

Modeling Uncertainty with Propositional Assumption-Based Systems*

Rolf Haenni

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

Phone: +41 (0)26 300 83 31
Fax: +41 (0)26 300 97 26
E-Mail: rolf.haenni@unifr.ch
WWW: www-iiuf.unifr.ch/tcs

Abstract. This paper proposes assumption-based systems as an efficient and convenient way to encode uncertain information. Assumption-based systems are obtained from propositional logic by including a special type of propositional symbol called assumption. Assumptions are needed to express the uncertainty of the given information. Assumption-based systems can be used to judge hypotheses qualitatively or quantitatively. This paper shows how assumption-based systems are obtained from causal networks, it describes how symbolic arguments for hypotheses can be computed efficiently, and it presents ABEL, a modeling language for assumption-based systems and an interactive tool for probabilistic assumption-based reasoning.

1 Introduction

Propositional logic is an efficient and convenient way to encode knowledge or information. In particular, uncertainty can be incorporated into propositional knowledge by including assumptions. Judging a hypothesis in the light of the given assumption-based knowledge means finding arguments (i.e. a sufficient number of assumptions) which allow a prove of the hypothesis. Assumption-based reasoning is therefore the process of computing arguments for which a given hypothesis can be deduced from the available knowledge.

The computation of arguments for certain hypotheses is essentially a task that involves deduction and theorem proving. A variety of established methods and procedures exist for deduction in propositional logic. In particular, the concept of **assumption-based truth maintenance systems** (ATMS) provides the basic elements for assumption-based reasoning (de Kleer, 1986; Reiter & de Kleer, 1987). A traditional ATMS is restricted to Horn clauses (clauses with at most one non-negated atom) and it accepts only single literals as queries. The

* Research supported by grant No.2100-042927.95 of the Swiss National Foundation for Research.

fundamental ATMS problem is to identify the contexts of assumptions in which queries hold. R. Reiter proposed the concept of a **clause management systems** (CMS) – a generalization of de Kleer’s original ATMS – which is no longer restricted to Horn clauses (Reiter & de Kleer, 1987). Other interesting works about truth maintenance systems, clause management systems, and probabilistic assumption-based reasoning are (Laskey & Lehner, 1989), (Provan, 1990), (Inoue, 1991), (Siegel, 1987), and (Kohlas & Monney, 1993).

This paper uses the notion of **assumption-based systems** (ABS) (Kohlas & Monney, 1993; Haenni, 1996), which turns out to be a special case of the general evidence theory (Kohlas, 1995; Haenni, 1996). Section 2 starts with an intuitive prescription of transforming uncertain causal relations into assumption-based propositional logic. Causal networks are often used to represent patterns of influence among variables. Representing knowledge by causal relations covers therefore a number of different applications in the domain of uncertain reasoning. Then, Section 3 introduces assumption-based systems formally. It describes how arguments for hypotheses can be computed efficiently. Finally, Section 4 presents ABEL, an interactive modeling tool for probabilistic assumption-based reasoning.

2 Modeling Uncertain Causal Relations

Causal or inference networks are used in a number of areas to represent patterns of influence among variables. They consist of connected causal relations. A causal relation can be regarded as a rule of the form “if *cause*, then *effect*”. Examples of causal relations are: “if there is some rain tonight, then the grass will be wet tomorrow”, “if the next bingo-number is 7, then my wife wins 100 dollars”, or “if my father returns late at night, then my mother gets angry”. Thus, causality can be seen as any natural ordering in which knowledge of an event influences the opinion concerning another event. This influence can be logical, physical, temporal, or simply conceptual (Lauritzen & Spiegelhalter, 1988).

2.1 From Causal Networks to Propositional Logic

Causes and effects can be modeled as variables with finite domains. These variables are the nodes of the network. Causal relations between two variables are the edges. Figure 2.1 depicts a causal network consisting of only two nodes C and E . Usually, nodes are represented as circles and causal relations between them as arrows (Kohlas & Monney, 1995).



Fig. 2.1. Causal relation between two variables c and e .

The concept of directed acyclic graphs is an appropriate mathematical structure to describe causal networks. A pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is called **directed graph** if \mathcal{N} is a set of nodes and $E \in \mathcal{E}$ are **directed edges**, i.e. pairs (n_i, n_j) of nodes, $n_i, n_j \in \mathcal{N}$. Directed edges (n_i, n_j) are depicted as arrows from n_i to n_j .

In a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ a **directed chain** from n_1 to n_{q+1} is a sequence of edges E_1, \dots, E_q such that $E_k = (n_k, n_{k+1})$ for $k = 1, \dots, q$. n_1 is called the **initial endpoint** and n_{q+1} the **terminal endpoint** of the directed chain. A **directed cycle** is a directed chain in which no edge appears twice in the sequence of edges and in which the two endpoints are the same. If a directed graph \mathcal{G} has no cycles, then it is called **directed acyclic graph**. Figure 2.2 shows two directed graphs. The one on the left is acyclic, the one on the right is not.

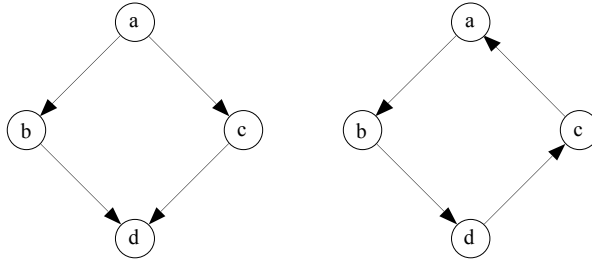


Fig. 2.2. Two directed graphs.

Generally, a variable of a causal network can have any domain. In the following, only problems with binary variables are considered, i.e. variables with possible values 1 (true) and 0 (false). If the cause c of a causal relation is true, then the effect e of the relation is also true. This can be expressed as a logical implication $c \rightarrow e$. Binary variables of a causal network are therefore represented by propositional symbols.

Consider the directed acyclic graph on the left side of Figure 2.2. If it is interpreted as a causal network, then a , for example, is cause for b and c , whereas b and c are causes of d . Such multiple causes or multiple effects can be interpreted as conjunctions or as disjunctions. Thus, four different cases are possible:

- (1) If cause c is true, then all effects e_1 to e_n are also true:

$$c \rightarrow e_1, c \rightarrow e_2, \dots, c \rightarrow e_n. \quad (2.1)$$

- (2) If cause c is true, then at least one effect e_1 to e_n is also true:

$$c \rightarrow e_1 \vee e_2 \vee \dots \vee e_n. \quad (2.2)$$

- (3) If at least one of the causes c_1 to c_n is true, then effect e is also true; besides c_1 to c_n there are no other causes of e :

$$\begin{aligned} c_1 \rightarrow e, c_2 \rightarrow e, \dots, c_n \rightarrow e, \\ e \rightarrow c_1 \vee c_2 \vee \dots \vee c_n. \end{aligned} \quad (2.3)$$

- (4) If all causes c_1 to c_n are true, then effect e is also true; this is the only way to cause e :

$$\begin{aligned} c_1 \wedge c_2 \wedge \dots \wedge c_n \rightarrow e, \\ e \rightarrow c_1, e \rightarrow c_2, \dots, e \rightarrow c_n. \end{aligned} \quad (2.4)$$

According to Figure 2.3, different graphical representations are used to distinguish these cases.

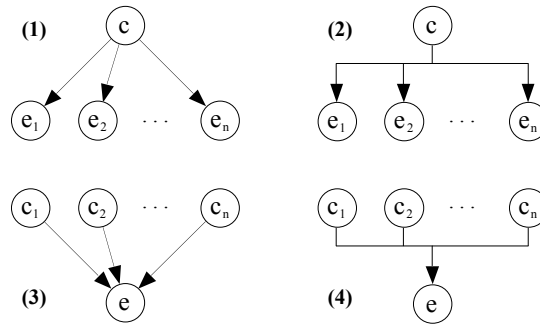


Fig. 2.3. Multiple causes and multiple effects.

In (3) and in (4) it is assumed that besides c_1 to c_n there are no other causes for e . This is called **accountability condition** (Pearl, 1988). It states that the model explicitly contains all causes that may produce e . Logically, this can be expressed through implications from e to c_i . Such implications pointing to the opposite direction are important especially for diagnostics analyses.

Some or all of the causal relations of a causal network may be uncertain. In such cases, effects are only caused under some circumstances. This type of uncertainty can be expressed by **assumptions**. A rule like “if cause c is true, then effect e is true under some circumstances a ” can be transformed into $c \wedge a \rightarrow e$. a represents the assumption that the necessary circumstances are present. If all causal relations of Figure 2.3 are assumed to be uncertain, then the following logical implications are produced:

- (1') If cause c is true, then – under some circumstances a_i – effect e_i is also true:

$$c \wedge a_1 \rightarrow e_1, c \wedge a_2 \rightarrow e_2, \dots, c \wedge a_n \rightarrow e_n. \quad (2.5)$$

- (2') If cause c is true, then – under some circumstances a – at least one effect e_1 to e_n is also true:

$$c \wedge a \rightarrow e_1 \vee e_2 \vee \dots \vee e_n. \quad (2.6)$$

- (3') If cause c_i is true, then – under some circumstances a_i – effect e is also true; besides c_1 to c_n there are no other causes of e :

$$\begin{aligned} c_1 \wedge a_1 \rightarrow e, c_2 \wedge a_2 \rightarrow e, \dots, c_n \wedge a_n \rightarrow e, \\ e \rightarrow (c_1 \wedge a_1) \vee (c_2 \wedge a_2) \vee \dots \vee (c_n \wedge a_n). \end{aligned} \quad (2.7)$$

- (4') If all causes c_1 to c_n are true, then – under some circumstances a – effect e is also true; this is the only way to cause e :

$$\begin{aligned} c_1 \wedge c_2 \wedge \dots \wedge c_n \wedge a \rightarrow e, \\ e \rightarrow c_1, e \rightarrow c_2, \dots, e \rightarrow c_n, e \rightarrow a. \end{aligned} \quad (2.8)$$

If in case (3') other (unknown) causes of e are possible, then an additional assumption a_{n+1} has to be introduced. a_{n+1} describes the uncertainty whether unknown causes of e are present.

- (3'') If cause c_i is true, then – under some circumstances a_i – effect e is also true; besides c_1 to c_n other causes of e are possible:

$$\begin{aligned} c_1 \wedge a_1 \rightarrow e, c_2 \wedge a_2 \rightarrow e, \dots, c_n \wedge a_n \rightarrow e, a_{n+1} \rightarrow e \\ e \rightarrow (c_1 \wedge a_1) \vee (c_2 \wedge a_2) \vee \dots \vee (c_n \wedge a_n) \vee a_{n+1}. \end{aligned} \quad (2.9)$$

According to this intuitive prescription, all relations of a causal network can be transformed into a set of material implications. This system of logical formulas can be used to judge hypotheses about variables of the causal network. Arguments for such hypotheses are expressions consisting of assumptions about the uncertain circumstances of the causal relations. They can be used to make **predictions** of effects or to give **explanations** (diagnoses) of observed effects.

2.2 Example 1: Chest Clinic

A doctor has to decide whether a patient, who complains about shortness-of-breath, suffers from bronchitis, lung cancer, or tuberculosis. This fictitious example was first mentioned in (Lauritzen & Spiegelhalter, 1988) in order to illustrate the use of Bayesian networks. Here, the same example is applied to the field of assumption-based reasoning.

The doctor's medical knowledge helps him to find the causes of the patient's symptoms. His knowledge is composed of what he learned at medical school and of what he knows from his practical experience. It can be summarized as follows (Lauritzen & Spiegelhalter, 1988):

“Shortness-of-breath (dyspnoea) may be due to tuberculosis, lung cancer, or bronchitis, or none of them, or more than one of them. A recent visit to an under-developed country increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and tuberculosis. The results of a single chest X-ray do not discriminate between lung cancer and tuberculosis; nor does the presence or absence of dyspnoea.”

In Figure 2.4 this small piece of fictitious medical knowledge is shown as causal network. Direction of causality is from top to bottom. Note that some effects have more than one cause and that some causes produce more than one effect. Thus, multiple causes are interpreted as disjunction, whereas multiple effects are interpreted as conjunction.

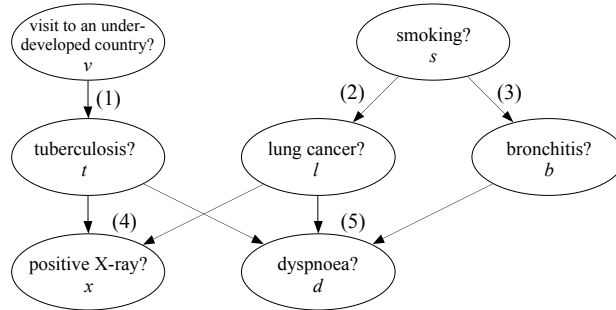


Fig. 2.4. Causal network for the fictitious example.

All causal relations of Figure 2.4 are uncertain. Furthermore, it is assumed that for all effects there are other (unknown) causes. Thus, the strategy described in (3'') has to be used in order to obtain the following set of logical implications:

$$(1) \quad v \wedge a_1 \rightarrow t, \quad a_2 \rightarrow t, \quad t \rightarrow (v \wedge a_1) \vee a_2; \quad (2.10)$$

$$(2) \quad s \wedge a_3 \rightarrow l, \quad a_4 \rightarrow l, \quad l \rightarrow (s \wedge a_3) \vee a_4; \quad (2.11)$$

$$(3) \quad s \wedge a_5 \rightarrow b, \quad a_6 \rightarrow b, \quad b \rightarrow (s \wedge a_5) \vee a_6; \quad (2.12)$$

$$(4) \quad t \wedge a_7 \rightarrow x, \quad l \wedge a_8 \rightarrow x, \quad a_9 \rightarrow x, \\ x \rightarrow (t \wedge a_7) \vee (l \wedge a_8) \vee a_9; \quad (2.13)$$

$$(5) \quad t \wedge a_{10} \rightarrow d, \quad l \wedge a_{11} \rightarrow d, \quad b \wedge a_{12} \rightarrow d, \quad a_{13} \rightarrow d, \\ d \rightarrow (t \wedge a_{10}) \vee (l \wedge a_{11}) \vee (b \wedge a_{12}) \vee a_{13}. \quad (2.14)$$

2.3 Example 2: Burglary

The example considered here is a small story around the alarm system of Mr. Holmes' house (Pearl, 1988):

“A burglary in Mr. Holmes’ house generates an alarm if the alarm system is functioning. But the alarm may also be caused by an earthquake or by other (unspecified) reasons. The neighbors of Mr. Holmes, Dr. Watson and Mrs. Gibbons, phone Mr. Holmes in the case of an alarm. Possibly, Dr. Watson may also phone Mr. Holmes as a joke. Mrs. Gibbons is hard of hearing, and she may possibly not be able to hear the alarm. Furthermore, if Mr. Holmes’ daughter is at home, then she surely will phone too in the case of an alarm. Finally, if there is an earthquake and if the earthquake is registered, then there is a confirmation of it on the radio.”

In Figure 2.5 the above story is shown as a causal network.

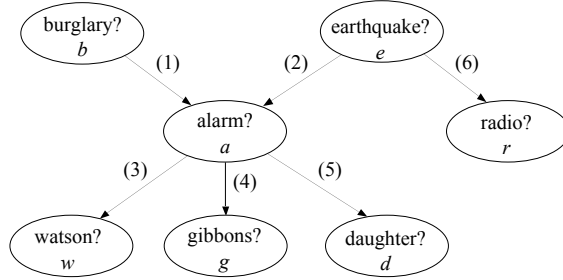


Fig. 2.5. Causal network for the burglary example.

Using the strategy described in (3'') leads to the following set of logical implications:

$$(1) \quad b \wedge a_1 \rightarrow a, \quad e \wedge a_2 \rightarrow a, \quad a_3 \rightarrow a, \\ a \rightarrow (b \wedge a_1) \vee (e \wedge a_2) \vee a_3; \quad (2.15)$$

$$(2) \quad a \wedge a_4 \rightarrow w, \quad a_5 \rightarrow w, \quad w \rightarrow (a \wedge a_4) \vee a_5; \quad (2.16)$$

$$(3) \quad a \wedge a_6 \rightarrow g, \quad a_7 \rightarrow g, \quad g \rightarrow (a \wedge a_6) \vee a_7; \quad (2.17)$$

$$(4) \quad a \wedge a_8 \rightarrow d, \quad a_9 \rightarrow d, \quad d \rightarrow (a \wedge a_8) \vee a_9; \quad (2.18)$$

$$(5) \quad e \wedge a_{10} \rightarrow r, \quad a_{11} \rightarrow r, \quad r \rightarrow (e \wedge a_{10}) \vee a_{11}. \quad (2.19)$$

3 Assumption-Based Systems

This section introduces the notion of assumption-based systems formally. Basically, an assumption-based system is a set of propositional formulae, where a subset of the propositions is declared as assumptions. Therefore, this technique profits from the expressiveness of propositional logic and the advantages of existing deduction techniques based on resolution. Assumption-based systems can be used to treat queries about the given knowledge, i.e. to find arguments in favor or against certain hypotheses. The examples of the previous section show how assumption-based systems are obtained from causal networks.

3.1 Representing Uncertainty by Propositional Logic

Propositional logic with uncertain assumptions provides a concise language for describing uncertain information. Let $P = \{p_1, \dots, p_q\}$ and $A = \{a_1, \dots, a_r\}$ be two disjoint sets of propositional symbols. The elements of P are called **propositions** and the elements of A **assumptions**. $N = P \cup A$ is the entire set of symbols considered. Let $Q \subseteq N$ be a subset of N . The corresponding set of negated symbols is denoted by $\sim Q$. $Q^\pm = Q \cup \sim Q$ represents the set of all literals of Q . The corresponding set of all well-formed logical formulae in Q is denoted by \mathcal{L}_Q .

The set $\Sigma = \{\xi_1, \dots, \xi_s\}$ is a set of clauses on the literals N^\pm . Note that a more general framework in which Σ consists of arbitrary logical formulas can always be transformed into an equivalent framework consisting only of clauses. Σ is called a **knowledge base** and it represents the available information. The conjunction $\xi = \xi_1 \wedge \dots \wedge \xi_s$ is a logical representation of Σ . If Σ is empty, then $\xi = \top$. A clause $\xi_i \in \Sigma$ is a statement of the available information. It restricts the possible truth values of the propositional symbols appearing in Σ . A triple $\mathcal{A} = (P, A, \Sigma)$, where P and A are disjoint sets of propositional symbols and Σ is a set of clauses on the literals in N^\pm , is called **assumption-based system** (ABS).

The assumptions involved in the clauses of Σ are necessary to express the uncertainty of the corresponding statements. The examples of Section 2 show how assumptions differ from “normal” propositions, and how they are used to represent uncertainty.

3.2 Hypotheses and Arguments in Assumption-Based Systems

Given an assumption-based system $\mathcal{A} = (P, A, \Sigma)$, one may be interested in certain hypotheses h about the knowledge contained in \mathcal{A} . Such hypotheses can be expressed as logical formulae in \mathcal{L}_N . What can be learned from Σ about the possible truth of h ? If some assumptions are considered to be either true or false, then h can possibly be deduced from Σ . Such a combination of true and false assumptions can be regarded as an **argument** in favor of h . Arguments are therefore conjunctions of literals of assumptions, or, more generally, logical formulae in \mathcal{L}_A . This is the link to the general evidence theory (Haenni, 1996): the possible hypotheses are logical formulae in \mathcal{L}_N , and the possible arguments are logical formulae in \mathcal{L}_A .

In view of these remarks consider a logical formula $a \in \mathcal{L}_A$, such that $a \wedge \xi \models h$, where ξ is the conjunction of the clauses contained in Σ , and $h \in \mathcal{L}_N$ represents a hypothesis. The formula a allows us to deduce h from Σ . It is called **supporting arguments** for h relative to Σ . If a is the coarsest (least precise) supporting argument of h , i.e. if there is no other supporting argument a' of h with $a \models a'$ and $a \neq a'$, then a is called **quasi-support** of h relative to Σ , denoted by $qs(h, \Sigma)$. It can be shown (Kohlas & Monney, 1995) that quasi-support satisfies the following conditions (“ \equiv ” means “logically equivalent”):

$$(Q1) \quad qs(\top, \Sigma) \equiv \top,$$

- (Q2) $qs(h_1 \wedge h_2, \Sigma) \equiv qs(h_1, \Sigma) \wedge qs(h_2, \Sigma)$,
- (Q3) if $h_1 \models h_2$, then $qs(h_1, \Sigma) \models qs(h_2, \Sigma)$,
- (Q4) $qs(h_1, \Sigma) \vee qs(h_2, \Sigma) \models qs(h_1 \vee h_2, \Sigma)$.

In many cases $qs(\perp, \Sigma)$ will be different from \perp . If a is a supporting argument for \perp , i.e. if $a \wedge \xi \models \perp$, then a is in contradiction to the knowledge base Σ and should be eliminated from quasi-support. The **support** of h relative to Σ is defined by

$$sp(h, \Sigma) \equiv qs(h, \Sigma) \wedge \sim qs(\perp, \Sigma), \quad (3.1)$$

and an argument a with $a \models sp(h, \Sigma)$ and $a \not\models qs(\perp, \Sigma)$ is called **proper argument** of h . Support satisfies the following properties (S1) to (S4):

- (S1) $sp(\top, \Sigma) \equiv \sim qs(\perp, \Sigma)$,
- (S1') $sp(\perp, \Sigma) \equiv \perp$,
- (S2) $sp(h_1 \wedge h_2, \Sigma) \equiv sp(h_1, \Sigma) \wedge sp(h_2, \Sigma)$,
- (S3) if $h_1 \models h_2$, then $sp(h_1, \Sigma) \models sp(h_2, \Sigma)$,
- (S4) $sp(h_1, \Sigma) \vee sp(h_2, \Sigma) \models sp(h_1 \vee h_2, \Sigma)$.

It may also be interesting to consider arguments against a hypothesis $h \in \mathcal{L}_N$, in other words, arguments in favor of $\sim h$. A formula $a \in \mathcal{L}_A$ is called a **refuting argument** of h if $a \wedge \xi \models \sim h$. If a is a refuting argument of h , and there is no other refuting argument a' of h with $a' \models a$ and $a' \neq a$, then a is called the **doubt** in h relative to Σ , denoted by $db(h, \Sigma)$. Doubt is obtained from quasi-support and vice versa:

$$db(h, \Sigma) \equiv qs(\sim h, \Sigma), \quad (3.2)$$

$$qs(h, \Sigma) \equiv db(\sim h, \Sigma). \quad (3.3)$$

From (Q1) to (Q4) it follows that the following conditions (D1) are valid for doubt:

- (D1) $db(\perp, \Sigma) \equiv \top$,
- (D2) $db(h_1 \vee h_2, \Sigma) \equiv db(h_1, \Sigma) \wedge db(h_2, \Sigma)$,
- (D3) if $h_1 \models h_2$, then $db(h_2, \Sigma) \models db(h_1, \Sigma)$,
- (D4) $db(h_1, \Sigma) \vee db(h_2, \Sigma) \models db(h_1 \wedge h_2, \Sigma)$.

A formula $a \in \mathcal{L}_A$ is called a **possible argument** of h if it is not a refuting argument of h , i.e. if $a \wedge \xi \not\models \sim h$. If a is a possible argument of h , and if there is no other possible argument a' of h with $a \models a'$ and $a' \neq a$, then a is called the **plausibility** of h relative to Σ , denoted by $pl(h, \Sigma)$. It corresponds to the negated quasi-support of the negated hypothesis,

$$pl(h, \Sigma) \equiv \sim qs(\sim h, \Sigma), \quad (3.4)$$

$$qs(h, \Sigma) \equiv \sim pl(\sim h, \Sigma), \quad (3.5)$$

and it satisfies the conditions (P1) to (P4):

- (P1) $pl(\perp, \Sigma) \equiv \perp$,

- (P2) $pl(h_1 \vee h_2, \Sigma) \equiv pl(h_1, \Sigma) \vee pl(h_2, \Sigma)$,
(P3) if $h_1 \models h_2$, then $pl(h_1, \Sigma) \models pl(h_2, \Sigma)$,
(P4) $pl(h_1 \wedge h_2, \Sigma) \models pl(h_1, \Sigma) \wedge pl(h_2, \Sigma)$.

If a formula $a \in \mathcal{L}_A$ is either a supporting argument, a proper argument, a refuting argument, or a possible argument of a hypotheses h , then it is called a **symbolic argument** of h . \mathcal{L}_A is therefore the set of all possible symbolic arguments.

If the symbolic arguments are computed for a given hypothesis, then it is also possible to transform this qualitative measure into a corresponding quantitative measure by assigning prior probabilities to the assumptions (Kohlas & Monney, 1994). Then, instead of support, doubt, and plausibility we speak of **degree of support**, **degree of doubt**, and **degree of plausibility**. Note that degree of support correspond to the notion of **belief** in Dempster-Shafer's theory of evidence (Dempster, 1967; Shafer, 1976). The technique used for this computation is not discussed in this paper, see (Kohlas & Monney, 1994; Kohlas, 1994; Bertschy & Monney, 1996). Examples of so-called **numerical arguments** are given in Subsections 4.1 & 4.2.

To illustrate the idea of symbolic arguments, consider three implications $a_1 \rightarrow p$, $a_2 \rightarrow q$, and $p \rightarrow \sim q$. Let a_1, a_2 be assumptions and p, q propositions, i.e. $A = \{a_1, a_2\}$ and $P = \{p, q\}$. The knowledge base is therefore given by $\Sigma = \{\sim a_1 \vee p, \sim a_2 \vee q, \sim p \vee \sim q\}$. Let q be the hypothesis to be judged. For this situation we get the following symbolic arguments:

$$qs(q, \Sigma) = a_2, \quad (3.6)$$

$$qs(\perp, \Sigma) = a_1 \wedge a_2, \quad (3.7)$$

$$sp(q, \Sigma) \equiv qs(q, \Sigma) \wedge \sim qs(\perp, \Sigma) = a_2 \wedge \sim a_1, \quad (3.8)$$

$$db(q, \Sigma) \equiv qs(\sim q, \Sigma) = a_1, \quad (3.9)$$

$$pl(q, \Sigma) \equiv \sim db(q, \Sigma) = \sim a_1. \quad (3.10)$$

3.3 Representing and Computing Symbolic Arguments

The previous subsections introduced an evidence theory based on propositional logic. It assigns arguments (represented by formulae in \mathcal{L}_A) to hypotheses (represented by formulae in \mathcal{L}_N) in the sense of the general evidence theory (Kohlas, 1995). Now, there are at least two important questions to be answered:

- (1) How can symbolic arguments be represented or stored efficiently?
- (2) How are arguments computed from the knowledge base and the given hypothesis?

These questions are discussed in the following two subsections.

Representing Symbolic Arguments. How can symbolic arguments be represented efficiently? Note that in all cases (quasi-support, support, etc.) symbolic

arguments are ordinary logical formulae $a \in \mathcal{L}_A$. Thus, the question to be discussed here can be answered more generally by considering how logical formulae (i.e. their corresponding equivalence classes) can be represented efficiently.

Let f be a logical formula in \mathcal{L} . To represent f efficiently, a semantically equivalent formula $f' \equiv f$ is needed such that the number of propositions or logical operators contained in f' is as small as possible. To simplify the problem, f' can be restricted to consist only of conjunctions \wedge , disjunctions \vee , and negations \sim . Furthermore, in order to obtain conformity, f' is assumed to be either in disjunctive normal form (DNF) or in conjunctive normal form (CNF). Thus, the problem is to transform an arbitrary logical formula f into an equivalent DNF or CNF f' such that the number of propositional symbols or the number of terms in f' is small or minimal. The disjunction $\psi(f)$ of all non-trivial prime implicants of f or the conjunction $\varphi(f)$ of all non-trivial prime implicates of f are usually good solutions (Haenni, 1996). The minimal or the shortest DNF or CNF is obtained from $\psi(f)$ or $\varphi(f)$ by eliminating different combinations of redundant terms. This is often difficult and expensive, and in most cases it is not worthwhile. Thus, either $\psi(f)$ or $\varphi(f)$ is selected to represent a formula f . An algorithm for computing prime implicants and prime implicates is described in (Haenni, 1996).

In the case of f being a symbolic argument, i.e. $f \in \mathcal{L}_A$, the disjunction $\psi(f)$ of all non-trivial prime implicants is the more interesting representation than the conjunction $\varphi(f)$. If for example f is the quasi-support of h , then the conjunctions contained in $\psi(f)$, i.e. the prime implicants $\Psi(f)$, are always supporting arguments of h . Thus, it is often more convenient to treat a symbolic argument f as a set, namely the set $\Psi(f)$ of non-trivial prime implicants of f . From this point of view, a symbolic argument is a set of conjunctions of literals of different assumptions.

Let C_A denote the set of conjunctions of zero, one, or more literals of different assumptions in A^\pm . The elements of C_A represent equivalence classes of arbitrary conjunctions of literal of assumptions. The “empty” conjunction, i.e. a conjunction with zero literals, represents the tautology \top . C_A is a partially ordered set with the logical entailment \models as its ordering relation. If A contains r elements, then C_A contains 3^r different conjunctions. Figure 3.1 shows the partially ordered set C_A for $A = \{a_1, a_2\}$.

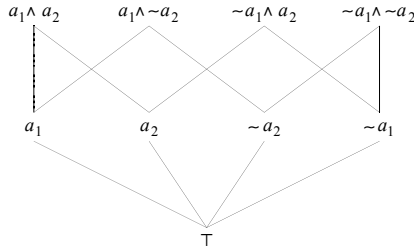


Fig. 3.1. Example of a partially ordered set.

It proposes a method that generates the quasi-support of a hypothesis $h \in \sim C_N$, in which $\sim C_N$ denotes the set of possible clauses over N^\pm , $N = P \cup A$. This method is based on a theorem stated in (Reiter & de Kleer, 1987). The theorem assumes that the set $\Phi(\xi)$ of prime implicates of ξ is known. ξ is the logical form of Σ and instead of $\Phi(\xi)$ one can also write $\Phi(\Sigma)$.

Reiter's and de Kleer's theorem uses a set-oriented view for the clauses contained in $\Phi(\Sigma)$ and for the conjunctions involved in the symbolic arguments. Thus, consider the following notation: if f and g are two clauses, then $f - g$ denotes the subclause of f , which is obtained when all literals present both in f and g are eliminated from f .

Theorem 1. (Reiter & de Kleer, 1987) *Let $\Phi(\Sigma)$ be the set of prime implicates of a given knowledge base Σ . If h is a clause in $\sim C_N$, then the prime implicants of the quasi-support $\Psi(qs(h, \Sigma))$ are given by*

$$\Psi(qs(h, \Sigma)) = \mu\{\sim(f - h) \in C_A : f \in \Phi(\Sigma)\}. \quad (3.11)$$

According to this theorem, one way to determine quasi-supports for clauses is to compute the prime implicates of Σ and then to filter the prime implicants of the quasi-support out of them.

If the hypothesis h is not restricted to be a clause, then property (Q2) helps to determine the quasi-support. First, h has to be transformed into a conjunctive normal form $h = h_1 \wedge \dots \wedge h_k$, in which h_i , $i = 1, \dots, k$, are all clauses. Then $qs(h_i, \Sigma)$, $i = 1, \dots, k$, for all these clauses can be obtained through (3.11), and from (Q2) follows that the quasi-support of h is simply the conjunction

$$qs(h, \Sigma) \equiv qs(h_1, \Sigma) \wedge \dots \wedge qs(h_k, \Sigma). \quad (3.12)$$

Although this method generates the quasi-support of a hypothesis h , it can also be used to generate support, doubt, and plausibility. They are obtained indirectly from quasi-support and equations 3.1 to 3.4.

The method described in this subsection to generate symbolic arguments like quasi-support is only appropriate for a knowledge base Σ with a relatively small number of prime implicates. Generally, the time needed to compute prime implicates grows exponentially with the size of Σ . Thus, the computation of the prime implicates for a large knowledge base will be too expensive or even impossible. Nevertheless, it is often possible to decompose a large set Σ into small sets $\Sigma_1, \dots, \Sigma_r$ with $\Sigma = \Sigma_1 \cup \dots \cup \Sigma_r$. Such a decomposition puts the problem of computing symbolic arguments into the framework of valuation networks (Shenoy & Shafer, 1990). Instead of working on a large knowledge base, valuation networks allow local computations on smaller sets. This idea is further developed in the following subsection.

3.4 Assumption-Based Systems in Valuation Networks

An assumption-based system $\mathcal{A} = (P, A, \Sigma)$ represents some information or knowledge relative to the statements expressed by the propositional symbols

contained in P . Sometimes it will be useful or necessary to marginalize (focus) the assumption-based system \mathcal{A} to a smaller set $P' \subseteq P$ of propositional symbols. If two or more assumption-based systems are given, then it will be important to combine them into a new combined assumption-based system as well. This subsection discusses two operations for assumption-based systems called **marginalization**, and **combination**. This puts assumption-based systems into the framework of **valuation networks** (Shenoy & Shafer, 1990) where operations of marginalization and combination are important. If a decomposition $\Sigma_1, \dots, \Sigma_r$ is given, then every sub-knowledge Σ_i (i.e. its corresponding assumption-based system \mathcal{A}_i) forms a **valuation** in the sense of Shenoy's idea of valuation networks. It is easy to show that assumption-based systems satisfy the necessary axioms that allow local computation in valuation networks (Haenni, 1996). Therefore, the idea how to compute symbolic arguments efficiently is the following:

- (1) Find a decomposition $\Sigma_1, \dots, \Sigma_r$ and construct a corresponding valuation network.
- (2) Propagate the valuations locally through the network (inward and outward phase).
- (3) Compute symbolic arguments from the resulting marginals using the methods of Subsection 3.3.

Marginalization. Let $\mathcal{A} = (P, A, \Sigma)$ be an assumption-based system. The marginalization of the information contained in \mathcal{A} to a subset $P' \subseteq P$ can be defined in terms of the given knowledge base Σ . This leads to a new assumption-based system $\mathcal{A}' = (P', A, \Sigma')$. Σ' is derived from Σ by eliminating sequentially all propositional symbols not belonging to P' (Kohlas & Moral, 1995).

Let $P^* = P - P' = \{p_1, \dots, p_k\}$ be the set of proposition to be eliminated from Σ . Generally, the elimination of a proposition $p \in P^*$ from Σ involves the computations of all possible resolvents for p . If ξ_i is a clause containing the positive literal p and ξ_j a clause containing the negative clause $\sim p$, then the **resolvent** $\rho(\xi_i, \xi_j)$ can be formed by eliminating p , $\sim p$ and all multiple occurrences of literals from $\xi \vee \xi_j$. $\rho(\xi_i, \xi_j)$ is set to \top if ξ_i and ξ_j contain other pairs of negated literals besides p and $\sim p$.

To describe the elimination of a proposition $p \in P^*$ from Σ , three sets Σ_p , $\Sigma_{\sim p}$, and Σ_{-p} , all subsets of Σ , are defined as follows:

$$\Sigma_p = \{\xi_i \in \Sigma : p \in \xi_i\}, \quad (3.13)$$

$$\Sigma_{\sim p} = \{\xi_i \in \Sigma : \sim p \in \xi_i\}, \quad (3.14)$$

$$\Sigma_{-p} = \{\xi_i \in \Sigma : p, \sim p \notin \xi_i\} = \Sigma - (\Sigma_p \cup \Sigma_{\sim p}). \quad (3.15)$$

Now, the new set $\Sigma_{-\{p\}}$ obtained after eliminating p is:

$$\Sigma'_{-\{p\}} = \mu(\Sigma_{-p} \cup \{\rho(\xi_i, \xi_j) : \xi_i \in \Sigma_p, \xi_j \in \Sigma_{\sim p}\}). \quad (3.16)$$

The marginalization $\Sigma' = \Sigma_{-P^*}$ is obtained from Σ by eliminating the propositions $p_i \in P^*$, $i = 1, \dots, k$, in any order. The new assumption-based system

$\mathcal{A}' = (P', A, \Sigma')$ is called **marginal** of \mathcal{A} to P' , denoted by $\mathcal{A}^{\downarrow P'}$. It satisfies the property of transitivity (Shenoy & Shafer, 1990), i.e. for $P'' \subseteq P' \subseteq P$

$$(\mathcal{A}^{\downarrow P'})^{\downarrow P''} \equiv \mathcal{A}^{\downarrow P''}. \quad (3.17)$$

Combination. Up to now, an assumption-based system $\mathcal{A} = (P, A, \Sigma)$ was fixed. Now, suppose that there are two (or more) assumption-based systems $\mathcal{A}_1 = (P_1, A_1, \Sigma_1)$ and $\mathcal{A}_2 = (P_2, A_2, \Sigma_2)$, $P_1 \cap P_2 = \emptyset$, $P_2 \cap A_1 = \emptyset$, which may be combined into a new system $\mathcal{A} = \mathcal{A}_1 \otimes \mathcal{A}_2 = (P, A, \Sigma)$. The symbol \otimes denotes the combination. The new system is clearly determined by $P = P_1 \cup P_2$, $A = A_1 \cup A_2$ and $\Sigma = \mu(\Sigma_1 \cup \Sigma_2)$. Thus, to combine two assumption-based systems essentially means to form the union between Σ_1 and Σ_2 and to eliminate all subsuming clauses.

The combination of assumption-based systems is idempotent (i.e. $\mathcal{A} \otimes \mathcal{A} = \mathcal{A}$), commutative (i.e. $\mathcal{A}_1 \otimes \mathcal{A}_2 = \mathcal{A}_2 \otimes \mathcal{A}_1$), and associative (i.e. $(\mathcal{A}_1 \otimes \mathcal{A}_2) \otimes \mathcal{A}_3 = \mathcal{A}_1 \otimes (\mathcal{A}_2 \otimes \mathcal{A}_3)$). Another important property is the distributivity of marginalization over combination (Shenoy & Shafer, 1990):

$$(\mathcal{A}_1 \otimes \mathcal{A}_2)^{\downarrow P_1} \equiv \mathcal{A}_1 \otimes (\mathcal{A}_2)^{\downarrow P_1 \cap P_2}. \quad (3.18)$$

3.5 Assumptions as Hypotheses

Generally, the hypotheses to judge are formulae in \mathcal{L}_N , $N = P \cup A$, i.e. they can contain both propositions and assumptions. Now, the special case will be studied in which the hypothesis h consists only of assumptions. This simplifies some computations. Let h_1 and h_2 be two formulae in \mathcal{L}_A , i.e. expressions composed of assumptions. In this case the following properties for quasi-support, support, doubt, and plausibility are satisfied (Kohlas *et al.*, 1996):

$$\begin{aligned} (Q4') \quad qs(h_1 \vee h_2, \Sigma) &\equiv qs(h_1, \Sigma) \vee qs(h_2, \Sigma), \\ (S4') \quad sp(h_1 \vee h_2, \Sigma) &\equiv sp(h_1, \Sigma) \vee sp(h_2, \Sigma), \\ (P4') \quad pl(h_1 \wedge h_2, \Sigma) &\equiv pl(h_1, \Sigma) \wedge pl(h_2, \Sigma), \\ (D4') \quad db(h_1 \wedge h_2, \Sigma) &\equiv db(h_1, \Sigma) \vee db(h_2, \Sigma). \end{aligned}$$

These properties replace (Q4), (S4), (D4), and (P4) from Subsection 3.2. Note that any logical formula f is equivalent to $f \vee \perp$ and to $f \wedge \top$. This leads to the following important theorem:

Theorem 2. (Kohlas *et al.*, 1996) *If $\mathcal{A} = (P, A, \Sigma)$ is an assumption-based system and h_A a formula in \mathcal{L}_A , then the symbolic arguments are given by:*

$$qs(h_A, \Sigma) \equiv h_A \vee qs(\perp, \Sigma), \quad (3.19)$$

$$sp(h_A, \Sigma) \equiv h_A \wedge \sim qs(\perp, \Sigma), \quad (3.20)$$

$$db(h_A, \Sigma) \equiv h_A \wedge qs(\perp, \Sigma), \quad (3.21)$$

$$pl(h_A, \Sigma) \equiv h_A \wedge \sim qs(\perp, \Sigma). \quad (3.22)$$

Thus, the problem of finding the symbolic arguments for a hypothesis $h_A \in \mathcal{L}_A$ mainly consists in finding the quasi-support of the contradiction. In many applications (especially in diagnostics problems) the interesting question can be expressed by a formula in \mathcal{L}_A . In such cases it is enough to have efficient methods to determine $qs(\perp, \Sigma)$.

From Theorem 2 follows that for hypotheses $h_A \in \mathcal{L}_A$, support and plausibility are always the same:

$$sp(h_A, \Sigma) \equiv pl(h_A, \Sigma). \quad (3.23)$$

The results of Theorem 2 can also be applied in a more general case, where a hypothesis $h \in \mathcal{L}_N$ has to be judged. Often, h can be decomposed into two hypotheses $h_N \in \mathcal{L}_N$ and $h_A \in \mathcal{L}_A$, such that $h = h_N \wedge h_A$. Thus, quasi-support and support are obtained through:

$$qs(h, \Sigma) \equiv (h_A \wedge qs(h_N, \Sigma)) \vee qs(\perp, \Sigma), \quad (3.24)$$

$$sp(h, \Sigma) \equiv h_A \wedge qs(h_N, \Sigma) \wedge \sim qs(\perp, \Sigma). \quad (3.25)$$

Similarly, if the hypothesis $h \in \mathcal{L}_N$ can be decomposed into $h_N \in \mathcal{L}_N$ and $h_A \in \mathcal{L}_A$, such that $h = h_N \vee h_A$, then doubt and plausibility are obtained through:

$$db(h, \Sigma) \equiv (\sim h_A \wedge qs(\sim h_N, \Sigma)) \vee qs(\perp, \Sigma), \quad (3.26)$$

$$pl(h, \Sigma) \equiv h_A \wedge \sim qs(\sim h_N, \Sigma) \wedge \sim qs(\perp, \Sigma). \quad (3.27)$$

In all of these cases it is sufficient to use the methods presented in Subsection 3.3 in order to determine the quasi-support of h_N (or $\sim h_N$) and the quasi-support of the contradiction. If h_N is much simpler than the original h , then it is often less efficient to apply the methods directly on h . For example, if $h = a_1 \wedge p \vee a_2 \wedge p$ is the hypothesis to be judged, $a_1, a_2 \in A$, $p \in P$, then $h = (a_1 \vee a_2) \wedge p = h_A \wedge h_N$ with $h_A = a_1 \vee a_2$ and $h_N = p$. In this example it would be easier to determine $qs(p, \Sigma)$ and $qs(\perp, \Sigma)$ instead of $qs(a_1 \wedge p \vee a_2 \wedge p, \Sigma)$.

4 ABEL – a New Language for Assumption-Based Reasoning

This section describes ABEL (Anrig *et al.*, 1997a; Anrig *et al.*, 1997b) which is both, a new modeling language for assumption-based systems and an interactive tool for assumption-based reasoning¹. Given an assumption-based system and some additional facts or observations, the aim of ABEL is to compute symbolic and numerical arguments for the user's hypotheses (see Figure 4.1). ABEL was designed and implemented at the University of Fribourg for the Macintosh platform. It results from a research project led by the author of this paper. The program is written in CLOS (Common Lisp Object System), and it is therefore easily portable to different platforms. Examples of ABEL models can be found in the following two subsections and in (Anrig *et al.*, 1997a).

¹ The software can be obtained through the author's home page.

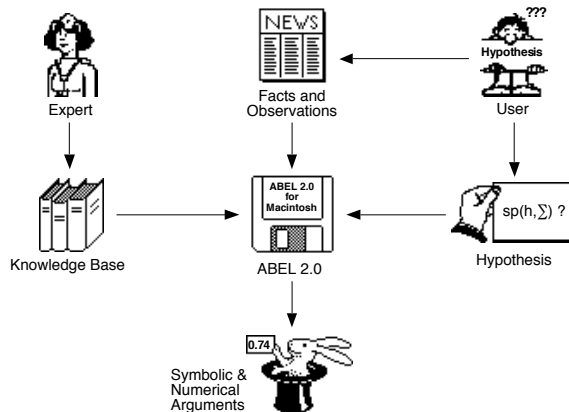


Fig. 4.1. Using ABEL to compute symbolic and numerical arguments.

The user interface of ABEL is a multiple-window-environment based on the Macintosh Common Lisp (MCL) programming interface. There are two different window types (Anrig *et al.*, 1997a):

- (1) **Modeling windows** are editors to model and develop assumption-based systems. Usual Macintosh editor features, such as text scrolling and mouse-based editing, are included. In addition, parenthesis matching and smart indentation are possible. The compilation of the ABEL model contained in a modeling window can be started by an appropriate command on the ABEL menu bar.
- (2) A special window called **transcript** is used for most interaction between the user and the system. As soon as the user types an ABEL command or a query in the transcript window, ABEL reads and interprets the expression, prints the result, and detects and reports possible errors.

Figure 4.2 shows a screenshot of the ABEL environment showing a modeling and a transcript window for the example discussed in Subsection 4.2.

ABEL is based on three other computer languages: (1) from **Common Lisp** (Steele, 1990) it adopts **prefix notation** and therewith a number of opening and closing parentheses; (2) from **Pulcinella** (Saffiotti & Umkehrer, 1991) it uses the idea of the commands **tell**, **ask**, and **empty**; and (3) from an existing ABEL prototype (Lehmann, 1994; Haenni, 1996) it inherits the concept of **modules** and the syntax of the queries.

Working with ABEL usually involves three sequential steps:

- (1) The given information is expressed using the command **tell**. The resulting model is called **basic knowledge base**. It describes the part of the available information that is relatively constant and static in course of time such as rules, relations, or dependencies between different statements. It consists of

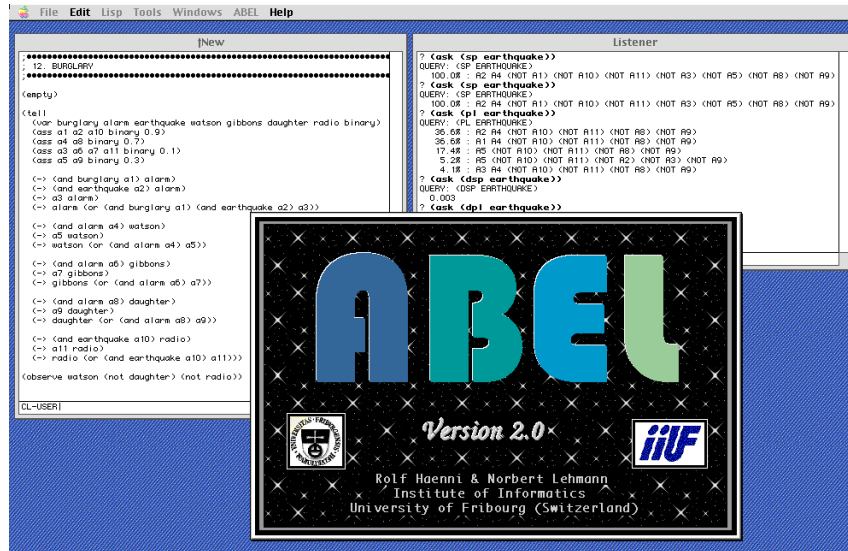


Fig. 4.2. A screenshot of the ABEL environment.

one or several lines called **instructions**. The contents of an instruction can be a **definition** of types, variables, assumptions, or modules, a **statement**, i.e. a rule or another part of the basic knowledge base, or an application of a **module**. The sequence of instructions is interpreted as a conjunction. The syntax of a **tell**-command is the following:

```
(tell <instr-1>
  ...
  <instr-n>)
```

Every instruction can be seen as a piece of information. Therefore, the command **tell** is used to add new pieces of information to the existing basic knowledge base. Note that the instructions of a **tell**-command can be distributed among several **tell**-commands.

- (2) In order to complete the model, **observations** or **facts** are added to the basic knowledge base. Observations describe the actual situation or the concrete circumstances of the problem. Note that observations may change in course of time. It is therefore important to separate observations from the basic knowledge base. ABEL provides a command **observe** to specify observations. It expects a sequence of ABEL statements. The sequence is interpreted as a conjunction.

```
(observe <stm-1>
  ...)
```

<stm-n>)

Statements given by **observe**- and **tell**-commands are treated similarly. The difference is that **observe**-statements can be deleted or changed independently when new observations were made. To change an observation, the same statement with the new values has to be re-written. The **empty**-command can be used to delete observations.

- (3) Queries about the actual knowledge base are expressed using the command **ask**. Generally, there are two different types of queries: (a) it can be interesting to get the available information about certain variables; and (b) it can be interesting to get symbolic or numerical arguments in favor or against certain hypotheses. In both cases, several queries can be treated at once:

```
(ask <query-1>
  ...
  <query-n>)
```

In the first case, a query is simply an ABEL expression. This type of query is useful, for example, in constraint satisfaction problems (consider (Anrig *et al.*, 1997a) for examples). The second way to state queries is important in problems of assumption-based reasoning, when different types of arguments may be of interest. The respective keywords in ABEL are **sp**, **qs**, **p1**, **db**. A hypothesis is an ABEL statement. Arguments are conjunctions of normal or negated assumptions. It is also possible to ask for numerical arguments like **degree of support**, **degree of quasi-support**, **degree of plausibility**, or **degree of doubt** (Haenni, 1996); the respective keywords in ABEL are **dsp**, **dqs**, **dp1**, **ddb**. A numerical argument is obtained by computing the probability for the corresponding symbolic argument. This computation is based on prior probabilities specified in the definition of the assumptions.

In this paper the language is not discussed further. For a precise language description refer to (Anrig *et al.*, 1997a).

In the case of a query, i.e. if the user formulates a hypothesis, then ABEL starts the inference mechanism and computes corresponding symbolic or numerical arguments. The inference mechanism is based on modules for the following seven tasks:

- (1) Compile the ABEL model into a corresponding set of clauses. Transform the clauses into an appropriate internal representation. Report possible errors.
- (2) Apply a method called Tree-Alg to transform the hypergraph given by the variables contained in the clauses into a covering hypertree (Kohlas & Monney, 1995).
- (3) Construct from the hypertree a corresponding valuation network (Shenoy & Shafer, 1990) and choose an arbitrary root. Distribute the clauses among the nodes of the network, i.e. decompose the knowledge base.

- (4) Propagate the clauses through the network towards the root using Shenoy's message-passing scheme (Shenoy & Shafer, 1990; Kohlas & Moral, 1995) (only inward phase). This delivers the marginal for the root of the network.
- (5) Select from the network an appropriate node that can be used to answer the query. Perform the outward propagation from the root to this node.
- (6) Use the resulting clauses obtained from the node selected in step (5) to compute the symbolic arguments for the hypothesis. Display the results in a convenient form.
- (7) Possibly, transform the symbolic arguments into corresponding numerical arguments.

The techniques implemented in ABEL for performing step (1) to step (7) are not discussed further. In the two remaining subsections, in order to illustrate the use of ABEL, we present the corresponding ABEL models for the examples introduced in Section 2.

4.1 Example 1: Chest Clinic

Consider the chest clinic example from Section 2.2. The knowledge base is a set of material implications resulting from the corresponding causal network. In ABEL, material implications can be expressed as follows (Anrig *et al.*, 1997a):

```
(-> <condition> <conclusion>)
```

The two expressions <condition> and <conclusion> are either atomic symbols (propositions or assumptions) or composed logical formulas. Furthermore, if we suppose the probabilities

$$\begin{aligned}
 p(a_1) &= 0.1, & p(a_2) &= 0.01, & p(a_3) &= 0.2, & p(a_4) &= 0.1, & p(a_5) &= 0.3, \\
 p(a_6) &= 0.1, & p(a_7) &= 0.9, & p(a_8) &= 0.8, & p(a_9) &= 0.1, & p(a_{10}) &= 0.9, \\
 p(a_{11}) &= 0.8, & p(a_{12}) &= 0.7, & p(a_{13}) &= 0.1,
 \end{aligned}$$

for the assumptions involved in the model, then the knowledge base can be encoded as follows:

```
(tell
  (var visit tuberculosis x-ray lung-cancer smoker
    bronchitis dyspnoea binary)
  (ass a1 a6 a9 a13 binary 0.1)
  (ass a2 a4 binary 0.01)
  (ass a3 binary 0.2)
  (ass a5 binary 0.3)
  (ass a7 a10 binary 0.9)
  (ass a8 a11 binary 0.8)
  (ass a12 binary 0.7)

  (-> (and visit a1) tuberculosis)
  (-> a2 tuberculosis)
```

```

(-> tuberculosis (or (and visit a1) a2))

(-> (and smoker a3) lung-cancer)
(-> a4 lung-cancer)
(-> lung-cancer (or (and smoker a3) a4))

(-> (and smoker a5) bronchitis)
(-> a6 bronchitis)
(-> bronchitis (or (and smoker a5) a6))

(-> (and lung-cancer a7) x-ray)
(-> (and tuberculosis a8) x-ray)
(-> a9 x-ray)
(-> x-ray (or (and lung-cancer a7) (and tuberculosis a8) a9))

(-> (and tuberculosis a10) dyspnoea)
(-> (and lung-cancer a11) dyspnoea)
(-> (and bronchitis a12) dyspnoea)
(-> a13 dyspnoea)
(-> dyspnoea (or (and tuberculosis a10) (and lung-cancer a11)
                 (and bronchitis a12) a13)))

```

If the doctor observes that the patient who suffers from dyspnoea is a smoker and that he has visited an under-developed country recently, then d , s , and v can be added as observations to the basic knowledge base:

```
(observe dyspnoea smoker visit)
```

Now, symbolic or numerical arguments for hypotheses like “Does the patient suffer from tuberculosis?”, “Does the patient suffer from bronchitis?”, etc., may be of interest. For the hypotheses t (tuberculosis) and b (bronchitis) the system reports the following results (Anrig *et al.*, 1997a):

? (ask (sp tuberculosis))	? (ask (sp bronchitis))
QUERY: (SP TUBERCULOSIS)	QUERY: (SP BRONCHITIS)
56.5% : A1 A10	49.2% : A12 A5
13.2% : A1 A12 A5	16.4% : A12 A6
10.0% : A1 A11 A3	11.2% : A11 A3 A5
6.3% : A1 A13	7.0% : A13 A5
5.7% : A10 A2	6.3% : A1 A10 A5
4.4% : A1 A12 A6	3.7% : A11 A3 A6
1.3% : A12 A2 A5	2.3% : A13 A6
1.0% : A11 A2 A3	2.1% : A1 A10 A6
0.6% : A13 A2	0.6% : A10 A2 A5
0.5% : A1 A11 A4	0.6% : A11 A4 A5
0.4% : A12 A2 A6	0.2% : A10 A2 A6
0.1% : A11 A2 A4	0.2% : A11 A4 A6
? (ask (dsp tuberculosis))	? (ask (dsp bronchitis))
QUERY: (DSP TUBERCULOSIS)	QUERY: (DSP BRONCHITIS)
0.206	0.591

From symbolic support the doctor gets configurations of assumptions that allow him to explain or deduce the hypothesis. In contrast, numerical degrees of support help to judge and compare the results quantitatively. Here, bronchitis is almost three times as credible as tuberculosis.

4.2 Example 2: Burglary

Consider the burglary example from Section 2.3. Again, if we suppose probabilities for the assumptions a_1, \dots, a_{11} , for example

$$p(a_1) = 0.9, p(a_2) = 0.9, p(a_3) = 0.1, p(a_4) = 0.7, p(a_5) = 0.3, p(a_6) = 0.1, \\ p(a_7) = 0.1, p(a_8) = 0.7, p(a_9) = 0.3, p(a_{10}) = 0.9, p(a_{11}) = 0.1,$$

then the problem can be modeled as follows:

```
(tell
  (var burglary alarm earthquake watson gibbons daughter radio binary)
  (ass a1 a2 a10 binary 0.9)
  (ass a4 a8 binary 0.7)
  (ass a3 a6 a7 a11 binary 0.1)
  (ass a5 a9 binary 0.3)

  (-> (and burglary a1) alarm)
  (-> (and earthquake a2) alarm)
  (-> a3 alarm)
  (-> alarm (or (and burglary a1) (and earthquake a2) a3))

  (-> (and alarm a4) watson)
  (-> a5 watson)
  (-> watson (or (and alarm a4) a5))

  (-> (and alarm a6) gibbons)
  (-> a7 gibbons)
  (-> gibbons (or (and alarm a6) a7))

  (-> (and alarm a8) daughter)
  (-> a9 daughter)
  (-> daughter (or (and alarm a8) a9))

  (-> (and earthquake a10) radio)
  (-> a11 radio)
  (-> radio (or (and earthquake a10) a11)))
```

Suppose now that Mr. Holmes receives a phone call from his neighbor Dr. Watson who tells him that there is an alarm sound from the direction of Mr. Holmes's house. Furthermore, suppose that there is no announcement of an earthquake on the radio.

```
(observe watson (not radio))
```

Now, symbolic and numerical arguments for some hypotheses may be of interest. Here, Mr. Holmes will be mostly interested in whether there is a burglary or not. Some of the possible queries produce the following results:

```
? (ask (sp burglary))
QUERY: (SP BURGLARY)
  90.0% : A1 A10 A4 (NOT A11) (NOT A3) (NOT A5)
  10.0% : A1 A4 (NOT A11) (NOT A2) (NOT A3) (NOT A5)
? (ask (sp earthquake))
QUERY: (SP EARTHQUAKE)
 100.0% : A2 A4 (NOT A1) (NOT A10) (NOT A11) (NOT A3) (NOT A5)
? (ask (dsp burglary))
QUERY: (DSP BURGLARY)
  0.482
? (ask (dpl burglary))
QUERY: (DPL BURGLARY)
  1.000
? (ask (dsp earthquake))
QUERY: (DSP EARTHQUAKE)
  0.005
? (ask (dpl earthquake))
QUERY: (DPL EARTHQUAKE)
  0.105
```

There is a relatively strong degree of support of 0.482 for a burglary, but only a very weak support for an earthquake. Mr. Holmes should therefore immediately call the police and return to his house.

5 Summary

This paper introduces the framework of propositional assumption-based reasoning. It shows how a set of uncertain causal relations can be transformed into a corresponding assumption-based system. This technique is illustrated by two examples which are often used in AI literature. Then, the paper describes a valuation-based method for computing symbolic arguments for a given hypothesis. This method is implemented in ABEL, an interactive tool for probabilistic assumption-based reasoning. The paper describes ABEL and demonstrates how it can be used for solving the examples mentioned before.

The actual work consists in improving the modeling language and the ABEL solver. The language has already been extended and today it supports also discrete and numerical variables (Anrig *et al.*, 1997a). For example, ABEL allows now mixed expressions of the form $w \in \{q, r, s\} \wedge \sim a \rightarrow (x^2 + 3y \leq z)$. Clearly, this improves the expressiveness of the formalism and enlarges the field of possible applications significantly. But the idea behind is still the same: finding symbolic or numerical arguments for hypotheses given some knowledge.

Future work will mainly focus on approximation strategies for big models with modular structures.

References

- Anrig, B., Haenni, R., & Lehmann, N. 1997a. *ABEL – A New Language for Assumption-Based Evidential Reasoning under Uncertainty*. Tech. Rep. 97–01. University of Fribourg, Institute of Informatics.
- Anrig, B., Haenni, R., Kohlas, J., & Lehmann, N. 1997b. Assumption-based Modeling using ABEL. In: Gabbay, D., Kruse, R., Nonnengart, A., & Ohlbach, H.J. (eds), *First International Joint Conference on Qualitative and Quantitative Practical Reasoning; ECSQARU–FAPR’97* Springer, for Lecture Notes in Artif. Intell.
- Bertschy, R., & Monney, P.A. 1996. A Generalization of the Algorithm of Heidtmann to Non-Monotone Formulas. *Journal of Computational and Applied Mathematics*, **76**, 55–76.
- de Kleer, J. 1986. An Assumption-based TMS. *Artificial Intelligence*, **28**, 127–162.
- Dempster, A. 1967. Upper and Lower Probabilities Induced by a Multivalued Mapping. *Ann. Math. Stat.*, **38**, 325–339.
- Haenni, R. 1996. *Propositional Argumentation Systems and Symbolic Evidence Theory*. Ph.D. thesis, Institute of Informatics, University of Fribourg.
- Inoue, K. 1991. An Abductive Procedure for the CMS/ATMS. *Pages 34–53 of: Martins, J.P., & Reinfrank, M. (eds), Truth Maintenance Systems, Lecture Notes in A.I.* Springer.
- Kohlas, J. 1994. *Mathematical Foundations of Evidence Theory*. Tech. Rep. 94–09. Institute of Informatics, University of Fribourg.
- Kohlas, J. 1995. *Mathematical Foundations of Evidence Theory. Pages 31–64 of: Coletti, G., Dubois, D., & Scozzafava, R. (eds), Mathematical Models for Handling Partial Knowledge in Artificial Intelligence*. Plenum Press.
- Kohlas, J., & Monney, P.A. 1993. Probabilistic Assumption-Based Reasoning. In: Heckerman, & Mamdani (eds), *Proc. 9th Conf. on Uncertainty in Artificial Intelligence*. Kaufmann, Morgan Publ.
- Kohlas, J., & Monney, P.A. 1994. *Probabilistic Assumption-Based Reasoning*. Tech. Rep. 94–22. Institute of Informatics, University of Fribourg.
- Kohlas, J., & Monney, P.A. 1995. *A Mathematical Theory of Hints. An Approach to the Dempster-Shafer Theory of Evidence*. Lecture Notes in Economics and Mathematical Systems, vol. 425. Springer.
- Kohlas, J., & Moral, S. 1995. *Propositional Information System*. Working Paper. Institute of Informatics, University of Fribourg.
- Kohlas, J., Monney, P.A., Anrig, B., & Haenni, R. 1996. *Model-Based Diagnostics and Probabilistic Assumption-Based Reasoning*. Tech. Rep. 96–09. University of Fribourg, Institute of Informatics.
- Laskey, K.B., & Lehner, P.E. 1989. Assumptions, Beliefs and Probabilities. *Artificial Intelligence*, **41**, 65–77.
- Lauritzen, S.L., & Spiegelhalter, D.J. 1988. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. *Journal of Royal Statistical Society*, **50**(2), 157–224.
- Lehmann, N. 1994. *Entwurf und Implementation einer annahmenbasierten Sprache*. Diplomarbeit. Institute of Informatics, University of Fribourg.
- Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publ. Inc.
- Provan, G.M. 1990. A Logic-Based Analysis of Dempster-Shafer Theory. *International Journal of Approximate Reasoning*, **4**, 451–495.

- Reiter, R., & de Kleer, J. 1987. Foundations of Assumption-Based Truth Maintenance Systems. *Proceedings of the American Association in AI*, 183–188.
- Saffiotti, A., & Umkehrer, E. 1991. *PULCINELLA: A General Tool for Propagating Uncertainty in Valuation Networks*. Tech. Rep. IRIDIA, Université de Bruxelles.
- Shafer, G. 1976. *The Mathematical Theory of Evidence*. Princeton University Press.
- Shenoy, P.P., & Shafer, G. 1990. Axioms for Probability and Belief Functions Propagation. *In*: Shachter, R.D., & al. (eds), *Uncertainty in Artificial Intelligence 4*. North Holland.
- Siegel, P. 1987. *Représentation et Utilisation de la Connaissance en Calcul Propositionnel*. Ph.D. thesis, Université d'Aix-Marseille II. Luminy, France.
- Steele, G. L. 1990. *Common Lisp – the Language*. Digital Press.