Probabilistic Argumentation Systems[*]

R. Haenni, J. Kohlas, & N. Lehmann

Institute of Informatics
University of Fribourg
CH–1700 Fribourg, Switzerland

Phone: +41 (26) 300 83 31
Fax: +41 (26) 300 97 26
E-Mail: rolf.haenni@unifr.ch

October 23, 1999

1 Introduction

Different formalisms for solving problems of inference under uncertainty have been developed so far. The most popular numerical approach is the theory of **Bayesian inference** [42]. More general approaches are the Dempster-Shafer **theory of evidence** [51], and **possibility theory** [16], which is closely related to fuzzy systems. For these systems computer implementations are available. In competition with these numerical methods are different symbolic approaches. Many of them are based on different types of non-monotonic logic.

From a practical point of view, De Kleer's idea of **assumption-based truth maintenance systems** (ATMS) gives a general architecture for problem solvers in the domain of reasoning under uncertainty [14, 15]. One of its advantages is that it is based on classical propositional logic. In contrast, most systems

[*]Research supported by grants No. 21-42927.95 and 21-342927.95 of the Swiss National Foundation for Research.

based on non-monotonic logic abandon the framework of classical logic. As a consequence, ATMS (easier than non-classical logical systems) can be combined with probability theory, which gives both probability theory as well as ATMS an interesting additional dimension. This has formerly been noted in [40] and [49].

The idea of combining classical logic with probability theory leads to a more general theory of **probabilistic argumentation systems** [3, 22]. This theory is an alternative approach for non-monotonic reasoning under uncertainty. It allows to judge open questions (hypotheses) about the unknown or future world in the light of the given knowledge. From a qualitative point of view, the problem is to derive **arguments** in favor and against the hypothesis of interest. An argument can be seen as a chain of possible events that makes the hypothesis true. Finally, a quantitative judgement of the situation is obtained by considering probabilities that the arguments are valid. The **credibility** of a hypothesis can then be measured by the total probability that it is supported by arguments. The resulting **degree of support** corresponds to (normalized) **belief** in the theory of evidence [35, 51, 55, 58]. A quantitative judgement is often more useful and can help to decide whether a hypothesis can be accepted, rejected, or whether the available knowledge does not permit to decide.

A fundamental property of the theory is that additional knowledge may cause the judgement of the situation to change non-monotonically. Clearly, the property of **non-monotonicity** is required in any mathematical formalism for reasoning under uncertainty. It reflects a natural property of how a human's conviction or belief can change when new information is added. The theory of probabilistic argumentation systems shows that non-monotonicity can be achieved without leaving the field of classical logic.

A particular system called ABEL [4, 5] is an example of a probabilistic argumentation systems. Examples from a broad spectrum of application domains show that ABEL is a general and powerful tool for reasoning under uncertainty [3]. It includes an appropriate modeling and query language, as well as corresponding inference mechanisms. The system has on an open architecture, that permits the later inclusion of further deduction techniques.

The purpose of this chapter is to present the main theoretical and computational concepts of probabilistic argumentation systems. The text is organized as follows: Subsection 1.1 explains the general idea and the motivation of the theory; Subsection 1.2 discusses from a practical point of view the problem of representing uncertainty by propositional logic; Subsection 1.3 uses an introductory example to illustrate the main terms and concepts; Section 2 introduces formally the theoretical model of probabilistic argumentation systems on the basis of propositional logic; Section 3 presents appropriate techniques for computing sets of arguments efficiently; Section 4 describes an alternative approach for computing numerical degrees of support by generating families of independent belief functions; Section 5 sketches the generalization of the theoretical model and the computational techniques to systems with non-binary variables; Section 6, finally, contains some concluding remarks and an outlook.

1.1 The General Idea

The basic idea of the theory is very simple. It is supposed that the part of the knowledge needed to solve a given problem can be described by a number of **variables**. Usually, every piece of information concerns only a restricted number of variables. The knowledge can then be encoded by **constraints**, **restrictions**, or, more generally, by **relations** on these variables. Relations are commonly described by **logical** expressions or **mathematical** equations or inequalities. For centuries, this way of encoding knowledge has been proved to be a convenient and efficient method for describing and solving problems in many conceivable fields.

The same idea of encoding knowledge is still applicable if parts of the available knowledge are uncertain. For example, it can happen that a given information is only valid if some additional conditions or circumstances are satisfied. Such a situation can be described by introducing a special type of so-called **environmental variables**. They are used to represent unpredictable conditions or circumstances, possible interpretations, unknown risks, interference factors, errors of measurement, and so on. The true value of an environmental variable is supposed to be determined by independent external circumstances or influences which are not further specified.

A collection of relations on "normal" and environmental variables can be useful for answering or judging open **questions** or **hypotheses** in the light of the given uncertain knowledge. Intuitively, judging a hypothesis means weighing the pros and cons. More precisely, the problem is to find **arguments** supporting the hypothesis, respectively **counter-arguments** refuting the hypothesis. An argument is like a chain of possible events or a particular combination of circumstances that allows to deduce the truth or the falsity of the hypothesis from the given knowledge. The basic elements for building arguments are the environmental variables. Restrictions on environmental variables are called **assumptions**. Finally, arguments are combinations of assumptions supposed to hold.

Example 1.1 *Suppose that John's birthday party only takes place if his fever has disappeared and if it is not raining. This piece of information can be modeled, for example, on the basis of three variables* $f \in \{\text{yes}, \text{no}\}$*,* $r \in \{\text{yes}, \text{no}\}$*, and* $p \in \{\text{yes}, \text{no}\}$ *(*f *stands for* "fever"*,* r *stands for* "rain"*,* p *stands for* "party"*).*

f and w are designated to be environmental variables. The relation between the variables can be expressed by a logical expression $\langle f = \text{no}\rangle \wedge \langle r = \text{no}\rangle \leftrightarrow \langle p = \text{yes}\rangle$*. Because only binary variables are used in the model, it is possible to consider f, r, and p as propositions and to express the given knowledge more conveniently as $\neg f \wedge \neg r \leftrightarrow p$. From this point of view, f and r are assumptions which are the basic elements for building arguments. The hypothesis to be judged, that is the question whether the party takes place or not, can be expressed by p. Clearly, $\neg f \wedge \neg r$ is a supporting argument for p, whereas f and r are counter-arguments.*

A more seizable judgement of the hypothesis can be obtained, if every environmental variable is an independent **random variable** with a corresponding probability distribution. An argument can then be weighed quantitatively by computing the probability that the argument is valid. Clearly, every argument or counter-argument provides an additional reason to believe or disbelieve the hypothesis. Therefore, the credibility of a hypothesis can be measured by the total probability that it is supported or refuted by arguments. More precisely, the **degree of support** of a hypothesis is the probability that at least one supporting argument is valid. Similarly, the **degree of possibility** of a hypothesis is the probability that no argument against the hypothesis is valid. Such a quantitative judgement of hypotheses is often more useful and can help to decide whether a hypothesis can be accepted, rejected, or whether the available knowledge does not permit to decide.

Example 1.2 *Consider the story of John's birthday party in Example 1.1. Suppose that the environmental variables f and r are random variables with the following estimated probabilities:* $p(f = \text{yes}) = 0.2$, $p(f = \text{no}) = 0.8$, $p(r = \text{yes}) = 0.4$ *and* $p(r = \text{no}) = 0.6$*. Clearly, the probability of the supporting argument is* $p(\neg f \wedge \neg r) = 0.8 \cdot 0.6 = 0.48$*. Note that $\neg f \wedge \neg r$ is the only supporting argument for p. The degree of support of p is therefore* 0.48.

The realization of these general ideas presupposes the specification of a **formal system** that allows to encode the given knowledge. The system must permit to define variables with corresponding sets of possible values. Somehow, a clear distinction between "normal" and environmental variables must be supported, and it should be possible to indicate corresponding probability distributions for the environmental variables. Furthermore, the systems must provide an adequate language for expressing all sorts of relations between the variables and also for the formulation of the hypotheses. Finally, an appropriate inference mechanism must allow to evaluate hypotheses qualitatively and quantitatively. A system that satisfies all these requirements is called a **probabilistic argumentation system**. ABEL is an example of a probabilistic argumentation system. The descriptive power of ABEL has been demonstrated by a rich collection of examples from different fields [3].

Often, it is convenient to restrict probabilistic argumentation systems to the case of **binary variables**. The reason for this is that the theoretical model becomes more compact and the computational effort remains relatively small. Moreover, there is still a number of interesting applications which are covered by this restricted case. Obviously, **propositional logic** is an expressive language for describing systems with binary variables. The main topic of this chapter is to develop a theory of argumentation systems on the basis of propositional logic. More general argumentation systems are based on the notion of **set constraints** [6, 23, 24]. This way of generalizing the theory is sketched in Section 5. Other systems are possible, for example those based on linear equations and inequalities, but they will not be discussed here (see [21]).

1.2 Representing Uncertainty by Propositional Logic

Propositional logic is one of the simplest and most convenient ways of encoding knowledge. The problem is that pure propositional logic, at first sight, seems to be unsuitable for representing uncertainty. However, uncertainty can be captured very easily by considering particular propositions called **assumptions**. In this subsection, representing uncertainty by assumptions will be discussed from a general point of view. Later on, in Subsection 1.3, the use of this technique will be illustrated by an introductory example.

The simplest cases of propositional knowledge are **facts** and **simple rules**. For example, if the proposition p_1 stands for an arbitrary statement, then p_1 represents the fact that the statement p_1 is true. Similarly, $\neg p_1$ represents that fact that the statement p_1 is false. Furthermore, if p_2 is a second proposition, then $p_1 \rightarrow p_2$ represent a simple rule of the form "if p_1 is true, then p_2 is also true". Similarly, $p_1 \rightarrow \neg p_2$, $\neg p_1 \rightarrow p_2$, and $\neg p_1 \rightarrow \neg p_2$ are other simple rules for p_1 and p_2. Thus, facts and simple rules can easily be handled by propositional logic . However, facts and rules depend often on unknown conditions or circumstances and are therefore not fully reliable. For example, if p_1 represents a testimony in court, then the truth of p_1 depends on the credibility of the witness. Such cases of **uncertain facts** and **uncertain simple rules** can be handled as shown in Table 1.1:

The additional proposition a_1 is an assumption. It represents the unknown conditions or circumstances on which the facts and rules depend on. Note that $a_1 \wedge p_1 \rightarrow p_2$ is an equivalent expression for uncertain simple rules.

More general cases of uncertain knowledge are handled similarly. Let γ be an arbitrary propositional formula that expresses somehow the relation between different propositions. The corresponding case where γ is not fully reliable can then be represented by $a_1 \rightarrow \gamma$. Furthermore, it may be possible to distinguish between independent circumstances. For example, $a_1 \wedge a_2 \rightarrow \gamma$ represents a

Type of Knowledge	**Logical Representation**	**Natural Language**
fact	p_1	p_1 is true
uncertain fact	$a_1 \rightarrow p_1$	p_1 is true under some circumstances a_1
simple rule	$p_1 \rightarrow p_2$	p_1 implies p_2
uncertain simple rule	$a_1 \rightarrow (p_1 \rightarrow p_2)$	p_1 implies p_2 under some circumstances a_1

Table 1.1: Representing uncertain facts and rules.

situation where γ depends simultaneously on different circumstances a_1 and a_2. From a general point of view, uncertainty is therefore captured by arbitrary propositional formulas containing assumptions.

1.3 Introductory Example

The general idea of probabilistic argumentation systems will now be illustrated more extensively by discussing the problem of detecting faulty components of a broken technical system. Suppose that the technical system to be considered is a logical circuit with four components C_1 (logical *or*-gate), C_2 (inverter), C_3 (first logical *and*-gate), and C_4 (second logical *and*-gate). Clearly, it is possible that any of the four components behaves abnormally with a certain prior probability. Such an abnormal behavior of the i-th component can be modeled by a proposition ab_i. $A = \{ab_1, ab_2, ab_3, ab_4\}$ is the set of all such propositions. The elements of A are also called **assumptions**. Furthermore, if a truth value $x_i \in \{0,1\}$ is associated with each assumption ab_i, indicating whether the component behaves abnormally ($x_i = 1$) or correctly ($x_i = 0$), then every Boolean vector $\mathbf{s} = (x_1, \ldots, x_4)$ represents a possible state of the system. In a more general framework, system states are also called **scenarios**. $N_A = \{0,1\}^4 = \{\mathbf{s}_0, \ldots, \mathbf{s}_{15}\}$ denotes the set of all possible scenarios:

$$\begin{array}{llll}
\mathbf{s}_0 = (0,0,0,0), & \mathbf{s}_4 = (0,1,0,0), & \mathbf{s}_8 = (1,0,0,0), & \mathbf{s}_{12} = (1,1,0,0), \\
\mathbf{s}_1 = (0,0,0,1), & \mathbf{s}_5 = (0,1,0,1), & \mathbf{s}_9 = (1,0,0,1), & \mathbf{s}_{13} = (1,1,0,1), \\
\mathbf{s}_2 = (0,0,1,0), & \mathbf{s}_6 = (0,1,1,0), & \mathbf{s}_{10} = (1,0,1,0), & \mathbf{s}_{14} = (1,1,1,0), \\
\mathbf{s}_3 = (0,0,1,1), & \mathbf{s}_7 = (0,1,1,1), & \mathbf{s}_{11} = (1,0,1,1), & \mathbf{s}_{15} = (1,1,1,1).
\end{array}$$

A particular scenario $\hat{\mathbf{s}} \in N_A$ is supposed to represent the true but unknown state of the system. Furthermore, if failure probabilities $\pi_i = p(x_i = 1)$ are known for the four components, then the prior probability of a particular scenario $\mathbf{s}$ to be the true scenario is given by

$$p(\mathbf{s}) \;=\; \prod_{i=1}^{4} {\pi_i}^{x_i} \cdot (1-\pi_i)^{(1-x_i)}.$$

For example, suppose that $\pi_1 = 0.1$, $\pi_2 = 0.2$, $\pi_3 = 0.3$, and $\pi_4 = 0.3$ are the failure probabilities of the components. The prior probabilities of the individual scenarios are then

$$\begin{array}{llll}
p(\mathbf{s}_0) = 0.3528, & p(\mathbf{s}_4) = 0.0648, & p(\mathbf{s}_8) = 0.0392, & p(\mathbf{s}_{12}) = 0.0098, \\
p(\mathbf{s}_1) = 0.1512, & p(\mathbf{s}_5) = 0.0378, & p(\mathbf{s}_9) = 0.0168, & p(\mathbf{s}_{13}) = 0.0042, \\
p(\mathbf{s}_2) = 0.1512, & p(\mathbf{s}_6) = 0.0378, & p(\mathbf{s}_{10}) = 0.0168, & p(\mathbf{s}_{14}) = 0.0042, \\
p(\mathbf{s}_3) = 0.0882, & p(\mathbf{s}_7) = 0.0162, & p(\mathbf{s}_{11}) = 0.0072, & p(\mathbf{s}_{15}) = 0.0018.
\end{array}$$

The situation becomes more interesting, if additional knowledge about the system is considered. Suppose that this additional knowledge can be expressed by propositional sentences. For example, if the four components C_1 to C_4 are connected as shown in Figure 1.1, then the system can be described by

$$\begin{array}{lllllll}
\xi_1 & = & \neg ab_1 \rightarrow (a \vee v \leftrightarrow x), & \qquad & \xi_3 & = & \neg ab_3 \rightarrow (c \wedge d \leftrightarrow w), \\
\xi_2 & = & \neg ab_2 \rightarrow (\neg x \leftrightarrow v), & & \xi_4 & = & \neg ab_4 \rightarrow (v \wedge w \leftrightarrow y),
\end{array}$$

where $P = \{a, b, c, u, v, x, y\}$ is a second set of propositions involved in the system description.

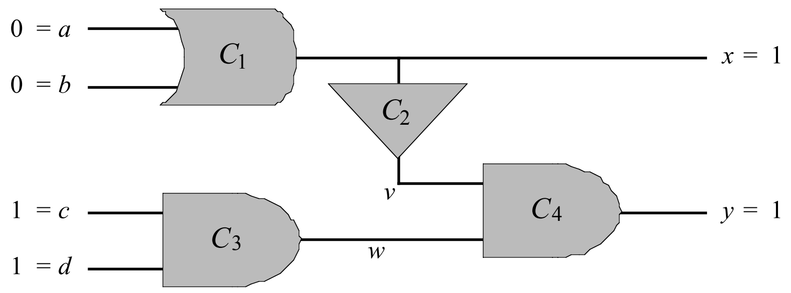

Figure 1.1: A faulty digital circuit with four components.

Additional information about the system comes from observing actual input and output values. In Figure 1.1, input and output values are observed for a, b, c, d, x, and y. This additional information can be encoded by the following propositional sentences:

$$\xi_5 \;=\; \neg a, \qquad \xi_6 \;=\; \neg b, \qquad \xi_7 \;=\; c, \qquad \xi_8 \;=\; x, \qquad \xi_9 \;=\; y.$$

The description of the system is now complete. The conjunction $\xi = \xi_1 \wedge \cdots \wedge \xi_9$ is called **knowledge base**, and the tuple $\mathcal{PAS}_P = (\xi, P, A, \Pi)$, $\Pi = \{\pi_1, \ldots, \pi_4\}$, is a **probabilistic argumentation system**. Note that for the given knowledge base, some **inconsistent** scenarios can be excluded. For example, the scenario $\mathbf{s}_0 = (0,0,0,0)$ is no longer allowed because it represents the case where all components are intact. This is clearly not possible in the situation of Figure 1.1, where the observed values for x and y are not compatible with the correct behavior of the system. For the same reason, it is possible to exclude the scenarios $\mathbf{s}_1$ to $\mathbf{s}_8$, $\mathbf{s}_{10}$, $\mathbf{s}_{13}$, and $\mathbf{s}_{14}$. Finally, a set $C_A(\xi) = \{\mathbf{s}_9, \mathbf{s}_{11}, \mathbf{s}_{12}, \mathbf{s}_{15}\}$ of **consistent** scenarios remains, and the true scenario $\hat{\mathbf{s}}$ is either

$$\mathbf{s}_9 = (1,0,0,1),\ \mathbf{s}_{11} = (1,0,1,1),\ \mathbf{s}_{12} = (1,1,0,0),\ \text{or } \mathbf{s}_{15} = (1,1,1,1).$$

Clearly, excluding inconsistent scenarios must also influence the quantitative judgement of the situation. In accordance with probability theory, this means that the prior probability measure on the set of scenarios N_A must be conditioned on the fact that the true scenario $\hat{\mathbf{s}}$ is in $C_A(\xi)$. This leads to the new probability measure p' given by

$$p'(\mathbf{s}) \ = \ p(\mathbf{s}|C_A(\xi)) \ = \ \begin{cases} k^{-1} \cdot p(\mathbf{s}), & \text{if } \mathbf{s} \in C_A(\xi), \\ 0, & \text{otherwise.} \end{cases}$$

The prior probabilities of the consistent scenarios $\mathbf{s}_9$, $\mathbf{s}_{11}$, $\mathbf{s}_{12}$, and $\mathbf{s}_{15}$ are therefore divided by the normalization factor

$$k \ = \sum_{\mathbf{s} \in C_A(\xi)} p(\mathbf{s}) \ = \ p(\mathbf{s}_9) + p(\mathbf{s}_{11}) + p(\mathbf{s}_{12}) + p(\mathbf{s}_{15}) \ = \ 0.0356.$$

This leads to the following posterior probabilities:

$$\begin{aligned} p'(\mathbf{s}_9) &= \tfrac{0.0168}{0.0356} = 0.472, & p'(\mathbf{s}_{12}) &= \tfrac{0.0098}{0.0356} = 0.275, \\ p'(\mathbf{s}_{11}) &= \tfrac{0.0072}{0.0356} = 0.202, & p'(\mathbf{s}_{15}) &= \tfrac{0.0018}{0.0356} = 0.051. \end{aligned}$$

The posterior failure probabilities of the components C_i can now be computed by considering the corresponding assumptions ab_i as **hypotheses** to be judged in the light of the given situation. More generally, hypotheses are arbitrary propositional sentences with symbols in A and P.

Scenarios for which a given hypothesis h certainly becomes true are called **supporting scenarios** for h. For example, $\mathbf{s}_9$ is a supporting scenario for ab_1 because $\mathbf{s}_9$ is consistent and it implies that C_1 is working abnormally. The complete sets of supporting scenarios are

$$\begin{aligned} SP_A(ab_1, \xi) &= \{\mathbf{s}_9, \mathbf{s}_{11}, \mathbf{s}_{12}, \mathbf{s}_{15}\}, & SP_A(ab_3, \xi) &= \{\mathbf{s}_{11}, \mathbf{s}_{15}\}, \\ SP_A(ab_2, \xi) &= \{\mathbf{s}_{12}, \mathbf{s}_{15}\}, & SP_A(ab_4, \xi) &= \{\mathbf{s}_9, \mathbf{s}_{11}, \mathbf{s}_{15}\}. \end{aligned}$$

Sets of scenarios can be represented more efficiently by minimal conjunctions of literals of assumptions. For example, $\{\mathbf{s}_9, \mathbf{s}_{11}, \mathbf{s}_{15}\}$ can be represented by two conjunctions $ab_1 \wedge \neg ab_2 \wedge ab_4$ and $ab_1 \wedge ab_3 \wedge ab_4$. Such conjunctions are called **supporting arguments** for h. A minimal supporting argument represents a sufficient condition or a minimal chain of possible events that makes the hypothesis true.

Finally, **degrees of support** for the hypotheses ab_1 to ab_4 are obtained by adding the corresponding posterior probabilities of the supporting scenarios:

$$\begin{aligned}
dsp(ab_1, \xi) &= p'(\mathbf{s}_9) + p'(\mathbf{s}_{11}) + p'(\mathbf{s}_{12}) + p'(\mathbf{s}_{15}) \;=\; 1,\\
dsp(ab_2, \xi) &= p'(\mathbf{s}_{12}) + p'(\mathbf{s}_{15}) \;=\; 0.326,\\
dsp(ab_3, \xi) &= p'(\mathbf{s}_{11}) + p'(\mathbf{s}_{15}) \;=\; 0.253,\\
dsp(ab_4, \xi) &= p'(\mathbf{s}_9) + p'(\mathbf{s}_{11}) + p'(\mathbf{s}_{15}) \;=\; 0.725.
\end{aligned}$$

The above results can be interpreted as posterior failure probabilities of the individual components given the system description and the observed input and output values. Component C_1 is certainly broken. Furthermore, $dsp(ab_2 \vee ab_3 \vee ab_4, \xi) = 1$ implies that one of the components C_2, C_3, or C_4 is broken simultaneously with C_1. Most probably, C_2 is the second broken component.

2 Propositional Argumentation Systems

This section introduces formally the fundamental concepts of probabilistic argumentation systems. The model will be based on propositional logic. Therefore, only binary variables will be allowed at the beginning. The reason for this restriction is to make the theory more compact. Later, in Section 5, it will be shown that the complete theoretical model and also the computational techniques of Section 3 can be generalized to systems with non-binary variables.

2.1 Propositional Logic

Propositional logic deals with declarative statements that can be either true or false. Such statements are called **propositions**. Let $P = \{p_1, \ldots, p_n\}$ be a finite set of propositions. The symbols $p_i \in P$ are called **atoms** or **atomic formulas**. The impossible statement (**contradiction** or **falsity**) is denoted by $\bot$, and $\top$ represents the statement that is always true (**tautology**). Compound formulas are built by the following syntactic rules:

(1) atoms, $\bot$, and $\top$ are formulas;

(2) if γ is a formula, then $\neg\gamma$ is a formula;

(3) if γ and δ are formulas, then $(\gamma \wedge \delta)$, $(\gamma \vee \delta)$, $(\gamma \rightarrow \delta)$, and $(\gamma \leftrightarrow \delta)$ are formulas.

Often, unnecessary parentheses can be omitted, e.g. $\gamma \wedge \delta$ instead of $(\gamma \wedge \delta)$. Furthermore, by assigning priority in decreasing ordering $\neg$, $\wedge$, $\vee$, $\rightarrow$, some other parentheses can be eliminated, e.g. $\gamma \rightarrow \delta \wedge \lambda$ instead of $\gamma \rightarrow (\delta \wedge \lambda)$. The set $\mathcal{L}_P$ of all formulas generated by the above recursive rules is called **propositional language** over P. A formula $\gamma \in \mathcal{L}_P$ is also called **propositional sentence**.

2.1.1 Semantics

The meaning of a propositional sentence is obtained by assigning truth values 0 (false) or 1 (true) to the propositions. The truth value of a compound formula can then be obtained according to Table 2.2:

An assignment of truth values to the elements of a set $P = \{p_1, \ldots, p_n\}$ is called **interpretation** relative to P. $N_P = \{0,1\}^n$ denotes the set of all 2^n different interpretations. Every interpretation $\mathbf{x} \in N_P$ can be seen as a point or a vector $\mathbf{x} = (x_1, \ldots, x_n)$ in the n-dimensional binary product space N_P. Each $x_i \in \{0,1\}$ denotes a binary variable that is associated with the corresponding proposition p_i.

γ	δ	$\bot$	$\top$	$\neg\gamma$	$\gamma \wedge \delta$	$\gamma \vee \delta$	$\gamma \rightarrow \delta$	$\gamma \leftrightarrow \delta$
0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0
1	0	0	1	0	0	1	0	0
1	1	0	1	0	1	1	1	1

Table 2.2: Truth values of compound formulas.

Let $\mathbf{x}$ be an arbitrary interpretation relative to P. If (according to Table 2.2) $\gamma \in \mathcal{L}_P$ evaluates to 1, then $\mathbf{x}$ is called a **model** of γ. Otherwise, $\mathbf{x}$ is a **counter-model** of γ. The set of all models of γ is denoted by $N_P(\gamma) \subseteq N_P$. If $N_P(\gamma) = \emptyset$, then γ is called **unsatisfiable**. Otherwise, it is called **satisfiable**.

The notions of models and counter-models links propositional logic to the algebra of subsets of interpretations:

$$\begin{aligned}
&(1)\ N_P(\bot) &=&\ \emptyset,\\
&(2)\ N_P(\top) &=&\ N_P,\\
&(3)\ N_P(\neg\gamma) &=&\ N_P - N_P(\gamma),\\
&(4)\ N_P(\gamma \wedge \delta) &=&\ N_P(\gamma) \cap N_P(\delta),\\
&(5)\ N_P(\gamma \vee \delta) &=&\ N_P(\gamma) \cup N_P(\delta).
\end{aligned}$$

A propositional sentence γ **entails** another sentence δ (denoted by $\gamma \models \delta$) if and only if $N_P(\gamma) \subseteq N_P(\delta)$. In that case, δ is also called a **logical consequence** of γ. For example, $\gamma \wedge \delta \models \gamma \vee \delta$. Sometimes, it is convenient to write $\mathbf{x} \models \gamma$ instead of $\mathbf{x} \in N_P(\gamma)$. Furthermore, two sentences γ and δ are **logically equivalent** (denoted by $\gamma \equiv \delta$), if and only if $N_P(\gamma) = N_P(\delta)$. For example, $\gamma \rightarrow \delta \equiv \neg\gamma \vee \delta$. Note that logically equivalent sentences represent exactly the same information. The set of all logically equivalent sentences of γ can therefore be considered as an **equivalence class** $[\gamma]$. Logical connectives applied on equivalence classes have the following meanings:

$$\begin{aligned}
&(1)\ \neg[\gamma] &=&\ [\neg\gamma],\\
&(2)\ [\gamma] \wedge [\delta] &=&\ [\gamma \wedge \delta],\\
&(3)\ [\gamma] \vee [\delta] &=&\ [\gamma \vee \delta],\\
&(4)\ [\gamma] \rightarrow [\delta] &=&\ [\gamma \rightarrow \delta].
\end{aligned}$$

Note that 2^{2^n} different equivalence classes exist for a set $P = \{p_1, \ldots, p_n\}$. Let $[\mathcal{L}_P]$ represent the set of all equivalence classes of $\mathcal{L}_P$. Obviously, $[\mathcal{L}_P]$ forms a finite **Boolean algebra** [27, 54] with $\wedge$ as meet, $\vee$ as join, $\neg$ as complement, $[\bot]$ as zero, and $[\top]$ as unit. $[\mathcal{L}_P]$ is also isomorph to the set algebra 2^{N_P} and is therefore a **Lindenbaum algebra** [46].

2.1.2 Normal Forms

Propositional logic can be simplified by using equivalent special forms into which any propositional sentence can be transformed. Examples of such forms are the conjunctive and the disjunctive normal form. These particular forms are based on the notions of literals, clauses, and terms.

A **positive literal** is simply an element of $P = \{p_1, \ldots, p_n\}$, while the elements of $\neg P = \{\neg p_1, \ldots, \neg p_n\}$ are **negative literals**. $P^{\pm} = P \cup \neg P$ denotes the set of all (positive and negative) literals. A **clause** is a finite disjunction $\ell_1 \vee \cdots \vee \ell_s$ of literals $\ell_i \in P^{\pm}$. The empty disjunction $\bot$ is called **empty clause**. Similarly, a **term** is a finite conjunction $\ell_1 \wedge \cdots \wedge \ell_s$ of literals $\ell_i \in P^{\pm}$, and the empty conjunction $\top$ is called **empty term**. A clause or a term is called **proper**, if every propositional symbol appears at most once. $\mathcal{D}_P$ and $\mathcal{C}_P$ represent the sets of all proper clauses (disjunctions) and proper terms (conjunctions), respectively.

A **Conjunctive normal form** (CNF for short) is a finite conjunction $\varphi_1 \wedge \cdots \wedge \varphi_r$ of proper clauses φ_i. Similarly, a **disjunctive normal form** (DNF for short) is a finite disjunction $\psi_1 \vee \cdots \vee \psi_r$ of proper terms ψ_i. Note that any propositional sentence can be transformed into an equivalent conjunctive or disjunctive normal form [11].

CNF and DNF formulas are often considered as sets of clauses and terms, respectively. For example, if $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_r$ is a CNF formula, then $\Phi = \{\varphi_1, \ldots, \varphi_r\}$ is the corresponding set of clauses. CNF and DNF formulas can therefore be seen as subsets of $\mathcal{D}_P$ and $\mathcal{C}_P$, respectively. If Γ and Δ are two sets of clauses or terms for γ and δ (it will always be clear from the context whether γ and δ are CNF or DNF formulas), then it is often convenient to write $\Gamma \models \Delta$, $\gamma \models \Delta$, or $\Gamma \models \delta$ instead of $\gamma \models \delta$. Similarly, $\Gamma \equiv \Delta$, $\gamma \equiv \Delta$, or $\Gamma \equiv \delta$ is sometimes used instead of $\gamma \equiv \delta$. Furthermore, $\neg\Gamma$ denotes the corresponding set of negated terms or clauses of $\neg\gamma$. Note that the the negation of a clause is a term and, similarly, the negation of a term is a clause.

Particular CNF and DNF formulas are connected to the notion of prime implicates and prime implicants. A clause $\varphi \in \mathcal{D}_P$ is called **implicate** of $\gamma \in \mathcal{L}_P$, if $\gamma \models \varphi$. An implicate φ of γ is called **prime implicate** of γ, if no proper sub-clause of φ is also an implicate of γ. The set of all prime implicates of γ defines a CNF denoted by $\Phi(\gamma)$. Similarly, a term $\psi \in \mathcal{C}_P$ is called **implicant** of γ, if $\psi \models \gamma$, and an implicant ψ of γ is called **prime implicant** of γ, if no sub-term of ψ is also an implicant of γ. The set of all prime implicants of γ defines a DNF denoted by $\Psi(\gamma)$.[1] Note that $\gamma \equiv \Phi(\gamma) \equiv \Psi(\gamma)$. If γ is a CNF (or a DNF) and Γ the corresponding set of clauses (or terms), then it is often more convenient to write $\Phi(\Gamma)$ and $\Psi(\Gamma)$ instead of $\Phi(\gamma)$ and $\Psi(\gamma)$.

The notions of prime implicates and prime implicants are closely connected. In fact, if φ is a (prime) implicate of γ, then $\neg\varphi$ is a (prime) implicant of $\neg\gamma$. Similarly, if ψ is a (prime) implicant of γ, then $\neg\psi$ is a (prime) implicate of $\neg\gamma$. Therefore, $\neg\Phi(\gamma) = \Psi(\neg\gamma)$ and $\neg\Psi(\gamma) = \Phi(\neg\gamma)$. The problems of computing prime implicates and prime implicants are therefore equivalent. A resolution-based approach for this is discussed in Section 3.

2.1.3 Sub-Languages

Sometimes, particular subsets of propositions $Q \subseteq \{p_1, \ldots, p_n\}$ with $|Q| = m$ are of interest. Clearly, any such Q defines a propositional language $\mathcal{L}_Q$ called **sub-language** of $\mathcal{L}_P$. Such a sub-language $\mathcal{L}_Q$ defines a corresponding m-dimensional space $N_Q = \{0,1\}^m$ of possible interpretations $\mathbf{x} = (x_1, \ldots, x_m)$ relative to Q.

The link between a sub-language $\mathcal{L}_Q$ with $\mathcal{L}_P$ is obtained by considering projection and extension of interpretations. If $\mathbf{x} \in N_P$ is an interpretation relative to P, then $\mathbf{x}^{\downarrow Q} \in N_Q$ denotes the **projection** of $\mathbf{x}$ to Q, obtained by removing the corresponding rows in $\mathbf{x}$ for all propositions in $P - Q$. More generally, $N^{\downarrow Q} = \{\mathbf{x}^{\downarrow Q} : \mathbf{x} \in N\}$ denotes the projection of an arbitrary set $N \subseteq N_P$ to Q. If γ is formula in $\mathcal{L}_P$, then $N_P(\gamma)^{\downarrow Q}$ is often abbreviated by $N_Q(\gamma)$.

Conversely, if $\mathbf{x} \in N_Q$ is an interpretation relative to Q, then the set of interpretations $\mathbf{x}^{\uparrow P} = \{\mathbf{y} \in N_P : \mathbf{y}^{\downarrow Q} = \mathbf{x}\}$ is called **extension** of $\mathbf{x}$ to P. Furthermore, the extension of an arbitrary set $N \subseteq N_Q$ to P is defined by $N^{\uparrow P} = \bigcup\{\mathbf{x}^{\uparrow P} : \mathbf{x} \in N\}$. Again, if $\gamma \in \mathcal{L}_Q$, then $N_P(\gamma)$ is an abbreviation for $N_Q(\gamma)^{\uparrow P}$.

Let γ be a propositional sentence in $\mathcal{L}_P$ and $\mathbf{x} \in N_Q$ an interpretation relative to $Q = \{q_1, \ldots, q_m\}$. For such a case, $\gamma_{Q \leftarrow \mathbf{x}}$ denotes the formula obtained from γ by replacing each occurrence of q_i by $\bot$ if $x_i = 0$ or by $\top$ if $x_i = 1$. Note that $N_P(\gamma_{Q \leftarrow \mathbf{x}}) = N_P(\gamma) \cap \mathbf{x}^{\uparrow P}$. Furthermore, if δ is another propositional sentence in $\mathcal{L}_P$, then $\mathbf{x} \models_\gamma \delta$ means that $\gamma_{Q \leftarrow \mathbf{x}} \models \delta$. In such a case, $\mathbf{x}$ is called **model** of δ relative to γ.

[1]In the literature, the sets of prime implicates or prime implicants are often denoted equally by $PI(\gamma)$. Here, $\Phi(\gamma)$ and $\Psi(\gamma)$ are preferred to distinguish properly between prime implicates and prime implicants.

2.2 Argumentation Systems

Argumentation systems are obtained from propositional logic by considering two disjoint sets $A = \{a_1, \ldots, a_m\}$ and $P = \{p_1, \ldots, p_n\}$ of propositions. The elements of A are called **assumptions**. $\mathcal{L}_{A \cup P}$ denotes the corresponding propositional language.

Definition 2.1 *Let A and P be two disjoint sets of propositions. If ξ is a propositional sentence in $\mathcal{L}_{A \cup P}$, then a triple $\mathcal{AS}_P = (\xi, P, A)$ is called* **propositional argumentation system**. *ξ is called the* **knowledge base** *of $\mathcal{AS}_P$.*

The knowledge base ξ is often assumed to be satisfiable. Furthermore, ξ is sometimes given as a conjunctive set $\Sigma = \{\xi_1, \ldots, \xi_r\}$ of sentences $\xi_i \in \mathcal{L}_{A \cup P}$ or, more specifically, clauses $\xi_i \in \mathcal{D}_{A \cup P}$. In such cases, it is always possible to use the corresponding conjunction $\xi = \xi_1 \wedge \cdots \wedge \xi_r$ instead. If $\xi \equiv \top$ (for example, if $\Sigma = \emptyset$), then ξ is called **vacuous** knowledge base. Similarly, ξ is called **contradictory**, if $\xi \equiv \bot$ (for example, if $\Sigma = \{\bot\}$).

The assumptions are essential for expressing uncertain information. They are used to represent uncertain events, unknown circumstances, or possible risks and outcomes. The set N_A of possible interpretations relative to A is therefore of particular interest. Such interpretations $\mathbf{s} \in N_A$ are called **scenarios**. They represent possible **states** of the unknown or future world. This is the fundamental notion in this theory.

2.2.1 Inconsistent and Consistent Scenarios

Evidently, some scenarios may become impossible with respect to the given knowledge base ξ. It is therefore necessary to distinguish two different types of scenarios.

Definition 2.2 *Let ξ be a propositional sentence in $\mathcal{L}_{A \cup P}$. A scenario $\mathbf{s} \in N_A$ is called*

(1) **inconsistent** *(or* **contradictory***) relative to ξ, if and only if $\mathbf{s} \models_\xi \bot$;*

(2) **consistent** *relative to ξ, otherwise.*

Suppose that $\mathbf{s} \in N_A$ is an inconsistent scenario relative to ξ. This means that ξ becomes unsatisfiable when all the assumptions are set according to $\mathbf{s}$. The set of all inconsistent scenarios is denoted by $I_A(\xi) = \{\mathbf{s} \in N_A : \mathbf{s} \models_\xi \bot\}$. Similarly, if $\mathbf{s}$ is supposed to be consistent relative to ξ, then ξ remains satisfiable when all the assumptions are set according to $\mathbf{s}$. The set $C_A(\xi) = \{\mathbf{s} \in N_A : \mathbf{s} \not\models_\xi \bot\}$ denotes the collection of all consistent scenarios relative to ξ. Evidently, $I_A(\xi)$ and $C_A(\xi)$ are complementary set, that is

$$C_A(\xi) \;=\; N_A - I_A(\xi). \tag{2.1}$$

Example 2.1 *Let $A = \{a_1, a_2\}$ and $P = \{p, q\}$ be two sets of propositions. If $\xi = (a_1 \rightarrow p) \wedge (a_2 \rightarrow q) \wedge (p \rightarrow \neg q)$ is a sentence in $\mathcal{L}_{A\cup P}$, then $I_A(\xi) = \{(1,1)\}$ is the set of inconsistent scenarios, and $C_A(\xi) = \{(0,0),(0,1),(1,0)\}$ is the set of consistent scenarios. The scenario $(1,1)$ is inconsistent because ξ is unsatisfiable when a_1 and a_2 are simultaneously true.*

The distinction between inconsistent and consistent scenarios can be seen as the main essence of argumentation systems. It introduces in a natural and convenient way non-monotonicity into propositional logic, while the richness of computational techniques for propositional logic is preserved. Non-monotonicity is the fundamental property of any formalism for dealing with uncertainty. The question, why and how the distinction between inconsistent and consistent scenarios leads to non-monotonicity, is discussed an the bottom of the following subsection and in Subsection 2.4.

2.2.2 Supporting Scenarios

The situation becomes more interesting when a second propositional sentence $h \in \mathcal{L}_{A\cup P}$ called **hypothesis** is given. Hypotheses represent open questions or uncertain statements about some of the propositions in $A \cup P$. What can be inferred from ξ about the possible truth of h with respect to the given set of unknown assumptions? Possibly, if the assumptions are set according to some scenarios $\mathbf{s} \in N_A$, then h may be a logical consequence of ξ. In other words, h is **supported** by certain scenarios.

Definition 2.3 *Let h and ξ be propositional sentences in $\mathcal{L}_{A\cup P}$. A scenario $\mathbf{s} \in N_A$ is called a*

(1) **quasi-supporting** *scenario for h relative to ξ, if and only if $\mathbf{s} \models_\xi h$;*

(2) **supporting** *scenario for h relative to ξ, if and only if $\mathbf{s} \models_\xi h$ and $\mathbf{s} \not\models_\xi \bot$;*

(3) **possibly supporting** *scenario for h relative to ξ, if and only if $\mathbf{s} \not\models_\xi \neg h$.*

The set $QS_A(h,\xi) = \{\mathbf{s} \in N_A : \mathbf{s} \models_\xi h\}$ denotes the collection of all quasi-supporting scenarios for h relative to ξ. Similarly, $SP_A(h,\xi) = \{\mathbf{s} \in N_A : \mathbf{s} \models_\xi$

$h,\ \mathbf{s} \not\models_\xi \bot\}$ denotes the set of all supporting scenarios and $PS_A(h,\xi) = \{\mathbf{s} \in N_A :\ \mathbf{s} \not\models_\xi \neg h\}$ the set of all possibly supporting scenarios for h relative to ξ.

The difference between quasi-supporting and supporting scenarios is that quasi-supporting scenarios are allowed to be inconsistent. This will be only convenient for technical reasons. However, inconsistency is usually excluded, and supporting scenarios are therefore more interesting.

Figure 2.2 illustrates the relation between different subsets of N_A with $I_A(\xi) = \text{A}$, $QS_A(h,\xi) = \text{A}+\text{B}$, $SP_A(h,\xi) = \text{B}$, $PS_A(h,\xi) = \text{B}+\text{C}$, and $C_A(\xi) = \text{B}+\text{C}+\text{D}$.

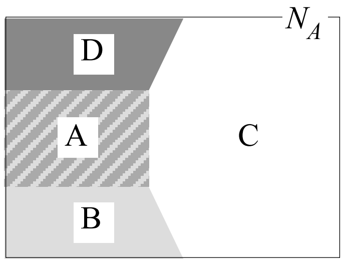

Figure 2.2: Different subsets of scenarios.

Note that $I_A(\xi) \subseteq QS_A(h,\xi)$, $SP_A(h,\xi) \subseteq C_A(\xi)$, and $PS_A(h,\xi) \subseteq C_A(\xi)$ for all hypotheses $h \in \mathcal{L}_{A\cup P}$. Furthermore, $SP_A(h,\xi) \subseteq QS_A(h,\xi)$, $SP_A(h,\xi) \subseteq PS_A(h,\xi)$.

Example 2.2 *Again, let $A = \{a_1,a_2\}$ and $P = \{p,q\}$. If $\xi = (a_1 \to p) \wedge (a_2 \to q) \wedge (p \to \neg q)$ is a sentence in $\mathcal{L}_{A\cup P}$, then $QS_A(p,\xi) = \{(1,0),(1,1)\}$ is the set of quasi-supporting scenarios, $SP_A(p,\xi) = \{(1,0)\}$ is the set of supporting scenarios, and $PS_A(p,\xi) = \{(0,0),(0,1),(1,0)\}$ is the set of possibly supporting scenarios for p. Similarly, $QS_A(q,\xi) = \{(0,1),(1,1)\}$, $SP_A(q,\xi) = \{(0,1)\}$, and $PS_A(q,\xi) = \{(0,0),(0,1)\}$ are the corresponding sets for q. Note that $(1,1)$ is inconsistent and therefore never a supporting or a possibly supporting scenario (see Example 2.1).*

The sets of inconsistent and consistent scenarios can be expressed in terms of quasi-supporting scenarios for $\bot$:

$$I_A(\xi) \ = \ QS_A(\bot,\xi), \tag{2.2}$$
$$C_A(\xi) \ = \ N_A - QS_A(\bot,\xi). \tag{2.3}$$

Similarly, the sets of supporting and possibly supporting scenarios for h can be determined via sets of quasi-supporting scenarios:

$$SP_A(h,\xi) \ = \ QS_A(h,\xi) - QS_A(\bot,\xi), \tag{2.4}$$

$$PS_A(h,\xi) \;=\; N_A - QS_A(\neg h,\xi). \tag{2.5}$$

The problem of computing sets of inconsistent, consistent, supporting, and possibly supporting scenarios can therefore be solved by computing solely sets of quasi-supporting scenarios. Hence, the importance of quasi-supporting scenarios relies mainly on technical reasons, and also on the fact that $QS_A(h,\xi)$ can be determined more easily than $SP_A(h,\xi)$ (see Section 3).

For a fixed knowledge base ξ, the notion of quasi-supporting scenarios defines a mapping $[\mathcal{L}_{A\cup P}] \longrightarrow 2^{N_A}$. Note that both sets $[\mathcal{L}_{A\cup P}] \cong 2^{N_{A\cup P}}$ and 2^{N_A} are finite Boolean algebras. In a more general theory [31, 33], such a mapping between two Boolean algebras is also called **allocation of support**. An allocation of support must satisfy some basic properties that are also valid in the case of quasi-supporting scenarios:

Theorem 2.1 *If h_1, h_2, and ξ are propositional sentences in $\mathcal{L}_{A\cup P}$, then*

(1) $QS_A(\bot,\xi) = I_A(\xi)$,
(2) $QS_A(\top,\xi) = N_A$,
(3) $QS_A(h_1 \wedge h_2,\xi) = QS_A(h_1,\xi) \cap QS_A(h_2,\xi)$,
(4) $QS_A(h_1 \vee h_2,\xi) \supseteq QS_A(h_1,\xi) \cup QS_A(h_2,\xi)$,
(5) $h_1 \models h_2$ *implies* $QS_A(h_1,\xi) \subseteq QS_A(h_2,\xi)$,
(6) $h_1 \equiv h_2$ *implies* $QS_A(h_1,\xi) = QS_A(h_2,\xi)$.

Similar considerations are possible for the case of supporting scenarios, for which corresponding basic properties exist:

Theorem 2.2 *If h_1, h_2, and ξ are propositional sentences in $\mathcal{L}_{A\cup P}$, then*

(1) $SP_A(\bot,\xi) = \emptyset$,
(2) $SP_A(\top,\xi) = C_A(\xi)$,
(3) $SP_A(h_1 \wedge h_2,\xi) = SP_A(h_1,\xi) \cap SP_A(h_2,\xi)$,
(4) $SP_A(h_1 \vee h_2,\xi) \supseteq SP_A(h_1,\xi) \cup SP_A(h_2,\xi)$,
(5) $h_1 \models h_2$ *implies* $SP_A(h_1,\xi) \subseteq SP_A(h_2,\xi)$,
(6) $h_1 \equiv h_2$ *implies* $SP_A(h_1,\xi) = SP_A(h_2,\xi)$.

Finally, corresponding properties exist also for sets of possibly supporting scenarios:

Theorem 2.3 *If h_1, h_2, and ξ are propositional sentences in $\mathcal{L}_{A\cup P}$, then*

$$
\begin{array}{ll}
(1) & PS_A(\bot,\xi) = \emptyset, \\
(2) & PS_A(\top,\xi) = C_A(\xi), \\
(3) & PS_A(h_1 \wedge h_2,\xi) \subseteq PS_A(h_1,\xi) \cap PS_A(h_2,\xi), \\
(4) & PS_A(h_1 \vee h_2,\xi) = PS_A(h_1,\xi) \cup PS_A(h_2,\xi), \\
(5) & h_1 \models h_2 \;\; \textit{implies} \;\; PS_A(h_1,\xi) \subseteq PS_A(h_2,\xi), \\
(6) & h_1 \equiv h_2 \;\; \textit{implies} \;\; PS_A(h_1,\xi) = PS_A(h_2,\xi).
\end{array}
$$

An important special case arises when the hypothesis consists only of assumptions, that is when h is a propositional sentence in $\mathcal{L}_A$ instead of $\mathcal{L}_{A\cup P}$. Such cases can be treated according to the following theorem:

Theorem 2.4 *If $h_A \in \mathcal{L}_A$ and $\xi \in \mathcal{L}_{A\cup P}$, then*

$$
\begin{array}{ll}
(1) & QS_A(h_A,\xi) = N_A(h_A) \cup I_A(\xi), \\
(2) & SP_A(h_A,\xi) = N_A(h_A) \cap C_A(\xi), \\
(3) & PS_A(h_A,\xi) = N_A(h_A) \cap C_A(\xi).
\end{array}
$$

Note that in such a case, the sets $SP_A(h_A,\xi)$ and $PS_A(h_A,\xi)$ are identical. Another interesting situation to be considered is the case where the knowledge base ξ changes to $\xi' = \xi \wedge \tilde{\xi}$ by adding new information. Then, the number of inconsistent and quasi-supporting scenarios is monotonically increasing, whereas the number of consistent and possibly supporting scenarios is monotonically decreasing.

Theorem 2.5 *If $h \in \mathcal{L}_{A\cup P}$ and $\xi' = \xi \wedge \tilde{\xi} \in \mathcal{L}_{A\cup P}$, then*

$$
\begin{array}{ll}
(1) & I_A(\xi') \supseteq I_A(\xi), \\
(2) & C_A(\xi') \subseteq C_A(\xi), \\
(3) & QS_A(h,\xi') \supseteq QS_A(h,\xi), \\
(4) & PS_A(h,\xi') \subseteq PS_A(h,\xi).
\end{array}
$$

In contrast, nothing can be said about the number of supporting scenarios. If new information is added, then the set of supporting scenarios behaves non-monotonically, that is it may either grow or shrink, both cases are possible. The reason for this is that according to (2.4), $SP_A(h,\xi)$ is a set difference of two monotonically growing sets $QS_A(h,\xi)$ and $QS_A(\bot,\xi)$. The size of the set $SP_A(h,\xi)$ has only two restrictions: $SP_A(h,\top) = N_A(h)$ and $SP_A(h,\bot) = \emptyset$. Between these two extreme cases, everything is possible.

The non-monotonicity of the set $SP_A(h, \xi)$ is an important property of argumentation systems. It reflects a natural property of how a human's conviction or belief can change when new information is given. Non-monotonicity is therefore a fundamental property for any mathematical formalism for reasoning under uncertainty. However, using propositional argumentation systems shows that non-monotonicity can be achieved without leaving the field of classical logic. Non-monotonicity will also be an important in Subsection 2.4, when probabilities are assigned to the assumptions and conditional probabilities $p(SP_A(h, \xi)|C_A(\xi))$ are considered.

2.2.3 Refuting Scenarios

Instead of considering scenarios in favor of the hypothesis, it is also reasonable to look at scenarios speaking against the hypothesis. In other words, such **refuting** scenarios are supporting the negated hypothesis $\neg h$.

Definition 2.4 *Let h and ξ be propositional sentences in $\mathcal{L}_{A \cup P}$. A scenario $\mathbf{s} \in N_A$ is called a*

(1) **quasi-refuting** *scenario for h relative to ξ, if and only if $\mathbf{s}$ is a quasi-supporting scenario for $\neg h$;*

(2) **refuting** *scenario for h relative to ξ, if and only if $\mathbf{s}$ is a supporting scenario for $\neg h$;*

(3) **possibly refuting** *scenario for h relative to ξ, if and only if $\mathbf{s}$ is a possibly supporting scenario for $\neg h$.*

The sets of all quasi-refuting, refuting, and possibly refuting scenarios are denoted by $QR_A(h, \xi) = QS_A(\neg h, \xi)$, $RF_A(h, \xi) = SP_A(\neg h, \xi)$, and $PR_A(h, \xi) = PS_A(\neg h, \xi)$, respectively. In Figure 2.2, these sets are composed as follows: $QR_A(h, \xi) = \text{A} + \text{D}$, $RF_A(h, \xi) = \text{D}$, $PR_A(h, \xi) = \text{C} + \text{D}$.

Example 2.3 *Again, let $A = \{a_1, a_2\}$ and $P = \{p, q\}$. If $\xi = (a_1 \rightarrow p) \wedge (a_2 \rightarrow q) \wedge (p \rightarrow \neg q)$ is a sentence in $\mathcal{L}_{A \cup P}$, then $QR_A(q, \xi) = \{(1,0), (1,1)\}$, $RF_A(q, \xi) = \{(1,0)\}$, and $PR_A(q, \xi) = \{(0,0), (1,0)\}$.*

Evidently, there is a strong duality between the notions of refuting and supporting scenarios. The theoretical results of the previous subsection can therefore be adapted to the case of refuting scenarios, simply by replacing h with $\neg h$. In the sequel, only the notion of support will be further investigated. However, refuting a hypothesis means always supporting its negation and vice versa.

2.3 Representing Sets of Scenarios

Sets of scenarios $S \subseteq N_A$ (such as $I_A(\xi)$, $C_A(\xi)$, $QS_A(h,\xi)$, $SP_A(h,\xi)$, etc.) tend to grow exponentially with the size of A. An explicit representation as a list of elements $\mathbf{s} \in S$ may therefore not be feasible. Thus, alternative representations are needed. An efficient representation is obtained by considering terms $\alpha \in \mathcal{C}_A$ for which $N_A(\alpha) \subseteq S$ holds. Let $T(S) = \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq S\}$ be the set of all terms for which this conditions holds. Such a set $T(S)$ is called **term representation** of S. Note that the terms $\alpha \in T(S)$ are implicants for any propositional sentence γ with $N_A(\gamma) = S$. Such propositional sentences γ are called **logical representations** of S.

Term representations are **upward-closed** sets. This means that $\alpha \in T(S)$ implies that every (longer) term $\alpha' \models \alpha$ is also in $T(S)$. It is sometimes convenient to consider terms as sets of literals and to write $\alpha' \supseteq \alpha$ instead of $\alpha' \models \alpha$. A property of upward-closed sets is that they can be represented by their minimal elements. A term $\alpha \in T(S)$ is called **minimal** in $T(S)$, if there is no other (shorter) term $\alpha' \subseteq \alpha$ in $T(S)$. The corresponding set $\mu T(S)$ of minimal terms is called **minimal term representation** of S. Clearly, the terms $\alpha \in T(S)$ are prime implicants for any logical representation γ of S, that is $\mu T(S) = \Psi(\gamma)$. Furthermore, note that

$$S = \bigcup_{\alpha \in T(S)} N_A(\alpha) = \bigcup_{\alpha \in \mu T(S)} N_A(\alpha). \tag{2.6}$$

Example 2.4 *If $S = \{(0,0),(1,0),(1,1)\}$ is a subset of N_A for $A = \{a_1, a_2\}$, then $T(S) = \{a_1, \neg a_2, a_1 \wedge a_2, a_1 \wedge \neg a_2, \neg a_1 \wedge \neg a_2\}$ is the term representation of S, and $\mu T(S) = \{a_1, \neg a_2\}$ is the corresponding minimal term representation of S. The propositional sentences $a_1 \vee \neg a_2$ and $a_2 \rightarrow a_1$, for example, are logical representations of S.*

The main operations for sets of scenarios can now be replaced by corresponding operations for minimal term representations:

- **Inclusion**: if S_1 and S_2 are two sets of scenarios, then $S_1 \subseteq S_2$ is true if and only if for each term $\alpha_1 \in \mu T(S_1)$ there is a term $\alpha_2 \in \mu T(S_2)$ such that $\alpha_1 \supseteq \alpha_2$.
- **Intersection**: if S_1 and S_2 are two sets of scenarios, then the minimal term representation of $S_1 \cap S_2$ is obtained from $\mu T(S_1)$ and $\mu T(S_2)$ by

$$\begin{aligned} \mu T(S_1 \cap S_2) &= \mu\{\alpha_1 \wedge \alpha_2 \in \mathcal{C}_A : \alpha_1 \in \mu T(S_1),\ \alpha_2 \in \mu T(S_2)\} \\ &= \mu(\{\alpha_1 \wedge \alpha_2 : \alpha_1 \in \mu T(S_1),\ \alpha_2 \in \mu T(S_2)\} \cap \mathcal{C}_A). \end{aligned} \tag{2.7}$$

- **Complement**: if S is a set of scenarios, then the minimal term representation of the complementary set $N_A - S$ is obtained from $\mu T(S)$ as follows:

$$N_A - S \;=\; N_A - \bigcup_{\alpha \in \mu T(S)} N_A(\alpha) \;=\; \bigcap_{\alpha \in \mu T(S)} (N_A - N_A(\alpha)) \;=\; \bigcap_{\alpha \in \mu T(S)} N_A(\neg\alpha). \quad (2.8)$$

  Note that $\neg\alpha = \neg\ell_1 \vee \cdots \vee \neg\ell_s$ is a clause, and the minimal term representation of $N_A(\neg\alpha)$ results when $\neg\alpha$ is considered as a set of literals, that is $\mu T(N_A(\neg\alpha)) = \{\neg\ell_1, \ldots, \neg\ell_s\}$. The final result $\mu T(N_A - S)$ is then obtained by repeatedly applying (2.7) for the intersection.

- **Difference**: if S_1 and S_2 are two sets of scenarios, then $S_1 - S_2$ is clearly the same as $S_1 - (N_A - S_2)$. Therefore, the the minimal term representation of $S_1 - S_2$ can be obtained in two steps: (1) compute the complement $N_A - S_2$ as described above; (2) use (2.7) to compute the intersection of S_1 and $N_A - S_2$.

Unfortunately, there is no corresponding simple operation for $S_1 \cup S_2$. This is not too disturbing as set union is only of minor importance for dealing with sets of scenarios. For example, deriving the sets $C_A(\xi)$, $I_A(\xi)$, $SP_A(h,\xi)$, and $PS_A(h,\xi)$ from $QS_A(h,\xi)$ and $QS_A(\bot,\xi)$ does not require set union (see Subsection 2.2). If necessary, however, the minimal term representation of $S_1 \cup S_2$ is obtained by computing prime implicants:

$$\mu T(S_1 \cup S_2) \;=\; \Psi(\mu T(S_1) \cup \mu T(S_2)). \quad (2.9)$$

From now on, sets S of scenarios will be represented by corresponding minimal term representations $\mu T(S)$, and the necessary set operations will be treated as described above. Note that $|\mu T(S)| \leq |S|$. Furthermore, if $\alpha \in \mu T(S)$ is a minimal term for S, then $|\alpha| \leq |A|$ whereas $|\mathbf{s}| = |A|$ for all $\mathbf{s} \in S$. Therefore, the total length of the representation $\mu T(s)$ is in general shorter than the total length of S, that is

$$\sum_{\alpha \in \mu T(S)} |\alpha| \;\leq\; |S| \cdot |A|. \quad (2.10)$$

Another advantage is that the same set $\mu T(S)$ is also the minimal term representation of $S^{\uparrow A'}$ for $A' \supseteq A$. Thus, the size of $\mu T(S)$ remains constant while $|S^{\uparrow A'}|$ is growing exponentially with the size of A'.

Representing sets of scenarios by sets of minimal terms has also an important semantical meaning. Let $\alpha \in \mu S$ be a minimal term and $Lit(\alpha) \subseteq A^{\pm}$ the corresponding set of literals. An assumption $a \in A$ is called **positive** relative to α, if $a \in Lit(\alpha)$, it is called **negative** relative to α, if $\neg a \in Lit(\alpha)$, and it is called **irrelevant** relative to α, if $a \notin Lit(\alpha)$ and $\neg a \notin Lit(\alpha)$. Therefore, every assumption $a \in A$ is either positive, negative, or irrelevant relative to α, and A can be decomposed into three sets $Lit^{+}(\alpha)$, $Lit^{-}(\alpha)$, and $Lit^{\pm}(\alpha)$ of positive, negative, and irrelevant assumptions relative to α, respectively. If α is a minimal term in $\mu T(QS_A(h,\xi))$, for example, then $\alpha \wedge \xi \models h$ (see Subsection 2.3.2). The term α can then be considered as a possible proof of h. Proving h by such a term α implies that the assumptions in $Lit^{+}(\alpha)$ are true, the assumptions in $Lit^{-}(\alpha)$ are false, and the assumptions in $Lit^{\pm}(\alpha)$ are either true or false (i.e. they are irrelevant for the proof). This point of view is of particular importance when arguments are considered as explanations (e.g. diagnostics of faulty technical systems).

2.3.1 Inconsistent and Consistent Terms

The above discussion about representing sets of scenarios can now be applied to the cases of inconsistent and consistent scenarios.

Definition 2.5 *Let ξ be a propositional sentence in $\mathcal{L}_{A\cup P}$. A term $\alpha \in \mathcal{C}_A$ is called*

(1) **inconsistent** *(or* **contradictory***) relative to ξ, if $N_A(\alpha) \subseteq I_A(\xi)$;*

(2) **consistent** *relative to ξ, if $N_A(\alpha) \subseteq C_A(\xi)$.*

Note that according to this definition there are possibly terms $\alpha \in \mathcal{C}_A$ that are neither inconsistent nor consistent relative to ξ. The term representations of $I_A(\xi)$ and $C_A(\xi)$ are denoted by

$$I(\xi) \;=\; T(I_A(\xi)) \;=\; \{\alpha \in \mathcal{C}_A :\; N_A(\alpha) \subseteq I_A(\xi)\}, \tag{2.11}$$

$$C(\xi) \;=\; T(C_A(\xi)) \;=\; \{\alpha \in \mathcal{C}_A :\; N_A(\alpha) \subseteq C_A(\xi)\}, \tag{2.12}$$

respectively. The sets $\mu I(\xi)$ and $\mu C(\xi)$ are the corresponding minimal term representations. Sometimes, $I(\xi)$ is called **contradiction** of ξ. Similarly, $\mu I(\xi)$ is called **minimal contradiction** of ξ.

The term representations of $I(\xi)$ and $C_A(\xi)$ can also be characterized without using the notions of inconsistent and consistent scenarios.

Theorem 2.6 *If $\xi \in \mathcal{L}_{A\cup P}$ is a propositional sentence, then*

$$\begin{array}{lrcl} (1) & I(\xi) & = & \{\alpha \in \mathcal{C}_A :\; \alpha \wedge \xi \models \bot\}, \\ (2) & C(\xi) & = & \{\alpha \in \mathcal{C}_A :\; \forall \alpha' \supseteq \alpha,\; \alpha' \in \mathcal{C}_A,\; \alpha' \wedge \xi \not\models \bot\}. \end{array}$$

This way of characterizing the set $I(\xi)$ may appear more obvious than the above definition in terms the set $I_A(\xi)$. However, in the authors opinion the notion of a scenario is the fundamental concept of this formalism. The terms $\alpha \in \mathcal{C}_A$ are mainly important for representing sets of scenarios efficiently. Furthermore, defining $I(\xi)$ according to Theorem 2.6 is somehow misleading as it may result in defining terms $\alpha \in \mathcal{C}_A$ with $\alpha \wedge \xi \not\models \perp$ to be consistent. This may appear reasonable, but it does not guarantee that $N_A(\alpha) \cap I_A(\xi) = \emptyset$. This leads then to a different set $C'(\xi) \supseteq C(\xi)$ that is no longer upward-closed. The problem then is that the set $C_A(\xi)$ of consistent scenarios is not determined by $C'(\xi)$ in an unequivocal way. The importance of an unequivocal set $C_A(\xi)$ will become clear in Subsection 2.4 when probabilities are assigned to the assumptions and conditional probabilities $p(S|C_A(\xi))$ are considered.

2.3.2 Supporting Arguments

The problem of representing sets of scenarios also appears in the case of quasi-supporting, supporting, and possibly supporting scenarios for a given hypothesis h. Here, the notion of **arguments** enters into the formalism.

Definition 2.6 *Let h and ξ be two propositional sentences in $\mathcal{L}_{A\cup P}$. A term $\alpha \in \mathcal{C}_A$ is called a*

(1) **quasi-supporting argument** *for h relative to ξ, if $N_A(\alpha) \subseteq QS_A(h,\xi)$;*

(2) **supporting argument** *for h relative to ξ, if $N_A(\alpha) \subseteq SP_A(h,\xi)$;*

(3) **possibly supporting argument** *for h relative to ξ, if $N_A(\alpha) \subseteq PS_A(h,\xi)$.*

The term representations of the sets $QS_A(h,\xi)$, $SP_A(h,\xi)$, and $PS_A(h,\xi)$, that is

$$\begin{aligned} QS(h,\xi) &= T(QS_A(h,\xi)) = \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq QS_A(h,\xi)\}, \\ SP(h,\xi) &= T(SP_A(h,\xi)) = \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq SP_A(h,\xi)\}, \\ PS(h,\xi) &= T(PS_A(h,\xi)) = \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq PS_A(h,\xi)\}, \end{aligned}$$

are called **quasi-support**, **support**, and **possibility** for h relative to ξ, respectively. Furthermore, the minimal term representations $\mu QS(h,\xi)$, $\mu SP(h,\xi)$, and $\mu PS(h,\xi)$ are called **minimal quasi-support**, **minimal support**, and **minimal possibility** for h relative to ξ, respectively. As above, $QS(h,\xi)$, $SP(h,\xi)$, and $PS(h,\xi)$ can be characterized without the notion of scenarios.

Theorem 2.7 *If h and ξ are two propositional sentences in $\mathcal{L}_{A\cup P}$, then*

$$\begin{aligned}
(1)\quad QS(h,\xi) &= \{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h\}, \\
(2)\quad SP(h,\xi) &= \{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h,\ \forall \alpha' \supseteq \alpha,\ \alpha' \in \mathcal{C}_A,\ \alpha' \wedge \xi \not\models \perp\}, \\
(3)\quad PS(h,\xi) &= \{\alpha \in \mathcal{C}_A : \forall \alpha' \supseteq \alpha,\ \alpha' \in \mathcal{C}_A,\ \alpha' \wedge \xi \not\models \neg h\}.
\end{aligned}$$

This way of characterizing the set $SP(h,\xi)$ may appear unusual. Often, support has been defined as the set of arguments α for which two conditions $\alpha \wedge \xi \models h$ and $\alpha \wedge \xi \not\models \perp$ hold [30, 34]. As argued before, the problem with excluding inconsistency by $\alpha \wedge \xi \not\models h$ is that, in fact, inconsistency is not completely excluded (there may be terms α with $N_A(\alpha) \cap I_A(\xi) \neq \emptyset$). Therefore, the resulting set $SP'(h,\xi) \supseteq SP(h,\xi)$ is no longer upward-closed, and the relation to the corresponding set of scenarios $SP_A(h,\xi)$ becomes ambiguous.

2.4 Probabilistic Argumentation Systems

So far, the problem of judging hypotheses has only been considered from a qualitative point of view. A more seizable judgement of the hypothesis can be obtained if every assumption $a_i \in A$ is linked to a corresponding prior probability π_i. Sometimes, π_i is understood as a probability in the sense of a proportion between "good" and possible events. However, in other cases π_i is simply an estimated value that expresses on a scale between 0 and 1 the subjective belief of a_i being true. In any case, the probabilities π_i are supposed to be stochastically independent.

Assigning probabilities to the assumption induces a probabilistic structure upon the symbolic argumentation system. A quadruple $\mathcal{PAS}_P = (\xi, P, A, \Pi)$, where $\Pi = \{\pi_1, \ldots, \pi_m\}$ denotes the set of probabilities assigned to the assumptions a_i, is therefore called **probabilistic argumentation system**.

2.4.1 Degree of Support and Possibility

Let $\mathbf{s} = (x_1, \ldots, x_m)$ be a scenario in N_A, then the prior probability of $\mathbf{s}$ is determined by

$$p(\mathbf{s}) = \prod_{i=1}^{m} {\pi_i}^{x_i} \cdot (1-\pi_i)^{(1-x_i)}. \tag{2.13}$$

If $S \subseteq N_A$ is an arbitrary set of scenarios, then the probability of S is simply the sum of the probabilities of its elements:

$$p(S) = \sum_{\mathbf{s} \in S} p(\mathbf{s}). \tag{2.14}$$

If $h \in \mathcal{L}_{A \cup P}$ is a hypothesis to be judged, then

$$dqs(h,\xi) \;=\; p(QS_A(h,\xi)) \tag{2.15}$$

is called **degree of quasi-support** of h relative to ξ. This measure corresponds to **unnormalized belief** in the Dempster-Shafer theory of evidence [51].

Supposing that inconsistent scenarios are not allowed means that the prior probability distribution on N_A must be conditioned on the fact that the true scenario is in $C_A(\xi)$. This leads to the new probability measure p' given by

$$p'(\mathbf{s}) \;=\; p(\mathbf{s}|C_A(\xi)) \;=\; \begin{cases} p(\mathbf{s})/p(C_A(\xi)), & \text{if } \mathbf{s} \in C_A(\xi), \\ 0, & \text{otherwise.} \end{cases} \tag{2.16}$$

Therefore, the prior probabilities $p(\mathbf{s})$ of consistent scenarios are multiplied by a **normalization factor** $k = p(C_A(\xi))^{-1}$. Note that $p(C_A(\xi))$ is the same as $1 - dqs(\bot,\xi)$. The new probability measure p' defines **posterior** probabilities for the scenarios given then the knowledge base ξ. If $h \in \mathcal{L}_{A \cup P}$ is a hypothesis, then

$$\begin{aligned} dsp(h,\xi) \;&=\; p'(SP_A(h,\xi)) \;=\; \sum_{\mathbf{s}\in SP_A(h,\xi)} p'(\mathbf{s}) \;=\; \frac{1}{p(C_A(\xi))} \cdot \sum_{\mathbf{s}\in SP_A(h,\xi)} p(\mathbf{s}) \\ &=\; \frac{p(SP_A(h,\xi))}{p(C_A(\xi))} \;=\; \frac{p(QS_A(h,\xi)) - p(QS_A(\bot,\xi))}{1 - p(QS_A(\bot,\xi))} \\ &=\; \frac{dqs(h,\xi) - dqs(\bot,\xi)}{1 - dqs(\bot,\xi)} \end{aligned} \tag{2.17}$$

is called **degree of support** of h relative to ξ. It corresponds to **normalized belief** in the Dempster-Shafer theory of evidence. Note that $dsp(h,\xi) = p(SP_A(h,\xi)|C_A(\xi))$. Degree of support can therefore be considered as the conditional probability of $SP_A(h,\xi)$ given $C_A(\xi)$. Note that $dsp(h,\bot)$ is undefined. Therefore, the knowledge base ξ is often supposed to be satisfiable. An important property of $dsp(h,\xi)$ is that it behaves non-monotonically when new knowledge is added. Further properties follow from Theorem 2.2 in Subsection 2.2.2:

(1) $dsp(\bot,\xi) = 0$,

(2) $dsp(\top,\xi) = 1$,

(3) $h_1 \models h_2$ implies $dsp(h_1,\xi) \leq dsp(h_2,\xi)$,

(4) $h_1 \equiv h_2$ implies $dsp(h_1,\xi) = dsp(h_2,\xi)$,

whenever $\xi \not\equiv \bot$. A second posterior measure for hypotheses is obtained by considering the corresponding conditional probability on the set of possibly supporting scenarios $PS_A(h,\xi)$. Therefore,

$$\begin{aligned} dps(h,\xi) &= p'(PS_A(h,\xi)) = \sum_{\mathbf{s}\in PS_A(h,\xi)} p'(\mathbf{s}) = \frac{1}{p(C_A(\xi))} \cdot \sum_{\mathbf{s}\in PS_A(h,\xi)} p(\mathbf{s}) \\ &= \frac{p(PS_A(h,\xi))}{p(C_A(\xi))} = \frac{1-p(QS_A(\neg h,\xi))}{1-p(QS_A(\bot,\xi))} \\ &= \frac{1-dqs(\neg h,\xi)}{1-dqs(\bot,\xi)} = 1-dsp(\neg h,\xi) \end{aligned} \tag{2.18}$$

is called **degree of possibility**. Again, $dps(h,\bot)$ is undefined. The corresponding notion in the context of the Dempster-Shafer theory is **plausibility**. Note that non-monotonicity is observed when new knowledge is added. Further properties can be derived from Theorem 2.3 in Subsection 2.2.2:

(1) $dps(\bot,\xi)=0$,

(2) $dps(\top,\xi)=1$,

(3) $h_1 \models h_2$ implies $dps(h_1,\xi) \leq dps(h_2,\xi)$,

(4) $h_1 \equiv h_2$ implies $dps(h_1,\xi) = dps(h_2,\xi)$,

whenever $\xi \not\equiv \bot$. Finally, an important property follows from the fact that $SP_A(h,\xi)$ is always a subset of $PS_A(h,\xi)$ (see Subsection 2.2.2):

$$dsp(h,\xi) \leq dps(h,\xi). \tag{2.19}$$

All this indicates that the framework of probabilistic argumentation systems constructed on propositional logic is a special case of Shafer's original evidence theory [51].

2.4.2 Computing Degree of Quasi-Support

According to (2.17) and (2.18), the problem of computing degree of support and degree of possibility involves the following three steps:

(1) determine $dqs(h,\xi)$, respectively $dqs(\neg h,\xi)$;

(2) determine $dqs(h,\bot)$;

(3) apply (2.17), respectively (2.18).

The problem to be solved is therefore the computation of $dqs(h,\xi)$ for arbitrary hypotheses $h \in \mathcal{L}_{A\cup P}$. Suppose that $QS_A(h,\xi)$ is represented by the set $\mu QS(h,\xi)$ of minimal quasi-supporting arguments. A method for computing minimal quasi-supports will be presented in Section 3. Clearly, a set $\mu QS(h,\xi) = \{\alpha_1,\ldots,\alpha_q\}$ defines a DNF $\alpha_1 \vee \cdots \vee \alpha_q$ with

$$QS_A(h,\xi) \;=\; N_A(\alpha_1 \vee \cdots \vee \alpha_q) \;=\; N_A(\alpha_1) \cup \cdots \cup N_A(\alpha_q). \tag{2.20}$$

The probability $p(QS_A(h,\xi))$ can therefore be seen as a probability of a union of events. This is a classical problem of probability theory. A first and simple approach is given by the so-called **inclusion-exclusion** formula [18]:

$$p(N_A(\alpha_1) \cup \cdots \cup N_A(\alpha_q)) \;=\; \sum_{\emptyset \neq I \subseteq \{1,\ldots,q\}} (-1)^{|I|+1} \cdot p(\bigcap_{i \in I} N_A(\alpha_i)). \tag{2.21}$$

Since the number of terms in the sum of Equation (2.21) grows exponentially with the number of elements in $\mu QS(h,\xi)$, the computational effort needed can quickly become prohibitive.

An alternative method consists in transforming the DNF $\alpha_1 \vee \cdots \vee \alpha_q$ into an equivalent disjunction $\gamma_1 \vee \ldots \vee \gamma_r$ with mutually disjoint formulas $\gamma_i \in \mathcal{L}_A$, that is $N_A(\gamma_i) \cap N_A(\gamma_j) = \emptyset$, whenever $i \neq j$. The probability of $QS_A(h,\xi)$ is then simply the sum of the probabilities of the individual formulas γ_i:

$$p(QS_A(h,\xi)) \;=\; p(N_A(\gamma_1) \cup \cdots \cup N_A(\gamma_r)) \;=\; \sum_{i=1}^{r} p(N_A(\gamma_i)). \tag{2.22}$$

The number of terms in such a sum is often much smaller than the number of terms in (2.21). However, the problem of computing such a disjoint representation of $QS_A(h,\xi)$ remains. In addition, the disjoint form must be such that $p(N_A(\gamma_i))$ can be computed easily.

Several methods for this problem have been developed especially in reliability theory. A simple method is due to Abraham [1]. The idea is that the new disjunction $\gamma_1 \vee \ldots \vee \gamma_r$ consists of disjoint conjunctions $\gamma \in \mathcal{C}_A$. The corresponding probabilities $p(N_A(\gamma_i))$ are therefore easily computed by

$$p(N_A(\gamma_i)) \;=\; \prod_{a_i \in Lit(\gamma_i)} \pi_i \;\cdot\; \prod_{\neg a_i \in Lit(\gamma_i)} (1-\pi_i), \tag{2.23}$$

where $Lit(\gamma_i) \subseteq A^{\pm}$ denotes the corresponding set of literals. Unfortunately, this method still tends to a relatively large number of terms. Therefore, Heidtmann [28] proposed a much better but more complex method. In Heidtmann's method, every γ_i is a conjunction of one conjunction of literals and a number of negations of conjunctions of literals. Moreover, the factors in γ_i are independent. The probabilities $p(N_A(\gamma_i))$ are therefore still easy to compute. The weakness of Heidtmann's method is its restriction to monotone formulas. For a generalization of Heidtmann's method to non-monotone formulas see [9].

3 Computing Minimal Arguments

The main problem of dealing with probabilistic argumentation systems is computing minimal quasi-supports $\mu QS(h, \xi)$ for arbitrary hypotheses $h \in \mathcal{L}_{A \cup P}$. Other symbolic representations like $\mu SP(h, \xi)$ or $\mu PS(h, \xi)$ can then be derived according to Subsection 2.2 and by the methods of Subsection 2.3. Furthermore, numerical results can be derived from quasi-supports by the methods of Subsection 2.4. This section addresses therefore the problem of computing quasi-supports.

First of all, suppose that the hypothesis $h \in \mathcal{L}_{A \cup P}$ is given as CNF of the form $h = h_1 \wedge \cdots \wedge h_n$ with clauses $h_i \in \mathcal{D}_{A \cup P}$. The problem can then be solved by computing independently the quasi-supports $\mu QS(h_i, \xi)$ for all clauses h_i. Finally, $\mu QS(h, \xi)$ is obtained by combining the individual results according to (2.8). This is possible because of property (3) in Theorem 2.1. From now on, hypotheses are therefore restricted to clauses of the form $h = \ell_1 \vee \cdots \vee \ell_m \in \mathcal{D}_{A \cup P}$.

In the following subsections, the negated hypothesis $\neg h$ will play an important role. Evidently, if h is a clause, then $\neg h = \neg \ell_1 \wedge \cdots \wedge \neg \ell_m$ is a term. The corresponding sets of literals are denoted by $H = \{\ell_1, \ldots, \ell_m\}$ and $\neg H = \{\neg \ell_1, \ldots, \neg \ell_m\}$, respectively.

Furthermore, suppose that the knowledge base $\xi \in \mathcal{L}_{A \cup P}$ is given as CNF of the form $\xi = \xi_1 \wedge \cdots \wedge \xi_r$ with clauses $\xi_i \in \mathcal{D}_{A \cup P}$. The corresponding set of clauses $\Sigma = \{\xi_1, \ldots, \xi_r\}$ is called **clause representation** of ξ. Finally, let $\Sigma_H = \mu(\Sigma \cup \neg H)$ be the clause representation of $\xi \wedge \neg h$.

3.1 Computing Minimal Quasi-Supports

The problem of computing minimal quasi-supports is closely related to the problem of computing prime implicants or prime implicates. According to Theorem 2.7, quasi-supporting arguments for h are terms $\alpha \in \mathcal{C}_A$ for which $\alpha \wedge \xi \models h$ holds. This condition can be rewritten as $\alpha \models \neg \xi \vee h$ or $\alpha \models \neg \Sigma_H$, respectively. Quasi-supporting arguments are therefore implicants of $\neg \Sigma_H$ which are in $\mathcal{C}_A$. In other words, if $\alpha \in \mathcal{D}_A$ is an implicate of Σ_H, then $\neg \alpha$ is a quasi-supporting argument for h. This reflection leads to the following theorem [45]:

Theorem 3.1 *If h and ξ are two propositional sentences in $\mathcal{L}_{A \cup P}$, then*

$$\mu QS(h, \xi) \; = \; \neg(\Phi(\Sigma_H) \cap \mathcal{D}_A).$$

Clearly, computing quasi-supports according to Theorem 3.1 is only feasible when Σ_H is relatively small. The problem is that the computation of $\Phi(\Sigma_H)$

is known to be NP-hard. However, when A is relatively small, many prime implicates of Σ_H are not in $\mathcal{D}_A$ and are therefore irrelevant for the minimal quasi-support. The following subsections present a method for computing minimal quasi-supports with the aim of avoiding generating such irrelevant prime implicates.

3.1.1 Computing Prime Implicates

The problem of computing the set $\Phi(\Sigma)$ for an arbitrary clause representation $\Sigma \subseteq \mathcal{D}_P$ is addressed first. Prime implicates can be obtained by an ordered procedure based on the **resolution** principle. Given a total ordering over P, at each step, all the possible resolvents (implicates) for the current proposition are generated and added to the set of clauses. Thus, all the possible resolvents for the first proposition are computed during the first step, then all the resolvents for the second proposition are computed during the second step, and so on. Non-minimal clauses are eliminated consecutively. The resulting set of clauses at the end of this procedure is the set $\Phi(\Sigma)$. The crucial point is that when all the resolvents for a proposition have been computed at a given step, it will never be necessary to compute resolvents for that proposition again [45, 57].

More formally, let $\Sigma \subseteq \mathcal{D}_P$ be a clause representation of ξ and $x \in P$ a proposition. The set Σ can then be decomposed into three sets Σ_x (the clauses containing x as a positive literal), $\Sigma_{\bar{x}}$ (the clauses containing x as a negative literal), and $\Sigma_{\dot{x}}$ (the clauses not containing x). If $Lit(\xi)$ denotes the set of literals of the clause ξ, then

$$\begin{aligned}
\Sigma_x &= \{\xi \in \Sigma : \ x \in Lit(\xi)\}, \\
\Sigma_{\bar{x}} &= \{\xi \in \Sigma : \ \neg x \in Lit(\xi)\}, \\
\Sigma_{\dot{x}} &= \{\xi \in \Sigma : \ x \notin Lit(\xi) \text{ and } \neg x \notin Lit(\xi)\}.
\end{aligned}$$

If $\xi_1 = x \vee \vartheta_1$ and $\xi_2 = \neg x \vee \vartheta_2$ are two clauses in Σ_x and $\Sigma_{\bar{x}}$, respectively, then the clause $\rho_x(\xi_1, \xi_2) = \vartheta_1 \vee \vartheta_2$ is called **resolvent** of ξ_1 and ξ_2. Note that $\Sigma \models \rho_x(\xi_1, \xi_2)$. Resolvents of two clauses of Σ are therefore implicates of Σ. The set of all resolvents for Σ_x and $\Sigma_{\bar{x}}$ is defined as

$$R_x(\Sigma_x, \Sigma_{\bar{x}}) \ = \ \{\rho_x(\xi_1, \xi_2) : \ \xi_1 \in \Sigma_x \text{ and } \xi_2 \in \Sigma_{\bar{x}}\}. \qquad (3.24)$$

Now, a single step of the procedure for computing prime implicates consists of adding $R_x(\Sigma_x, \Sigma_{\bar{x}})$ to Σ and removing the non-minimal clauses. The resulting set of clauses

$$Cons_x(\Sigma) \ = \ \mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}})) \qquad (3.25)$$

is called **minimal consequence** of Σ relative to x.

Example 3.1 *Let $\Sigma = \{x \vee y, \neg x \vee y, \neg x \vee z, y \vee \neg z\}$ be a set of clauses. Σ can then be decomposed into $\Sigma_x = \{x \vee y\}$, $\Sigma_{\bar{x}} = \{\neg x \vee y, \neg x \vee z\}$, and $\Sigma_{\dot{x}} = \{y \vee \neg z\}$. For that situation, two resolvents can be generated, that is $R_x(\Sigma_x, \Sigma_{\bar{x}}) = \{y, y \vee z\}$, and therefore $Cons_x(\Sigma) = \{y, \neg x \vee z\}$.*

Theorem 3.2 *Let $\Sigma \subseteq \mathcal{D}_P$ be a set of clauses. If x and y are propositions in P, then*

$$\begin{array}{lrcl} (1) & Cons_x(Cons_x(\Sigma)) & = & Cons_x(\Sigma), \\ (2) & Cons_x(Cons_y(\Sigma)) & = & Cons_y(Cons_x(\Sigma)). \end{array}$$

Let $Q = \{x_1, \ldots, x_q\}$ be a subset of P. According to Theorem 3.2, it is possible to compute the minimal consequences relative to the propositions $x_i \in Q$ according to an arbitrary ordering. If $x_1 x_2 \ldots x_q$, for example, is an arbitrary sequence of the proposition in Q, then it is possible to define

$$Cons_Q(\Sigma) \; = \; Cons_{x_1} \circ \cdots \circ Cons_{x_q}(\Sigma) \qquad (3.26)$$

as the **minimal consequence** of Σ relative to Q. However, although the sequence of the propositions does not influence the resulting set $Cons_Q(\Sigma)$, it determines critically the computational efficiency of the procedure. Heuristics for good sequences are discussed in Subsection 3.5.

The problem of finding prime implicates of Σ can now be solved by computing the minimal consequence of Σ relative to the complete set P of propositions [57]:

$$\Phi(\Sigma) \; = \; Cons_P(\Sigma). \qquad (3.27)$$

The expression $\Phi(\Sigma_H)$ in Theorem 3.1 can therefore be replaced by $Cons_{A \cup P}(\Sigma_H)$ or, for example, by $Cons_P(Cons_A(\Sigma_H))$:

$$\begin{aligned} \mu QS(h, \xi) \; &= \; \neg(Cons_{A \cup P}(\Sigma_H) \cap \mathcal{D}_A) \\ &= \; \neg(Cons_P(Cons_A(\Sigma_H)) \cap \mathcal{D}_A). \qquad (3.28) \end{aligned}$$

3.1.2 Deletion

The second problem of Theorem 3.1 is the intersection of the sets $\Phi(\Sigma_H)$ and $\mathcal{D}_A$. Obviously, this is the same as deleting from $\Phi(\Sigma_H)$ all the clauses containing propositions from P. More formally, consider a clause representation $\Sigma \subseteq \mathcal{D}_P$ and a single proposition $x \in P$. The deletion of the clauses containing x can then be defined as

$$Del_x(\Sigma) \; = \; \Sigma \cap \mathcal{D}_{P-\{x\}} \; = \; \Sigma - (\Sigma_x \cup \Sigma_{\bar{x}}) \; = \; \Sigma_{\dot{x}}. \qquad (3.29)$$

This simple operation is also called **deletion** of the proposition x.

Example 3.2 *If* $\Sigma = \{x \vee y, \neg x \vee y, \neg x \vee z, y \vee \neg z\}$ *is the same set of clauses as in Example 3.1, then* $Del_x(\Sigma) = \Sigma_{\dot{x}} = \{y \vee \neg z\}$.

Theorem 3.3 *Let* $\Sigma \subseteq \mathcal{D}_P$ *be a set of clauses. If* x *and* y *are distinct propositions in* P*, then*

$$Del_y(Cons_x(\Sigma)) \;=\; Cons_x(Del_y(\Sigma)).$$

Clearly, if $Q \subseteq P$ is a set of propositions to be deleted, then it is possible to delete them in an arbitrary sequence. Therefore,

$$Del_Q(\Sigma) \;=\; Del_{x_1} \circ \cdots \circ Del_{x_q}(\Sigma) \tag{3.30}$$

denotes the deletion of all the propositions $x_i \in Q$. The expression $\Phi(\Sigma_H) \cap \mathcal{D}_A$ in Theorem 3.1 can then be replaced by $Del_P(\Phi(\Sigma_H))$. Together with the result of the previous subsection, it is possible to specify the minimal quasi-support by

$$\mu QS(h, \xi) \;=\; \neg Del_P(Cons_P(Cons_A(\Sigma_H))). \tag{3.31}$$

3.1.3 Elimination

The last expression of the previous subsection can be further developed. Observe that the same set of propositions P appears twice. Thus, the idea is to merge the operations $Cons_P$ and Del_P. For that purpose, consider a single proposition $x \in P$ and a clause representation $\Sigma \subseteq \mathcal{D}_P$. The combined operation

$$Elim_x(\Sigma) \;=\; Del_x(Cons_x(\Sigma)) = \mu(\Sigma_{\dot{x}} \cup R_x(\Sigma_x, \Sigma_{\bar{x}})) \tag{3.32}$$

is called the **elimination** of the proposition x. It is also known as the **Davis-Putnam** procedure [12, 13].

Example 3.3 *Again, let* $\Sigma = \{x \vee y, \neg x \vee y, \neg x \vee z, y \vee \neg z\}$ *be the same set of clauses as in Example 3.1, that is* $\Sigma_{\dot{x}} = \{y \vee \neg z\}$ *and* $R_x(\Sigma_x, \Sigma_{\bar{x}}) = \{y, y \vee z\}$*, and therefore* $Elim_x(\Sigma) = \{y\}$.

Theorem 3.4 *Let* $\Sigma \subseteq \mathcal{D}_P$ *be a set of clauses. If* x *and* y *are propositions in* P*, then*

$$\begin{aligned} (1) \quad Elim_x(Elim_x(\Sigma)) \;&=\; Elim_x(\Sigma), \\ (2) \quad Elim_x(Elim_y(\Sigma)) \;&=\; Elim_y(Elim_x(\Sigma)). \end{aligned}$$

Again, if $Q \subseteq P$ is a subset of propositions, then the propositions in Q can be eliminated in an arbitrary sequence. It is therefore convenient to write

$$Elim_Q(\Sigma) \;=\; Elim_{x_1} \circ \cdots \circ Elim_{x_q}(\Sigma) \tag{3.33}$$

for the elimination of all propositions $x_i \in Q$.

Theorem 3.5 *Let $\Sigma \subseteq \mathcal{D}_P$ be a set of clauses. If $Q \subseteq P$ is a subset of propositions, then*

$$Del_Q(Cons_Q(\Sigma)) \;=\; Elim_Q(\Sigma).$$

Now, by Theorem 3.5 and the result of the previous subsection, it is possible to compute the minimal quasi-support by

$$\mu QS(h, \xi) \;=\; \neg Elim_P(Cons_A(\Sigma_H)). \tag{3.34}$$

This expression describes a concrete method for the computation of minimal quasi-supports. It consists of three successive steps:

(1) compute the minimal consequence of Σ_H relative to A;
(2) eliminate all propositions in P from the result of step (1);
(3) generate minimal terms by negating the clauses obtained from step (2).

This way of computing quasi-support will be the starting point of the approximation techniques of Subsection 3.3. However, by exchanging step (1) and step (2), an alternative method for computing exact solutions is obtained. The justification for exchanging the first two steps comes from the following theorem:

Theorem 3.6 *Let $\Sigma \subseteq \mathcal{D}_P$ be a set of clauses. If Q and R are disjoint subsets of P, that is $Q \cap R = \emptyset$, then*

$$Elim_Q(Cons_R(\Sigma)) \;=\; Cons_R(Elim_Q(\Sigma)).$$

Clearly, A and P are disjoint sets of propositions, and Theorem 3.6 can thus be applied to (3.34), that is

$$\mu QS(h, \xi) \;=\; \neg Cons_A(Elim_P(\Sigma_H)). \tag{3.35}$$

An alternative method for computing quasi-supports can therefore be described as follows:

(1') eliminate all propositions in P from Σ_H;

(2') compute the minimal consequence relative to A for the result of step (1').

(3') generate minimal terms by negating the clauses obtained from step (2').

Note that $Elim_P(\Sigma_H)$ and $Cons_A(Elim_P(\Sigma_H))$ are logically equivalent sets of clauses. Thus, the set of possible scenarios is already determined by the result of step (1'), that is

$$QS_A(h,\xi) \;=\; N_A(\neg Elim_P(\Sigma_H)). \tag{3.36}$$

This remark is of particular importance in Subsection 2.4 for the computation of numerical results. The point is that the probability $p(QS_A(h,\xi))$ is already determined by $\neg Elim_P(\Sigma_H)$, that is step (2') of the above procedure will not be necessary.

3.2 Example

In this subsection, a small exemplary argumentation system will be used to illustrate the idea of the procedure described in the previous subsection. Consider two sets $P=\{x,y,z\}$ of propositions and $A=\{a,b,c\}$ of assumptions. Furthermore, let $\xi=\gamma_1\wedge\gamma_2\wedge\gamma_3\wedge\gamma_4\wedge\gamma_5$ be the knowledge base with

$$\begin{array}{lll} \gamma_1 = a\wedge x\rightarrow y, & \gamma_2 = b\rightarrow z, & \gamma_3 = \neg(y\wedge z), \\ \gamma_4 = \neg c\rightarrow x, & \gamma_5 = c\rightarrow \neg z. & \end{array}$$

The knowledge base ξ can be transformed into a corresponding clause representation $\Sigma=\{\xi_1,\ldots,\xi_5\}$ with

$$\begin{array}{lll} \xi_1 = \neg a\vee\neg x\vee y, & \xi_2 = \neg b\vee z, & \xi_3 = \neg y\vee\neg z, \\ \xi_4 = c\vee x, & \xi_5 = \neg c\vee\neg z. & \end{array}$$

3.2.1 Minimal Quasi-Support

Let y be the hypothesis to be judged in the light of the given knowledge base. The negated hypothesis $\neg y$ can then be represented by $\neg H=\{\neg y\}$, and

$$\begin{array}{rcl} \Sigma_H & = & \mu(\Sigma\cup\neg H) \\ & = & \mu\{\neg a\vee\neg x\vee y,\ \neg b\vee z,\ \neg y\vee\neg z,\ c\vee x,\ \neg c\vee\neg z,\ \neg y\} \\ & = & \{\neg a\vee\neg x\vee y,\ \neg b\vee z,\ c\vee x,\ \neg c\vee\neg z,\ \neg y\} \end{array}$$

is the clause representation of $\xi \wedge \neg y$. In the following, the expression in (3.35) will be used for the computation of the minimal quasi-support of y. The sequence of eliminations and minimal consequences is defined by:

$$\begin{aligned}
\Sigma^{(0)} &= \Sigma_H, \\
\Sigma^{(1)} &= \mathit{Elim}_x(\Sigma^{(0)}) &&= \mathit{Elim}_{\{x\}}(\Sigma_H), \\
\Sigma^{(2)} &= \mathit{Elim}_y(\Sigma^{(1)}) &&= \mathit{Elim}_{\{x,y\}}(\Sigma_H), \\
\Sigma^{(3)} &= \mathit{Elim}_z(\Sigma^{(2)}) &&= \mathit{Elim}_{\{x,y,z\}}(\Sigma_H), \\
\Sigma^{(4)} &= \mathit{Cons}_a(\Sigma^{(3)}) &&= \mathit{Cons}_{\{a\}}(\mathit{Elim}_{\{x,y,z\}}(\Sigma_H)), \\
\Sigma^{(5)} &= \mathit{Cons}_b(\Sigma^{(4)}) &&= \mathit{Cons}_{\{a,b\}}(\mathit{Elim}_{\{x,y,z\}}(\Sigma_H)), \\
\Sigma^{(6)} &= \mathit{Cons}_c(\Sigma^{(5)}) &&= \mathit{Cons}_{\{a,b,c\}}(\mathit{Elim}_{\{x,y,z\}}(\Sigma_H)).
\end{aligned}$$

The first step is eliminating x from the initial set $\Sigma^{(0)}$. For that purpose, $\Sigma^{(0)}$ must be decomposed into three sets $\Sigma_x^{(0)}$, $\Sigma_{\bar{x}}^{(0)}$, and $\Sigma_{\dot{x}}^{(0)}$:

$$\begin{aligned}
\Sigma^{(0)} &= \{\neg a \vee \neg x \vee y,\ \neg b \vee z,\ c \vee x,\ \neg c \vee \neg z,\ \neg y\}, \\
\rightarrow\ \Sigma_x^{(0)} &= \{c \vee x\}, \\
\rightarrow\ \Sigma_{\bar{x}}^{(0)} &= \{\neg a \vee \neg x \vee y\}, \\
\rightarrow\ \Sigma_{\dot{x}}^{(0)} &= \{\neg b \vee z,\ \neg c \vee \neg z,\ \neg y\}.
\end{aligned}$$

Evidently, only one resolvent exists for that case, that is the set $R_x(\Sigma_x^{(0)}, \Sigma_{\bar{x}}^{(0)}) = \{\neg a \vee c \vee y\}$ must be added to $\Sigma_{\dot{x}}^{(0)}$. The resulting set $\Sigma^{(1)}$ can then be used for the second step of the procedure, the elimination of the proposition y from $\Sigma^{(1)}$:

$$\begin{aligned}
\Sigma^{(1)} &= \{\neg a \vee c \vee y,\ \neg b \vee z,\ \neg c \vee \neg z,\ \neg y\}, \\
\rightarrow\ \Sigma_y^{(1)} &= \{\neg a \vee c \vee y\}, \\
\rightarrow\ \Sigma_{\bar{y}}^{(1)} &= \{\neg y\}, \\
\rightarrow\ \Sigma_{\dot{y}}^{(1)} &= \{\neg b \vee z,\ \neg c \vee \neg z\}.
\end{aligned}$$

Again, there is only one resolvent, and $\Sigma^{(2)}$ (the result of the second step) is obtained by adding $R_y(\Sigma_y^{(1)}, \Sigma_{\bar{y}}^{(1)}) = \{\neg a \vee c\}$ to $\Sigma_{\dot{y}}^{(1)}$. Next, z must be eliminated from $\Sigma^{(2)}$:

$$\begin{aligned}
\Sigma^{(2)} &= \{\neg a \vee c,\ \neg b \vee z,\ \neg c \vee \neg z\}, \\
\rightarrow\ \Sigma_z^{(2)} &= \{\neg b \vee z\},
\end{aligned}$$

$$\begin{aligned} \rightarrow\ \Sigma^{(2)}_{\bar{z}} &= \{\neg c \vee \neg z\}, \\ \rightarrow\ \Sigma^{(2)}_{\dot{z}} &= \{\neg a \vee c\}. \end{aligned}$$

Again, only one resolvent exists for the actual situation, that is $R_z(\Sigma^{(2)}_z, \Sigma^{(2)}_{\bar{z}}) = \{\neg a \vee c\}$ must be added to $\Sigma^{(2)}_{\dot{z}}$. The result of this is the set $\Sigma^{(3)}$. Clearly, $\Sigma^{(3)}$ contains only assumptions, because the entire set of propositions $P = \{x, y, z\}$ has been eliminated. The remaining problem then is to compute $\Phi(\Sigma^{(3)})$. The first step for this is the computation of the minimal consequence of $\Sigma^{(3)}$ relative to a:

$$\begin{aligned} \Sigma^{(3)} &= \{\neg a \vee c,\ \neg b \vee \neg c\}, \\ \rightarrow\ \Sigma^{(3)}_{a} &= \{\}, \\ \rightarrow\ \Sigma^{(3)}_{\bar{a}} &= \{\neg a \vee c\}. \end{aligned}$$

In this situation, no resolutions are possible. Therefore, nothing is added to $\Sigma^{(3)}$, that is $\Sigma^{(4)} = \Sigma^{(3)}$. Next, the minimal consequence of $\Sigma^{(4)}$ relative to b must be computed:

$$\begin{aligned} \Sigma^{(4)} &= \{\neg a \vee c,\ \neg b \vee \neg c\}, \\ \rightarrow\ \Sigma^{(4)}_{b} &= \{\}, \\ \rightarrow\ \Sigma^{(4)}_{\bar{b}} &= \{\neg b \vee \neg c\}. \end{aligned}$$

Again, no resolutions are possible and therefore $\Sigma^{(5)} = \Sigma^{(4)}$. Finally, the minimal consequence of $\Sigma^{(5)}$ relative to c must be computed:

$$\begin{aligned} \Sigma^{(5)} &= \{\neg a \vee c,\ \neg b \vee \neg c\}, \\ \rightarrow\ \Sigma^{(5)}_{c} &= \{\neg a \vee c\}, \\ \rightarrow\ \Sigma^{(5)}_{\bar{c}} &= \{\neg b \vee \neg c\}. \end{aligned}$$

Clearly, one more resolvent exists for that case, and the final set $\Sigma^{(6)}$ is obtained by adding $R_c(\Sigma^{(5)}_c, \Sigma^{(5)}_{\bar{c}}) = \{\neg a \vee \neg b\}$ to $\Sigma^{(5)}$:

$$\Sigma^{(6)} = \{\neg a \vee c,\ \neg b \vee \neg c,\ \neg a \vee \neg b\}.$$

To complete the procedure, the minimal quasi-support for y relative to ξ can be obtained by negating the clauses of $\Sigma^{(6)}$. There are three minimal quasi-supporting arguments:

$$\mu QS(y, \xi) = \{a \wedge \neg c,\ b \wedge c,\ a \wedge b\}.$$

3.2.2 Minimal Contradiction

Consider the same knowledge base $\Sigma = \{\xi_1, \ldots, \xi_5\}$ as before and let $\bot$ be the hypothesis of interest. Note that $\bot$ is equivalent with the empty clause. The negated hypothesis $\top$ is therefore the empty term, and $\neg H = \{\}$ is the corresponding representation. The knowledge base Σ and the initial set for the procedure Σ_H are therefore identical:

$$\Sigma_H \;=\; \Sigma \;=\; \{\neg a \vee \neg x \vee y,\; \neg b \vee z,\; \neg y \vee \neg z,\; c \vee x,\; \neg c \vee \neg z\}.$$

Furthermore, suppose that the same procedure is used as in the previous subsection. The sequence of sets $\Sigma^{(0)}$ to $\Sigma^{(6)}$ is then as follows:

$$\begin{aligned}
\Sigma^{(0)} &= \{\neg a \vee \neg x \vee y,\; \neg b \vee z,\; \neg y \vee \neg z,\; c \vee x,\; \neg c \vee \neg z\},\\
\Sigma^{(1)} &= \{\neg a \vee c \vee y,\; \neg b \vee z,\; \neg y \vee \neg z,\; \neg c \vee \neg z\},\\
\Sigma^{(2)} &= \{\neg a \vee c \vee \neg z,\; \neg b \vee z,\; \neg c \vee \neg z\},\\
\Sigma^{(3)} &= \{\neg a \vee \neg b \vee c,\; \neg b \vee \neg c\},\\
\Sigma^{(4)} &= \{\neg a \vee \neg b \vee c,\; \neg b \vee \neg c\},\\
\Sigma^{(5)} &= \{\neg a \vee \neg b \vee c,\; \neg b \vee \neg c\},\\
\Sigma^{(6)} &= \{\neg b \vee \neg c,\; \neg a \vee \neg b\}.
\end{aligned}$$

Note that the clause $\neg a \vee \neg b \vee c \in \Sigma^{(5)}$ is not minimal in $\Sigma^{(6)}$ and has therefore been removed. Finally, the minimal contradiction of ξ is obtained by negating the clauses of $\Sigma^{(6)}$:

$$\mu QS(\bot, \xi) \;=\; \mu I(\xi) \;=\; \{b \wedge c,\; a \wedge b\}.$$

3.2.3 Minimal Support

Again, consider the same knowledge base $\Sigma = \{\xi_1, \ldots, \xi_5\}$ as before. The minimal support of the hypothesis y can then be derived from the results of the previous subsections. The relation between support, quasi-support, and contradiction is determined by (2.4) in Subsection 2.2.2. Therefore, the problem is to compute the set difference $QS_A(y,\xi) - I_A(\xi)$ in terms of the sets $\mu QS(y,\xi)$ and $\mu I(\xi)$. As described in Subsection 2.3, this problem can be solved in two steps. First, the minimal term representation of the complement $C_A(\xi) = N_A - I_A(\xi)$ must be computed according to (2.8):

$$\begin{aligned}
\mu C(\xi) &= \mu\{\neg a \wedge \neg b,\; \neg b,\; \neg a \wedge \neg c,\; \neg b \wedge \neg c\}\\
&= \{\neg b,\; \neg a \wedge \neg c\}.
\end{aligned}$$

Finally, the minimal support is obtained by computing the minimal term representation of the intersection $QS_A(y,\xi) \cap C_A(\xi)$ according to (2.7):

$$\begin{aligned}
\mu SP(y,\xi) &= \mu(\{\alpha_1 \wedge \alpha_2 :\ \alpha_1 \in \mu QS(y,\xi),\ \alpha_2 \in \mu C(\xi)\} \cap \mathcal{C}_A) \\
&= \mu(\{a \wedge \neg b \wedge \neg c,\ a \wedge \neg a \wedge \neg c,\ b \wedge \neg b \wedge c,\ \neg a \wedge b \wedge c \wedge \neg c, \\
&\qquad a \wedge b \wedge \neg b,\ a \wedge \neg a \wedge b \wedge \neg c\} \cap \mathcal{C}_A) \\
&= \{a \wedge \neg b \wedge \neg c\}.
\end{aligned}$$

Note that the process of computing support in two steps, first by computing the corresponding quasi-support and second by computing the contradiction, is not optimal because certain parts of the computation are repeated unnecessarily. The clause $\neg a \vee c \vee y$, for example, has been computed twice in the above example. It appears in the set $\Sigma^{(1)}$ in Subsection 3.2.1 as well as in Subsection 3.2.2. Such redundant computations can be avoided by so-called updating techniques (see Subsection 3.4).

3.3 Cost-Bounded Focusing

Representing sets of scenarios by sets of minimal terms (see Subsection 2.3) is important for improving the efficiency of propositional argumentation systems. However, computing and representing sets of minimal terms like $\mu QS(h,\xi)$, $\mu SP(h,\xi)$, or $\mu PS(h,\xi)$ is only feasible for relatively small knowledge bases and small sets of assumptions. For achieving reasonable time and space complexity, strategies for computing approximated solutions are needed.

3.3.1 Cost Functions

A promising approach is to concentrate on **important** terms only. A general approach for capturing the importance or the relevance of terms is to consider a **cost function** $c : \mathcal{C}_A \longrightarrow \mathbb{R}^+$ that expresses somehow the price to pay for obtaining a term $\alpha \in \mathcal{C}_A$. Terms α with low values $c(\alpha)$ are preferred and therefore more relevant. It is assumed that $\alpha \subseteq \alpha'$ implies $c(\alpha) \leq c(\alpha')$. This condition is called the **monotonicity criterion**. Examples of common cost functions for terms $\alpha \in \mathcal{C}_A$ are

- the **length** of the term (number of literals): $c(\alpha) = |Lit(\alpha)|$,
- the **probability** of the negated term: $c(\alpha) = 1 - p(N_A(\alpha))$.

The idea of using the length of the terms as cost function is that short terms are supposed to be more weighty arguments. Clearly, if α is a term in $\mathcal{C}_A$, then an additional literal $\ell \in A^{\pm}$ is a supplementary condition to be satisfied, and $\alpha \wedge \ell$ is therefore less probable than α. From this point of view, the length of a term correlates somehow with its probability. However, if probabilities are assigned to the assumptions, then it is possible to specify the probability of a term more precisely. This is the idea behind the second cost function suggested above.

Alternatively, instead of working with cost functions, it is also possible to consider **utility functions**. However, there is a strong duality between cost and utility functions: the negation of a cost function is an utility function and vice versa. Therefore, the probability $p(N_A(\alpha))$ of a term $\alpha \in \mathcal{C}_A$, for example, is a possible utility function. If represents the idea that a higher probability increases the utility and therefore the relevance of an argument. In the sequel, only cost functions will be used.

Definition 3.1 *Let $\beta \in \mathbb{R}^+$ be a fixed bound for a monotone cost function $c(\alpha)$, then a term $\alpha \in \mathcal{C}_A$ is called*

(1) β-**relevant**, *if and only if $c(\alpha) \leq \beta$;*

(2) β-**irrelevant**, *if and only if $c(\alpha) > \beta$.*

The set of all β-relevant terms for a fixed cost bound β is denoted by

$$\mathcal{C}_A^{\beta} \;=\; \{\alpha \in \mathcal{C}_A : \; c(\alpha) \leq \beta\}. \tag{3.37}$$

Such a set is called **stable production field** [53]. Note that $\mathcal{C}_A^{\beta}$ is a **downward-closed** set. This means that $\alpha \in \mathcal{C}_A^{\beta}$ implies that every (shorter) term $\alpha' \subseteq \alpha$ is also in $\mathcal{C}_A^{\beta}$. Clearly, $\mathcal{C}_A^0 = \emptyset$ and $\mathcal{C}_A^{\infty} = \mathcal{C}_A$.

3.3.2 Cost-Bounded Quasi-Support

The problem now is to compute sets of β-relevant terms. Again, sets of minimal quasi-supporting arguments play an important role and the approximated computation of $\mu QS(h, \xi)$ is therefore considered first. For that purpose, let

$$\mu QS(h, \xi, \beta) \;=\; \mu(QS(h, \xi) \cap \mathcal{C}_A^{\beta}) \;=\; \mu QS(h, \xi) \cap \mathcal{C}_A^{\beta} \tag{3.38}$$

denote the β-relevant subset of $\mu QS(h, \xi)$ for a given cost bound β. The set $\mu QS(h, \xi, \beta)$ is called **β-relevant minimal quasi-support** for h relative to ξ. Compared to the complete set $\mu QS(h, \xi)$, it represents a possibly much smaller set of quasi-supporting scenarios. In fact, if

$$QS_A(h, \xi, \beta) \;=\; N_A(\mu QS(h, \xi, \beta)) \tag{3.39}$$

denotes the set of quasi-supporting scenarios represented by $\mu QS(h,\xi,\beta)$, then $QS_A(h,\xi,\beta) \subseteq QS_A(h,\xi)$ for any cost bound β. Further properties of the sets $QS_A(h,\xi,\beta)$ are described by the following theorem.

Theorem 3.7 *Let h and ξ propositional sentences in $\mathcal{L}_{A\cup P}$. If β_1 and β_2 are cost bounds for a monotone cost function $c(\alpha)$, then*

$$\begin{array}{ll}
(1) & QS_A(h,\xi,0) = \emptyset, \\
(2) & QS_A(h,\xi,\infty) = QS_A(h,\xi), \\
(3) & \beta \leq \beta' \ \textit{ implies } \ QS_A(h,\xi,\beta) \subseteq QS_A(h,\xi,\beta').
\end{array}$$

The hope of computing β-relevant minimal quasi-supports instead of exact solutions is that $p(QS_A(h,\xi,\beta)) \approx p(QS_A(h,\xi))$ for a reasonably small β (see Subsection 3.3.5).

3.3.3 Cost-Bounded Elimination

The computation of β-relevant minimal quasi-support can be developed on the basis of the expression in (3.34). The point is that the resolvents generated during the elimination of the propositions in P always contain more assumptions than the clauses used for the resolution. More formally, let $\xi = \ell_1 \vee \cdots \vee \ell_m$ be an arbitrary clause in $\mathcal{D}_{A\cup P}$. Then ξ can always be decomposed into sub-clauses ξ_A and ξ_P, say

$$\xi \;=\; \underbrace{\ell_1 \vee \cdots \vee \ell_k}_{\in A^{\pm}} \vee \underbrace{\ell_{k+1} \vee \cdots \vee \ell_m}_{\in P^{\pm}} \;=\; \xi_A \vee \xi_P. \qquad (3.40)$$

Note that such a clause can also be written as an implication $\neg\xi_A \rightarrow \xi_P$ where $\neg\xi_A$ is a term in $\mathcal{C}_A$. The set of clauses ξ for which $\neg\xi_A$ is in $\mathcal{C}_A^{\beta}$ can then be defined as

$$\mathcal{D}_{A\cup P}^{\beta} \;=\; \{\xi \in \mathcal{D}_{A\cup P} : \; c(\neg\xi_A) \leq \beta\}. \qquad (3.41)$$

If Σ is an arbitrary set of clauses in $\mathcal{D}_{A\cup P}$, then the intersection of the sets Σ and $\mathcal{D}_{A\cup P}^{\beta}$,

$$Cut_{\beta}(\Sigma) \;=\; \Sigma \cap \mathcal{D}_{A\cup P}^{\beta}, \qquad (3.42)$$

is called the β-**cut** of Σ. The expression in (3.38) for the β-relevant minimal quasi-support can then be rewritten with the help of (3.34) and (3.42):

$$\begin{array}{rcl}
\mu QS(h,\xi,\beta) & = & \mu QS(h,\xi) \cap \mathcal{C}_A^{\beta} \\
& = & \neg Elim_P(Cons_A(\Sigma_H)) \cap \mathcal{C}_A^{\beta} \\
& = & \neg(Elim_P(Cons_A(\Sigma_H)) \cap \mathcal{D}_{A\cup P}^{\beta}) \\
& = & \neg(Cut_{\beta}(Elim_P(Cons_A(\Sigma_H)))). \qquad (3.43)
\end{array}$$

Furthermore, consider an arbitrary proposition $x \in P$. If Σ is a set of clauses, then the combined operation

$$Elim_x^\beta(\Sigma) \;=\; Cut_\beta(Elim_x(\Sigma)) \tag{3.44}$$

is called **β-elimination** of x. Two basic properties of this operation are described by the following theorem:

Theorem 3.8 *Let $\Sigma \subseteq \mathcal{D}_{A \cup P}$ be a set of clauses and β a cost bound for a monotone cost function $c(\alpha)$. If x and y are propositions in P, then*

$$\begin{aligned}
&(1)\quad Elim_x^\beta(Elim_x^\beta(\Sigma)) \;=\; Elim_x^\beta(\Sigma),\\
&(2)\quad Elim_x^\beta(Elim_y^\beta(\Sigma)) \;=\; Elim_y^\beta(Elim_x^\beta(\Sigma)).
\end{aligned}$$

Therefore, β-elimination can be performed with an arbitrary sequence of propositions. Again, it is convenient to write

$$Elim_Q^\beta(\Sigma) \;=\; Elim_{x_1}^\beta \circ \cdots \circ Elim_{x_q}^\beta(\Sigma) \tag{3.45}$$

for the β-elimination of all propositions $x_i \in Q \subseteq P$.

Theorem 3.9 *Let $\Sigma \subseteq \mathcal{D}_{A \cup P}$ be a set of clauses and β a cost bound for a monotone cost function $c(\alpha)$. If $Q \subseteq P$ is a subset of propositions, then*

$$Cut_\beta(Elim_Q(\Sigma)) \;=\; Elim_Q^\beta(\Sigma).$$

This theorem states that instead of removing clauses which are not in $\mathcal{D}_{A\cup P}^\beta$ (i.e. clauses leading to β-irrelevant terms) at the end of the elimination process, it is also possible to remove them consecutively during the elimination process. This is the crucial point that keeps the process under control. It can be applied to (3.43) for the computation of the β-relevant minimal quasi-support:

$$\mu QS(h, \xi, \beta) \;=\; \neg Elim_P^\beta(Cons_A(\Sigma_H)). \tag{3.46}$$

The method described by (3.46) can be optimized because the result of $Cons_A(\Sigma_H)$ may already contain clauses which are not in $\mathcal{D}_{A\cup P}^\beta$. Such clauses can be eliminated immediately. The above expression can therefore be rewritten as

$$\mu QS(h, \xi, \beta) \;=\; \neg Elim_P^\beta(Cut_\beta(Cons_A(\Sigma_H))). \tag{3.47}$$

It describes now a method with four successive steps:

(1) compute the minimal consequence of Σ_H relative to A;

(2) remove from the result of step (1) clauses which are not in $\mathcal{D}^{\beta}_{A \cup P}$;

(3) with the result of step (2), perform β-elimination for all propositions in P;

(4) generate minimal terms by negating the clauses obtained from step (3).

If the cost bound increases from β to β', then only step (2), step (3), and step (4) of the above procedure must be repeated. However, instead of completely repeating step (3), it is also possible to exploit intermediate results of foregoing computations. This will be is described in Subsection 3.4.2.

3.3.4 Cost-Bounded Support and Possibility

As shown by Theorem 3.7, the approximated set $QS_A(h, \xi, \beta)$ behaves monotonically when the cost bound β changes to $\beta' \geq \beta$. This is a desirable property since it allows to improve the approximation of the exact set $QS_A(h, \xi)$ by increasing the cost bound step by step. Altering the cost bound is therefore a tool to control both the level of approximation as well as the time and space complexity of the computation.

A similar problem is the approximation of the set $SP_A(h, \xi)$. Intuitively, the approach is the same as in Subsection 3.3.2. Therefore, let

$$\mu SP(h, \xi, \beta) \;=\; \mu(SP(h, \xi) \cap \mathcal{C}^{\beta}_{A}) \;=\; \mu SP(h, \xi) \cap \mathcal{C}^{\beta}_{A}$$

be the β-relevant subset of $\mu SP(h, \xi)$ for a given cost bound β. The corresponding set of supporting scenarios $SP_A(h, \xi, \beta) \;=\; N_A(\mu SP(h, \xi, \beta))$ is then an approximation of $SP_A(h, \xi)$ with the same desirable property of monotonicity relative to an increasing cost bound β.

The problem is the computation of the set $\mu SP(h, \xi)$. So far, support has always been derived from quasi-support. As shown by (2.4) in Subsection 2.2.2, $SP_A(h, \xi)$ is determined by the set difference $QS_A(h, \xi) - QS_A(\perp, \xi)$. Unfortunately, the corresponding set difference of two approximated sets $QS_A(h, \xi, \beta)$ and $QS_A(\perp, \xi, \beta)$ does not coincide with the set $SP_A(h, \xi, \beta)$ as proposed above. It is therefore not possible to derive approximated support from approximated quasi-support as it is possible with exact sets. Note that a solution for computing $\mu SP(h, \xi)$ as defined above is not known at the moment.

Alternatively, it is possible to define $SP^*_A(h, \xi, \beta)$ as the set difference $QS_A(h, \xi, \beta) - QS_A(\perp, \xi, \beta)$. Note that $SP^*_A(h, \xi, \beta) \subseteq SP_A(h, \xi)$ for any cost bound β. Furthermore, $SP^*_A(h, \xi, 0) = \emptyset$ and $SP^*_A(h, \xi, \infty) = SP_A(h, \xi)$ (compare with (1) and (2) in Theorem 3.7). The problem is the monotonicity property of Theorem 3.7. Unfortunately, $\beta \leq \beta'$ does not necessarily imply $SP^*_A(h,\xi,\beta) \subseteq SP^*_A(h,\xi,\beta')$. A set $SP^*_A(h,\xi,\beta)$ can therefore be seen as an approximation of $SP_A(h,\xi)$, but it is not possible to control the level of approximation monotonically by altering the cost bound β.

Another problem is the approximation of the possibly supporting scenarios $PS_A(h,\xi)$. Again, it is intuitive to define

$$\mu PS(h,\xi,\beta) \;=\; \mu(PS(h,\xi) \cap \mathcal{C}^\beta_A) \;=\; \mu PS(h,\xi) \cap \mathcal{C}^\beta_A$$

as the β-relevant subset of $\mu PS(h,\xi)$. The corresponding set of possibly supporting scenarios $PS_A(h,\xi,\beta) = N_A(\mu PS(h,\xi,\beta))$ is then an approximation of $PS_A(h,\xi)$ with the same desirable property of monotonicity. The problem is again the computation of the set $\mu PS(h,\xi,\beta)$, for which no solution is known at the moment.

However, by defining $PS^*_A(h,\xi,\beta) = N_A - QS_A(\neg h,\xi,\beta)$ (compare with (2.5) in Subsection 2.2.2), it is possible to approximate the exact set $PS_A(h,\xi)$ monotonically from the other side, since $PS^*_A(h,\xi,\beta) \supseteq PS_A(h,\xi)$ for any cost bound β. Furthermore, note that $PS^*_A(h,\xi,0) = N_A$ and $PS^*_A(h,\xi,\infty) = PS_A(h,\xi)$ (compare with (1) and (2) in Theorem 3.7). Finally, $\beta \leq \beta'$ implies $PS^*_A(h,\xi,\beta) \supseteq PS^*_A(h,\xi,\beta')$.

3.3.5 Cost-Bounded Degrees of Support and Possibility

In Subsection 2.4.1, degree of support has been defined in terms of degree of quasi-support. If quasi-support is approximated by $QS_A(h,\xi,\beta)$, then it is evident to approximate degree of quasi-support by

$$dqs(h,\xi,\beta) \;=\; p(QS_A(h,\xi,\beta)). \tag{3.48}$$

Note that $dqs(h,\xi,\beta) \leq dqs(h,\xi)$. In other words, $dqs(h,\xi,\beta)$ is a lower bound for the exact value $dqs(h,\xi)$. Furthermore, increasing the cost bound from β to β' implies that $dqs(h,\xi,\beta) \leq dqs(h,\xi,\beta')$. The requirement of monotonicity relative to β is therefore satisfied for degree of quasi-support. However, by defining degree of support and degree of possibility in the same way as in Subsection 2.4.1, that is by

$$dsp(h,\xi,\beta) \;=\; \frac{dqs(h,\xi,\beta) - dqs(\bot,\xi,\beta)}{1 - dqs(\bot,\xi,\beta)}, \tag{3.49}$$

$$dps(h,\xi,\beta) \;=\; 1 - dsp(\neg h,\xi,\beta), \tag{3.50}$$

then monotonicity relative to β is no longer observed. At the moment, no solution is known for this problem. Another problem is that $dsp(h,\xi,\beta)$ is only a lower bound for the exact value $dsp(h, \xi)$. Similarly, $dps(h, \xi, \beta)$ is only an upper bound for the exact value $dps(h, \xi)$. It is therefore not possible to judge the level of approximation for a given cost bound β.

3.4 Updating

The request of computing a set of minimal arguments for a given hypothesis (e.g. $\mu QS(h, \xi)$, $\mu QS(h, \xi, \beta)$, $\mu SP(h, \xi)$, etc.) is called a **query**. If several queries for the same or a similar knowledge base are of interest, then the results of foregoing queries may be helpful to reduce computational costs. The process of exploiting intermediate results of previous computations is called **updating**. The discussion of updating techniques can be focused on (β-relevant) quasi-support. Note that processing a query $\mu QS(h, \xi, \beta)$ can be seen as computing a function with three parameters h, ξ, and β. Therefore, there are three cases where updating techniques are useful:

(1) the knowledge base ξ changes to $\xi' = \xi \wedge \tilde{\xi}$ by adding new information (respectively Σ changes to $\Sigma' = \Sigma \cup \tilde{\Sigma}$);

(2) the hypothesis h changes to $h' = h \vee \tilde{h}$ (respectively H changes to $H' = H \cup \tilde{H}$);

(3) the cost bound β increases to $\beta' \geq \beta$ in order to improve the approximation.

The second case, apparently of minor significance, is in fact crucial for computing support or degree of support for a given hypothesis h, that is when two queries $\mu QS(\bot, \xi)$ and $\mu QS(h, \xi)$ are required (respectively $\mu QS(\bot, \xi, \beta)$ and $\mu QS(h, \xi, \beta)$). The point is that $\mu QS(h, \xi)$ can be obtained from $\mu QS(\bot, \xi)$ through updating from $\bot$ to h (respectively from $\emptyset$ to H). Therefore, computing $\mu QS(\bot, \xi)$ is only necessary once at the beginning. Queries for different hypothesis can then treated through updating.

Another important remark is that quasi-support mainly depends on the set $\Sigma_H = \mu(\Sigma \cup \neg H)$ (see Subsections 3.1 and 3.3). The first two cases of the above list can therefore be considered as a single case where Σ_H changes to $\Sigma'_H = \Sigma_H \cup \tilde{\Sigma}_H$. From this point of view, $\tilde{\Sigma}_H = \tilde{\Sigma} \cup \neg\tilde{H}$ represents simultaneously new knowledge $\tilde{\xi}$ and an extension $\tilde{h}$ of the hypothesis. This case is discussed in the following subsection. Finally, case (3) is treated in Subsection 3.4.2.

3.4.1 Adding New Knowledge

The idea for the updating procedure is that every (minimal) quasi-supporting argument for h relative to ξ is also a quasi-supporting argument for $h' = h \vee \tilde{h}$

relative to $\xi' = \xi \wedge \tilde{\xi}$, that is

$$\mu QS(h,\xi) \;\subseteq\; QS(h',\xi'), \tag{3.51}$$
$$\mu QS(h,\xi,\beta) \;\subseteq\; QS(h',\xi',\beta). \tag{3.52}$$

Therefore, it is always possible to derive the new set of quasi-supporting arguments $\mu QS(h',\xi')$ from the old set $\mu QS(h,\xi)$ by

$$\mu QS(h',\xi') \;=\; \mu(\mu QS(h,\xi) \cup New), \tag{3.53}$$

where *New* is a set of new (minimal) quasi-supporting arguments for h'. Similarly, $\mu QS(h',\xi',\beta)$ can be obtained from $\mu QS(h,\xi,\beta)$ by

$$\mu QS(h',\xi',\beta) \;=\; \mu(\mu QS(h,\xi,\beta) \cup New). \tag{3.54}$$

From this point of view, the problem to be solved is to find such a set *New* of new (minimal) quasi-supporting arguments.

According to (3.34) and (3.35) in Subsection 3.1.3, the computation of minimal quasi-support is based on two interchangeable operations $Cons_A$ and $Elim_P$. Similarly, computing β-relevant minimal quasi-support involves mainly $Cons_A$, Cut_β, and $Elim_P^\beta$ (see (3.47) in Subsection 3.3.3). Therefore, updating will be investigated independently for the operations $Cons_Q(\Sigma')$, $Elim_Q(\Sigma')$, $Cut_\beta(\Sigma')$, and $Elim_Q^\beta(\Sigma')$, where Q is an arbitrary subset of propositions and $\Sigma' = \Sigma \cup \tilde{\Sigma}$ the updated set of of clauses.

Consequence: The same idea of updating quasi-support by (3.53) or (3.54) can also be applied for updating from $Cons_Q(\Sigma)$ to $Cons_Q(\Sigma \cup \tilde{\Sigma})$, that is

$$Cons_Q(\Sigma \cup \tilde{\Sigma}) \;=\; \mu(Cons_Q(\Sigma) \cup New), \tag{3.55}$$

where *New* is the new set of clauses to be computed. This problem is most easily understood if the case of a single propositions $x \in Q$ is considered first. Suppose that $Cons_x(\Sigma)$ has already been computed, then $Cons_x(\Sigma \cup \tilde{\Sigma})$ can be obtained from (3.25) by

$$\begin{aligned}
Cons_x(\Sigma \cup \tilde{\Sigma}) \;&=\; \mu(\Sigma \cup \tilde{\Sigma} \cup R_x(\Sigma_x \cup \tilde{\Sigma}_x, \Sigma_{\bar{x}} \cup \tilde{\Sigma}_{\bar{x}}))\\
=\;& \mu(\Sigma \cup \tilde{\Sigma} \cup R_x(\Sigma_x, \Sigma_{\bar{x}}) \cup R_x(\Sigma_x, \tilde{\Sigma}_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \Sigma_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \tilde{\Sigma}_{\bar{x}}))\\
=\;& \mu(\underbrace{\mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}}))}_{Cons_x(\Sigma)} \cup \underbrace{\mu(\tilde{\Sigma} \cup R_x(\Sigma_x, \tilde{\Sigma}_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \Sigma_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \tilde{\Sigma}_{\bar{x}}))}_{New_x}),
\end{aligned}$$

where New_x is the set of new consequences relative to x. Clearly, New_x depends on $\tilde{\Sigma}$, Σ_x, and $\Sigma_{\bar{x}}$. It is therefore convenient to define

$$Newcons_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}) \;=\; \mu(\tilde{\Sigma} \cup R_x(\Sigma_x, \tilde{\Sigma}_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \Sigma_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \tilde{\Sigma}_{\bar{x}})) \tag{3.56}$$

as the set of new minimal consequences relative to x. Note that the sets Σ_x and $\Sigma_{\bar{x}}$ have been used in the foregoing computation of $Cons_x(\Sigma)$.

The same idea for updating a single proposition can now be applied repeatedly to all the propositions of the set Q. If, for example, x_1 and x_2 are the first two propositions of the same sequence of propositions as used for the foregoing computation, then $Newcons_{x_1}(\tilde{\Sigma}, \Sigma_{x_1}, \Sigma_{\overline{x}_1})$ can be considered as the new set of clauses for the second step, $Newcons_{x_2}(Newcons_{x_1}(\tilde{\Sigma}, \Sigma_{x_1}, \Sigma_{\overline{x}_1}), \Sigma_{x_2}, \Sigma_{\overline{x}_2}))$ as the new set of clauses for the third step, and so on. Finally, after computing the new consequences for the entire sequence of propositions, the set *New* of new clauses is obtained for (3.55). Note that the sets Σ_{x_i} and $\Sigma_{\overline{x}_i}$ must have been stored for all $x_i \in Q$.

Elimination: The idea here is the same as above. The new result $Elim_Q(\Sigma \cup \tilde{\Sigma})$ can again be derived from the old result $Elim_Q(\Sigma)$ by

$$Elim_Q(\Sigma \cup \tilde{\Sigma}) \;=\; \mu(Elim_Q(\Sigma) \cup New), \tag{3.57}$$

where *New* is a new set of clauses to be determined. In this situation, the case of a single proposition $x \in Q$ can be derived from (3.32) by

$$\begin{aligned}
Elim_x(\Sigma \cup \tilde{\Sigma}) \;&=\; \mu(\Sigma_{\dot{x}} \cup \tilde{\Sigma}_{\dot{x}} \cup R_x(\Sigma_x \cup \tilde{\Sigma}_x, \Sigma_{\bar{x}} \cup \tilde{\Sigma}_{\bar{x}}))\\
&=\; \mu(\Sigma_{\dot{x}} \cup \tilde{\Sigma}_{\dot{x}} \cup R_x(\Sigma_x, \Sigma_{\bar{x}}) \cup R_x(\Sigma_x, \tilde{\Sigma}_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \Sigma_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \tilde{\Sigma}_{\bar{x}}))\\
&=\; \mu(\underbrace{\mu(\Sigma_{\dot{x}} \cup R_x(\Sigma_x, \Sigma_{\bar{x}}))}_{Elim_x(\Sigma)} \cup \underbrace{\mu(\tilde{\Sigma}_{\dot{x}} \cup R_x(\Sigma_x, \tilde{\Sigma}_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \Sigma_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \tilde{\Sigma}_{\bar{x}}))}_{New_x}),
\end{aligned}$$

where New_x is the new set of clauses obtained after eliminating x from $\Sigma \cup \tilde{\Sigma}$. Again, the set New_x depends on the sets $\tilde{\Sigma}$, Σ_x, and $\Sigma_{\bar{x}}$, and it can therefore be defined as

$$Newelim_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}) \;=\; \mu(\tilde{\Sigma}_{\dot{x}} \cup R_x(\Sigma_x, \tilde{\Sigma}_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \Sigma_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \tilde{\Sigma}_{\bar{x}})). \tag{3.58}$$

As above, the final result *New* is obtained by repeatedly applying (3.58) for every proposition $x_i \in Q$ and by following the sequence of propositions of the foregoing computation.

β-Cut: Updating from $Cut_\beta(\Sigma)$ to $Cut_\beta(\Sigma \cup \tilde{\Sigma})$ is simple. The procedure can be derived from (3.42) by replacing Σ with $\Sigma \cup \tilde{\Sigma}$:

$$\begin{aligned}
Cut_\beta(\Sigma \cup \tilde{\Sigma}) \;&=\; (\Sigma \cup \tilde{\Sigma}) \cap \mathcal{D}^\beta_{A \cup P} \;=\; (\Sigma \cap \mathcal{D}^\beta_{A \cup P}) \cup (\tilde{\Sigma} \cap \mathcal{D}^\beta_{A \cup P})\\
&=\; Cut_\beta(\Sigma) \cup Cut_\beta(\tilde{\Sigma}).
\end{aligned}$$

Therefore, the only thing to do is to compute $Cut_\beta(\tilde{\Sigma})$ for the new set of clauses $\tilde{\Sigma}$ and to add the result to the previously computed set $Cut_\beta(\Sigma)$.

β-Elimination: The idea here again is that the new result $Elim_Q^\beta(\Sigma \cup \tilde{\Sigma})$ can be derived from the old result $Elim_Q^\beta(\Sigma)$ by

$$Elim_Q^\beta(\Sigma \cup \tilde{\Sigma}) \;=\; \mu(Elim_Q^\beta(\Sigma) \cup New), \tag{3.59}$$

where New is a new set of clauses to be determined. If $x \in Q$ is a single proposition to be eliminated, then (3.44) can be developed as follows:

$$\begin{aligned} Elim_x^\beta(\Sigma \cup \tilde{\Sigma}) \;&=\; Cut_\beta(Elim_x(\Sigma \cup \tilde{\Sigma})) \\ =\;& Cut_\beta(\mu(Elim_x(\Sigma) \cup Newelim_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}))) \\ =\;& \mu(Cut_\beta(Elim_x(\Sigma) \cup Newelim_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}))) \\ =\;& \mu(\underbrace{Cut_\beta(Elim_x(\Sigma))}_{Elim_x^\beta(\Sigma)} \cup \underbrace{Cut_\beta(Newelim_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}))}_{New_x}). \end{aligned}$$

Clearly, the new set of clauses New_x depends on $\tilde{\Sigma}$, Σ_x, and $\Sigma_{\bar{x}}$. It can therefore be defined as

$$\begin{aligned} Newelim_x^\beta(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}) \;&=\; Cut_\beta(Newelim_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}})) \\ =\;& \mu(\tilde{\Sigma}_{\dot{x}} \cup R_x(\Sigma_x, \tilde{\Sigma}_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \Sigma_{\bar{x}}) \cup R_x(\tilde{\Sigma}_x, \tilde{\Sigma}_{\bar{x}})) \cap \mathcal{D}_{A \cup P}^\beta. \end{aligned} \tag{3.60}$$

Finally, the resulting set New in (3.59) is obtained by repeatedly applying (3.60) for every proposition $x_i \in Q$. Again, it is important to respect the sequence of propositions of the foregoing computation.

3.4.2 Increasing the Cost Bound

Computing β-relevant minimal quasi-support is described by (3.47) in Subsection 3.3.3. It involves mainly $Cons_A$, Cut_β, and $Elim_P^\beta$. Clearly, increasing the cost bound from β to β' does not affect $Cons_A$. Therefore, updating from β to β' will only be investigated for the operations $Cut_{\beta'}(\Sigma)$ and $Elim_Q^{\beta'}(\Sigma)$, where Σ is an arbitrary set of clauses and Q a subset of propositions in Σ.

β-Cut: Depending on the actual cost bound, $Cut_\beta(\Sigma)$ selects a particular subset of Σ. The set of remaining clauses, that is clauses not selected by $Cut_\beta(\Sigma)$, is defined as

$$\overline{Cut}_\beta(\Sigma) \;=\; \Sigma - Cut_\beta(\Sigma) \;=\; \Sigma - \mathcal{D}_{A \cup P}^\beta. \tag{3.61}$$

Evidently, $Cut_\beta(\Sigma)$ and $\overline{Cut}_\beta(\Sigma)$ are disjoint and complementary sets with respect to Σ. The new situation, after the cost bound β has increased to β', can now be described as

$$\begin{aligned} Cut_{\beta'}(\Sigma) &= Cut_{\beta'}(Cut_{\beta}(\Sigma) \cup \overline{Cut}_{\beta}(\Sigma)) \\ &= Cut_{\beta'}(Cut_{\beta}(\Sigma)) \cup Cut_{\beta'}(\overline{Cut}_{\beta}(\Sigma)) \\ &= Cut_{\beta}(\Sigma) \cup Cut_{\beta'}(\overline{Cut}_{\beta}(\Sigma)). \end{aligned} \tag{3.62}$$

The problem of updating from $Cut_{\beta}(\Sigma)$ to $Cut_{\beta'}(\Sigma)$ can therefore be solved by retrieving the set $\overline{Cut}_{\beta}(\Sigma)$ of previously irrelevant clauses.

β-Elimination: The problem here is similar as in Subsection 3.4.1. Note that according (3.47), β-elimination is required after $Cut_{\beta}(Cons_A(\Sigma_H)))$. If the cost bound has increased from β to β', then by (3.62)

$$Cut_{\beta'}(Cons_A(\Sigma_H))) \;=\; Cut_{\beta}(Cons_A(\Sigma_H)) \cup Cut_{\beta'}(\overline{Cut}_{\beta}(Cons_A(\Sigma_H))).$$

Clearly, the second part of this expression, that is $Cut_{\beta'}(\overline{Cut}_{\beta}(Cons_A(\Sigma_H)))$, can be viewed as an additional set of clauses $\tilde{\Sigma}$. Therefore, the problem to be considered is a problem of updating from $Elim_Q^{\beta}(\Sigma)$ to $Elim_Q^{\beta'}(\Sigma \cup \tilde{\Sigma})$, where Σ is an arbitrary set of clauses, Q a subset of propositions, β' the new cost bound, and $\tilde{\Sigma}$ a new set of clauses obtained from updating $Cut_{\beta}(Cons_A(\Sigma_H)))$ to $Cut_{\beta'}(Cons_A(\Sigma_H)))$. As in the previous subsection, the idea is that the new result $Elim_Q^{\beta'}(\Sigma \cup \tilde{\Sigma})$ can be derived from the old result $Elim_Q^{\beta}(\Sigma)$ by

$$Elim_Q^{\beta'}(\Sigma \cup \tilde{\Sigma}) \;=\; \mu(Elim_Q^{\beta}(\Sigma) \cup New), \tag{3.63}$$

where *New* is an additional set of clauses to be determined. The case of eliminating a single proposition $x \in Q$ can be derived from (3.44) as follows:

$$\begin{aligned} Elim_x^{\beta'}(\Sigma \cup \tilde{\Sigma}) \;&=\; Cut_{\beta'}(Elim_x(\Sigma \cup \tilde{\Sigma})) \\ &= Cut_{\beta'}(\mu(Elim_x(\Sigma) \cup Newelim_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}))) \\ &= \mu(Cut_{\beta'}(Elim_x(\Sigma) \cup Newelim_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}))) \\ &= \mu(Cut_{\beta'}(Elim_x(\Sigma)) \cup Cut_{\beta'}(Newelim_x(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}}))) \\ &= \mu(Cut_{\beta'}(Elim_x(\Sigma)) \cup Newelim_x^{\beta'}(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}})) \\ &= \mu(Elim_x^{\beta'}(\Sigma) \cup Newelim_x^{\beta'}(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}})). \end{aligned} \tag{3.64}$$

Furthermore, if $\overline{Elim}_x^{\beta}(\Sigma)) = \overline{Cut}_{\beta}(Elim_x(\Sigma))$ denotes the complementary set of clauses relative to $Elim_x^{\beta}(\Sigma)$ and with respect to $Elim_x(\Sigma)$, then it is possible to rewrite $Elim_x^{\beta'}(\Sigma)$ of the above expression as follows:

$$Elim_x^{\beta'}(\Sigma) \;=\; Elim_x^{\beta}(\Sigma) \cup Cut_{\beta'}(\overline{Elim}_x^{\beta}(\Sigma)). \tag{3.65}$$

Finally, updating from $Elim_x^{\beta}(\Sigma)$ to $Elim_x^{\beta'}(\Sigma \cup \tilde{\Sigma})$ is described by the following expression:

$$Elim_x^{\beta'}(\Sigma \cup \tilde{\Sigma}) \;=\; \mu(Elim_x^{\beta}(\Sigma) \cup \underbrace{Cut_{\beta'}(\overline{Elim}_x^{\beta}(\Sigma)) \cup Newelim_x^{\beta'}(\tilde{\Sigma}, \Sigma_x, \Sigma_{\bar{x}})}_{New_x}).$$

The new set of clauses New_x depends on four sets of clauses $\tilde{\Sigma}$, Σ_x, $\Sigma_{\bar{x}}$, and $\overline{Elim}_x^{\beta}(\Sigma)$. Note that Σ_x, $\Sigma_{\bar{x}}$, and $\overline{Elim}_x^{\beta}(\Sigma)$ are intermediate results to be stored during the computation of $Elim_x^{\beta}(\Sigma)$. Finally, the idea for computing the resulting set *New* of (3.59) is the same as in the previous subsection: it is obtained by repeatedly applying the above expression for every proposition $x_i \in Q$ and by respecting the same elimination sequence.

3.5 Elimination Sequences

The efficiency of the procedures for computing (β-relevant) minimal quasi-support depends strongly on the sequence of how the propositions $x \in P$ are eliminated. More generally, if Σ is an arbitrary set of clauses and Q a subset of propositions, then the choice of a good sequence of propositions is crucial for computing $Elim_Q(\Sigma)$ and $Elim_Q^{\beta}(\Sigma)$ efficiently. Note that a good sequence of propositions is also important for computing minimal consequences $Cons_Q(\Sigma)$. However, only the case of finding good elimination sequences will be considered below.

The problem with some elimination sequences is that too many superfluous clauses are produced during the elimination process. A clause is superfluous if it is detected to be non-minimal later in the elimination process. Finding a good elimination sequence means to keep the number of superfluous clauses as small as possible. A class of simple heuristics for finding good elimination sequences is given by the following rule: if $\Sigma^{(i)}$ is the actual set of clauses at the i-th step of the elimination process, then select a proposition $x \in Q$ such that (alternatively)

(1) the number $|\Sigma_x^{(i)}| + |\Sigma_{\bar{x}}^{(i)}|$ of clauses containing x or $\neg x$,

(2) the number $|R_x(\Sigma_x^{(i)}, \Sigma_{\bar{x}}^{(i)})| = |\Sigma_x^{(i)}| \cdot |\Sigma_{\bar{x}}^{(i)}|$ of resolvents,

(3) or the balance $|\Sigma_x^{(i)}| \cdot |\Sigma_{\bar{x}}^{(i)}| - (|\Sigma_x^{(i)}| + |\Sigma_{\bar{x}}^{(i)}|)$ between added and removed clauses

is minimal. The idea behind these heuristics is to eliminate those propositions first for which the size of the actual set of clauses either decreases or remains more or less constant.

The problem with these heuristics is that the above numbers have to be recalculated after every step of the elimination process and for each of the remaining propositions. This additional computation at each step can become too expensive. An alternative approach is to fix the complete elimination sequence at the beginning of the elimination process, that is to compute the above numbers only once for the initial set of clauses $\Sigma = \Sigma^{(0)}$. Clearly, computing such a fixed elimination sequence is less expensive, but the results are correspondingly worse. However, in combination with the approximation techniques of Subsection 3.3, this method turns out to be satisfactory for most practical cases.

Another class of heuristics for finding elimination sequences is based on the following rule: at each step of the elimination process select the shortest clause $\xi_s \in \Sigma^{(i)}$ and choose arbitrarily a proposition appearing in ξ_s. The idea here is that resolutions on short clauses are also producing short resolvents for which the chance of being superfluous is rather small. Again, instead of looking for the shortest clause at each step of the process, it is possible to apply the above rule to the initial set of clauses $\Sigma = \Sigma^{(0)}$ and to fix the complete elimination sequence at the beginning of the process.

A third class of heuristics comes from the fact that a set of clauses can always be considered as a hypergraph structure. This point of view will be discussed in the following subsection.

3.6 Hypergraph and Hypertree Structures

Hypergraphs and hypertrees are suitable underlying structures for computing marginals of probability distributions and belief functions . An abstract theory of **local computation** in **valuation networks** based on two basic operations of **combination** and **marginalization** can be developed for computations on hypertrees [41, 52]. This general framework can be applied to any mathematical formalism satisfying a certain set of basic axioms. The same idea also leads to a more general theory of **information systems** [32, 37, 38].

Propositional logic satisfies the requirements of the theory of valuation networks. If two sets of propositional clauses Σ and Σ' are considered as valuations ϑ and ϑ', respectively, then $\Sigma \cup \Sigma'$ or $\mu(\Sigma \cup \Sigma')$ are possible representatives of the **combined valuation** $\vartheta \otimes \vartheta'$. Furthermore, if P is the set of propositions appearing in Σ and $Q \subseteq P$, then $Elim_{P-Q}(\Sigma)$ represents the **marginalized valuation** $\vartheta^{\downarrow Q}$. It can then be shown that the basic axioms for the local computation in valuation networks are satisfied [36]:

(1) **Commutativity and associativity of combination**: if ϑ, ϑ', and ϑ'' are valuations, then $\vartheta \otimes \vartheta' = \vartheta' \otimes \vartheta$ and $\vartheta \otimes (\vartheta' \otimes \vartheta'') = (\vartheta \otimes \vartheta') \otimes \vartheta''$.

(2) **Consonance of marginalization**: if ϑ is a valuation on P and $R \subseteq Q \subseteq P$, then $(\vartheta^{\downarrow Q})^{\downarrow R} = \vartheta^{\downarrow R}$.

(3) **Distributivity of marginalization over combination**: if ϑ and ϑ' are valuations on P and P', respectively, then $(\vartheta \otimes \vartheta')^{\downarrow P} = \vartheta \otimes (\vartheta'^{\downarrow P})$.

In the case of propositional logic, an additional basic axiom is satisfied:

(4) **Idempotency of combination**: if ϑ is a valuation on P and $Q \subseteq P$, then $\vartheta \otimes (\vartheta^{\downarrow Q}) = \vartheta$.

The idempotency axiom is the main difference between the theory of information systems and the framework of valuation networks. Thus, propositional logic leads to information systems with variables [36], for which **hypergraphs** and **hypertrees** are known to be appropriate underlying structures. Therefore, it is possible to apply the computational theory of information systems for the elimination problem in propositional logic. In this way, the knowledge base is first distributed on a corresponding propagation network, and the elimination process is then replaced by a corresponding propagation process from the leaves of the network to the root. Note that such a propagation process on a corresponding network is also generated during the elimination process as described in Subsection 3.1.3, but the nodes of the network and the messages of the propagation process are not stored explicitly. However, it is important to realize that the computational structure behind an elimination process in propositional logic is always a hypertree with a corresponding propagation network.

Hypergraphs which are not hypertrees are not directly applicable for the local computation scheme. However, it is always possible to construct corresponding **covering hypertrees**. Several heuristics for finding good covering hypertrees have been developed for that purpose [2, 8, 29, 39, 50, 56, 59]. A comparison of different heuristics can be found in [10], and an efficient algorithm is discussed in [20]. The heuristics are all based on the selection of an appropriate **constructing sequence**, which is similar to the problem of finding efficient elimination sequences as discussed in the previous subsection.

More formally, a **hypergraph** on a finite set N is defined as a collection of non-empty subsets of N [7]. Therefore, if clauses are considered as sets of literals, then a set of clauses $\Sigma \subseteq \mathcal{D}_P$ can be regarded as a hypergraph on $P^{\pm}$. Alternatively, it is possible to focus on a subset of propositions $Q \subseteq P$. If the set of literals of each clause of Σ is intersected with $Q^{\pm}$, then Σ defines a corresponding hypergraph on $Q^{\pm}$. A reversed constructing sequence for such a hypergraph can then be used as elimination sequence for $Elim_Q(\Sigma)$. In fact, the heuristics for constructing good covering hypertrees turn out to be useful alternatives for finding efficient elimination sequences.

4 Numerical Computation

The topic of this section is to show how exact numerical degrees of support and possibility can be obtained more efficiently. The method presented so far includes two computational phases. First, the sets $\mu QS(h,\xi)$, $\mu QS(\neg h,\xi)$, and $\mu QS(\bot,\xi)$ are computed as presented in Section 3 by an ordered elimination procedure. Second, according to the methods of Subsection 2.4.2, corresponding numerical values $dqs(h,\xi)$, $dqs(\neg h,\xi)$, and $dqs(\bot,\xi)$ are derived from the given prior probabilities. The resulting values $dsp(h,\xi)$ and $dps(h,\xi)$ are then determined by (2.17) and (2.18), respectively. Clearly, this approach is not very efficient, if only numerical results are required. In such a case, the sets of quasi-supporting arguments are only used as intermediate results. Note that computing exact sets of quasi-supporting arguments is often very expensive.

A more efficient approach is based on the fact that degree of quasi-support corresponds to the notion of **unnormalized belief** in the Dempster-Shafer theory of evidence [35, 51, 55, 58].Therefore, the idea of the method presented in this section is to transform the given probabilistic argumentation system (ξ, P, A, Π) into a family of independent belief functions $b_1, \ldots, b_q$. The final result can then be derived from the combined belief function $b_1 \otimes \cdots \otimes b_q$. The following picture illustrates the different ways of computing $dsp(h,\xi)$ and $dps(h,\xi)$.

$$\begin{tikzcd}
(\xi, P, A, \Pi) \arrow[r, dash, "\text{Elimination}"] \arrow[d, dash] & \begin{matrix} \mu QS(h,\xi) \\ \mu QS(\neg h,\xi) \\ \mu QS(\bot,\xi) \end{matrix} \arrow[d, dash] \\
b_1, \ldots, b_q \arrow[r, dash, "\text{Combination } \otimes"] & \begin{matrix} dsp(h,\xi) \\ dps(h,\xi) \end{matrix}
\end{tikzcd}$$

Usually, belief functions are represented more conveniently by so-called mass functions. If $\Theta = \{\theta_1, \ldots, \theta_k\}$ is an arbitrary finite set, then a **mass function** over Θ is a function $m : 2^{\Theta} \rightarrow [0,1]$ such that $\sum_{G \subseteq \Theta} m(G) = 1$. A set $G \subseteq \Theta$ with $m(G) \neq 0$ is called **focal set**. Θ is called **frame of discernment**. The combination $m_1 \otimes m_2$ of two mass functions is obtained by **Dempster's rule** of combination:

$$(m_1 \otimes m_2)(H) \;=\; \sum_{\substack{G_1, G_2 \subseteq \Theta \\ G_1 \cap G_2 = H}} m_1(G_1) \cdot m_2(G_2). \tag{4.66}$$

Furthermore, **unnormalized belief functions** $b : 2^\Theta \rightarrow [0,1]$ are constructed from mass functions by

$$b(H) \;=\; \sum_{G \subseteq H} m(G). \tag{4.67}$$

Finally, **normalized belief functions** $Bel : 2^\Theta \rightarrow [0,1]$ and **plausibility functions** $Pl : 2^\Theta \rightarrow [0,1]$ are obtained from b by:

$$Bel(H) \;=\; \frac{b(H) - b(\emptyset)}{1 - b(\emptyset)}, \tag{4.68}$$

$$Pl(H) \;=\; 1 - b(\Theta - H). \tag{4.69}$$

Note the the above above expressions are similar to (2.17) and (2.18) in Subsection 2.4.1. Therefore, if unnormalized belief and degree of quasi-support are equivalent measures, then normalized belief and degree of support are the same, as well as plausibility and degree of possibility. The equivalence between unnormalized belief and degree of quasi-support is demonstrated in the following subsection.

4.1 Constructing Independent Belief Functions

Let (ξ, P, A, Π) be a probabilistic argumentation system. The problem then is to find a belief function b such that $dqs(h,\xi) = b(N_{A\cup P}(h))$ for all $h \in \mathcal{L}_{A\cup P}$. Suppose that the knowledge base ξ is given as a set of clauses Σ. The problem can then be solved in two consecutive steps. First, by imposing certain conditions, Σ is transformed into an equivalent set of implications $\vec{\Sigma}'$. An algorithm for this will be discussed in the following subsection. Second, a corresponding mass function m is derived from $\vec{\Sigma}'$. It determines b according to (4.67).

4.1.1 The Partition Algorithm

The **partition algorithm** transforms a set of clauses $\Sigma = \{\xi_1, \ldots, \xi_r\}$ into a logically equivalent set of implications $\vec{\Sigma}'$. The idea of the algorithm is to partition the set of scenarios N_A into disjoint subsets. Every individual piece of the partition is then automatically linked to a corresponding focal set. The masses of the focal sets are determined by the probabilities π_i. First of all, note that every clause $\xi_i \in \Sigma$ can also be written as an implication

$$\xi_i \;=\; \underbrace{l_1 \vee \cdots \vee l_k}_{\in A^\pm} \vee \underbrace{l_{k+1} \vee \cdots \vee l_m}_{\in P^\pm} \;\equiv\; \neg\alpha \vee \beta \;\equiv\; \alpha \rightarrow \beta, \tag{4.70}$$

where $\alpha \in \mathcal{D}_A$ is a term and $\beta \in \mathcal{C}_P$ a clause. Therefore, let $\vec{\Sigma} = \{\alpha_1 \rightarrow \beta_1, \ldots, \alpha_r \rightarrow \beta_r\}$ denote the corresponding set of implications obtained from Σ. Clearly, Σ and $\vec{\Sigma}$ are logically equivalent. Two distinct implications of $\vec{\Sigma}$ can always be replaced by three other implications such that the new set of implications is logically equivalent to $\vec{\Sigma}$. The following **production rule** describes the procedure:

$$\left.\begin{array}{l} \alpha_i \rightarrow \beta_i \\ \alpha_j \rightarrow \beta_j \end{array}\right\} \Longrightarrow \left\{\begin{array}{l} (\alpha_i \wedge \neg\alpha_j) \rightarrow \beta_i \\ (\alpha_i \wedge \alpha_j) \rightarrow (\beta_i \wedge \beta_j)\,. \\ (\neg\alpha_i \wedge \alpha_j) \rightarrow \beta_j \end{array}\right. \tag{4.71}$$

Clearly, the application of the production rule makes only sense if $N_A(\alpha_i) \cap N_A(\alpha_j) \neq \emptyset$. Furthermore, note that $N_A(\alpha_i \wedge \neg\alpha_j)$, $N_A(\alpha_i \wedge \alpha_j)$, and $N_A(\neg\alpha_i \wedge \alpha_j)$ are mutually disjoint.

Possibly, two types of **simplification rules** can help to reduce the number of implications of $\vec{\Sigma}$. First, implications of the form $\perp \rightarrow \beta$ can always be eliminated from $\vec{\Sigma}$. Second, two implications with the same conclusion can be simplified as follows:

$$\left.\begin{array}{l} \alpha_i \rightarrow \beta \\ \alpha_j \rightarrow \beta \end{array}\right\} \Longrightarrow \quad (\alpha_i \vee \alpha_j) \rightarrow \beta. \tag{4.72}$$

The repeated application of the production and simplification rules produces a new set of implications $\{\alpha'_1 \rightarrow \beta'_1, \ldots, \alpha'_s \rightarrow \beta'_s\}$. If an additional implication $\alpha'_0 \rightarrow \beta'_0$ of the form $\neg(\vee_{i=1}^{s}\alpha_i) \rightarrow \top$ is adjoined, then the resulting set of implications $\vec{\Sigma} = \{\alpha'_0 \rightarrow \beta'_0, \ldots, \alpha'_s \rightarrow \beta'_s\}$ satisfies the following four conditions:

(1) $N_A(\alpha'_i) \cap N_A(\alpha'_j) = \emptyset$ for all $i \neq j$,

(2) $\bigcup_{i=0}^{s} N_A(\alpha'_i) = N_A$,

(3) $N_A(\alpha'_i) \neq \emptyset$,

(4) $N_P(\beta'_i) \neq N_P(\beta'_j)$ for all $i \neq j$.

From (1) and (2) follows that the implications form a **partition** on N_A, whereas (3) and (4) reveal that it is not possible to perform further simplifications.

The pseudo code of the partition algorithm is given below. It starts with $\vec{\Sigma}' = \{\top \rightarrow \top\}$ and adjoins successively every implication of $\vec{\Sigma}$. The resulting set $\vec{\Sigma}'$ is logically equivalent to $\vec{\Sigma}$ and satisfies the conditions (1) to (4):

```
Input: $\vec{\Sigma} = \{\alpha_1 \rightarrow \beta_1, \ldots, \alpha_r \rightarrow \beta_r\}$
$\vec{\Sigma}' = \{\top \rightarrow \top\}$
For each $\alpha_i \rightarrow \beta_i$ in $\vec{\Sigma}$ do
```

```
        $\Psi = \emptyset$
        **For each** $\alpha_j \rightarrow \beta_j$ in $\vec{\Sigma}'$ **do**
            $\Psi = \Psi \cup \{(\alpha_j \wedge \neg\alpha_i) \rightarrow \beta_j, (\alpha_j \wedge \alpha_i) \rightarrow (\beta_j \wedge \beta_i)\}$
        **Next**
        $\vec{\Sigma}' = Simplify(\Psi)$
    **Next**
    **Output:** $\vec{\Sigma}' = \{\alpha'_0 \rightarrow \beta'_0, \ldots, \alpha'_s \rightarrow \beta'_s\}$.
```

The purpose of *Simplify*(Ψ) is to perform all possible simplifications. In the worst case, that is if no simplifications are possible, then $\vec{\Sigma}'$ contains 2^r implications. The partition algorithm is therefore only applicable for a relatively small initial set $\vec{\Sigma}$. However, as it will be shown in Subsection 4.2, $\vec{\Sigma}$ can often be decomposed into smaller sets and the computation can be distributed on a corresponding propagation network.

4.1.2 Constructing the Mass Function

The set of implications produced by the partition algorithm is a preparation for constructing a corresponding mass function m. Clearly, the construction depends on the choice of the frame of discernment Θ. In the most general case, Θ is given by $N_{A\cup P}$. However, it is often useful to restrict Θ to N_P, that is to allow only hypotheses $h \in \mathcal{L}_P$. The benefit of this restriction is a smaller frame of discernment Θ. This can be advantageous for the problem of representing mass functions efficiently. In the following, Θ will always be restricted to N_P.

Let $\vec{\Sigma}' = \{\alpha'_0 \rightarrow \beta'_0, \ldots, \alpha'_s \rightarrow \beta'_s\}$ be a set of implications derived from Σ by the partition algorithm. A corresponding mass function $m : 2^{N_P} \rightarrow [0,1]$ for $\vec{\Sigma}'$ can then be constructed as follows:

$$m(H) = \begin{cases} p(N_A(\alpha'_i)), & \text{if } \exists \alpha'_i \rightarrow \beta'_i \in \vec{\Sigma}' \text{ such that } H = N_P(\beta'_i), \\ 0, & \text{otherwise.} \end{cases}$$

Note that the mass function is unequivocally determined for a given set of clauses Σ. Conversely, several sets of clauses may lead to the same mass function.

4.1.3 Computing Degrees of Quasi-Support

Let h be an arbitrary hypothesis in $\mathcal{L}_P$. Since Σ, $\vec{\Sigma}$, and $\vec{\Sigma}'$ are logically equivalent, it is possible to define the set $QS_A(h,\xi)$ of quasi-supporting scenarios for h in terms of the implications in $\vec{\Sigma}'$:

$$\begin{aligned} QS_A(h,\xi) &= \{\mathbf{s} \in N_A : \mathbf{s} \models_\xi h\} \\ &= \bigcup \{N_A(\alpha'_i) : \alpha'_i \rightarrow \beta'_i \in \vec{\Sigma}', \beta'_i \models h\}. \end{aligned} \quad (4.73)$$

As mentioned before, the sets $N_A(\alpha'_i)$ are mutually disjoint. The degree of quasi-support can therefore be written as a corresponding sum of probabilities of sets $N_A(\alpha'_i)$. Furthermore, if $H = N_A(h)$ represent the hypothesis h in Θ, then the equivalence between degree of quasi-support and unnormalized belief can be demonstrated as follows:

$$\begin{aligned} dqs(h,\xi) &= p(QS_A(h,\xi)) \\ &= \sum \{p(N_A(\alpha'_i)) : \ \alpha'_i \to \beta'_i \in \Sigma', \ \beta_i \models h\} \\ &= \sum_{G \subseteq H} m(G) \ = \ b(H). \end{aligned}$$

As a consequence, normalized belief and degree of support are the same, as well as plausibility and degree of possibility.

4.2 Decomposition and Local Computations

As mentioned in the previous section, the partition algorithm is only applicable for relatively small sets Σ. If the size of Σ exceeds a certain range, it may be preferable to decompose Σ into several smaller sets $\Sigma_1, \ldots, \Sigma_q$. The decomposition must be such that every assumption $a_i \in A$ occurs in at most one of these smaller sets. This requirement is needed because it allows to compute independent mass functions for each of the smaller sets. Independency is a basic requirement for Dempster's rule of combination. Therefore, if A_i denotes the set of assumptions appearing in Σ_i, then $A_i \cap A_j = \emptyset$ whenever $i \neq j$.

The simple case where $\Sigma = \{\xi_1, \ldots, \xi_r\}$ is decomposed only into two parts Σ_1 and Σ_2 will be studied first. This is not really a restriction since every decomposition $\Sigma_1, \ldots, \Sigma_q$ can be obtained by repeatedly decomposing Σ into two parts. Let A_1 and A_2 be the disjoint sets of assumptions for Σ_1 and Σ_2, respectively. Furthermore, let $\vec{\Sigma}'_1 = \{\alpha'_0 \to \beta'_0, \ldots, \alpha'_s \to \beta'_s\}$ and $\vec{\Sigma}'_2 = \{\gamma'_0 \to \delta'_0, \ldots, \gamma'_t \to \delta'_t\}$ be the corresponding sets of implications obtained from the partition algorithm. A consequence of $A_1 \cap A_2 = \emptyset$ is that

$$\vec{\Sigma}' \ = \ \{(\alpha'_i \wedge \gamma'_j) \to (\beta'_i \wedge \delta'_j) : \ \alpha_i \to \beta_i \in \vec{\Sigma}'_1, \ \gamma_j \to \delta_j \in \vec{\Sigma}'_2\}$$

is the resulting set of implications obtained from applying the partition algorithm to the initial set Σ. The set of quasi-supporting scenarios can therefore be written as

$$QS_A(h,\xi) \ = \ \bigcup \{N_A(\alpha'_i \wedge \gamma'_j) : \ (\alpha'_i \wedge \gamma'_j) \to (\beta'_i \wedge \delta'_j) \in \vec{\Sigma}', \ \beta'_i \wedge \delta'_j \models h\}.$$

Again, the sets $N_A(\alpha'_i \wedge \gamma'_j)$ are mutually disjoint. The degree of quasi-support is therefore a corresponding sum of probabilities $p(N_A(\alpha'_i \wedge \gamma'_j))$. Note that

$A_1 \cap A_2 = \emptyset$ implies that

$$p(N_A(\alpha'_i \wedge \gamma'_j)) \;=\; p(N_A(\alpha'_i)) \cdot p(N_A(\gamma'_j)).$$

As a consequence, $dqs(h, \xi)$ can be expressed in terms of the individual mass functions m_1 and m_2:

$$\begin{aligned} dqs(h,\xi) &= p(QS_A(h,\xi)) \\ &= \sum \{p(N_A(\alpha'_i \wedge \gamma'_j)) : \; (\alpha'_i \wedge \gamma'_j) \rightarrow (\beta'_i \wedge \delta'_j) \in \vec{\Sigma}',\; \beta'_i \wedge \delta'_j \models h\} \\ &= \sum \{p(N_A(\alpha'_i)) \cdot p(N_A(\gamma'_j)) : \; (\alpha'_i \wedge \gamma'_j) \rightarrow (\beta'_i \wedge \delta'_j) \in \vec{\Sigma}',\; \beta'_i \wedge \delta'_j \models h\} \\ &= \sum_{G_1 \cap G_2 \subseteq H} m_1(G_1) \cdot m_2(G_2). \end{aligned} \tag{4.74}$$

Furthermore, if $m = m_1 \otimes m_2$ denotes the combined mass functions, then it is also possible to express $b(H)$ in terms of m_1 and m_2:

$$\begin{aligned} b(H) &= \sum_{G \subseteq H} m(G) \;=\; \sum_{G \subseteq H} (m_1 \otimes m_2)(G) \\ &= \sum_{G \subseteq H} \Big(\sum_{G_1 \cap G_2 = G} m_1(G_1) \cdot m_2(G_2)\Big) \\ &= \sum_{G_1 \cap G_2 \subseteq H} m_1(G_1) \cdot m_2(G_2). \end{aligned} \tag{4.75}$$

Clearly, the expression obtained in (4.74) and (4.75) are the same. Therefore, $dqs(h, \xi)$ can be obtained by (1) decomposing Σ into Σ_1 and Σ_2, (2) applying the partition algorithm to Σ_1 and Σ_2, (3) constructing corresponding mass functions m_1 and m_2, (4) combining m_1 and m_2 by Dempster's rule, and (5) computing $b(H)$ according to (4.67).

The same procedure is also applicable if Σ is decomposed into several smaller sets $\Sigma_1, \ldots, \Sigma_q$. However, the problem is the computation of $m = m_1 \otimes \cdots \otimes m_q$, because the number of focal sets grows exponentially with q. This problem can be avoided, if every mass function is considered as a **valuation** in the sense of [41, 52]. The result can then be computed locally on a corresponding **propagation network**. The connection to the valuation framework has already been mentioned in Subsection 3.6.

5 Building Argumentation Systems on Set Constraint Logic

So far, the theory of argumentation systems and the corresponding computational techniques have been developed on the basis of propositional logic. The problem is that propositional logic is a formal language for describing statements about binary variables. This is sufficient for expressing a certain class of problems. However, describing the world on the basis of binary variables is sometimes not very convenient. For that reason, propositional logic can be generalized to **set constraint logic** (SCL) [6, 24]. The idea is that arbitrary variables are allowed, each of them having an individual set of possible values. Constraints about the possible true value of a variable are then the atoms of the language. They replace somehow the notion of a literal in propositional logic.

The SCL-framework is closely related to the the domain of **many-valued logic** (MVL) [25, 43, 44, 48]. The idea behind the MVL approach is that the set of possible truth values is extended from $\{0, 1\}$ (classical propositional logic) to an arbitrary set Θ. Depending on properties of the set Θ (finite, infinite, unordered, partially ordered, totally ordered, etc.), various classes of many-valued logics can be defined [26]. The case of a finite and unordered set Θ leads to the concept of **signed logic**, for which the resolution principle corresponds to SCL-resolution. The main difference between signed logic and the SCL-framework is that for signed logic the same number of possible values is used for all variables. From this point of view, SCL is a more general approach, since different sets of values are possible for different variables.

5.1 Set Constraint Logic

The alphabet of set constraint logic is a finite set $V = \{v_1, \ldots, v_n\}$ of variables. The true value of a variable $v \in V$ is supposed to be exactly one value of a given set of values Θ_v. Θ_v is called **frame** of v. An expression $\langle v \in X \rangle$, where X is a subset of Θ_v, is called a **set constraint**. It can be seen as a predicate that becomes true if and only if the true value of the variable v is in X. A set constraint $\langle v \in \{\theta_i\} \rangle$, $\theta_i \in \Theta_v$, is called **assignment** and is often abbreviated by $\langle v = \theta_i \rangle$. Set constraints together with the symbols $\perp$ and $\top$ can then be used to build compound **SCL-formulas**:

(1) set constraints, $\perp$ and $\top$ are SCL-formulas;

(2) if γ is a SCL-formula, then $\neg\gamma$ is a SCL-formula;

(3) if γ and δ are SCL-formulas, then $(\gamma \wedge \delta)$, $(\gamma \vee \delta)$, $(\gamma \rightarrow \delta)$, and $(\gamma \leftrightarrow \delta)$ are SCL-formulas.

Note that a fixed precedence relation on $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ allows to omit unnecessary parentheses. If V is a given set of variables, then $\mathcal{L}_V$ denotes the set of SCL-formulas over V.

The assignment of a specific value to every variable $v_i \in V$ is called an **interpretation**. The set of all possible interpretations is denoted by $N_V = \Theta_{v_1} \times \cdots \times \Theta_{v_n}$. Every interpretation $\mathbf{x} \in N_V$ can be seen as a point or a vector $\mathbf{x} = (x_1, \ldots, x_n)$ in the n-dimensional product space N_V. For a given interpretation $\mathbf{x}$, the truth value of a set constraint $\langle v_i \in X \rangle$ is 1 (true) whenever $x_i \in X$ and 0 (false) otherwise. Given the truth values of the set constraints contained in a formula, the truth value of the formula itself can be determined in the same way as in propositional logic (see Subsection 2.1.1).

$N(\gamma) \subseteq N_V$ denotes the set of all interpretations for which a formula γ is true. A SCL-formula γ **entails** a SCL-formula δ (denoted by $\gamma \models \delta$), if and only if δ is true under all interpretations for which γ is true, that is if $N(\gamma) \subseteq N(\delta)$. Furthermore, γ and δ are **equivalent** (denoted by $\gamma \equiv \delta$), if and only if the truth values of γ and δ are the same under all possible interpretations, that is if $N(\gamma) = N(\delta)$. Note that equivalent SCL-formulas represent exactly the same information or knowledge.

Let γ be a SCL-formula in $\mathcal{L}_V$ and $\mathbf{x} \in N_Q$ an interpretation relative to $Q \subseteq V$. For such a case, $\gamma_{Q \leftarrow \mathbf{x}}$ denotes the formula obtained from γ by replacing each set constraint $\langle v \in X \rangle$ by $\top$ if $x_i \in X$ or by $\perp$ if $x_i \notin X$. If δ is another SCL-formula in $\mathcal{L}_V$, then $\mathbf{x} \models_\gamma \delta$ means that $\gamma_{Q \leftarrow \mathbf{x}} \models \delta$.

Some basic properties of SCL-formulas are given by the axioms of **set theory** [19]. They are important for simplifying SCL-formulas:

$$
\begin{aligned}
&(1)\ \langle v \in \emptyset \rangle \ \equiv\ \perp, \\
&(2)\ \langle v \in \Theta_s \rangle \ \equiv\ \top, \\
&(3)\ \neg\langle v \in X \rangle \ \equiv\ \langle v \in \Theta_v - X \rangle, \\
&(4)\ \langle v \in X_1 \rangle \vee \langle v \in X_2 \rangle \ \equiv\ \langle v \in X_1 \cup X_2 \rangle, \\
&(5)\ \langle v \in X_1 \rangle \wedge \langle v \in X_2 \rangle \ \equiv\ \langle v \in X_1 \cap X_2 \rangle.
\end{aligned}
$$

For example, property (3) can be used to eliminate negations. A set constraint $\langle v \in X \rangle$ is called **proper**, if $X \neq \emptyset$ and $X \neq \Theta_s$. A disjunction of proper set constraints $\langle v_1 \in X_1 \rangle \vee \cdots \vee \langle v_q \in X_q \rangle$ where every variable occurs at most once is called a **SCL-clause**. Similarly, a conjunction of proper set constraints $\langle v_1 \in X_1 \rangle \wedge \cdots \wedge \langle v_q \in X_q \rangle$ where every variable occurs at most once is called a **SCL-term**. Arbitrary disjunctions or conjunctions of set constraints can be transformed into corresponding SCL-clauses or SCL-terms by applying properties (1) to (5).

Let φ_1 and φ_2 be SCL-clauses. φ_1 **absorbs** φ_2 whenever $N(\varphi_1) \subseteq N(\varphi_2)$. Absorption of SCL-clauses can be tested as follows: φ_1 absorbs φ_2 if and only if for every $\langle v \in X_1 \rangle$ in φ_1 there is a $\langle v \in X_2 \rangle$ in φ_2 such that $X_1 \subseteq X_2$. Similarly, a SCL-term ψ_1 **absorbs** another SCL-term ψ_2 whenever $N(\psi_1) \supseteq N(\psi_1)$. This is the case if and only if for every $\langle v \in X_1 \rangle$ in ψ_1 there is a $\langle v \in X_2 \rangle$ in ψ_2 such that $X_1 \supseteq X_2$. If Γ is a set of SCL-clauses or SCL-term, then the result of removing all absorbed SCL-clauses or SCL-term from Γ is denoted by $\mu(\Gamma)$.

A **conjunctive normal form** (CNF for short) is a finite conjunction $\varphi_1 \wedge \cdots \wedge \varphi_r$ of SCL-clauses φ_i. Note that any SCL-formula can be transformed into an equivalent conjunctive CNF. Particular CNF formulas are connected to the notion of prime implicates. A SCL-clause φ is called **implicate** of $\gamma \in \mathcal{L}_S$, if $\gamma \models \varphi$. An implicate φ of γ is called **prime implicate** of γ, if it is not absorbed by another implicate of γ. Clearly, the set of all prime implicates $\Phi(\gamma)$ defines a corresponding CNF. Note that $\gamma \equiv \Phi(\gamma)$. If Γ is the corresponding set of clauses for a CNF γ, then it is often more convenient to write $\Phi(\Gamma)$ instead of $\Phi(\gamma)$.

5.2 Constraint-Based Argumentation Systems

The concept of propositional argumentation systems can now be generalized to the SCL-framework. Let $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$ be two sets of variables with corresponding frames Θ_{v_i} and Θ_{e_i}. The elements of E are called **environmental variables**. $\mathcal{L}_{E \cup V}$ denotes the corresponding set constraint language. If ξ is a propositional sentence in $\mathcal{L}_{E \cup V}$, then a triple $\mathcal{AS}_C = (\xi, V, E)$ is called **constraint-based argumentation system**. Again, the knowledge base ξ is often given as a conjunctive set $\Sigma = \{\xi_1, \ldots, \xi_r\}$ of SCL-formulas $\xi_i \in \mathcal{L}_{E \cup V}$, or, more specifically, SCL-clauses $\xi_i \in \mathcal{D}_{E \cup V}$, where $\mathcal{D}_{E \cup V}$ denotes the set of all possible SCL-clauses over $E \cup V$.

In the propositional case, the uncertainty is captured by a special set of propositions called assumptions. The same role is now played by the environmental variables. Clearly, an assumption a_i in the propositional case can also be seen as an assignment $\langle e_i = 1 \rangle$, where e_i is a binary environmental variable with a set $\Theta_{e_i} = \{0, 1\}$ of possible values. Similarly, the negated literal $\neg a_i$ corresponds to $\langle e_i = 0 \rangle$. Propositional argumentation systems are therefore special cases of constraint-based argumentation systems. The following subsections show how the basic notions of propositional argumentation systems can be generalized.

5.2.1 Scenarios

As mentioned above, uncertainty is now captured by the environmental variables. Therefore, the possible interpretations $\mathbf{s} \in N_E$ relative to E are the

scenarios to be considered. In the same way as in the propositional case, it is possible to distinguish between **consistent** and **inconsistent** scenarios. For example, $C_E(\xi) = \{\mathbf{s} \in N_E : \mathbf{s} \not\models_\xi \perp\}$ denotes the set of all consistent scenarios relative to ξ (see Subsection 2.2).

Furthermore, when a hypothesis $h \in \mathcal{L}_{E\cup V}$ is given, particular scenarios of $\mathbf{s} \in N_E$ can be defined as **quasi-supporting**, **supporting**, **possibly supporting**, **quasi-refuting**, **refuting**, and **possibly refuting** scenarios for h relative to ξ. For example, $QS_E(h, \xi) = \{\mathbf{s} \in N_E : \mathbf{s} \models_\xi h\}$ denotes the set of all quasi-supporting scenarios for h.

5.2.2 Arguments

The idea of representing sets of scenarios by corresponding (minimal) sets of terms remains the same. Let $\mathcal{C}_E$ denote the set of all SCL-terms on E. If $S \subseteq N_E$ is a set of scenarios, then $T(S) = \{\alpha \in \mathcal{C}_E : N_E(\alpha) \subseteq S\}$ is the **term representation** and $\mu T(S)$ the **minimal term representation** of S (see Subsection 2.3). The notions of **consistent** and **inconsistent** terms are therefore defined in the same way as in the propositional case. For example, $C(\xi) = T(C_E(\xi))$ is the set of consistent terms relative to ξ. Similarly, it is possible to generalize the concept of **arguments**. For example, $QS(h, \xi) = T(QS_E(h, \xi))$ defines the set of quasi-supporting arguments for a hypothesis h. Arguments are therefore SCL-terms containing set constraints on environmental variables.

5.2.3 Probabilistic Argumentation Systems

Suppose that the set of possible values Θ_{e_i} is finite for every environmental variable $e_i \in E$. Furthermore, let $\pi_{ij} = p(e_i = \theta_{ij})$ denote the probability that the value of e_i is θ_{ij} with $\theta_{ij} \in \Theta_{e_i}$ and $\sum_j \pi_{ij} = 1$. The probability distribution associated to e_i is then represented by π_i. Finally, a tuple $\mathcal{PAS}_C = (\xi, V, E, \Pi)$, where $\Pi = \{\pi_1, \ldots, \pi_m\}$ denotes the set of probability distributions on the environmental variables, is a **probabilistic constraint-based argumentation system**.

Let $\mathbf{s} = (\theta_{1j}, \ldots, \theta_{mj})$, $\theta_{ij} \in \Theta_{e_i}$, be a particular scenario in N_E. The probability of $\mathbf{s}$ is then given by

$$p(\mathbf{s}) \;=\; \prod_{i=1}^{m} p(e_i = \theta_{ij}) \;=\; \prod_{i=1}^{m} \pi_{ij}. \tag{5.76}$$

Furthermore, if $S \subseteq N_E$ is an arbitrary set of scenarios, then the probability of S is obtained in the same way as in Subsection 2.4.1, that is by summing the probabilities of the elements of S:

$$p(S) \;=\; \sum_{\mathbf{s}\in S} p(\mathbf{s}). \tag{5.77}$$

Finally, **degree of quasi-support**, **degree of support**, and **degree of possibility** are defined as before in (2.15), (2.17), and (2.18). The remaining problem then is to compute the probability $p(QS_E(h,\xi))$. The idea of computing a disjoint form of $\mu QS(h,\xi)$ remains the same as in the propositional case. The general case of this problem has been studied in [47].

5.3 Computing Quasi-Support

The technique for computing minimal quasi-supporting arguments is based on the same idea as before. If $Cons_Q(\Sigma)$ represents the set of SCL-clauses obtained after computing all minimal consequences relative to Q and $Elim_Q(\Sigma)$ the result of eliminating from Σ the variables in Q, then $\mu QS(h,\xi)$ can be computed by

$$\begin{aligned} \mu QS(h,\xi) &= \neg(\Phi(\Sigma_H) \cap \mathcal{D}_E) \\ &= \neg Elim_V(Cons_E(\Sigma_H)), \end{aligned} \tag{5.78}$$

where Σ_H is a set of SCL-clauses representing $\xi \wedge \neg h$. The following subsections show how $Cons_Q(\Sigma)$ and $Elim_Q(\Sigma)$ can be generalized from propositional logic to set constraint logic.

5.3.1 Computing Consequences by SCL-Resolution

As before, the problem of computing $Cons_Q(\Sigma)$ for a set $Q = \{x_1,\ldots,x_q\}$ can be decomposed into a sequence $Cons_{x_1} \circ \cdots \circ Cons_{x_q}(\Sigma)$, where at each step only a single variable is considered. Therefore, if $x \in Q$ is the variable to be considered, then Σ can then be decomposed into two sets Σ_x (the clauses containing x) and $\Sigma_{\dot{x}}$ (the clauses not containing x),

$$\begin{aligned} \Sigma_x &= \{\xi \in \Sigma :\; x \in Var(\xi)\}, \\ \Sigma_{\dot{x}} &= \{\xi \in \Sigma :\; x \notin Var(\xi)\}, \end{aligned}$$

where $Var(\xi)$ denotes the set of variables of the clause ξ. If $\xi_1 = \langle x \in X_1\rangle \vee \vartheta_1$ and $\xi_2 = \langle x \in X_1\rangle \vee \vartheta_2$ are two SCL-clauses in Σ_x, then the clause

$$\rho_x(\xi_1,\xi_2) \;=\; \langle x \in X_1 \cap X_2\rangle \vee \vartheta_1 \vee \vartheta_2 \tag{5.79}$$

is called **resolvent** of ξ_1 and ξ_2. The idea is the same as in [17] where resolution is defined for a general logic. The set constraint $\langle x \in X_1 \cap X_2\rangle$ is called **residue**.
SCL-resolution is illustrated in Figure 5.3.

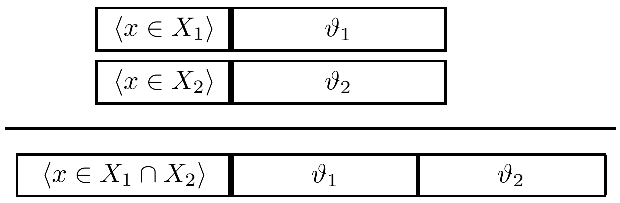
$\langle x \in X_1 \rangle$
ϑ_1
$\langle x \in X_2 \rangle$
ϑ_2
$\langle x \in X_1 \cap X_2 \rangle$
ϑ_1
ϑ_2

Figure 5.3: The resolution principle for two SCL-clauses.

If the two sets X_1 and X_2 in (5.79) are disjoint, then the residue $\langle x \in \emptyset \rangle \equiv \perp$ can be eliminated, and $\vartheta_1 \vee \vartheta_2$ is therefore the remaining resolvent. Note that this special case corresponds to the resolution principle in propositional logic. However, the situation now is more complicated. Clearly, if $X_1 \cap X_2 \neq \emptyset$, then $\rho_x(\xi_1, \xi_2)$ can be used for other resolutions with other SCL-clauses in Σ_x. Therefore, a more general resolution rule $\hat{\rho}_x(\Upsilon)$ for subsets $\Upsilon \subseteq \Sigma_x$, $|\Upsilon| \geq 2$, must be considered. If $\xi_i = \langle x \in X_i \rangle \vee \vartheta_i$ represents a SCL-clause in Υ, then $\hat{\rho}_x(\Upsilon)$ can be defined as

$$\hat{\rho}_x(\Upsilon) \;=\; \langle x \in \bigcap_{\xi_i \in \Upsilon} X_i \rangle \vee \left(\bigvee_{\xi_i \in \Upsilon} \vartheta_i \right). \tag{5.80}$$

Such a general resolution rule is possible, because $\rho_x(\xi_1, \xi_2)$ for a fixed variable x is **commutative**, **associative** and **idempotent**. Figure 5.4 illustrates SCL-resolution for a set $\Upsilon = \{\xi_1, \ldots, \xi_r\}$ of SCL-clauses.

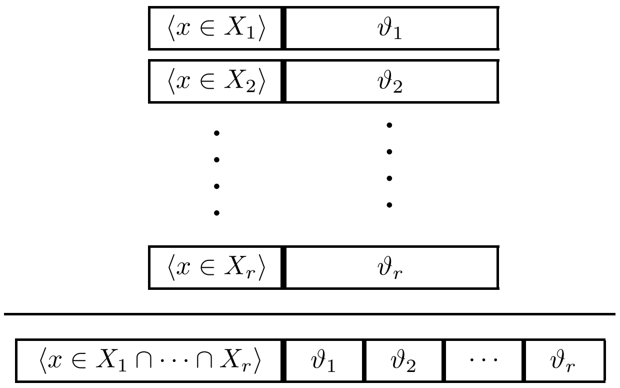
$\langle x \in X_1 \rangle$
ϑ_1
$\langle x \in X_2 \rangle$
ϑ_2
$\langle x \in X_r \rangle$
ϑ_r
$\langle x \in X_1 \cap \cdots \cap X_r \rangle$
ϑ_1
ϑ_2
$\cdots$
ϑ_r

Figure 5.4: The resolution principle for a set of SCL-clauses.

Clearly, every resolvent $\hat{\rho}_x(\Upsilon)$ is an implicate of Υ and therefore also an implicate of Σ. The set of all resolvents for Σ_x is defined as

$$R_x(\Sigma_x) \;=\; \mu\{\hat{\rho}_x(\Upsilon) : \; \Upsilon \subseteq \Sigma_x, \; |\Upsilon| \geq 2\}. \tag{5.81}$$

There are efficient algorithms for computing $R_x(\Sigma_x)$. They are based on the idea that every resolvent $\hat{\rho}_x(\Upsilon)$ can be computed as a sequence of simple resolutions $\rho_x(\xi_i, \xi_j)$ on two clauses $\xi_i, \xi_j \in \Upsilon$, and on several criterions to reduce the number of necessary sequences to be considered [6].

Finally, in accordance with (3.25) of Subsection 3.1.1, the **minimal consequence** of Σ relative to x is given by

$$Cons_x(\Sigma) \;=\; \mu(\Sigma \cup R_x(\Sigma_x)). \tag{5.82}$$

This expression describes a single step of the procedure for computing minimal consequences of Σ relative to a set of variables Q. Again, any ordering of the variables in Q can be used for the procedure. This is possible because the properties of Subsection 3.1.1 are still valid.

5.3.2 Deletion and Elimination

The problem of computing $Elim_Q(\Sigma)$ for a set $Q = \{x_1, \ldots, x_q\}$ can also be solved by decomposing the procedure into a sequence $Elim_{x_1} \circ \cdots \circ Elim_{x_q}(\Sigma)$, where at each step only a single variable is eliminated. Therefore, if $x \in Q$ is such a variable to be eliminated, then

$$\begin{aligned} Elim_x(\Sigma) \;&=\; Del_x(Cons_x(\Sigma)) \;=\; Del_x(\mu(\Sigma_{\dot{x}} \cup R_x(\Sigma_x))) \\ &=\; \mu(Del_x(\Sigma_{\dot{x}} \cup R_x(\Sigma_x))) \\ &=\; \mu(Del_x(\Sigma_{\dot{x}}) \cup Del_x(R_x(\Sigma_x)))) \\ &=\; \mu(\Sigma_{\dot{x}} \cup Del_x(R_x(\Sigma_x))). \end{aligned} \tag{5.83}$$

The difference between (3.32) and (5.83) is that the variable x may still be contained in some of the clauses in $R_x(\Sigma_x)$. Clearly, clauses containing the variable x are not allowed in $Elim_x(\Sigma)$ and are therefore deleted from $R_x(\Sigma_x)$. Note that the corresponding algorithms for computing $R_x(\Sigma_x)$ can be adapted for the purpose of deleting such clauses automatically [6].

6 Conclusion

This chapter presents the framework of probabilistic argumentation systems from a theoretical and computational point of view. It shows the state of the art by introducing properly the main theoretical concepts and by summarizing the computational methods. The aim of the theory is to judge open questions about the unknown or future world in the light of uncertain knowledge. The theory supports both a qualitative and a quantitative judgement. Compared with other theories for solving problems of inference under uncertainty, this is one of the main advantages of probabilistic argumentation systems, since most other approaches are either restricted to qualitative or quantitative aspects only.

Probabilistic argumentation systems are based on a novel combination of classical logic and probability theory. In this way, non-monotonicity is obtained in a natural and convenient way without leaving the field of classical logic. The advantage is that the richness of computational techniques for classical logic is preserved. Furthermore, the theory of probabilistic argumentation systems shows that probability theory, which is fundamental for the Bayesian approach, can also be used to obtain a more general way of reasoning under uncertainty. As shown in this chapter, the result of this more general use of probability theory is equivalent with the Dempster-Shafer theory of evidence. Therefore, an important contribution of probabilistic argumentation systems is to demonstrate how probability theory is linked with the Dempster-Shafer theory.

The main computational concept of probabilistic argumentation systems is the idea of eliminating variables or propositions. The strength of this approach is that the same principle is applicable for different formalisms such as propositional logic, set constraint logic, and also for systems with linear equations or inequalities. The theory presented in this chapter is therefore based on generic computational methods.

So far, probabilistic argumentation systems have been developed for the purpose of judging questions under uncertainty. Future work will focus on integrating the related problems of decision and action under uncertainty.

A Proofs

Remark: If γ is a propositional sentence in $\mathcal{L}_{A\cup P}$, then the corresponding set of models $N_{A\cup P}(\gamma) \subseteq N_{A\cup P}$ of γ will often be abbreviated as $N(\gamma)$.

Proof of Theorem 2.1

(1) $QS_A(\bot,\xi) \;=\; \{\mathbf{s}\in N_A :\; \mathbf{s}\models_\xi \bot\} \;=\; I_A(\xi).$ $\Box$

(2) $QS_A(\top,\xi) \;=\; \{\mathbf{s}\in N_A :\; \mathbf{s}\models_\xi \top\} \;=\; \{\mathbf{s}\in N_A :\; \xi_{A\leftarrow\mathbf{s}} \models \top\}$

$= \; \{\mathbf{s}\in N_A :\; \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N(\top)\ \}$

$= \; \{\mathbf{s}\in N_A :\; \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N_{A\cup P}\} \;=\; N_A.$ $\Box$

(3) $QS_A(h_1\wedge h_2,\xi) \;=\; \{\mathbf{s}\in N_A :\; \mathbf{s}\models_\xi h_1\wedge h_2\}$

$= \; \{\mathbf{s}\in N_A :\; \xi_{A\leftarrow\mathbf{s}}\models h_1\wedge h_2\}$

$= \; \{\mathbf{s}\in N_A :\; \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N(h_1\wedge h_2)\}$

$= \; \{\mathbf{s}\in N_A :\; \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N(h_1)\cap N(h_2)\}$

$= \; \{\mathbf{s}\in N_A :\; \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N(h_1),\ \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N(h_2)\}$

$= \; \{\mathbf{s}\in N_A :\; \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N(h_1)\}\;\cap$

$= \; \{\mathbf{s}\in N_A :\; \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N(h_2)\}$

$= \; QS_A(h_1,\xi)\cap QS_A(h_2,\xi).$ $\Box$

(4) Clearly, $h_1 \models h_1\vee h_2$ and $h_2\models h_1\vee h_2$. Thus, from (5) follows that $QS_A(h_1,\xi)\subseteq QS_A(h_1\vee h_2,\xi)$ and $QS_A(h_2,\xi)\subseteq QS_A(h_1\vee h_2,\xi)$. Therefore, $QS_A(h_1,\xi)\cup QS_A(h_2,\xi)\subseteq QS_A(h_1\vee h_2,\xi)$. $\Box$

(5) If $h_1\models h_2$ then $\exists h^*\in\mathcal{L}_{A\cup P}$ such that $h_1\equiv h_2\wedge h^*$. From (3) and (6) follows that $QS_A(h_1,\xi)=QS_A(h_2,\xi)\cap QS_A(h^*,\xi)$. Therefore, $QS_A(h_1,\xi)\subseteq QS_A(h_2,\xi)$. $\Box$

(6) This property follows immediately from $h_1\equiv h_2 \iff N(h_1)=N(h_2)$. $\Box$

Proof of Theorem 2.2

Some of the properties follow from Theorem 2.1 and (2.4):

(1) $SP_A(\bot,\xi) \;=\; QS_A(\bot,\xi)-QS_A(\bot,\xi) \;=\; \emptyset.$ $\Box$

(2) $SP_A(\top,\xi) \;=\; QS_A(\top,\xi)-QS_A(\bot,\xi) \;=\; N_A - I_A(\xi) \;=\; C_A(\xi).$ $\Box$

(3) $SP_A(h_1 \wedge h_2, \xi) \; = \; QS_A(h_1 \wedge h_2, \xi) - QS_A(\bot, \xi)$

$= \; (QS_A(h_1, \xi) \cap QS_A(h_2, \xi)) - QS_A(\bot, \xi)$

$= \; (QS_A(h_1, \xi) - QS_A(\bot, \xi)) \cap (QS_A(h_2, \xi) - QS_A(\bot, \xi))$

$= \; SP_A(h_1, \xi) \cap SP_A(h_2, \xi).$ □

(4) Clearly, $h_1 \models h_1 \vee h_2$ and $h_2 \models h_1 \vee h_2$. Thus, from (5) follows that $SP_A(h_1, \xi) \subseteq SP_A(h_1 \vee h_2, \xi)$ and $SP_A(h_2, \xi) \subseteq SP_A(h_1 \vee h_2, \xi)$. Therefore, $SP_A(h_1, \xi) \cup SP_A(h_2, \xi) \subseteq SP_A(h_1 \vee h_2, \xi)$. □

(5) If $h_1 \models h_2$ then $\exists h^* \in \mathcal{L}_{A \cup P}$ such that $h_1 \equiv h_2 \wedge h^*$. From (3) and (6) follows that $SP_A(h_1, \xi) = SP_A(h_2, \xi) \cap SP_A(h^*, \xi)$. Therefore, $SP_A(h_1, \xi) \subseteq SP_A(h_2, \xi)$. □

(6) This property follows immediately from $h_1 \equiv h_2 \iff N(h_1) = N(h_2)$. □

Proof of Theorem 2.3

Some of the properties follow from Theorem 2.1 and (2.5):

(1) $PS_A(\bot, \xi) \; = \; N_A - QS_A(\top, \xi) \; = \; N_A - N_A \; = \; \emptyset.$ □

(2) $PS_A(\top, \xi) \; = \; N_A - QS_A(\bot, \xi) \; = \; N_A - I_A(\xi) \; = \; C_A(\xi).$ □

(3) Clearly, $h_1 \wedge h_2 \models h_1$ and $h_1 \wedge h_2 \models h_2$. Thus, from (5) follows that $PS_A(h_1 \wedge h_2, \xi) \subseteq PS_A(h_1, \xi)$ and $PS_A(h_1 \wedge h_2, \xi) \subseteq PS_A(h_2, \xi)$. Therefore, $PS_A(h_1 \wedge h_2, \xi) \subseteq PS_A(h_1, \xi) \cap PS_A(h_2, \xi)$. □

(4) $PS_A(h_1 \vee h_2, \xi) \; = \; N_A - QS_A(\neg h_1 \wedge \neg h_2, \xi)$

$= \; N_A - (QS_A(\neg h_1, \xi) \cap QS_A(\neg h_2, \xi))$

$= \; (N_A - QS_A(\neg h_1, \xi)) \cup (N_A - QS_A(\neg h_2, \xi))$

$= \; PS_A(h_1, \xi) \cup PS_A(h_2, \xi).$ □

(5) If $h_1 \models h_2$ then $\exists h^* \in \mathcal{L}_{A \cup P}$ such that $h_1 \vee h^* \equiv h_2$. From (4) and (6) follows that $PS_A(h_2, \xi) = PS_A(h_1, \xi) \cup PS_A(h^*, \xi)$. Therefore, $PS_A(h_1, \xi) \subseteq PS_A(h_2, \xi)$. □

(6) This property follows immediately from $h_1 \equiv h_2 \iff N(h_1) = N(h_2)$. □

Proof of Theorem 2.4

Properties (2) and (3) follow from (2.4), (2.5), and property (1):

(1) $QS_A(h_A,\xi) \;=\; \{\mathbf{s}\in N_A:\ \mathbf{s}\models_\xi h_A\} \;=\; \{\mathbf{s}\in N_A:\ \xi_{A\leftarrow\mathbf{s}}\models h_A\}$

$$\begin{aligned}
&= \{\mathbf{s}\in N_A:\ \mathbf{s}^{\uparrow A\cup P}\cap N_{A\cup P}(\xi)\subseteq N_{A\cup P}(h_A)\}\\
&= \{\mathbf{s}\in N_A:\ (\mathbf{s}^{\uparrow A\cup P}\cap N_{A\cup P}(\xi))^{\downarrow A}\subseteq N_{A\cup P}(h_A)^{\downarrow A}\}\\
&= \{\mathbf{s}\in N_A:\ \{\mathbf{s}\}\cap N_{A\cup P}(\xi)^{\downarrow A}\subseteq N_A(h_A)\}\\
&= \{\mathbf{s}\in N_A:\ \mathbf{s}\in N_A(h_A)\text{ or }\mathbf{s}\notin N_{A\cup P}(\xi)^{\downarrow A}\}\\
&= \{\mathbf{s}\in N_A:\ \mathbf{s}\in N_A(h_A)\}\cup\{\mathbf{s}\in N_A:\ \mathbf{s}\notin N_{A\cup P}(\xi)^{\downarrow A}\}\\
&= N_A(h_A)\cup\{\mathbf{s}\in N_A:\ \{\mathbf{s}\}\cap N_{A\cup P}(\xi)^{\downarrow A}=\emptyset\}\\
&= N_A(h_A)\cup\{\mathbf{s}\in N_A:\ \mathbf{s}^{\uparrow A\cup P}\cap N_{A\cup P}(\xi)\subseteq N_{A\cup P}(\bot)\}\\
&= N_A(h_A)\cup QS_A(\bot,\xi) \;=\; N_A(h_A)\cup I_A(\xi). \qquad\square
\end{aligned}$$

(2) $SP_A(h_A,\xi) \;=\; QS_A(h_A,\xi)-QS_A(\bot,\xi)$

$$\begin{aligned}
&= (N_A(h_A)\cup QS_A(\bot,\xi))-QS_A(\bot,\xi)\\
&= N_A(h_A)-QS_A(\bot,\xi) \;=\; N_A(h_A)\cap C_A(\xi). \qquad\square
\end{aligned}$$

(3) $PS_A(h_A,\xi) \;=\; N_A-QS_A(\neg h_A,\xi) \;=\; N_A-(N_A(\neg h_A)\cup I_A(\xi))$

$$= (N_A-N_A(\neg h_A))\cap(N_A-I_A(\xi)) \;=\; N_A(h_A)\cap C_A(\xi). \qquad\square$$

Proof of Theorem 2.5

(1) This property follows immediately from (1) in Theorem 2.1 and property (3). □

(2) This property follows immediately from (2) in Theorem 2.2 and property (4). □

(3) $QS_A(h,\xi') \;=\; QS_A(h,\xi\wedge\tilde{\xi}) \;=\; \{\mathbf{s}\in N_A:\ \mathbf{s}\models_{\xi\wedge\tilde{\xi}} h\}$

$$\begin{aligned}
&= \{\mathbf{s}\in N_A:\ (\xi\wedge\tilde{\xi})_{A\leftarrow\mathbf{s}}\models h\}\\
&= \{\mathbf{s}\in N_A:\ \mathbf{s}^{\uparrow A\cup P}\cap N(\xi\wedge\tilde{\xi})\subseteq N(h)\}\\
&= \{\mathbf{s}\in N_A:\ \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\cap N(\tilde{\xi})\subseteq N(h)\}\\
&\supseteq \{\mathbf{s}\in N_A:\ \mathbf{s}^{\uparrow A\cup P}\cap N(\xi)\subseteq N(h)\} \;=\; QS_A(h,\xi). \qquad\square
\end{aligned}$$

(4) This property follows immediately from (2.5) and property (3). □

Proof of Theorem 2.6

(1) This property follows immediately from (1) in Theorem 2.1 and from (1) in Theorem 2.7. □

(2) This property follows immediately from (2) in Theorem 2.2 and from (2) in Theorem 2.7. □

Proof of Theorem 2.7

(1) $$\begin{aligned} QS(h,\xi) &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq QS_A(h,\xi)\} \\ &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq \{\mathbf{s} \in N_A : \mathbf{s} \models_\xi h\}\} \\ &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq \{\mathbf{s} \in N_A : \mathbf{s}^{\uparrow A\cup P} \cap N(\xi) \subseteq N(h)\}\} \\ &= \{\alpha \in \mathcal{C}_A : N(\alpha) \cap N(\xi) \subseteq N(h)\} = \{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h\}. \qquad \square \end{aligned}$$

(2) $$\begin{aligned} SP(h,\xi) &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq SP_A(h,\xi)\} \\ &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq (QS_A(h,\xi) - QS_A(\bot,\xi))\} \\ &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq QS_A(h,\xi),\ N_A(\alpha) \subseteq (N_A - QS_A(\bot,\xi))\} \\ &= \{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h,\ N_A(\alpha) \subseteq (N_A - N_A(\hat\alpha)),\ \forall \hat\alpha \in QS(\bot,\xi)\} \\ &= \{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h,\ N_A(\alpha) \subseteq (N_A - N_A(\hat\alpha)),\ \forall \hat\alpha \in \mathcal{C}_A,\ \hat\alpha \wedge \xi \models \bot\} \\ &= \{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h,\ N_A(\alpha) \supseteq N_A(\alpha'),\ \forall \alpha' \in \mathcal{C}_A,\ \alpha' \wedge \xi \not\models \bot\} \\ &= \{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h,\ \forall \alpha' \supseteq \alpha,\ \alpha' \in \mathcal{C}_A,\ \alpha' \wedge \xi \not\models \bot\} \qquad \square \end{aligned}$$

(3) $$\begin{aligned} PS(h,\xi) &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq PS_A(h,\xi)\} \\ &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq (N_A - QS_A(\neg h,\xi))\} \\ &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq (N_A - N_A(\hat\alpha)),\ \forall \hat\alpha \in QS(\neg h,\xi)\} \\ &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq (N_A - N_A(\hat\alpha)),\ \forall \hat\alpha \in \mathcal{C}_A,\ \hat\alpha \wedge \xi \models \neg h\} \\ &= \{\alpha \in \mathcal{C}_A : N_A(\alpha) \supseteq N_A(\alpha'),\ \forall \alpha' \in \mathcal{C}_A,\ \alpha' \wedge \xi \not\models \neg h\} \\ &= \{\alpha \in \mathcal{C}_A : \forall \alpha' \supseteq \alpha,\ \alpha' \in \mathcal{C}_A,\ \alpha' \wedge \xi \not\models \neg h\}. \qquad \square \end{aligned}$$

Proof of Theorem 3.1

This theorem can be proved with the aid of Theorem 2.7:

$$\begin{aligned} \mu QS(h,\xi) &= \mu\{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h\} = \mu\{\alpha \in \mathcal{C}_A : \alpha \models \neg\xi \vee h\} \\ &= \mu(\{\alpha \in \mathcal{C}_{A\cup P} : \alpha \models \neg\xi \vee h\} \cap \mathcal{C}_A) = \mu(\{\alpha \in \mathcal{C}_{A\cup P} : \alpha \models \neg\xi \vee h\}) \cap \mathcal{C}_A \\ &= \Psi(\neg\xi \vee h) \cap \mathcal{C}_A = \neg\Phi(\xi \wedge \neg h) \cap \mathcal{C}_A = \neg(\Phi(\Sigma_H) \cap \mathcal{D}_A). \qquad \square \end{aligned}$$

Proof of Theorem 3.2

(1) Let $\Sigma' = Cons_x(\Sigma) = \mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}}))$ be the set of clauses after the first step. Clearly, $\Sigma'_x \subseteq \Sigma_x$, $\Sigma'_{\bar{x}} \subseteq \Sigma_{\bar{x}}$, and thus $R_x(\Sigma'_x, \Sigma'_{\bar{x}}) \subseteq R_x(\Sigma_x, \Sigma_{\bar{x}})$. Therefore,

$$\begin{aligned} Cons_x(Cons_x(\Sigma)) \ &= \ Cons_x(\Sigma') \ = \ \mu(\Sigma' \cup R_x(\Sigma'_x, \Sigma'_{\bar{x}})) \\ &= \ \mu(\mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}})) \cup R_x(\Sigma'_x, \Sigma'_{\bar{x}})) \\ &= \ \mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}}) \cup R_x(\Sigma'_x, \Sigma'_{\bar{x}})) \\ &= \ \mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}})) \ = \ Cons_x(\Sigma). \end{aligned}$$

□

(2) Suppose that ξ is a clause in $Cons_y(Cons_x(\Sigma))$. Clearly, there are six possible reasons for this:

1) $\xi \in \Sigma$,
2) $\xi = \rho_x(\xi_1, \xi_2)$,
3) $\xi = \rho_y(\xi_1, \xi_2)$,
4) $\xi = \rho_y(\xi_1, \rho_x(\xi_2, \xi_3))$,
5) $\xi = \rho_y(\rho_x(\xi_1, \xi_2), \xi_3)$,
6) $\xi = \rho_y(\rho_x(\xi_1, \xi_2), \rho_x(\xi_3, \xi_4))$,

with $\xi_1, \xi_2, \xi_3, \xi_4 \in \Sigma$. Obviously, the first three cases are the same for clauses in $Cons_x(Cons_y(\Sigma))$. Furthermore, case 4) can be divided into three sub-cases, depending whether $\neg y$ is only in ξ_2, only in ξ_3, or in ξ_2 and ξ_3. Each of these sub-cases has corresponding case in $Cons_x(Cons_y(\Sigma))$:

4a) $$\begin{aligned} \xi &= \rho_y(y \vee \varphi_1, \rho_x(x \vee \neg y \vee \varphi_2, \neg x \vee \varphi_3)) \\ &= \rho_x(\rho_y(y \vee \varphi_1, x \vee \neg y \vee \varphi_2), \neg x \vee \varphi_3) \\ &= \varphi_1 \vee \varphi_2 \vee \varphi_3; \end{aligned}$$

4b) $$\begin{aligned} \xi &= \rho_y(y \vee \varphi_1, \rho_x(x \vee \varphi_2, \neg x \vee \neg y \vee \varphi_3)) \\ &= \rho_x(x \vee \varphi_2, \rho_y(y \vee \varphi_1, \neg x \vee \neg y \vee \varphi_3)) \\ &= \varphi_1 \vee \varphi_2 \vee \varphi_3; \end{aligned}$$

4c) $$\begin{aligned} \xi &= \rho_y(y \vee \varphi_1, \rho_x(x \vee \neg y \vee \varphi_2, \neg x \vee \neg y \vee \varphi_3)) \\ &= \rho_x(\rho_y(y \vee \varphi_1, x \vee \neg y \vee \varphi_2), \rho_y(y \vee \varphi_1, \neg x \vee \neg y \vee \varphi_3)) \\ &= \varphi_1 \vee \varphi_2 \vee \varphi_3; \end{aligned}$$

Similarly, case 5) can also be divided into three sub-cases, depending whether y is only in ξ_1, only in ξ_2, or in ξ_1 and ξ_2:

5a) $$\begin{aligned} \xi &= \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee \varphi_2), \neg y \vee \varphi_3) \\ &= \rho_x(\rho_y(x \vee y \vee \varphi_1, \neg y \vee \varphi_3), \neg x \vee \varphi_2) \\ &= \varphi_1 \vee \varphi_2 \vee \varphi_3; \end{aligned}$$

$$\begin{aligned}
\text{5b)}\quad \xi &= \rho_y(\rho_x(x \vee \varphi_1, \neg x \vee y \vee \varphi_2), \neg y \vee \varphi_3) \\
&= \rho_x(x \vee \varphi_1, \rho_y(\neg x \vee y \vee \varphi_2, \neg y \vee \varphi_3)) \\
&= \varphi_1 \vee \varphi_2 \vee \varphi_3;
\end{aligned}$$

$$\begin{aligned}
\text{5c)}\quad \xi &= \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee y \vee \varphi_2), \neg y \vee \varphi_3) \\
&= \rho_x(\rho_y(x \vee y \vee \varphi_1, \neg y \vee \varphi_3), \rho_y(\neg x \vee y \vee \varphi_2, \neg y \vee \varphi_3)) \\
&= \varphi_1 \vee \varphi_2 \vee \varphi_3;
\end{aligned}$$

Finally, case 6) can be divided into nine sub-cases, depending whether y is only in ξ_1, only in ξ_2, or in ξ_1 and ξ_2, and $\neg y$ is only in ξ_3, only in ξ_4, or in ξ_4 and ξ_5. Note that some of these sub-cases can be simplified in a first step:

$$\begin{aligned}
\text{6a)}\quad \xi &= \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee \varphi_2), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \varphi_4)) \\
&\Dashv \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee \varphi_2), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \varphi_2)) \\
&= \rho_x(\rho_y(x \vee y \vee \varphi_1, x \vee \neg y \vee \varphi_3), \neg x \vee \varphi_2) \\
&= \varphi_1 \vee \varphi_2 \vee \varphi_3;
\end{aligned}$$

$$\begin{aligned}
\text{6c)}\quad \xi &= \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee \varphi_2), \rho_x(x \vee \varphi_3, \neg x \vee \neg y \vee \varphi_4)) \\
&\Dashv \rho_x(x \vee \varphi_3, \neg x \vee \varphi_2) \\
&= \varphi_2 \vee \varphi_3;
\end{aligned}$$

$$\begin{aligned}
\text{6d)}\quad \xi &= \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee \varphi_2), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \neg y \vee \varphi_4)) \\
&\Dashv \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee \varphi_2), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \varphi_2)) \\
&= \rho_x(\rho_y(x \vee y \vee \varphi_1, x \vee \neg y \vee \varphi_3), \neg x \vee \varphi_2) \\
&= \varphi_1 \vee \varphi_2 \vee \varphi_3;
\end{aligned}$$

$$\begin{aligned}
\text{6e)}\quad \xi &= \rho_y(\rho_x(x \vee \varphi_1, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \varphi_4)) \\
&\Dashv \rho_x(x \vee \varphi_1, \neg x \vee \varphi_4) \\
&= \varphi_1 \vee \varphi_4;
\end{aligned}$$

$$\begin{aligned}
\text{6f)}\quad \xi &= \rho_y(\rho_x(x \vee \varphi_1, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \varphi_3, \neg x \vee \neg y \vee \varphi_4)) \\
&\Dashv \rho_y(\rho_x(x \vee \varphi_1, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \varphi_1, \neg x \vee \neg y \vee \varphi_4)) \\
&= \rho_x(x \vee \varphi_1, \rho_y(\neg x \vee y \vee \varphi_2, \neg x \vee \neg y \vee \varphi_4)) \\
&= \varphi_1 \vee \varphi_2 \vee \varphi_4;
\end{aligned}$$

$$\begin{aligned}
\text{6g)}\quad \xi &= \rho_y(\rho_x(x \vee \varphi_1, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \neg y \vee \varphi_4)) \\
&\Dashv \rho_y(\rho_x(x \vee \varphi_1, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \varphi_1, \neg x \vee \neg y \vee \varphi_4)) \\
&= \rho_x(x \vee \varphi_1, \rho_y(\neg x \vee y \vee \varphi_2, \neg x \vee \neg y \vee \varphi_4)) \\
&= \varphi_1 \vee \varphi_2 \vee \varphi_4;
\end{aligned}$$

$$\begin{aligned}
\text{6h)}\quad \xi &= \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \varphi_4)) \\
&\Dashv \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee \varphi_4), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \varphi_4)) \\
&= \rho_x(\rho_y(x \vee y \vee \varphi_1, x \vee \neg y \vee \varphi_3), \neg x \vee \varphi_4) \\
&= \varphi_1 \vee \varphi_3 \vee \varphi_4;
\end{aligned}$$

$$\begin{aligned}
\text{6i)}\quad \xi &= \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \varphi_3, \neg x \vee \neg y \vee \varphi_4)) \\
&\Dashv \rho_y(\rho_x(x \vee \varphi_3, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \varphi_3, \neg x \vee \neg y \vee \varphi_4)) \\
&= \rho_x(x \vee \varphi_3, \rho_y(\neg x \vee y \vee \varphi_2, \neg x \vee \neg y \vee \varphi_4)) \\
&= \varphi_2 \vee \varphi_3 \vee \varphi_4;
\end{aligned}$$

6j) $$\begin{aligned}\xi &= \rho_y(\rho_x(x \vee y \vee \varphi_1, \neg x \vee y \vee \varphi_2), \rho_x(x \vee \neg y \vee \varphi_3, \neg x \vee \neg y \vee \varphi_4)) \\ &= \rho_x(\rho_y(x \vee y \vee \varphi_1, x \vee \neg y \vee \varphi_3), \rho_y(\neg x \vee y \vee \varphi_2, \neg x \vee \neg y \vee \varphi_4)) \\ &= \varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4;\end{aligned}$$

Therefore, every possible clause in $Cons_y(Cons_x(\Sigma))$ has a corresponding clause in $Cons_x(Cons_y(\Sigma))$. Symmetrically, every clause in $Cons_x(Cons_y(\Sigma))$ has also a corresponding clause in $Cons_y(Cons_x(\Sigma))$. $\square$

Proof of Theorem 3.3

Let $\Sigma' = Del_y(\Sigma) = \Sigma \cap \mathcal{D}_{P-\{y\}}$ denote the set of clauses after deleting y from Σ. Therefore,

$$\begin{aligned}&Del_y(Cons_x(\Sigma)) \;=\; Del_y(\mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}})) \\ &\quad= \mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}})) \cap \mathcal{D}_{P-\{y\}} \\ &\quad= \mu(\Sigma \cup R_x(\Sigma_x, \Sigma_{\bar{x}}) \cap \mathcal{D}_{P-\{y\}}) \\ &\quad= \mu((\Sigma \cap \mathcal{D}_{P-\{y\}}) \cup (R_x(\Sigma_x, \Sigma_{\bar{x}}) \cap \mathcal{D}_{P-\{y\}})) \\ &\quad= \mu((\Sigma \cap \mathcal{D}_{P-\{y\}}) \cup R_x(\Sigma_x \cap \mathcal{D}_{P-\{y\}}, \Sigma_{\bar{x}} \cap \mathcal{D}_{P-\{y\}})) \\ &\quad= \mu(\Sigma' \cup R_x(\Sigma'_x, \Sigma'_{\bar{x}})) \;=\; Cons_x(\Sigma') \;=\; Cons_x(Del_y(\Sigma)).\end{aligned}$$ $\square$

Proof of Theorem 3.4

(1) Let $\Sigma' = Elim_x(\Sigma) = \mu(\Sigma_{\dot{x}} \cup R_x(\Sigma_x, \Sigma_{\bar{x}}))$ be the set of clauses after the first step. Clearly, $\Sigma'_x = \emptyset$, $\Sigma'_{\bar{x}} = \emptyset$, and thus $\Sigma'_{\dot{x}} = \Sigma'$. Therefore,

$$\begin{aligned}&Elim_x(Elim_x(\Sigma)) \;=\; Elim_x(\Sigma') \;=\; \mu(\Sigma'_{\dot{x}} \cup R_x(\Sigma'_x, \Sigma'_{\bar{x}})) \\ &\quad= \mu(\Sigma'_{\dot{x}} \cup R_x(\emptyset, \emptyset)) \;=\; \mu(\Sigma'_{\dot{x}} \cup \emptyset) \;=\; \Sigma'_{\dot{x}} \;=\; \Sigma_{\dot{x}} \\ &\quad= Elim_x(\Sigma).\end{aligned}$$ $\square$

(2) This property follows from (3.32), property (2) in Theorem 3.2, and Theorem 3.3:

$$\begin{aligned}&Elim_x(Elim_y(\Sigma)) \;=\; Del_x(Cons_x(Del_y(Cons_y(\Sigma)))) \\ &\quad= Del_x(Del_y(Cons_x(Cons_y(\Sigma)))) \\ &\quad= Del_y(Del_x(Cons_y(Cons_x(\Sigma)))) \\ &\quad= Del_y(Cons_y(Del_x(Cons_x(\Sigma)))) \\ &\quad= Elim_y(Elim_x(\Sigma)).\end{aligned}$$ $\square$

Proof of Theorem 3.5

Let $Q = \{x_1, \ldots, x_q\} \subseteq P$ be the propositions considered. The theorem can then be proved by repeatedly applying the result of Theorem 3.3:

$$\begin{aligned}
&Del_Q(Cons_Q(\Sigma)) \;=\; Del_{x_1} \circ \cdots \circ Del_{x_q} \circ Cons_{x_1} \circ \cdots \circ Cons_{x_q}(\Sigma)\\
&\quad=\; Del_{x_1} \circ \cdots \circ Del_{x_{q-1}} \circ Cons_{x_1} \circ Del_{x_q} \circ \cdots \circ Cons_{x_q}(\Sigma)\\
&\quad=\; \cdots \;=\; Del_{x_1} \circ \cdots Del_{x_{q-1}} \circ Cons_{x_1} \circ \cdots \circ Del_{x_q} \circ Cons_{x_q}(\Sigma)\\
&\quad=\; \cdots \;=\; Del_{x_1} \circ Cons_{x_1} \circ \cdots \circ Del_{x_q} \circ Cons_{x_q}(\Sigma)\\
&\quad=\; Elim_{x_1} \circ \cdots \circ Elim_{x_q} \;=\; Elim_Q(\Sigma). \qquad \Box
\end{aligned}$$

Proof of Theorem 3.6

Clearly, Theorem 3.2 implies that $Cons_Q(Cons_R(\Sigma)) = Cons_R(Cons_Q(\Sigma))$ for disjoint sets Q and R. Similarly, Theorem 3.3 implies that $Del_Q(Cons_R(\Sigma)) = Cons_R(Del_Q(\Sigma))$ for $Q \cap R = \emptyset$. The theorem can therefore be proved with the aid of Theorem 3.5:

$$\begin{aligned}
&Elim_Q(Cons_R(\Sigma)) \;=\; Del_Q(Cons_Q(Cons_R(\Sigma)))\\
&\quad=\; Del_Q(Cons_R(Cons_Q(\Sigma))) \;=\; Cons_R(Del_Q(Cons_Q(\Sigma)))\\
&\quad=\; Cons_R(Elim_Q(\Sigma)). \qquad \Box
\end{aligned}$$

Proof of Theorem 3.7

(1) $QS_A(h, \xi, 0) \;=\; N_A(\mu QS(h, \xi, 0)) \;=\; N_A(\mu QS(h, \xi) \cap \mathcal{C}_A^0)$
$\;=\; N_A(\mu QS(h, \xi) \cap \emptyset) \;=\; N_A(\emptyset) \;=\; \emptyset.$ $\Box$

(2) $QS_A(h, \xi, \infty) \;=\; N_A(\mu QS(h, \xi, \infty)) \;=\; N_A(\mu QS(h, \xi) \cap \mathcal{C}_A^\infty)$
$\;=\; N_A(\mu QS(h, \xi) \cap \mathcal{C}_A) \;=\; N_A(\mu QS(h, \xi)) \;=\; QS_A(h, \xi).$ $\Box$

(3) Clearly, $\beta_1 \leq \beta_2$ implies $\mathcal{C}_A^{\beta_1} \subseteq \mathcal{C}_A^{\beta_2}$. Therefore,
$\mu QS(h, \xi, \beta_1) = \mu QS(h, \xi) \cap \mathcal{C}_A^{\beta_1} \;\subseteq\; \mu QS(h, \xi) \cap \mathcal{C}_A^{\beta_2} = \mu QS(h, \xi, \beta_2),$
and consequently, $QS_A(h, \xi, \beta_1) \subseteq QS_A(h, \xi, \beta_2)$. $\Box$

Proof of Theorem 3.8

These properties follow from (3.42), (3.44), and Theorem 3.4:

(1) $Elim^\beta_x(Elim^\beta_x(\Sigma)) \;=\; Cut_\beta(Elim_x(Cut_\beta(Elim_x(\Sigma))))$

$\quad = \; Elim_x(Elim_x(\Sigma) \cap \mathcal{D}^\beta_{A\cup P}) \cap \mathcal{D}^\beta_{A\cup P} \;=\; Elim_x(Elim_x(\Sigma)) \cap \mathcal{D}^\beta_{A\cup P}$

$\quad = \; Elim_x(\Sigma) \cap \mathcal{D}^\beta_{A\cup P} \;=\; Cut_\beta(Elim_x(\Sigma)) \;=\; Elim^\beta_x(\Sigma).$ $\square$

(2) $Elim^\beta_x(Elim^\beta_y(\Sigma)) \;=\; Cut_\beta(Elim_x(Cut_\beta(Elim_y(\Sigma))))$

$\quad = \; Elim_x(Elim_y(\Sigma) \cap \mathcal{D}^\beta_{A\cup P}) \cap \mathcal{D}^\beta_{A\cup P} \;=\; Elim_x(Elim_y(\Sigma)) \cap \mathcal{D}^\beta_{A\cup P}$

$\quad = \; Elim_y(Elim_x(\Sigma)) \cap \mathcal{D}^\beta_{A\cup P} \;=\; Elim_y(Elim_x(\Sigma) \cap \mathcal{D}^\beta_{A\cup P}) \cap \mathcal{D}^\beta_{A\cup P}$

$\quad = \; Cut_\beta(Elim_y(Cut_\beta(Elim_x(\Sigma)))) \;=\; Elim^\beta_y(Elim^\beta_x(\Sigma)).$ $\square$

Proof of Theorem 3.9

Let $Q = \{x_1, \ldots, x_q\} \subseteq P$ be the propositions to be eliminated.

$Cut_\beta(Elim_Q(\Sigma)) \;=\; Elim_Q(\Sigma) \cap \mathcal{D}^\beta_{A\cup P}$

$\quad = \; [Elim_{x_1} \circ \cdots \circ Elim_{x_q}(\Sigma)] \cap \mathcal{D}^\beta_{A\cup P}$

$\quad = \; Elim_{x_1}([Elim_{x_2} \circ \cdots \circ Elim_{x_q}(\Sigma)] \cap \mathcal{D}^\beta_{A\cup P}) \cap \mathcal{D}^\beta_{A\cup P}$

$\quad = \; \cdots \;=\; Elim_{x_1}(Elim_{x_2}(\cdots(Elim_{x_q}(\Sigma) \cap \mathcal{D}^\beta_{A\cup P})\cdots) \cap \mathcal{D}^\beta_{A\cup P}) \cap \mathcal{D}^\beta_{A\cup P}$

$\quad = \; \cdots \;=\; Elim^\beta_{x_1} \circ \cdots \circ Elim^\beta_{x_q} \;=\; Elim^\beta_Q(\Sigma).$ $\square$

References

[1] J.A. Abraham. An improved algorithm for network reliability. *IEEE Transactions on Reliability*, 28:58–61, 1979.

[2] R.G. Almond and A. Kong. Optimality issues in constructing a markov tree from graphical models. Research Report A-3, Department of Statistics, Harvard University, November 1991.

[3] B. Anrig, R. Bissig, R. Haenni, J. Kohlas, and N. Lehmann. Probabilistig argumentation systems: Introduction to assumption-based modeling with ABEL. Technical Report 99-1, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1999.

[4] B. Anrig, R. Haenni, J. Kohlas, and N. Lehmann. Assumption-based modeling using ABEL. In D. Gabbay, R. Kruse, A. Nonnengart, and H.J. Ohlbach, editors, *First International Joint Conference on Qualitative and Quantitative Practical Reasoning; ECSQARU–FAPR'97*. Lecture Notes in Artif. Intell., Springer, 1997.

[5] B. Anrig, R. Haenni, and N. Lehmann. ABEL – a new language for assumption-based evidential reasoning under uncertainty. Technical Report 97–01, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1997.

[6] B. Anrig, N. Lehmann, and R. Haenni. Reasoning with finite set constraints. Working Paper 97–11, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1997.

[7] C. Berge. *Hypergraphs*. North Holland, 1989.

[8] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

[9] R. Bertschy and P.A. Monney. A generalization of the algorithm of Heidtmann to non-monotone formulas. *Journal of Computational and Applied Mathematics*, 76:55–76, 1996.

[10] A. Cano and S. Moral. Heuristic algorithms for the triangulation of graphs. In B. Bouchon-Meunier, R.R. Yager, and L.A. Zadeh, editors, *Proceedings of the Fifth IPMU Conference*, pages 166–171. Springer, 1995.

[11] C.L. Chang and R.C.T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Boston, 1973.

[12] M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 5:394–397, 1962.

[13] M. Davis and H. Putnam. A computing procedure for quantification theory. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning 1: Classical Papers on Computational Logic 1957-1966*, pages 125–139. Springer, Berlin, Heidelberg, 1983.

[14] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.

[15] J. de Kleer. Extending the ATMS. *Artificial Intelligence*, 28:163–196, 1986.

[16] D. Dubois and H. Prade. An introduction to possibilistic and fuzzy logics. In G. Shafer and J. Pearl, editors, *Readings in Uncertain Reasoning*, pages 742–761. Kaufmann, San Mateo, CA, 1990.

[17] N. Eisinger and H.J. Ohlbach. Deduction systems based on resolution. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 184–263. Oxford Science Publications, 1993.

[18] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley and Sons, 3rd edition, 1968.

[19] D. Gries and F.B. Schneider. *A Logical Approach to Discrete Mathematic.* Springer, 1993.

[20] R. Haenni, , and N. Lehmann. Efficient hypertree construction. Technical Report 99-2, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1999.

[21] R. Haenni. Assumption-based reasoning with algebraic clauses. In H.J. Zimmermann, editor, *EUFIT'97, 5th European Congress on Intelligent Techniques and Soft Computing*, pages 958–961. Verlag Mainz, 1997.

[22] R. Haenni. Modeling uncertainty with propositional assumption-based systems. In S. Parson and A. Hunter, editors, *Applications of Uncertainty Formalisms*, Lecture Notes in Artifical Intelligence 1455, pages 446–470. Springer, 1998.

[23] R. Haenni and N. Lehmann. Assumption-based reasoning with finite set constraints. In *IPMU'98, Proceedings of the seventh international conference, Paris, France*, pages 1289–1295, 1998.

[24] R. Haenni and N. Lehmann. Reasoning with finite set constraints. In *ECAI'98, Workshop W17: Many-valued logic for AI application*, pages 1–6, 1998.

[25] R. Hähnle. Short conjunctive normal forms in finitely-valued logics. *Journal of Logic and Computation*, 4(6):905–927, 1994.

[26] R. Hähnle and G. Escalada-Imaz. Deduction in many-valued logics: a survey. *Mathware & Soft Computing*, IV(2):69–97, 1997.

[27] P. Halmos. *Lectures on Boolean Algebras.* Van Nostrand-Reinhold, London, 1963.

[28] K.D. Heidtmann. Smaller sums of disjoint products by subproduct inversion. *IEEE Transactions on Reliability*, 38(3):305–311, August 1989.

[29] U. Kjærulff. Triangulation of graphs – Algorithms giving total state space. Technical Report R 90–09, Department of Mathematics and Computer Science, Aalborg University, 1990.

[30] J. Kohlas. Symbolic evidence, arguments, supports and valuation networks. In M. Kruse M. Clarke and S. Moral, editors, *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 186–198. Springer, 1993.

[31] J. Kohlas. Mathematical foundations of evidence theory. In G. Coletti, D. Dubois, and R. Scozzafava, editors, *Mathematical Models for Handling Partial Knowledge in Artificial Intelligence*, pages 31–64. Plenum Press, 1995.

[32] J. Kohlas. Computational theory for information systems. Technical Report 97–07, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1997.

[33] J. Kohlas and P. Besnard. An algebraic study of argumentation systems and evidence theory. Technical Report 95–13, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1995.

[34] J. Kohlas and R. Haenni. Assumption-based reasoning and probabilistic argumentation systems. Technical Report 96–07, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1996.

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

[36] J. Kohlas, S. Moral, and R. Haenni. Propositional information systems. *Journal of Logic and Computation*, 1999.

[37] J. Kohlas and R. Stärk. Eine mathematische Theorie von Informationssystemen. Technical Report 96–12, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1996.

[38] J. Kohlas and R. Stärk. Information algebras and information systems. Technical Report 96–14, University of Fribourg, Institute of Informatics, Theoretical Computer Science, 1996.

[39] A. Kong. *Multivariate Belief Functions and Graphical Models.* PhD thesis, Department of Statistics, Harvard University, 1986.

[40] K.B. Laskey and P.E. Lehner. Assumptions, beliefs and probabilities. *Artificial Intelligence*, 41:65–77, 1989.

[41] S.L. Lauritzen and P.P. Shenoy. Computing marginals using local computation. In J. Kohlas and S. Moral, editors, *Algorithms for Uncertainty and Defeasible Reasoning.* Kluwer Academic Publishers, 1999.

[42] S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistical Society*, 50(2):157–224, 1988.

[43] J. Lu. Logic programming with signs and annotations. *Journal of Logic and Computation*, 6(6):755–778, 1996.

[44] J. Lu, N. Murray, and E. Rosenthal. A framework for automated reasoning in multiple-valued logics. Technical Report 94-04, State University of New York at Albany, 1994.

[45] P. Marquis. Consequence finding algorithms. In J. Kohlas and S. Moral, editors, *Algorithms for Uncertainty and Defeasible Reasoning.* Kluwer Academic Publishers, 1999.

[46] J. D. Monk. *Cardinal Functions on Boolean Algebras.* Lectures in Mathematics. Birkhäuser, 1990.

[47] P.A. Monney and B. Anrig. Computing the probability of formulas representing events in product spaces. In *IPMU'98, Proceedings of the seventh international conference, Paris, France*, pages 1724–1731, 1998.

[48] N.V. Murray and E. Rosenthal. Adapting classical inference techniques to multiple-valued logics using signed formulas. *Fundamenta Informaticae*, 21(3):237–253, 1994.

[49] G.M. Provan. A logic-based analysis of Dempster-Shafer theory. *International Journal of Approximate Reasoning*, 4:451–495, 1990.

[50] D.J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32:597–609, 1970.

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

[52] P.P. Shenoy and G. Shafer. Axioms for probability and belief functions propagation. In R.D. Shachter and al., editors, *Uncertainty in Artificial Intelligence 4.* North Holland, 1990.

[53] P. Siegel. *Représentation et Utilisation de la Connaissance en Calcul Propositionel.* PhD thesis, Université d'Aix-Marseille II. Luminy, France, 1987.

[54] R. Sikorski. *Boolean Algebras.* Springer, Berlin-Göttingen-Heidelberg, 1960.

[55] Ph. Smets. The transferable belief model for quantified belief representation. In D.M. Gabbay and Ph. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 1, pages 267–301. Kluwer Academic Publishers, 1998.

[56] R.E. Tarjan and M. Yannakakis. Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal of Computing*, 13:566–579, 1984.

[57] P. Tison. Generalization of consensus theory and application to the minimization of Boolean functions. *IEEE Transaction on Electronic Computers*, EC-16:446–456, 1967.

[58] N. Wilson. Algorithms for Dempster-Shafer theory. In J. Kohlas and S. Moral, editors, *Algorithms for Uncertainty and Defeasible Reasoning.* Kluwer Academic Publishers, 1999.

[59] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. of Algebraic and Discrete Methods*, 2:77–79, 1981.

Index