artif. Intell.*, **56**, 197–222. 96

De Kleer, J., Raiman, O., & Shirley, M. 1992b. One Step Lookahead is Pretty Good. *Pages 138–142 of:* Hamscher, W., Console, L., & De Kleer, J. (eds), *Readings in Model-based Diagnosis.* Morgan Kaufmann. 4, 185, 199, 200

Dempster, A. 1967. Upper and Lower Probabilities Induced by a Multivalued Mapping. *Ann. Math. Stat.*, **38**, 325–339. 2, 34, 39, 43

Dubois, D., & Prade, H. 1982. On Several Representations of an Uncertain Body of Evidence. *Pages 167–181 of:* Gupta, M.M., & Sanchez, E. (eds), *Fuzzy Information and Decision Processes.* North Holland. 186

Dubois, D., & Prade, H. 1990. An Introduction to Possibilistic and Fuzzy Logics. *Pages 742–761 of:* Shafer, G., & Pearl, J. (eds), *Readings in Uncertain Reasoning.* San Mateo, CA: Kaufmann. 211

Genesereth, M.R. 1984. The Use of Design Description in Automated Diagnosis. *Artif. Intell.*, **24**, 411–436. 1

Gilleron, R., Tison, S., & Tommasi, M. 1993. Solving systems of set constraints using tree automata. *Lecture Notes in Computer Science*, **665**. 66

Grätzer, G. 1978. *General Lattice Theory.* Academic Press. 59

Gries, D., & Schneider, F.B. 1993. *A Logical Approach to Discrete Math.* Springer-Verlag. 69

Guiasu, S. 1977. *Information theory with applications.* McGraw-Hill. 202

Haenni, R. 1995. A Valuation-Based Architecture for Assumption-Based Reasoning. *Pages 255–258 of:* Coletti, G., Dubois, D., & Scozzafava, R. (eds), *Mathematical Models for Handling Partial Knowledge in Artificial Intelligence.* Plenum Press. 123

Haenni, R. 1996. *Propositional Argumentation Systems and Symbolic Evidence Theory*. Ph.D. thesis, University of Fribourg, Institute of Informatics. 1, 74, 77, 123, 124, 156, 162, 212, 213

Haenni, R. 1997. Assumption-based Reasoning with Algebraic Clauses. *In:* Zimmermann, H.J. (ed), *EUFIT'97, 5th European Congress on Intelligent Techniques and Soft Computing*. Verlag Mainz. 83

Haenni, R. 1998. Generating Diagnoses from Conflict Sets. *Pages 120–124 of:* Cook, Diane J. (ed), *Proceedings of the FLAIRS-98 Conference*. AAAI Press. 77

Haenni, R., & Lehmann, N. 1998a. Assumption-Based Reasoning with Finite Set Constraints. *Pages 1289–1295 of: IPMU'98, Proc. 7th int. conf., Paris, France*. 66, 83

Haenni, R., & Lehmann, N. 1998b. Reasoning with Finite Set Constraints. *Pages 1–6 of: ECAI'98, Workshop W17: Many-valued logic for AI application*. 66

Haenni, R., & Lehmann, N. 1999. *Efficient Hypertree Construction*. Tech. Rep. 99-2. Institute of Informatics, University of Fribourg. 124

Haenni, R., Anrig, B., Bissig, R., & Lehmann, N. 1999a. *ABEL homepage*. `http://www-iiuf.unifr.ch/tcs/abel`. 211, 212, 218, 226

Haenni, R., Kohlas, J., & Lehmann, N. 1999b. *Probabilistic Argumentation Systems*. Tech. Rep. 99-9. Institute of Informatics, University of Fribourg. 82

Hähnle, R. 2000. Advanced Many-Valued Logics. *In:* Gabbay, Dov (ed), *Handbook of Philosophical Logic*, second edn., vol. 6: Alternatives to Classical Logic I. Kluwer, Dordrecht. 66

Hähnle, R., & Escalada-Imaz, G. 1997. Deduction in Many-Valued Logics: a Survey. *Mathware & Soft Computing*, **IV**(2), 69–97. 66

Halmos, P. 1963. *Lectures on Boolean Algebras*. Van Nostrand-Reinhold. 15, 54

Hänni, U. 1997. *Implementation und Anwendung einer logik-basierten, symbolischen Evidenz-Theorie*. Ph.D. thesis, University of Fribourg, Institute of Informatics. 169, 170, 212

Heidtmann, K.D. 1989. Smaller Sums of Disjoint Products by Subproduct Inversion. *IEEE Transactions on Reliability*, **38**(3), 305–311. 143

Hertelendy, P. 1997. *Resolution of Sparse Systems of Linear Equations using Local Computations*. Student Project. University of Fribourg, Institute of Informatics. 58, 64

Hou, A. 1994. A Theory of Measurement in Diagnosis from First Principles. *Artif. Intell.*, **65**, 281–382. 151

Howard, R.A., & Matheson, J.E. 1981. Influence Diagrams. *Pages 720–762 of: Principles and Applications of Decision Analysis.* Strategic Dec. Group, Menlo Park CA. 178

Imbert, J.L. 1995. Fourier's Elimination: Which to Choose? *Pages 245–268 of:* Saraswat, V., & van Heutenryck, P. (eds), *Principles and Practice of Constraint Programming.* MIT Press. 64

Inoue, K. 1991. An Abductive Procedure for the CMS/ATMS. *Pages 34–53 of:* Martins, J.P., & Reinfrank, M. (eds), *Truth Maintenance Systems, Lecture Notes in Artif. Intell.* Springer. 169, 170

Inoue, K. 1992. Linear Resolution for Consequence Finding. *Artif. Intell.*, **56**, 301–353. 96, 169, 170

Kalt, M. 1997. *Lösen von linearen Ungleichungssystemen in Valuations-Netzen.* Diplomarbeit. University of Fribourg, Institute of Informatics. 58, 64

Kohlas, J. 1987. *Zuverlässigkeit und Verfügbarkeit.* Teubner. 95, 223

Kohlas, J. 1993a. Support-and Plausibility Functions Induced by Filter-Valued Mappings. *Int. J. of General Systems*, **21**(4), 343–363. 33, 34

Kohlas, J. 1993b. Symbolic Evidence, Arguments, Supports and Valuation Networks. *Pages 186–198 of:* Clarke, M., Kruse, M., & Moral, S. (eds), *Symbolic and Quantitative Aproaches to Reasoning and Uncertainty.* Springer. 123

Kohlas, J. 1995. Mathematical Foundations of Evidence Theory. *Pages 31–64 of:* Coletti, G., Dubois, D., & Scozzafava, R. (eds), *Mathematical Models for Handling Partial Knowledge in Artificial Intelligence.* Plenum Press. 2, 9, 17, 18, 20, 32, 34, 39, 41, 53, 54, 55, 186

Kohlas, J. 1997a. Allocation of Arguments and Evidence Theory. *Theoretical Computer Science*, **171**, 221–246. 2, 46, 49, 61, 94

Kohlas, J. 1997b. *Computational Theory for Information Systems.* Tech. Rep. 97–07. University of Fribourg, Institute of Informatics. 3, 47, 57, 58, 61, 62, 63, 64, 123, 124, 126, 131

Kohlas, J. 1997c. *Uncertainty in Information Systems.* Internal Paper, University of Fribourg, Institute of Informatics. 7, 127

Kohlas, J., & Brachinger, H.W. 1995. Argumentation Systems and Evidence Theory. *In:* Bouchon-Meunier, B., Yager, R.R., & Zadeh, L.A. (eds), *Advances in Intelligent Computing.* Springer Verlag. 2, 47

Kohlas, J., & Monney, P.A. 1993. Probabilistic Assumption-Based Reasoning. *In:* Heckermann, & Mamdani (eds), *Proc. 9th Conf. on Uncertainty in Artif. Intell.* Kaufmann, Morgan Publ. 1, 2, 96, 169, 170, 212

Kohlas, J., & Monney, P.A. 1994. Theory of Evidence – a Survey of its Mathematical Foundations, Applications and Computational Analysis. *ZOR – Mathematical Methods of Operations Research*, **39**, 35–68. 2, 204

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. 1, 2, 3, 39, 40, 41, 43, 44, 75, 124, 139, 212

Kohlas, J., & Stärk, R.F. 1996a. *Eine mathematische Theorie der Informationssysteme*. Tech. Rep. 96–12. University of Fribourg, Institute of Informatics. 2, 3, 7, 8, 10, 12, 13, 57, 90, 123

Kohlas, J., & Stärk, R.F. 1996b. *Information Algebras and Information Systems*. Tech. Rep. 96–14. University of Fribourg, Institute of Informatics. 2, 3, 7, 8, 10, 11, 13, 14, 15, 34, 35, 57, 59, 60, 62, 90, 123

Kohlas, J., Monney, P.A., Haenni, R., & Lehmann, N. 1995. Model-Based Diagnostics Using Hints. *Pages 259–266 of:* Fridevaux, Ch., & Kohlas, J. (eds), *Symbolic and Quantitative Approaches to Uncertainty, European Conference ECSQARU'95, Fribourg*. Springer. 2

Kohlas, J., Anrig, B., Haenni, R., & Monney, P.A. 1998. Model-Based Diagnostics and Probabilistic Assumption-Based Reasoning. *Artif. Intell.*, **104**, 71–106. 1, 2, 4, 75, 96, 97, 98, 101, 105, 186

Kohlas, J., Haenni, R., & Lehmann, N. 1999a. *Computing Probabilities of Events in Bayesian Networks*. Internal paper, University of Fribourg, Institute of Informatics. 154

Kohlas, J., Haenni, R., & Moral, S. 1999b. Propositional Information Systems. *Journal of Logic and Computation*, **9 (5)**, 651–681. 58, 64, 123, 154, 157

Kohlas, J., Haenni, R., & Lehmann, N. 2000. Probabilistic Argumentation Systems. *In:* Kohlas, J., & Moral, S. (eds), *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, vol. 5: Algorithms for Uncertainty and Defeasible Reasoning. Kluwer, Dordrecht. 1, 44, 45, 63, 75, 82, 96, 124

Kozen, D. 1994. Logical Aspects of Set Constraints. *Lecture Notes in Computer Science*, **832**. 66

Laskey, K.B., & Lehner, P.E. 1989. Assumptions, Beliefs and Probabilities. *Artif. Intell.*, **41**, 65–77. 97, 211

Lauritzen, S.L., & Shenoy, P.P. 1995. *Computing Marginals Using Local Computation*. Working Paper 267. School of Business, University of Kansas. 123

Lauritzen, S.L., & Spiegelhalter, D.J. 1988. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. *J. of Royal Stat. Soc.*, **50**(2), 157–224. 4, 123, 211

Lehmann, N. 1994. *Entwurf und Implementation einer annahmenbasierten Sprache.* Diplomarbeit. Institute of Informatics, University of Fribourg. 212, 213

Lehmann, N. 2000. *Probabilistic Approaches to Assumption-Based Reasoning.* Ph.D. thesis *in preparation*, University of Fribourg, Institute of Informatics. 2, 134, 212

Lehmann, N., & Haenni, R. 1999. An Alternative to Outward Propagation for Dempster-Shafer Belief Functions. *Pages 256–267 of:* Hunter, A., & Parsons, S. (eds), *European Conf. ECSQARU'99, London*, Springer, for Lecture Notes in Artif. Intell. 134, 154

Lepar, V., & Shenoy, P.P. 1998. A Comparison of Lauritzen-Spiegelhalter, Hugin and Shenoy-Shafer Architectures for Computing Marginals of Probability Distributions. *Uncertainty in Artif. Intell.*, **14**, 328–337. 133

Miller, A.C., Merkhofer, M.M., Howard, R.A., Matheson, J.E., & Rice, T.R. 1976. *Development of Automated Aids for Decision Analysis.* Technical Report. SRI International, Menlo Park, California. 178

Monney, P.A. 1994. *An Abstract Theory of Argumentation.* Tech. Rep. University of Fribourg, Institute of Informatics. 40, 82

Monney, P.A. 2000. *Assumption-Based Reasoning with Functional Models.* Habilitation Thesis *submitted*, University of Fribourg, Seminar of Statistics. 41

Monney, P.A., & Anrig, B. 1998. Computing the Probability of Formulas Representing Events in Product Spaces. *Pages 1724–1731 of: IPMU'98, Proc. 7th int. conf., Paris, France.* 137, 139, 140, 143, 147

Monney, P.A., & Anrig, B. 2000. Computing the Probability of Formulas Representing Events in Product Spaces. *Pages 197–208 of:* Bouchon-Meunier, B., Yager, R.R., & Zadeh, L.A. (eds), *Information, Uncertainty and Fusion.* Kluwer Academic Publishers. 139

Ngair, T.H. 1992. *Convex Spaces as an Order-theoretic Basis for Problem Solving.* Ph.D. thesis, University of Pennsylvania. 75

Olmsted, S. 1983. *On representing and solving decision problems.* Ph.D. thesis, Department of Engineering-Economic Systems, Stanford University. 178

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publ. Inc. 133

Provan, G.M. 1990. A Logic-Based Analysis of Dempster-Shafer Theory. *Int. J. of Approximate Reasoning*, **4**, 451–495. 97, 211

Raiffa, H. 1968. *Decision Analysis, Introductory Lectures on Choices under Uncertainty.* Addison-Wesley, Massachusetts. 178, 179, 189

Reggia, J.A., Nau, D.S., & Wang, Y. 1983. Diagnostic Expert Systems Based on Set Covering Model. *Int. J. Man-Machine Stud.*, **19**, 437–460. 1

Reggia, J.A., Nau, D.S., & Wang, Y. 1985. A Formal Model of Diagnostic Inference: 1. Problem Formulation and Decomposition. *Inf. Sci.*, **37**, 227–256. 1

Reiter, R. 1987. A Theory of Diagnosis From First Principles. *Artif. Intell.*, **32**, 57–95. 1, 4, 96, 97, 100, 105, 175

Reiter, R., & De Kleer, J. 1987. Foundations of Assumption-Based Truth Maintenance Systems. *Proc. of the American Association in Artif. Intell.*, 183–188. 77, 156, 162

Roy, B. 1985. *Méthodologie Multicritère d'Aide à la Décision*. Ed. Economica, Paris. 225

Saffiotti, A., & Umkehrer, E. 1991. *PULCINELLA: A General Tool for Propagating Uncertainty in Valuation Networks*. Tech. Rep. IRIDIA, Université de Bruxelles. 213

Schachter, R.D. 1986a. Evaluating Influence Diagrams. *Operations Research*, **33**(6), 871–882. 178

Schachter, R.D. 1986b. Probabilistic Inference and Influence Diagrams. *Operations Research*, **36**, 589–605. 178

Scott, D.S. 1982. Domains for denotational semantics. *Pages 577–613 of:* Nielsen, M., & Schmitt, E.M. (eds), *Automata, Languages and Programming*. Springer. 57, 58

Shafer, G. 1976. *The Mathematical Theory of Evidence*. Princeton University Press. 2, 3, 32, 34, 186, 211

Shafer, G. 1979. Allocations of Probability. *Ann. of Prob.*, **7**, 827–839. 18, 34, 36, 46, 49

Shafer, G. 1991. *An Axiomatic Study of Computation in Hypertrees*. Working Paper 232. School of Business, University of Kansas. 7, 123, 124

Shannon, C.E. 1948. A Mathematical Theory of Communications. *The Bell System Technical Journal*, **27**, 379–432. 199, 202

Shenoy, P.P. 1989. A Valuation-Based Language For Expert Systems. *Int. J. of Approximate Reasoning*, **3**, 383–411. 4, 123

Shenoy, P.P. 1992a. Valuation-Based Systems: A Framework for Managing Uncertainty in Expert Systems. *Pages 83–104 of:* Zadeh, L.A., & Kacprzyk, J. (eds), *Fuzzy Logic for the Management of Uncertainty*. John Wiley & Sons. 178

Shenoy, P.P. 1992b. Valuation-Based Systems for Bayesian Decision Analysis. *Operations Research*, **40**(3), 463–484. 178

Shenoy, P.P. 1995. *Binary Join Trees*. Tech. Rep. 270. School of Business, University of Kansas. 133

Shenoy, P.P., & Kohlas, J. 2000. Computation in Valuation Algebras. *In:* Kohlas, J., & Moral, S. (eds), *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, vol. 5: Algorithms for Uncertainty and Defeasible Reasoning. Kluwer, Dordrecht. 123

Shenoy, P.P., & Shafer, G. 1990. Axioms for Probability and Belief Functions Propagation. *In:* Shachter, R.D., Levitt, T.S., Kanal, L.N., & Lemmer, J.F. (eds), *Uncertainty in Artif. Intell. 4*. North Holland. 7, 123, 125, 127

Siegel, P. 1987. *Représentation et Utilisation de la Connaissance en Calcul Propositionel*. Ph.D. thesis, Université d'Aix-Marseille II. Luminy, France. 96, 169, 170

Sikorski, R. 1960. *Boolean Algebras*. Springer, Berlin-Göttingen-Heidelberg. 16

Simonaitus, D.F., Anderson, R.T., & Kaye, M.P. 1972. Reliability evaluation of a heart assist system. *In: Proc. of the 1972 Annual Reliability and Maintainability Symposium, San Francisco*. 117

Smets, P. 1990. Decisions and Belief Functions. *In: Joint National Meeting of TIMS/ORSA, New York, 1990*. 186, 187

Smets, P. 1999. *Transferable Belief Model*. Unpublished Notes of a Seminar at the Institute of Informatics, University of Friborg, Switzerland. 2

Smets, P., & Kennes, R. 1990. Constructing the Pignistic Probability Function in a Context of Uncertainty. *Uncertainty in Artif. Intell.*, **5**, 29–39. 186, 187

Smets, P., & Kennes, R. 1994. The Transferable Belief Model. *Artif. Intell.*, **66**, 191–234. 2, 186, 187

Stärk-Lepar, V. 1999. *Performance of Architectures for Local Computations in Bayesian Networks*. Ph.D. thesis, University of Fribourg, Institute of Informatics. 133

Steele, G. L. 1990. *Common Lisp – the Language*. Digital Press. 212, 213, 214

Strat, T.M. 1990. Decision Analysis using Belief Functions. *Int. J. of Approximate Reasoning*, **4**, 391–418. 187

Tarski, A. 1957. Über einige fundamentale Begriffe der Metamathematik. *Transl. in:* Woodger, J.H. (ed), *Logic, Semantics, Metamathematics*. Kluwer Academic Press, Dordrecht. 57, 59

Williams, H.P. 1976. Fourier-Motzkin Elimination Extension to Integer Programming Problems. *J. Combinatorial Theory*, **21**, 118–123. 64

Xu, H. 1995. Computing Marginals for Arbitrary Subsets from Marginal Representation in Markov Trees. *Artif. Intell.*, **74**, 177–189. 163, 164, 165, 167

Xu, H. 2000. Network-Based Decision Algorithms. *In:* Kohlas, J., & Moral, S. (eds), *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, vol. 5: Algorithms for Uncertainty and Defeasible Reasoning. Kluwer, Dordrecht. 177

Index

Curriculum Vitae

Geboren wurde ich am 21. August 1969 in Schaffhausen, Schweiz.

Nach dem Besuch der Primar- und Sekundarschulen von 1976 bis 1984 in Thayngen, bin ich im Frühjahr 1984 in die Kantonsschule in Schaffhausen aufgenommen worden und habe diese im Sommer 1989 mit einer Matura Typus B abgeschlossen.

Von 1990 bis 1994 habe ich an der Universität von Freiburg i.Ue. Mathematik studiert und im Dezember 1994 wurde mir das Diplom in Mathematik mit Nebenfach Informatik überreicht. Das Thema meiner Diplomarbeit, welche von Prof. Holmann betreut wurde, war die Newton-Iteration in der komplexen Ebene.

Von 1993 bis 1995 war ich Unterassistent, anschliessend Assistent am interfakultären Institut für Informatik der Universität Freiburg i.Ue. in der Gruppe für Theoretische Informatik unter der Leitung von Prof. Kohlas, welcher auch meine Forschung und Doktorarbeit betreut hat.