Assumption-Based Reasoning with LPL [*]

N. Lehmann, Ch. Eichenberger, T. Hürlimann

Department of Informatics
University of Fribourg
CH–1700 Fribourg, Switzerland
`norbert.lehmann@unifr.ch`
`christianmarkus.eichenberger@unifr.ch`
`tony.huerlimann@unifr.ch`
`http://diuf.unifr.ch/tcs`

21st September 2005

Abstract

Probabilistic argumentation systems combine classical logic and probability theory. In order to demonstrate the expressive power of probabilistic argumentation systems the language and software system ABEL was developed. Among other features, the language ABEL incorporates a *module concept* allowing to perform abstractions. However, something missing in ABEL for years is an *index mechanism*, which gives the user a powerful weapon for modeling. LPL, a language and software system used above all for solving linear optimization problems includes such an index mechanism. So, why not extend the language and the corresponding software system LPL such that LPL could handle models from the domain of assumption-based reasoning? The purpose of this paper is to present the extended language LPL, where several new language constructs were integrated in order to treat models from the domain of assumption-based reasoning. Examples from different fields are modeled in this extended LPL language.

[*]Research supported by grants No. 20-67996.02 of the Swiss National Foundation for Research.

Contents

1 Introduction

Probabilistic argumentation systems combine in a new and convincing way classical logic and probability theory. A corresponding language and inference system called **ABEL** [2] is developed at the University of Fribourg. ABEL (Assumption-Based Evidential Language) is a general language, which includes among others a module concept allowing to perform abstractions.

LPL (Linear Programming Language) [12] is an other language developed at the University of Fribourg. It is a very general language which is mainly used to formulate optimization problems in the field of linear programming. Linear solvers and inference machines can be plugged to it. One of the main strong points of LPL is its integrated mechanism to deal with **index sets**, a concept still missing in ABEL.

LPL is not only a language, it is also an application which contains an inference machine that corresponding solvers can be connected to. However, in this paper we do not discuss the architecture of the LPL system nor the solvers which can be connected to it. Information about these topics can be found in [12].

This paper is based on [2], which contains many examples for ABEL. The main purpose of this paper is to present LPL language features which allow to model examples in the field of **probabilistic argumentation systems**. The LPL system includes a modeling and query language and a corresponding inference mechanism. It is based on an open architecture, which permits the later inclusion of specialized solvers.

Inference poses numerous computational problems, both for deduction or for the search of arguments as well as for determining the probabilities of these arguments. These computational aspects as well as the mathematical foundations of assumption-based reasoning will not be addressed here. We refer to [23], [8], [22], [10], and [9] for more information on the computational aspects. The mathematical foundations are discussed and developed in [17, 19, 14, 18, 7].

This paper is organized as follows: The basic idea of reasoning with arguments (sometimes called hypothetical reasoning) is developed in Chapter 2 by using a number of very simple, elementary problems. In addition, it is shown that among others **Dempster-Shafer theory** fits exactly into the framework and gets thereby a clear semantic which is often missing in the literature discussing this formalism. In Chapter 4, the elements of the LPL language are presented. In the subsequent Chapters, the LPL language is used to formulate a number of problems from different domains. These examples are grouped under the headings of model-based prediction and diagnostics, causal modeling and uncertain systems, and evidential reasoning.

2 Reasoning with Arguments

Argumentation systems as understood here are designed to cope with uncertain, partial, incomplete, and contradictory knowledge and information. Such systems provide a formalism to draw inferences from a given knowledge description. This is done by combining logic and probability theory. Logic is used for deduction, and probabilities are important for weighing the possible conclusions. Argumentation systems are therefore based on a novel combination of logic and probability theory. This allows to derive arguments in favor of and against hypotheses. These arguments may be weighted by their reliability, expressed by the probability that the arguments are valid. The **degree of support** of a hypothesis can be measured by the probability of the supporting arguments. Additionally, the **degree of possibility** of a hypothesis can be expressed by the probability of the arguments refuting the hypothesis.

2.1 The Basic Idea

A detailed description of argumentation systems can be found in [7]. The basic idea is quite simple and is based on the fact that two different kinds of variables are distinguished in argumentation systems: *assumption variables* (in the following called **assumptions**) and *ordinary variables* (in the following called **variables**). Assumptions are not certain to hold and are used to represent possible interpretations, unknown risks, or unpredictable circumstances. That is how uncertainty is captured.

Inference concerns queries about hypotheses. **Arguments** in favor of a hypothesis are sets of assumptions which permit to infer the hypothesis from the given knowledge. Under such assumptions the hypothesis must necessarily be true in the light of the available knowledge. Arguments in favor of the hypothesis form the **support** of it. Combinations of assumptions which permit to deduce the falsity of the hypothesis are counter-arguments, which increase the **doubt** on the hypothesis. Arguments which are not counter-arguments do not refute the hypothesis and contribute to the **possibility** of the hypothesis.

Arguments for and against a hypothesis form the **qualitative** part of an argumentation system. They provide reasons to believe or disbelieve a hypothesis. However, this will not be sufficient in most practical cases. The arguments must be weighed by their likelihood or reliability. This can be done based on prior probabilities of the assumptions, which measure how likely an assumption is to hold. The reliability of an argument is the probability that sufficient assumptions hold to make the argument valid. The **credibility** or **degree of support** of a hypothesis is then 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. This **quantitative** judgment 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 (see [19, 20]).

Certainly, the realization of these general ideas presupposes the specification of a formal system that allows to encode the knowledge, including the assumptions and their probabilities, and to perform deduction of hypotheses. In the following section such systems are described based on propositional logic. Of course, other systems are possible, for example systems based on linear equations and inequalities. Such systems will not be treated here, more information on this topic can be found in [15, 26].

2.2 Propositional Assumption-Based Reasoning

First, some very simple but typical situations will be considered. These situations are depicted in Figure 1. Dotted circles represent uncertain evidences, and dotted lines are uncertain implications. In addition, we write $P(\alpha)$ for the probability of a propositional formula α.

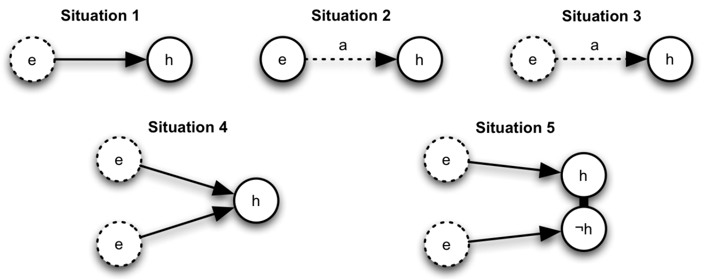


Figure 1: Different situations of propositional assumption-based models.

Situation 1: Suppose there is an uncertain evidence e. If e is valid, then it implies a hypothesis h necessarily. In terms of propositional logic, this situation can be expressed by

$$e \rightarrow h.$$

The proposition e is considered as an assumption, which may be valid or not. Let $p(e)$ be its probability. Modus ponens permits to deduce h if e is valid. Thus, e is an argument for h, and there are no arguments against h. The degrees of support and possibility for h are therefore

$$dsp(h) = p(e) \text{ and } dpl(h) = 1.$$

If $p(e)$ is sufficiently close to 1, for example if $p(e) = 0.9$, then we have sufficient reason to believe that h is true.

Situation 2: Suppose now that evidence e holds for sure, but that it implies h only contingently. This can be encoded by two propositional statements

$$\begin{gathered} e, \\ a \rightarrow (e \rightarrow h). \end{gathered}$$

In this situation a is an assumption which is only valid with a certain probability $p(a)$. Note that the second statement can also be written as $a \wedge e \rightarrow h$. Again, a is an argument for h, and there are no arguments against h. The degrees of support and possibility for h are again

$$dsp(h) = p(a) \text{ and } dpl(h) = 1.$$

This situation changes, if the following statement is added:

$$\sim\! a \rightarrow (e \rightarrow \sim\! h).$$

Now, $\sim\! a$ is an argument against h and therefore

$$dpl(h) = 1 - (1 - p(a)) = p(a).$$

In this case degrees of support and possibility are equal.

Situation 3: If both the evidence e as well as the implication are uncertain, then the knowledge is encoded simply as

$$a \rightarrow (e \rightarrow h).$$

Both propositions e and a are considered as assumptions with probabilities $p(e)$ and $p(a)$. In this situation, both assumptions e and a must be valid in order to deduce h. The set of literals $\{e, a\}$ is therefore an argument for h. If independence of the assumptions is assumed, then its degree of support is

$$dsp(h) = p(e) \cdot p(a).$$

There are no arguments against h unless a second statement such as

$$\sim\! a \rightarrow (e \rightarrow \sim\! h)$$

is added. Then, $\{e, \sim\! a\}$ is an argument against h and the degree of possibility of h becomes

$$dpl(h) = 1 - p(e) \cdot (1 - p(a)).$$

Note that support and possibility are different here. In general, we have $dsp(h) \leq dpl(h)$ [**?**].

Situation 4: Consider the following situation with two uncertain evidences e_1 and e_2, each of them implying the hypothesis h:

$$e_1 \rightarrow h, \quad e_2 \rightarrow h.$$

e_1 and e_2 are considered to be assumptions which hold with probability $p(e_1)$ and $p(e_1)$, respectively. Every assumption e_1 and e_2 is then an argument for h. The degree of support for h is the probability that at least one of these arguments is valid, that is

$$\begin{aligned} dsp(h) &= P(e_1 \vee e_2) \\ &= 1 - (1 - p(e_1)) \cdot (1 - p(e_2)). \end{aligned}$$

This formula shows how multiple evidence can reinforce the credibility of a hypothesis. The situation becomes even more interesting if we add the following two statements

$$\sim e_1 \rightarrow \sim h, \quad \sim e_2 \rightarrow \sim h.$$

The remarkable fact is that now either both literals e_1 and e_2 or both literals $\sim e_1$ and $\sim e_2$ must be valid. All other cases lead to a contradiction. In the first case we have an argument in favor of h, and in the second case an argument against h. Contradictory combinations of assumptions are not possible, since they contradict the given knowledge. Therefore, the results must be conditioned to the fact that

$$(e_1 \wedge e_2) \vee (\sim e_1 \wedge \sim e_2)$$

must be true. The degree of support of h can then be computed as follows:

$$\begin{aligned} dsp(h) &= P(e_1 \vee e_2 | (e_1 \wedge e_2) \vee (\sim e_1 \wedge \sim e_2)) \\ &= \frac{P(e_1 \wedge e_2)}{P((e_1 \wedge e_2) \vee (\sim e_1 \wedge \sim e_2))} \\ &= \frac{p(e_1) \cdot p(e_2)}{p(e_1) \cdot p(e_2) + (1 - p(e_1)) \cdot (1 - p(e_2))}. \end{aligned}$$

The degree of support for $\sim h$ can be determined in a similar way, that is we get

$$dsp(\sim h) = \frac{(1 - p(e_1)) \cdot (1 - p(e_2))}{p(e_1) \cdot p(e_2) + (1 - p(e_1)) \cdot (1 - p(e_2))}.$$

Note that this situation implies $dsp(h) + dsp(\sim h) = 1$, and consequently we have $dsp(h) = dpl(h)$.

Situation 5: The following situation of contradictory evidence must be treated similarly. Suppose that the statements

$$\begin{aligned} e_1 &\rightarrow h, \\ e_2 &\rightarrow {\sim}h, \end{aligned}$$

are given. Here e_1 and e_2 are considered as assumptions with corresponding probabilities $p(e_1)$ and $p(e_2)$. Clearly, e_1 is an argument in favor of h, and e_2 is an argument against h. Therefore, if both assumptions e_1 and e_2 are valid, then a contradictory situation arises. Again, contradictions must be eliminated by conditioning the results.

$$\begin{aligned} dsp(h) &= P(e_1 | {\sim}(e_1 \wedge e_2)) = \frac{P(e_1 \wedge {\sim}(e_1 \wedge e_2))}{P({\sim}(e_1 \wedge e_2))} = \frac{P(e_1 \wedge {\sim}e_2)}{1 - P(e_1 \wedge e_2)} \\ &= \frac{p(e_1) \cdot (1 - p(e_2))}{1 - p(e_1) \cdot p(e_2)}. \end{aligned}$$

Contradictory evidence weakens the strength of belief in the hypothesis. In fact, suppose, for example, $p(e_1) = p(e_2) = 0.9$. Then e_1 alone gives h a credibility of 0.9. However, in view of the above formula we obtain only $dsp(h) = 0.474$.

General Situation: Now, consider a more general situation where an arbitrary set of propositional formulae

$$\Sigma = \{\xi_1, \xi_2, \ldots, \xi_n\}$$

is given. These formulae contain propositional symbols from two different sets $A = \{a_1, a_2, \ldots, a_n\}$ (assumptions) and $P = \{p_1, p_2, \ldots, p_m\}$ (ordinary propositions). Let $A^{\pm} = \{a_1, \ldots, a_n, {\sim}a_1, \ldots, {\sim}a_n\}$ be the set of all literals formed by elements of A. A subset $a \subseteq A^{\pm}$ is called a **contradiction**, if a together with Σ is not satisfiable, that is

$$a, \Sigma \models \perp.$$

If h is an arbitrary propositional formula, then a subset $a \subseteq A^{\pm}$ is called **quasi-support** of h if a together with Σ entails h, that is

$$a, \Sigma \models h.$$

The subset $a \subseteq A^{\pm}$ is called **support** of h, if it is a quasi-support of h and there is no subset $a' \subseteq A^{\pm}$ containing a which is a contradiction. If we assume probabilities $p(a_i)$ for the corresponding assumptions a_i, then the **degree of support** of a hypothesis h can be defined as conditional probability

$$dsp(h) = P(\text{at least one support of } h \text{ is valid} \mid \text{no contradiction}).$$

Similarly, the **degree of possibility** of ${\sim}h$ can be defined as

$$dpl(h) = P(\text{no support of } {\sim}h \text{ is valid} \mid \text{no contradiction}).$$

For a rigorous discussion of this formalism we refer to [19].

2.3 General Framework of Assumption-Based Reasoning

Consider a non-empty set of variables $\{X_1, \dots, X_n\}$ which take values in the (not necessarily finite) sets $\Theta_1, \dots, \Theta_n$, called *frames*. Define the Cartesian product $\Theta = \times_{i=1}^{n} \Theta_i$. The set $X = \{X_1, \dots, X_n\}$ is called a *domain* and represents a precise question. Θ is called the *frame of discernment* and represents the set of possible answers to the question X (this is a closed world assumption). Let Ω be a non-empty set and suppose a function

$$\Gamma : \Omega \to \mathcal{P}(\Theta)$$

which represents uncertain information, as the information depends on the unknown $\omega \in \Omega$; therefore, Ω is called set of *assumptions* or *arguments*. A probabilistic argumentation system $\mathcal{P}$ is then a quartuple

$$\mathcal{P} = (\Theta, \Omega, \Gamma, \mu)$$

where μ is a prior probability distribution on a σ-algebra $\mathcal{F} \subseteq \mathcal{P}(\Omega)$. Note that Γ is generally represented by a set of sentences in a formal language including the variables $X_1, \dots, X_n$ and variables for the assumptions. Similarly, the sets $\Gamma(\omega)$ are then the sentences obtained by replacing the assumption variables by a configuration $\omega \in \Omega$, i.e. by substituting each variable for a value.

An assumption $\omega \in \Omega$ is *contradictory* if $\Gamma(\omega) = \emptyset$ since nothing holds under this assumption. Define Ω^* to be the set of non-contradictory arguments,

$$\Omega^* = \{\omega \in \Omega : \Gamma(\omega) \neq \emptyset\}.$$

If $\Omega^* = \emptyset$, then $\mathcal{P}$ yields a contradiction. If $\Omega^* \neq \emptyset$, let μ^* be the conditional probability measure on $\mathcal{F}$. The existence of μ^* can be assumed in many important cases under reasonable conditions, such as propositional argumentation systems, finite-set constraints and Gaussian linear systems [22, 19, 15, 26].

Let $H \subseteq \Theta$, then H represents a *hypothesis* on Θ. If $\Gamma(\omega) \subseteq H$, then the answer is a fortiori also in H. Hence, such an assumption $\omega \in \Omega$ is called *supporting argument* for H. The *support* of H is then the set of all supporting arguments,

$$sp(H) \quad = \quad \{\omega \in \Omega : \Gamma(\omega) \subseteq H\}.$$

The *degree of support* of H is defined by

$$dsp(H) \quad = \quad \mu^*(sp(H)), \qquad sp(H) \in \mathcal{F}.$$

The *possibility* of H is the set of arguments which do no support H^c,

$$\begin{aligned} pl(H) &= \{\omega \in \Omega : \Gamma(\omega) \not\subseteq H^c\} \\ &= \{\omega \in \Omega : \Gamma(\omega) \cap H \neq \emptyset\}, \end{aligned}$$

and the *degree of possibility*

$$\begin{aligned} dpl(H) &= \mu^*(pl(H)) \\ &= 1 - dsp(H^c), \qquad pl(H) \in \mathcal{F}, sp(H^c) \in \mathcal{F}. \end{aligned}$$

The result of assumption-based reasoning can be summarized in a structure called *hint*, a quintuple

$$\mathcal{H} = (\Omega^*, \mathcal{F}, \mu^*, \Gamma_{|\Omega^*}, \Theta)$$

where $\Gamma_{|\Omega^*}$ denotes the restriction of Γ to Ω^*. The functions dsp and dpl can then be derived from such a hint.

3 Software Environment

Section 4 contains a short introduction to the LPL language. The rest of the paper then contains different models from the field of assumption-based reasoning which are formulated in the LPL language. The user who wants to play around with these models has two possibilities to execute these models: the first possibility is to use the LPL *online-solver* on the internet, the second possibility is to download a *standalone application* of LPL.

3.1 Online-Solver on the Internet

The first possibility requires a computer with a connection to the internet. By using a standard internet browser, the user can go to

```
http://diufpc03.unifr.ch/lpl/abel.html
```

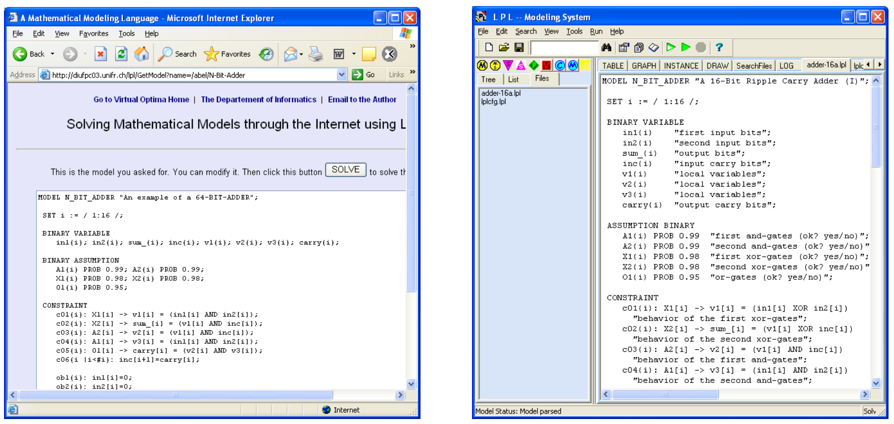
or

```
http://diufpc03.unifr.ch/lpl/gauss.html
```

where an online solver allows to solve models over the internet. On the left side of Figure 2 the corresponding page is shown.

3.2 Standalone Application

The second possibility for playing around with the models described in the following sections is to download the free version of the LPL application from the following URL:


A Mathematical Modeling Language - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://diufpc03.unifr.ch/lpl/GetModel?name=/abel/N-Bit-Adder
Go Links
Go to Virtual Optima Home | The Departement of Informatics | Email to the Author
SOLVE
Internet
L P L -- Modeling System
File Edit Search View Tools Run Help
TABLE GRAPH INSTANCE DRAW SearchFiles LOG adder-16a.lpl lplc
Tree List Files
adder-16a.lpl
lplcfg.lpl
Solv

Solving Mathematical Models through the Internet using L

This is the model you asked for. You can modify it. Then click this button SOLVE to solve th

```
MODEL N_BIT_ADDER "An example of a 64-BIT-ADDER";

 SET i := / 1:16 /;

 BINARY VARIABLE
    in1{i}; in2{i}; sum_{i}; inc{i}; v1{i}; v2{i}; v3{i}; carry{i};

 BINARY ASSUMPTION
    A1{i} PROB 0.99; A2{i} PROB 0.99;
    X1{i} PROB 0.98; X2{i} PROB 0.98;
    O1{i} PROB 0.95;

 CONSTRAINT
    c01{i}: X1[i] -> v1[i] = (in1[i] AND in2[i]);
    c02{i}: X2[i] -> sum_[i] = (v1[i] AND inc[i]);
    c03{i}: A2[i] -> v2[i] = (v1[i] AND inc[i]);
    c04{i}: A1[i] -> v3[i] = (in1[i] AND in2[i]);
    c05{i}: O1[i] -> carry[i] = (v2[i] AND v3[i]);
    c06{i |i<#i}: inc[i+1]=carry[i];

    ob1{i}: in1[i]=0;
    ob2{i}: in2[i]=0;
```

```
MODEL N_BIT_ADDER "A 16-Bit Ripple Carry Adder (I)";

 SET i := / 1:16 /;

 BINARY VARIABLE
    in1{i}    "first input bits";
    in2{i}    "second input bits";
    sum_{i}   "output bits";
    inc{i}    "input carry bits";
    v1{i}     "local variables";
    v2{i}     "local variables";
    v3{i}     "local variables";
    carry{i}  "output carry bits";

 ASSUMPTION BINARY
    A1{i} PROB 0.99  "first and-gates (ok? yes/no)";
    A2{i} PROB 0.99  "second and-gates (ok? yes/no)"
    X1{i} PROB 0.98  "first xor-gates (ok? yes/no)";
    X2{i} PROB 0.98  "second xor-gates (ok? yes/no)"
    O1{i} PROB 0.95  "or-gates (ok? yes/no)";

 CONSTRAINT
    c01{i}: X1[i] -> v1[i] = (in1[i] XOR in2[i])
      "behavior of the first xor-gates";
    c02{i}: X2[i] -> sum_[i] = (v1[i] XOR inc[i])
      "behavior of the second xor-gates";
    c03{i}: A2[i] -> v2[i] = (v1[i] AND inc[i])
      "behavior of the first and-gates";
    c04{i}: A1[i] -> v3[i] = (in1[i] AND in2[i])
      "behavior of the second and-gates";
```

Model Status: Model parsed

Figure 2: Online solver on the internet and standalone application.

```
http://www.virtual-optima.com
```

Then, after installing this software it is possible to write and solve models using the standalone application shown on the right side of Figure 2.

4 Syntax Elements of LPL

This section describes the main concepts and syntax elements of LPL (Linear Programming Language). We will concentrate on the syntax elements which were added to LPL in order to deal with models from the domain of *assumption-based reasoning*. A more detailed description of LPL can be found in [12].

In order to explain the new syntax elements introduced into LPL, we start by looking at the following example given in the ABEL language:

```
(tell
  (var b binary)
  (ass a1 binary 0.6)
  (ass a2 binary 0.8))

(tell
  (-> a1 b)
  (-> a2 (not b)))

(ask
  (qs b)
  (sp b))
```

Hereby, a binary variable `b` and two binary assumptions `a1`, `a2` are defined. In addition, two implications state the relationship between them. Finally, there are two queries for computing quasi-supporting and supporting arguments of b. In LPL, the above example is written as follows:

```
MODEL INTRODUCTION "An introductory example";
  BINARY VARIABLE
    b;
  ASSUMPTION BINARY VARIABLE
    a1 PROB 0.6;
    a2 PROB 0.8;
  CONSTRAINT
    c1: a1 -> b;
    c2: a2 -> ~b;
  QUERY
    q1: ABEL_QS(b), ABEL_SP(b);
END
```

The entire model in LPL is given as one single piece which contains also the queries. In order to answer the queries, the LPL application calls an appropriate solver and forwards the model to that solver. After the solver has performed its computations, the results are transmitted back to the LPL application, which finally displays them to the user. In contrast, the model in ABEL is constructed *incrementally*. This means that the knowledge base is empty at the beginning and each `tell` command then adjoins new pieces of knowledge or information to the knowledge base. Queries can be performed at any moment using the `ask` command.

As it can be seen, LPL has a clear and well-defined underlying structure. First, all variables and assumptions are defined. Then, all constraints follow. Finally, there are the queries.

Let us mention already now the main syntactical differences between the LPL language and the ABEL language:

- models always start with the keyword `MODEL` and terminate with the keyword `END`;
- each command is terminated by a *semicolon*;
- *infix notation* is used, whereas ABEL uses *prefix notation*;
- each constraint and the queries must have a *label*.

In the following, we will look at the individual parts of LPL models. For each part we give its corresponding syntax, following the convention that curly braces are used for grouping, brackets denote zero or one occurrence of the given entity, and terminal symbols are written in bold-faced capital letters. Commas, (semi-)colons and the terminal symbols correspond to the equally written tokens of the LPL language.

4.1 Definining Variables

Syntax : [`type`] **VARIABLE** $\{$`name ;`$\}^+$

`type` has to be an element one of the LPL keywords $\{$`BINARY`, `INTEGER`, `REAL`$\}$. The default type `INTEGER` is assumed when `type` is missing. Afterwards, one or several variables can be declared, each of them followed by a *semicolon*.

Example :

```
...
  BINARY VARIABLE
    a; b; c;
  INTEGER VARIABLE
    year; age;
  REAL VARIABLE
    amount;
...
```

In the above example, three binary variables, two integer variables and one real variable are defined.

4.2 Definining Assumptions

Syntax : **ASSUMPTION** [`type`] [**VARIABLE**] $\{$`name` [**PROB** `prob`] `;`$\}^+$

`type` can for the moment be one of the LPL keywords `BINARY` or `REAL`. The default type `BINARY` is assumed when `type` is missing. Afterwards, one or several assumptions can be declared. Probabilities can be assigned to binary assumptions using the keyword `PROB`, thus `type` is a number between 0 and 1. If no probability is assigned, then the uniform distribution is assumed when numerical queries are performed. Finally, each assumption is followed by a semicolon.

Example :

```
...
  ASSUMPTION BINARY VARIABLE
    a1; a2; a3 PROB 0.9;
  ASSUMPTION REAL VARIABLE
    omega;
```

```
...
    omega  := 5;
...
```

In the above example, first three binary assumptions are defined where `a3` is the only assumption which has a probability assigned to it. In addition, a real assumption is defined which is normally distributed with mean 0 and variance 5.

4.3 Definining Constraints

Syntax : **CONSTRAINT** {`label` : `constraint` `;`}$^+$

Each constraint must be introduced with a *label* followed by a *colon*. Then, the constraint can be formulated, again terminated with a *semicolon*. What concerns the concrete syntax for formulating queries, we refer to examples in the remainder of this paper and to the Reference Manual of LPL [12].

Example :

```
...
  CONSTRAINT
    c1: a1 -> b;
    c2: a2 -> ~b;
...
```

In the above example, two constraints are formulated. The first constraint expresses that `a1` implies `b`, whereas the second states that `a2` implies the negation of `b`.

4.4 Stating Queries

Syntax : **QUERY** {`label` : `query` {`, query`} `;`}$^+$

Each group of queries must be introduced with a label followed by a colon. Then, the queries can be formulated and must again be terminated with a semicolon. Each query in a group of queries is separated by a comma. There are three categories of queries, namely symbolic queries, numerical queries and queries for linear models.

Symbolic Queries

These queries allow to compute quasi-supporting, supporting and plausible arguments. The corresponding keywords are

- `ABEL_QS`,
- `ABEL_SP`,
- `ABEL_PL`.

Numerical Queries

These queries allow to compute the corresponding degrees for the arguments. The keywords to use for these queries are

- `ABEL_DQS`,
- `ABEL_DSP`,
- `ABEL_DPL`.

Linear Models

For linear models, a set of variables has to be provided. The returned result is then a representation of the non-contradictory arguments (called *Gaussian Hint*) on the given variables.

Example:

```
...
  QUERY
    q1: ABEL_QS(b), ABEL_DSP(b);
...
```

In the example above, the *quasi-support* of `b` and the *degree of support* of `b` are computed.

4.5 Freezing constraints

It was already mentioned that one of the main differences of LPL and ABEL is that, in LPL, the entire model is given as one single piece which contains also the queries. In contrast, the model in ABEL is constructed *incrementally* such that at each moment queries can be executed. This *interactive* aspect is one of the key features of ABEL.

Freezing constraints is a feature of LPL which is related to *interactivity*. It allows to switch on or off individual constraints. This is shown by the following example:

```
MODEL FREEZING "Another introductory example";
  BINARY VARIABLE
    x; y;
  ASSUMPTION BINARY VARIABLE
    a1 PROB 0.6;
    a2 PROB 0.8;
  CONSTRAINT
    c1: a1 -> x;
    c2: x AND a2 -> y;
    obs1 FREEZE: x;
  QUERY
    q1: ABEL_SP(y);
  UNFREEZE obs1;
  QUERY
    q2: ABEL_SP(y);
END
```

By executing this model the solver is called twice. The first time, the knowledge base consists of the two constraints `c1` and `c2` and the query `q1` is solved. The second time, the query `q2` is solved. But here, the knowledge base consists of three constraints because in addition to `c1` and `c2` the constraint `obs1` is also taken into account.

Therefore, the execution of the above model gives the following result:

```
/*-- query q1: --*/
ABEL_SP(y)=(a1 AND a2)

/*-- query q2: --*/
ABEL_SP(y)=a2
```

Obviously, the same query gives two different results. This is due to the fact that the knowledge base is different for the two calls of the solver.

4.6 Using Index Sets

The use of index sets can shorten and simplify the model description quite a lot. It would however be too much to explain here in detail the use of index sets. We refer to [11] and to the reference manual of LPL [12] for more information on the index mechanism of LPL. In the following, we will therefore only show a small fragment of the possibilities of the powerful index mechanism of LPL. For this purpose, consider the following small example:

Example:

Suppose that there are 5 binary assumptions $a_1, \ldots, a_5$ and a binary variable x such that $a_i \rightarrow x$ for $i \in \{1, \ldots, 5\}$. What is the modeling of this example in LPL if we would like to compute the set of supporting arguments of x?

Without using any index mechanism, the above information can be modeled as follows:

```
MODEL INDEX "Without using Index Sets";
  BINARY VARIABLE
    x;
  ASSUMPTION BINARY VARIABLE
    a1 PROB 0.7;
    a2 PROB 0.7;
    a3 PROB 0.7;
    a4 PROB 0.7;
    a5 PROB 0.7;
  CONSTRAINT
    c1: a1 -> x;
    c2: a2 -> x;
    c3: a3 -> x;
    c4: a4 -> x;
    c5: a5 -> x;
  QUERY
    q1: ABEL_SP(x);
END
```

However, by using the powerful index mechanism of LPL, the same example can be modeled in a much shorter and conciser way as follows:

```
MODEL INDEX "Using Index Sets";
  SET
    i := / 1:5 /;
  BINARY VARIABLE
    x;
  ASSUMPTION BINARY VARIABLE
    a{i} PROB 0.7;
  CONSTRAINT
    c{i}: a[i] -> x;
  QUERY
    q1: ABEL_SP(x);
END
```

4.7 Other Elements of LPL

Another feature of LPL is the possibility to have a **parameter** section where integer and real variables can be declared. The advantage is then that values can later be attributed to these variables in a separate **model data** section of the LPL model. This is shown by the following example.

```
...
  PARAMETER
    Bilbo    "the age of Bilbo";
    Frodo    "the age of Frodo";
  MODEL DATA myData "sample data from the book";
  BEGIN
    Bilbo := 111;
    Frodo :=  33;
  END
...
```

Of course, the LPL language has many additional features which were not presented here. We refer to [12] for a more complete and more detailed presentation.

5 Model-Based Prediction and Diagnostics

The problem of technical systems (airplanes, nuclear power stations, television sets, cars, computers, etc.) is that a correct functioning cannot be guaranteed. In this sense, technical systems are more or less unreliable. The degree of the reliability depends on different factors such as the quality, the age, the complexity, the maintenance, and the service of the system. For a given technical system, two important questions are of interest:

(1) How reliable is the system? (Prediction)

(2) Which is the component to be repaired or replaced when the system is not working correctly? (Diagnostics)

If a complete description or a model of the system's functionality is given, then argumentation systems can help answering the above questions. The following subsections discuss a number of concrete examples of model-based prediction and diagnostics.

5.1 A Communication Network (commNetwork)

The Description

Propositional assumption-based systems are well suitable for computing reliabilities of **communication networks** [13]. Such a network is composed of **nodes** which are connected by **communication wires**. If one or several nodes or wires are broken, then some point-to-point connections may be impossible. In such a case, the system is not functioning according to its communication functionality between the two nodes.

More generally, a system whose actual state is classified as *functioning* or *not functioning* is called a **binary system**. If a broken component of a binary system is always reducing the reliability of the entire system, then the system is called **monotone**. This condition seems to be reasonable for most cases. In fact, communication networks are monotone systems, since a broken node or wire is always decreasing the network reliability.

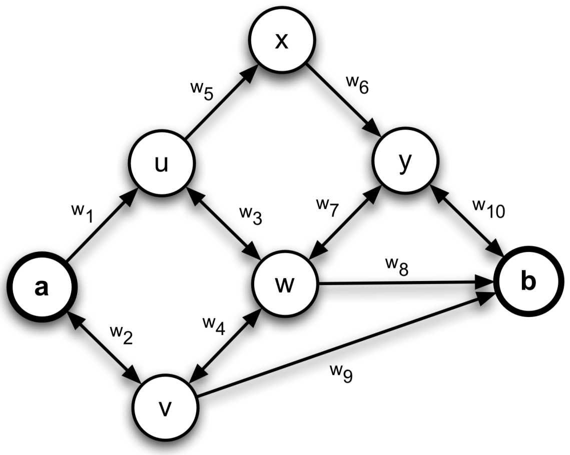


Figure 3: Example of a communication network

Communication networks can be described by directed or undirected graphs. For example, consider a communication network consisting of the nodes a, b, u, v, w, x, and y as shown in Figure 3. The nodes are connected by wires $w_1, w_2, \ldots, w_{10}$. A point-to-point communication has to be guaranteed between the nodes a and b. The reliability for this depends on the failure probabilities of the network components. For the purpose of simplification, we suppose that only the communication wires can break down and ok_i represents the assumption of w_i functioning correctly, $i = 1, \ldots, 10$. Furthermore, we assume the following independent probabilities for the wires:

$$\begin{array}{llll} p(ok_1) = 0.6, & p(ok_2) = 0.9, & p(ok_3) = 0.9, & p(ok_4) = 0.3, \\ p(ok_5) = 0.5, & p(ok_6) = 0.8, & p(ok_7) = 0.9, & p(ok_8) = 0.4, \\ p(ok_9) = 0.2, & p(ok_{10}) = 0.7. & & \end{array}$$

Note that some of the communication wires (that is $w_2, w_3, w_4, w_7, w_{10}$) are bi-directed, whereas others (that is w_1, w_5, w_6, w_8, w_9) are unidirectional.

The problem of the given example is to compute the reliability of the communication between the nodes a and b. For that purpose, at least one of the possible communication paths between a and b must be intact. Examples of communication paths are w_2–w_9, w_1–w_3–w_8, w_1–w_3–w_4–w_9, etc. The problem of computing the reliability of the network can therefore be divided into two sequential steps:

(1) determine all possible communication paths;

(2) derive the network reliability from the set of possible communication paths and the given failure probabilities.

The intactness of communication paths can be modeled in the following way: Each node is characterized by a binary variable $a, b, \ldots$. Two nodes are connected if and only if a binary signal is propagated correctly between them. The set of all possible communication paths between a and b can then be computed using the hypothesis $a \rightarrow b$. This is equivalent to require that there is at least one working path between the two nodes.

The Model

The corresponding model is as follows:

```
MODEL CommNetwork "A Communication Network";
  SET
    i := / 1:10 /   "indices for the edges";
  PARAMETER
    pr{i} := [0.6 0.9 0.9 0.3 0.5 0.8 0.9 0.4 0.2 0.7];
  BINARY VARIABLE
    a  "signal state of the corresponding node";
    b  "signal state of the corresponding node";
    u  "signal state of the corresponding node";
    v  "signal state of the corresponding node";
    w  "signal state of the corresponding node";
    x  "signal state of the corresponding node";
    y  "signal state of the corresponding node";
  ASSUMPTION BINARY VARIABLE
    ok{i} PROB pr "assumptions with probabilities";
  CONSTRAINT
    c01: ok[1] -> (a -> u);
    c02: ok[2] -> (a <-> v);
    c03: ok[3] -> (u <-> w);
    c04: ok[4] -> (v <-> w);
    c05: ok[5] -> (u -> x);
    c06: ok[6] -> (x -> y);
    c07: ok[7] -> (w <-> y);
    c08: ok[8] -> (w -> b);
    c09: ok[9] -> (v -> b);
    c10: ok[10] -> (y <-> b);
  QUERY
    q1: ABEL_SP(a->b), ABEL_SP(b->a);
  QUERY
    q2: ABEL_DSP(a->b),ABEL_DSP(b->a);
END
```

The Result

By executing this model, the solver is called twice and the results for the queries `q1` and `q2` are returned. There is only one communication path from b to a, whereas there are 10 communication paths from a to b. The second query shows that the reliability of the connection from b to a is much lower than that of the connection from a to b.

```
/*-- query q1: --*/
```

```
ABEL_SP(b->a)=(ok[2] AND ok[4] AND ok[7] AND ok[10]) ,
ABEL_SP(a->b)=(
 ok[2] AND ok[9] OR
 ok[2] AND ok[4] AND ok[8] OR
 ok[1] AND ok[3] AND ok[8] OR
 ok[1] AND ok[3] AND ok[4] AND ok[9] OR
 ok[2] AND ok[3] AND ok[4] AND ok[5] AND ok[6] AND ok[10] OR
 ok[1] AND ok[5] AND ok[6] AND ok[7] AND ok[8] OR
 ok[1] AND ok[4] AND ok[5] AND ok[6] AND ok[7] AND ok[9] OR
 ok[1] AND ok[5] AND ok[6] AND ok[10] OR
 ok[1] AND ok[3] AND ok[7] AND ok[10] OR
 ok[2] AND ok[4] AND ok[7] AND ok[10])

/*-- query q2: --*/
ABEL_DSP(b->a)=0.1701 ,
ABEL_DSP(a->b)=0.62316456
```

5.2 Communication Network (commnetX)

The Description

This communication network is based on an example from [4]. The network consists of 20 nodes and 57 undirected edges. The nodes are considered to be always working, the edges can fall down with a certain probability.

The Model

The 20 nodes are modeled by variables `k[a]` to `k[u]`. For every edge there is a corresponding assumption *ok* which denotes its working mode.

```
MODEL CommNetX "Communication Network";
  SET
    nodes ALIAS i,j :=
      / a b c d e f g h i j l m n o p q r s t u /;
    links{i,j} :=
      / a b, a c, a d, a e, b d, b e, b f, b k, c d
        c g, c h, d h, d e, d i, e g, e h, e i, e j,
        f j, f k, g h, g m, g n, h i, h j, h l, h n,
        h o, h r, i j, i k, i l, j k, j l, k l, k o,
        k p, l q, l o, l p, m n, m q, m r, n o, n q,
        n t, o p, o r, o s, o u, p s, q t, r t, r u,
        r s, s u, t u /;
  BINARY VARIABLE
    k{nodes}  "state of the nodes";
  ASSUMPTION BINARY VARIABLE
    ok{links} PROB 0.8  "working mode for the edges";
  CONSTRAINT
    c{links[i,j]}: ok[i,j] -> (k[i] <-> k[j]);
  QUERY
    q1: ABEL_DSP( k['a'] -> k['u'] );
END
```

Note the use of the two index sets `nodes` and `links`. The former is used to define the variables for the nodes, whereas the latter is used to define the assumptions corresponding to the edges of the network. The definition of the 20 variables and 57 assumptions takes only one single line in both cases. Also, the 57 constraints are generated by a single line in the model above. This shows again the descriptive power of index sets.

The Result

Note that the execution of the above model does not give any result with the current version of the ABEL solver because the computation would take too much time. Nevertheless, in a future version of the ABEL Solver, approximation algorithms will be implemented to compute an approximative result.

Note also that if the standalone application of the software system LPL is used (see Section 3.2) then the execution of the solver can be aborted by clicking with the mouse on the small button *Stop Running* or by selecting the corresponding menu item of the *Run* menu.

5.3 Communication Grid (commgrid)

The Description

A very simple structure of a communication network is the so called “grid”, i.e. a structure of the form of a chessboard [4], where a node is connected to its neighbor nodes on the right and above. This means, that a node at position (i, j) is connected to the nodes at position $(i, j+1)$ and $(i+1, j)$. Figure 4 contains a picture of such a specially structured communication network.

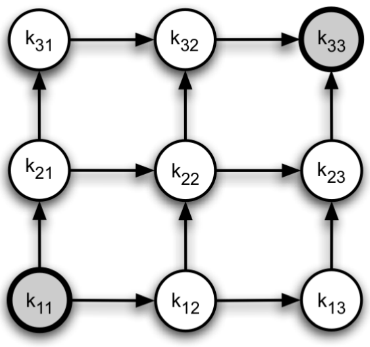


Figure 4: Communication network as a grid

The Model

Here, we describe a grid of size three, $i, j = 1, \ldots, 3$. For each node we define a variable and for every edge an assumption for modeling the respective state.

```
MODEL Communication_Grid "Communication Grid";
  SET
```

```
    n ALIAS i,j := /1:3/;
    arcs{i,j,r=i,s=j} := i=r and s=j+1 or j=s and r=i+1
        "connection to nodes below and on the right";
  BINARY VARIABLE
    k{i,j}  "state for node (i,j) of the grid";
  ASSUMPTION BINARY VARIABLE
    ok{arcs} PROB 0.8  "horizontal and vertical connections";
  CONSTRAINT
    c{arcs[i,j,r,s]}: ok[i,j,r,s] -> (k[i,j] -> k[r,s]);
  QUERY
    q1: ABEL_DSP(k[1,1] -> k[#i,#j]),
        ABEL_SP(k[1,1] -> k[#i,#j]);
END
```

Note the use of the two index sets `n` and `arcs`. The former is used to define the variables for the nodes, whereas the latter is used to define the assumptions corresponding to the edges of the network. For a communication grid of size 3 as shown above, there are 9 nodes and 12 edges which have to be defined. The definition of the nodes and assumptions takes only one single line in both cases. Also, the 12 constraints are generated by a single line in the model above. Once again, this shows again the descriptive power of index sets.

It is also worth noticing the use of the operator # in the query part. This operator returns the cardinality of an index set. In the example above, `#i` and `#j` are both equal to 3 because each of these two index sets are composed of 3 elements. In that way, it is possible to change the size of the grid without the need to change the query part as well.

The Result

We are interested in the question whether a message sent from the node left below arrives at the node on the right side on top. For that, all communication paths between these two nodes are computed. In addition, the degree of support is computed as well.

Executing the above model gives the following result:

```
/*-- query q1: --*/
ABEL_SP(k[1,1]->k[3,3])=(
 ok[1@1@2@1] AND ok[2@1@3@1] AND ok[3@1@3@2] AND ok[3@2@3@3] OR
 ok[1@1@2@1] AND ok[2@1@2@2] AND ok[2@2@3@2] AND ok[3@2@3@3] OR
 ok[1@1@1@2] AND ok[1@2@2@2] AND ok[2@2@3@2] AND ok[3@2@3@3] OR
 ok[1@1@2@1] AND ok[2@1@2@2] AND ok[2@2@2@3] AND ok[2@3@3@3] OR
 ok[1@1@1@2] AND ok[1@2@2@2] AND ok[2@2@2@3] AND ok[2@3@3@3] OR
 ok[1@1@1@2] AND ok[1@2@1@3] AND ok[1@3@2@3] AND ok[2@3@3@3]) ,
ABEL_DSP(k[1,1]->k[3,3])=0.85816089
```

All 6 communication paths have a length of 4 and it can easily be verified that these are indeed valid paths from the node k_{11} to the node k_{33}. For example, the first of the above communication paths goes through the nodes k_{11}–k_{21}–k_{31}–k_{32}–k_{33}.

Several larger communication grids have been modeled and tested. However, as size increases, much more time is needed for the computation. This is due to the fact that a communication grid of size n has $\binom{2n}{n}$ communication paths from k_{11} to k_{nn}. This means that we have an exponential growth of the number of communication paths.

5.4 Availability of Energy Distribution Systems (energy)

The Description

The function of an energy distribution system as depicted in Figure 5 is to supply a high-tension line L_3 from one of two incoming high-tension lines L_1 and L_2 over the bus A_1 [13]. The lines L_1, L_2, and L_3 are protected by corresponding high-tension switches S_1, S_2, and S_3. In case of a broken switch it is possible to redirect the electric current on a second bus A_2, which is protected by another high-tension switch S_4. The corresponding switches T_1, T_2, and T_3 can only be manipulated if no tension is present.

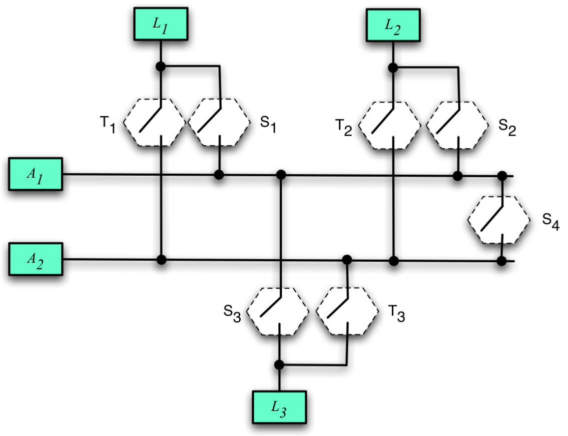


Figure 5: Example of an energy distribution system

The energy distribution system is considered to be operating if L_3 is linked to L_1 or L_2 over at least one protecting high-tension switch S_i. The problem is to determine the availability of an operating system.

The lines L_1, L_2, and L_3 have one binary attribute (tension *yes*/*no*) and the switches have two attributes (state *on*/*off*, intact *yes*/*no*). It is uncertain whether the switches are intact or faulty. For that purpose, we assume the following probabilities:

$$\begin{aligned} p(intact(S_i)) &= 0.8, \quad i = 1,2,3,4, \\ p(intact(T_i)) &= 0.95, \quad i = 1,2,3. \end{aligned}$$

In the following, we discuss how this system can be modeled. Note that for this purpose, the model is developed in several steps and is given as a sequence of blocks. Clearly, all these blocks have to be appended in order to obtain the entire model which can be executed.

The Model

The attributes of the lines and switches can be modeled by the following definitions of binary variables and assumptions:

```
MODEL Energy "Availability of Energy Distribution Systems";
  BINARY VARIABLE
    a1  "high-tension bus (tension yes/no)";
    a2  "high-tension bus (tension yes/no)";
    l1  "incoming line (tension yes/no)";
    l2  "incoming line (tension yes/no)";
    l3  "outgoing line (tension yes/no)";
    s1  "switch to protect line l1 (state on/off)";
    s2  "switch to protect line l2 (state on/off)";
    s3  "switch to protect line l3 (state on/off)";
    s4  "switch to protect second bus (state on/off)";
    t1  "switch for line l1 and second bus (state on/off)";
    t2  "switch for line l2 and second bus (state on/off)";
    t3  "switch for line l3 and second bus (state on/off)";
  ASSUMPTION BINARY VARIABLE
    ok_s1 PROB 0.8  "is the switch intact? (yes/no)";
    ok_s2 PROB 0.8  "is the switch intact? (yes/no)";
    ok_s3 PROB 0.8  "is the switch intact? (yes/no)";
    ok_s4 PROB 0.8  "is the switch intact? (yes/no)";
    ok_t1 PROB 0.95 "is the switch intact? (yes/no)";
    ok_t2 PROB 0.95 "is the switch intact? (yes/no)";
    ok_t3 PROB 0.95 "is the switch intact? (yes/no)";
...
```

The energy can only pass through a switch if the switch is on and intact. Then, the component after the switch is supplied. The circuit shown in Figure 5 can therefore be modeled by the following implications:

```
...
  CONSTRAINT
    c1: (l1 AND s1 AND ok_s1) -> a1;
    c2: (l1 AND t1 AND ok_t1) -> a2;
    c3: (l1 AND s2 AND ok_s2) -> a1;
    c4: (l1 AND t2 AND ok_t2) -> a2;
    c5: (a1 AND s3 AND ok_s3) -> l3;
    c6: (a2 AND t3 AND ok_t3) -> l3;
    c7: (a1 AND s4 AND ok_s4) -> a2;
    c8: (a2 AND s4 AND ok_s4) -> a1;
...
```

We assume that T_1, T_2, and T_3 are only switched on if the corresponding high-tension switches S_1, S_2, and S_3 are faulty. Furthermore, we suppose that all high-tension switches are on:

```
...
    d1: (~ ok_s1) -> t1;
    d2: (~ ok_s2) -> t2;
    d3: (~ ok_s3) -> t3;
    d4: s1 AND s2 AND s3 AND s4;
...
```

Another important part of the model is the fact that T_1 and T_3, as well as T_2 and T_3, are not allowed to be switched on at the same time because this would supply L_3 without protection:

```
...
    d5: ~ (t1 AND t3);
    d6: ~ (t2 AND t3);
...
```

Finally, suppose that the incoming lines L_1 and L_2 are under tension:

```
...
    o1: l1;
    o2: l2;
...
```

The model is now almost complete, what remains is the part with the queries. We are interested in the reliability of L_3 being under tension, in the symbolic arguments, as well as in the numerical strength:

```
...
  QUERY
    q1: ABEL_SP(l3), ABEL_DSP(l3);
END
```

Executing the entire model gives the following result: The Result

```
/*-- query q1: --*/
```

```
ABEL_DSP(l3)=0.959931034 ,
ABEL_SP(l3)=(
 ok_s2 AND ok_s3 OR
 ok_s1 AND ok_s3 OR
 ok_t2 AND ok_s4 AND ok_s3 OR
 ok_t3 AND ok_s4 AND ok_s1 AND ok_s2 OR
 ok_t1 AND ok_s4 AND ok_s3)
```

We have strong evidence that L_3 is under tension. There are 5 supporting arguments for L_3.

5.5 An Arithmetical Network (arinet)

The Description

In the previous subsection, a digital circuit consisting of binary variables has been considered. Now, the more general case of integer variables will be discussed. The following example has been introduced in [5]. Subsequently, it was used in several papers on model-based diagnostics [28, 6, 16].

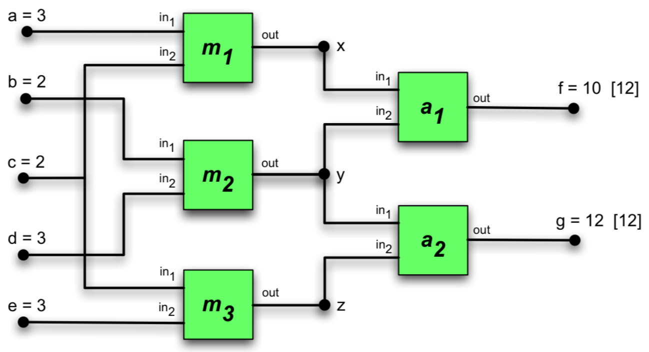


Figure 6: An arithmetical network

The network consists of three multipliers m_1, m_2, m_3, and two adders a_1, a_2. These components are connected as shown in Figure 6. If it is assumed that the components work either correctly or fail, then a binary variable can be used to represent the two working modes of a component. The behavior of the first adder a_1, for example, can then be expressed by

$$ok_{a_1} \quad \rightarrow \quad (f = x + y).$$

In the following, we discuss how this system can be modeled. For that purpose, the model is developed in several steps and is given as a sequence of blocks. Clearly, all these blocks have to be adjoined in order to obtain the entire model, which can be executed.

The Model

The state of each wire is modeled by an integer variable, whether a component works correctly by a binary assumption. The correct behavior of each component is modeled by a constraint. The reliability of a component is measured by a probability 0.97 for adders and 0.95 for multipliers. The behavior of a faulty component is not explicitly specified.

The modules are then instantiated according to the network topology of Figure 6. The global variable `ok` reflects the state of the whole system.

```
MODEL ArithNetwork "An Arithmetical Network";
  INTEGER VARIABLE
    a; b; c; d; e; x; y; z; f; g;
  BINARY VARIABLE
    ok;
  ASSUMPTION BINARY VARIABLE
    ok_m1 PROB 0.95; ok_m2 PROB 0.95; ok_m3 PROB 0.95;
    ok_a1 PROB 0.97; ok_a2 PROB 0.97;
  CONSTRAINT
    c1: ok_m1 -> (x = a * c);
    c2: ok_m2 -> (y = b * d);
    c3: ok_m3 -> (z = c * e);
    c4: ok_a1 -> (f = x + y);
    c5: ok_a2 -> (g = y + z);
    c6: ok -> ok_m1 AND ok_m2 AND ok_m3 AND ok_a1 AND ok_a2;
  ...
```

Following the example presented in [6], we first consider the situation where input and output measurements are available. The observations $g = 12$ and $x = 6$ will be used only later. This leads to the following:

```
  ...
    obs1: a = 3;
    obs2: b = 2;
    obs3: c = 2;
    obs4: d = 3;
    obs5: e = 3;
    obs6: f = 10;
    obs7 FREEZE: g = 12;
    obs8 FREEZE: x = 6;
  ...
```

Note the use of the keyword `FREEZE`. This implies that the corresponding observations are not yet active. They become active only when they are explicitly triggered using the keyword `UNFREEZE`.

The input observations and the output observation $f = 10$ together with the given knowledge imply that at least one component is faulty. The following query computes the minimal diagnoses of this situation and the degree of support for each of the five components to be faulty:

```
  ...
  QUERY
    q1: ABEL_SP(~ok);
    q2: ABEL_DSP(~ok_m1),ABEL_DSP(~ok_m2),ABEL_DSP(~ok_m3),
        ABEL_DSP(~ok_a1),ABEL_DSP(~ok_a2);
  ...
```

The following results are then obtained:

```
/*-- query q1: --*/
ABEL_SP(~ok)=(
 ~ok_m1 OR
 ~ok_a1 OR
 ~ok_m2)

/*-- query q2: --*/
ABEL_DSP(~ok_a2)=0.0301 ,
ABEL_DSP(~ok_a1)=0.241446594 ,
ABEL_DSP(~ok_m3)=0.05 ,
ABEL_DSP(~ok_m2)=0.401074076 ,
ABEL_DSP(~ok_m1)=0.401074076
```

The three diagnoses are more or less equally important and therefore, we cannot say for sure which of the components is faulty. Also, the numerical queries do not allow us to discriminate between the first and the second multiplier. This changes if a further observation is supplied. Suppose therefore that the observation $g = 12$ is added to the model. For that, we have to *unfreeze* the corresponding observation:

```
  ...
  UNFREEZE obs7;
  QUERY
    q3: ABEL_SP(~ok);
  QUERY
    q4: ABEL_DSP(~ok_m1),ABEL_DSP(~ok_m2),ABEL_DSP(~ok_m3),
        ABEL_DSP(~ok_a1),ABEL_DSP(~ok_a2);
  ...
```

Now, four more precise minimal diagnoses are computed. The following results are obtained:

```
/*-- query q3: --*/
ABEL_SP(~ok)=(
 ~ok_a2 AND ~ok_m2 OR
 ~ok_m3 AND ~ok_m2 OR
 ~ok_a1 OR
 ~ok_m1)
```

```
/*-- query q4: --*/
ABEL_DSP(~ok_a2)=0.045641104 ,
ABEL_DSP(~ok_a1)=0.366109271 ,
ABEL_DSP(~ok_m3)=0.075815787 ,
ABEL_DSP(~ok_m2)=0.091839195 ,
ABEL_DSP(~ok_m1)=0.608154935
```

Note that the first two diagnoses are rather improbable. The faulty component is therefore either the multiplier m_1 or the adder a_1.

However, we are not able to discriminate between the two components m_1 and a_1. This situation changes when we observe $x = 6$ (see [1] for a discussion and further literature on how to select such a variable). For that, again, we have to *unfreeze* the corresponding observation:

```
  ...
  UNFREEZE obs8;
  QUERY
    q5: ABEL_SP(~ok);
  QUERY
    q6: ABEL_DSP(~ok_m1),ABEL_DSP(~ok_m2),ABEL_DSP(~ok_m3),
        ABEL_DSP(~ok_a1),ABEL_DSP(~ok_a2);
  END
```

Now, only three minimal diagnoses remain, and only one of them has a high probability:

```
/*-- query q5: --*/
ABEL_SP(~ok)=(
 ~ok_a1 OR
 ~ok_a2 AND ~ok_m2 OR
 ~ok_m3 AND ~ok_m2)

/*-- query q6: --*/
ABEL_DSP(~ok_a2)=0.069761349 ,
ABEL_DSP(~ok_a1)=0.887605428 ,
ABEL_DSP(~ok_m3)=0.115882639 ,
ABEL_DSP(~ok_m2)=0.156774843 ,
ABEL_DSP(~ok_m1)=0.05
```

The first adder is most probably the faulty component, as the result of the numerical query shows.

5.6 A Failure Tree for a Medical Example (failure)

The concept of **failure trees** (or **event trees**) can help to determine possible causes

The Description

of a system failure, and it provides a useful structure for the computation of the probability of the system's working mode. In the case of an ordinary tree, the possible causes are rather easy to detect. But in the literature, the term *failure tree* is somewhat misleading because, in general, the structure considered is a causal network and not a tree. In this more general case, the computation of the causes is more difficult.

The failure tree of the following medical example was originally modeled in [30] and described in [3]. The purpose is to analyze the electrical shock hazard to a patient using a certain heart assist device:

> The intra-aortic balloon (IAB) circulatory assist device is intended to provide temporary circulatory assistance following obstruction of blood circulation from the heart. In its application, a balloon-catheter positioned in the thoracic aorta is synchronously inflated and deflated with the action of the heart, resulting in decreased heart work, increased coronary blood flow, and support of the general circulation.
>
> A synchronization of the balloon with the heart is obtained through monitoring of the electrocardiogram (ECG) of the patient by a control console. This console provides all of the control, monitoring display, and alarm functions required for operation of the balloon device. In addition, provision is made for automatic deflation of the balloon in the event of an anomaly in the ECG signal or functioning of the balloon.
>
> A major hazard to the patient is electrical shock, either macroshock or microshock. Macroshock can cause heart failure (ventricular fibrillation) by currents entering the body through connections at the skin. Microshock can cause heart failure by currents having a conductive path directly to the vicinity of the heart.
>
> Other hazards to the patient are balloon inflation during the contraction period of heart function and balloon overpressurization.

A detailed structure of the example is shown in Figure 7. Note that the example is only treated partially here. In order to reduce the tree's complexity, some terminal nodes have been introduced for hiding some subtrees. The abbreviations for the nodes introduced in the figure (e.g. `HP0` for the top node) will be used in order to keep the source code readable.

The Model

The structure of the tree of Figure 7 can easily be implemented in LPL. The terminal nodes of the tree are modeled as binary assumptions, and the non-terminal nodes as binary variables.

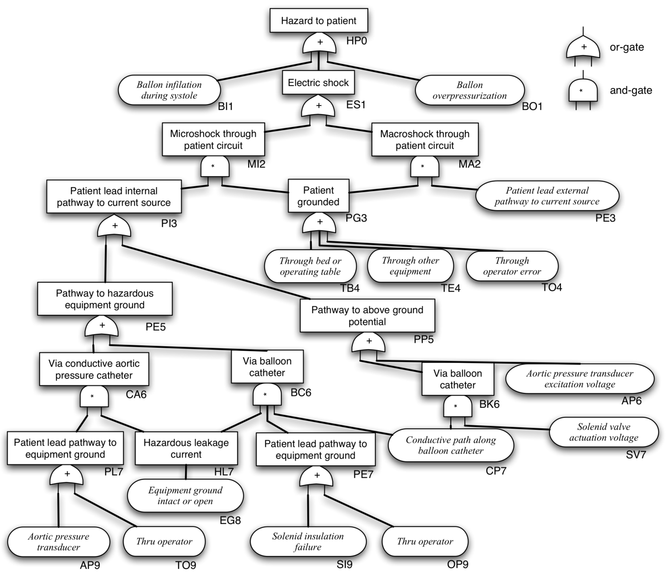

Hazard to patient
HP0
or-gate
and-gate
Ballon infilation during systole
BI1
Electric shock
ES1
Ballon overpressurization
BO1
Microshock through patient circuit
MI2
Macroshock through patient circuit
MA2
Patient lead internal pathway to current source
PI3
Patient grounded
PG3
Patient lead external pathway to current source
PE3
Through bed or operating table
TB4
Through other equipment
TE4
Through operator error
TO4
Pathway to hazardous equipment ground
PE5
Pathway to above ground potential
PP5
Via conductive aortic pressure catheter
CA6
Via balloon catheter
BC6
Via balloon catheter
BK6
Aortic pressure transducer excitation voltage
AP6
Patient lead pathway to equipment ground
PL7
Hazardous leakage current
HL7
Patient lead pathway to equipment ground
PE7
Conductive path along balloon catheter
CP7
Solenid valve actuation voltage
SV7
Equipment ground intact or open
EG8
Aortic pressure transducer
AP9
Thru operator
TO9
Solenid insulation failure
SI9
Thru operator
OP9

Figure 7: The failure tree of the heart assist device

```
MODEL FailureTree "A Failure Tree for a Medical Example";
  BINARY VARIABLE
    hp0 "Hazard to patient (yes/no)";
    es1 "Electric shock (yes/no)";
    mi2 "Microshock through patient circuit (yes/no)";
    ma2 "Macroshock through patient circuit (yes/no)";
    pi3 "Patient lead internal pathway to current source";
    pg3 "Patient grounded (yes/no)";
    pe5 "Pathway to hazardous equipment ground (yes/no)";
    pp5 "Pathway to above ground material (yes/no)";
    ca6 "Via conductive aortic pressure catheter (yes/no)";
    bc6 "Via balloon catheter (yes/no)";
    bk6 "Via balloon catheter (yes/no)";
    pl7 "Patient lead pathway to equipment ground  (yes/no)";
    hl7 "Hazardous leakage current (yes/no)";
    pe7 "Patient lead pathway to equipment ground  (yes/no)";
  ASSUMPTION BINARY VARIABLE
    bi1 PROB 0.01 "Balloon infiliation d. systole (yes/no)";
    bo1 PROB 0.01 "Balloon overpressurization (yes/no)";
    pe3 PROB 0.01 "Patient lead e. p. to c. source (yes/no)";
    tb4 PROB 0.08 "Through bet or operation table (yes/no)";
    te4 PROB 0.06 "Through other equipment (yes/no)";
    to4 PROB 0.02 "Through operator error (yes/no)";
    ap6 PROB 0.01 "Aortic pressure t. e. voltage (yes/no)";
    cp7 PROB 0.01 "Conductive path along balloon c.(yes/no)";
    sv7 PROB 0.05 "Solenid valve accuration voltage (yes/no)";
    eg8 PROB 0.05 "Equipment ground - intact/open (yes/no)";
    ap9 PROB 0.15 "Aortic pressure transducer (yes/no)";
    to9 PROB 0.01 "Through operator (yes/no)";
    si9 PROB 0.01 "Solenid insulation failure (yes/no)";
    op9 PROB 0.01 "Through operator (yes/no)";
  CONSTRAINT
    c01: hp0 <-> bi1 OR bo1 OR es1;
    c02: es1 <-> mi2 OR ma2;
    c03: mi2 <-> pi3 AND pg3;
    c04: ma2 <-> pe3 AND pg3;
    c05: pg3 <-> tb4 OR te4 OR to4;
    c06: pi3 <-> pe5 OR pp5;
    c07: pe5 <-> ca6 OR bc6;
    c08: pp5 <-> bk6 OR ap6;
    c09: ca6 <-> pl7 AND hl7;
    c10: bc6 <-> hl7 AND pe7 AND cp7;
    c11: bk6 <-> cp7 AND sv7;
    c12: hl7 <-> eg8;
    c13: pl7 <-> ap9 OR to9;
    c14: pe7 <-> si9 OR op9;
    obs1 FREEZE: hp0;
    obs2 FREEZE: ~bi1;
    obs3 FREEZE: ~bo1;
...
```

In reliability theory, minimal paths and reliabilities are calculated for such systems. This means to calculate the support for the top event hp0, whose results are the minimal paths, and the degree of support of hp0. This corresponds to the reliability of the system:

```
...
  QUERY
    q1: ABEL_SP(hp0);
  QUERY
    q2: ABEL_DSP(hp0), ABEL_DSP(~hp0);
...
```

The following 23 arguments supporting the hypothesis hp0 are obtained:

```
/*-- query q1: --*/
ABEL_SP(hp0)=(
 bo1 OR
 bi1 OR
 pe3 AND to4 OR
 pe3 AND tb4 OR
 pe3 AND te4 OR
 te4 AND cp7 AND eg8 AND si9 OR
 tb4 AND cp7 AND eg8 AND si9 OR
 to4 AND cp7 AND eg8 AND si9 OR
 te4 AND cp7 AND eg8 AND op9 OR
 tb4 AND cp7 AND eg8 AND op9 OR
 to4 AND cp7 AND eg8 AND op9 OR
 te4 AND eg8 AND to9 OR
 tb4 AND eg8 AND to9 OR
 to4 AND eg8 AND to9 OR
 te4 AND eg8 AND ap9 OR
 tb4 AND eg8 AND ap9 OR
 to4 AND eg8 AND ap9 OR
 te4 AND sv7 AND cp7 OR
 tb4 AND sv7 AND cp7 OR
 to4 AND sv7 AND cp7 OR
 te4 AND ap6 OR
 tb4 AND ap6 OR
 to4 AND ap6)
```

Nevertheless, the result for the numerical queries indicate that there is strong evidence that there is no hazard to the patient:

```
/*-- query q2: --*/
ABEL_DSP(~hp0)=0.975894194 ,
ABEL_DSP(hp0)=0.024105806
```

These calculations and results are well known in reliability theory, and a number of tools can do that. However, we can do more than that. Assume that there has been a hazard to the patient (`hp0`). Furthermore, suppose that neither balloon inflation during systole (`bi1`) nor balloon overpressurization (`bo1`) has been observed. The following query then computes supporting arguments for the top event `hp0`:

```
...
  UNFREEZE obs1, obs2, obs3;
  QUERY
    q3: ABEL_SP(hp0);
...
```

Now, there are 21 possible diagnoses for this new situation:

```
/*-- query q3: --*/
ABEL_SP(hp0)=(
 pe3 AND tb4 AND ~bi1 AND ~bo1 OR
 tb4 AND ap6 AND ~bi1 AND ~bo1 OR
 tb4 AND sv7 AND cp7 AND ~bi1 AND ~bo1 OR
 tb4 AND eg8 AND ap9 AND ~bi1 AND ~bo1 OR
 tb4 AND eg8 AND to9 AND ~bi1 AND ~bo1 OR
 tb4 AND cp7 AND eg8 AND op9 AND ~bi1 AND ~bo1 OR
 tb4 AND cp7 AND eg8 AND si9 AND ~bi1 AND ~bo1 OR
 pe3 AND te4 AND ~bi1 AND ~bo1 OR
 te4 AND ap6 AND ~bi1 AND ~bo1 OR
 te4 AND sv7 AND cp7 AND ~bi1 AND ~bo1 OR
 te4 AND eg8 AND ap9 AND ~bi1 AND ~bo1 OR
 te4 AND eg8 AND to9 AND ~bi1 AND ~bo1 OR
 te4 AND cp7 AND eg8 AND op9 AND ~bi1 AND ~bo1 OR
 te4 AND cp7 AND eg8 AND si9 AND ~bi1 AND ~bo1 OR
 pe3 AND to4 AND ~bi1 AND ~bo1 OR
 to4 AND ap6 AND ~bi1 AND ~bo1 OR
 to4 AND sv7 AND cp7 AND ~bi1 AND ~bo1 OR
 to4 AND eg8 AND ap9 AND ~bi1 AND ~bo1 OR
 to4 AND eg8 AND to9 AND ~bi1 AND ~bo1 OR
 to4 AND cp7 AND eg8 AND op9 AND ~bi1 AND ~bo1 OR
 to4 AND cp7 AND eg8 AND si9 AND ~bi1 AND ~bo1)
```

These diagnoses reflect the possible causes for the top level event, the hazard to the patient. Note that every diagnosis contains also the observed assumptions `bi1` and `bo1`.

In the present case, there are several causes within a certain interval of probability, so some mechanisms have to be applied to discriminate between these diagnoses. For example, another variable (or assumption) of the system can be measured. Furthermore, numerical queries are often helpful for a better analysis of the situation:

```
...
  QUERY
    q4: ABEL_DSP(tb4), ABEL_DSP(te4), ABEL_DSP(ap9),
        ABEL_DSP(pe3), ABEL_DSP(ap6), ABEL_DSP(eg8),
        ABEL_DSP(to4);
END
```

The following numerical results represent in fact the probabilities of the most probable events:

```
/*-- query q4: --*/
ABEL_DSP(to4)=0.131228567 ,
ABEL_DSP(eg8)=0.312037658 ,
ABEL_DSP(ap6)=0.355158956 ,
ABEL_DSP(pe3)=0.355158956 ,
ABEL_DSP(ap9)=0.369317193 ,
ABEL_DSP(te4)=0.393029558 ,
ABEL_DSP(tb4)=0.524914268
```

5.7 A Full Adder (adder-full)

The Description

Complex digital circuits can be built using very simple basic gates. In Figure 8, three types of basic gates are shown: An *and-gate*, an *xor-gate*, and an *or-gate*, respectively.

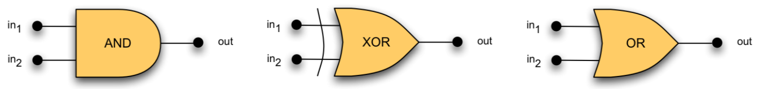


Figure 8: The three basic gates for more complex digital circuits

These gates implement a certain input-output behavior. For example, the output of an *and-gate* equals 1 if and only if both inputs are 1 too. Otherwise, the output equals 0. Such input-output behavior is usually represented by truth tables as shown in Table 1 for the three basic gates. If n denotes the number of input bits, then for each of the 2^n possible configurations of the input it can be seen what the corresponding output is.

in_1	in_2	out
0	0	0
0	1	0
1	0	0
1	1	1

(a) and-gate

in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	1

(b) or-gate

in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	0

(c) xor-gate

Table 1: Truth tables for the three basic gates

Using the three basic gates, we can build more complex digital circuits as for example the *full adder* shown in Figure 9, which implements the input-output behavior represented by the truth table of Table 2.

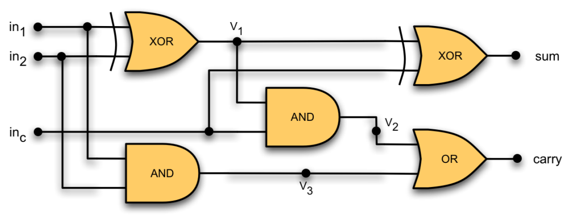


Figure 9: Full adder composed of basic gates

in_1	in_2	$carry\_in$	sum_out	$carry\_out$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2: Truth table for the full adder

However, what happens if one or several of the basic gates are not working? In that case, we cannot expect the input-output behavior of Table 2 for the *full adder* and we can also not expect the input-output behavior of Table 1 for *and-gates*, *or-gates*, and *xor-gates*, respectively.

In order to capture the situation where a basic gate is not working, we will assign an assumption to each basic gate. Then, the *normal* behavior is described using this assumption. For example, if `a1` is the assumption assigned to an *and-gate*, then we write

```
(a1 -> out = (in1 AND in2))
```

to express that if the *and-gate* is working correctly (`a1`), then `out` is the logical AND of `in1` and `in2`. Nothing is said here about the situation where the *and-gate* is not working correctly. In that way, we now have the input-output behavior shown in the truth table of Table 3 where an asterisk (*) means that either 0 or 1 is possible.

The Model

LPL can be used in order to answer this question. For that, the full adder is modeled as follows:

```
MODEL ADDER_FULL "A Full Adder";
  BINARY VARIABLE
    in1         "first input bit";
    in2         "second input bit";
    sum_out     "output bit";
    carry_in    "input carry bit";
    v1          "local variable";
    v2          "local variable";
    v3          "local variable";
    carry_out   "output carry bit";
  ASSUMPTION BINARY VARIABLE
    a1 PROB 0.99  "first and-gate (ok yes/no)";
    a2 PROB 0.99  "second and-gate (ok yes/no)";
```

$a1$	in_1	in_2	out
0	0	0	*
0	0	1	*
0	1	0	*
0	1	1	*
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(a) and-gate

$o1$	in_1	in_2	out
0	0	0	*
0	0	1	*
0	1	0	*
0	1	1	*
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(b) or-gate

$x1$	in_1	in_2	out
0	0	0	*
0	0	1	*
0	1	0	*
0	1	1	*
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(c) xor-gate

Table 3: New truth tables for the three basic gates

```
    x1 PROB 0.98  "first xor-gate (ok yes/no)";
    x2 PROB 0.98  "second xor-gate (ok yes/no)";
    o1 PROB 0.95  "or-gate (ok yes/no)";
  CONSTRAINT
    c1: x1 -> v1 = (in1 XOR in2);
    c2: x2 -> sum_out = (v1 XOR carry_in);
    c3: a2 -> v2 = (v1 AND carry_in);
    c4: a1 -> v3 = (in1 AND in2);
    c5: o1 -> carry_out = (v2 OR v3);
...
```

Suppose now that we make the following observations:

- `in1 = 1`,
- `in2 = 0`,
- `carry_in = 1`,
- `sum_out = 1`,
- `carry_out = 0`.

This can be modeled as follows:

```
...
    obs1: in1=1        "first input bit is 1";
    obs2: in2=0        "second input bit is 0";
    obs3: carry_in=1   "input carry bit is 1";
    obs4: sum_out=1    "output bit is 1";
    obs5: carry_out=0  "output carry bit is 0";
...
```

Obviously, some of the basic gates must be broken because otherwise we would obtain `sum_out = 0` and `carry_out = 1`. The question is now which gates are faulty. In order to answer this question we can in a first step compute the set of contradictory arguments:

```
...
  PARAMETER CONTRADICTION:=0; TAUTOLOGY:=1;
  QUERY
    q1: ABEL_QS(CONTRADICTION);
...
```

Note that the value 0 represents all contradictory logical formulas. Similarly, the value 1 represents all tautological logical formulas. Instead of writing `ABEL_QS(0)` we prefer here to first define the symbols `CONTRADICTION` and `TAUTOLOGY` in the parameter section and then to use these symbols later.

The following two contradictory arguments are obtained:

```
/*-- query q1: --*/
ABEL_QS(0)=(
 x1 AND a2 AND o1 OR
 x1 AND x2)
```

This result indicates that the knowledge base becomes contradictory if we add the fact that both *xor-gates* are working correctly (`x1 AND x2`) to the knowledge base. Another way to obtain a contradictory knowledge base is by adding the fact that the first *xor-gate*, the second *and-gate* and the *or-gate* are working correctly.

Possible *explanations* of the observations are given by the result of the following query.

```
...
  QUERY
    q2: ABEL_SP(TAUTOLOGY);
END
```

The following three explanations are obtained:

```
/*-- query q2: --*/
ABEL_SP(1)=(
 ~x2 AND ~o1 OR
 ~x2 AND ~a2 OR
 ~x1)
```

One explanation of the observations is that the first *xor-gate* is possibly not working correctly (`~x1`) . Another explanation is that the second *xor-gate* and the second *and-gate* are possibly not working correctly (`~x2 AND ~a2`). And there is yet a third explanation (`~x2 AND ~o1`).

5.8 A 4-Bit Ripple Carry Adder (I) (adder-4a)

The Description

Using and-gates, xor-gates and or-gates we can build an n-bit ripple carry adder. The purpose of this special kind of digital circuit is to compute the sum of two n-bit binary numbers. For that purpose, an n-bit ripple carry adder is composed of n binary one-bit adders.

Here, for the first version of an n-bit ripple carry adder, each binary one-bit adder is as shown in Figure 10. This corresponds therefore to the description in [2].

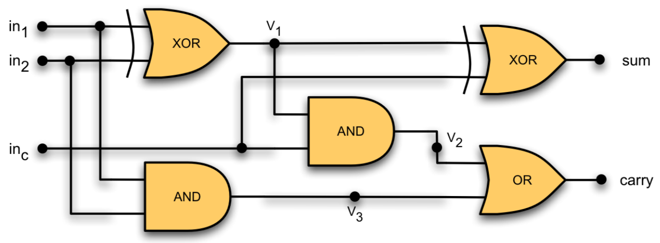


Figure 10: Version I of a binary one-bit adder

In the following, we describe a 4-bit ripple carry adder where both input numbers equal 0, however where the output equals 2^3. Obviously, one or several simple gates must be faulty. The question is then, which of the basic gates are most probably faulty and should be replaced.

The Model

In LPL, the modeling for a 4-bit ripple carry adder where both inputs equal 0 and output equals 2^3 is as follows:

```
MODEL N_BIT_ADDER "A Ripple Carry Adder (I)";
  SET i := / 1:4 /;
  BINARY VARIABLE
    in1{i}    "first input bits";
    in2{i}    "second input bits";
    out{i}    "output bits";
    inc{i}    "input carry bits";
    v1{i}     "local variables";
    v2{i}     "local variables";
    v3{i}     "local variables";
    carry{i}  "output carry bits";
  ASSUMPTION BINARY VARIABLE
    a1{i} PROB 0.99  "first and-gates (ok? yes/no)";
    a2{i} PROB 0.99  "second and-gates (ok? yes/no)";
    x1{i} PROB 0.98  "first xor-gates (ok? yes/no)";
    x2{i} PROB 0.98  "second xor-gates (ok? yes/no)";
    o1{i} PROB 0.95  "or-gates (ok? yes/no)";
  CONSTRAINT
    c1{i}: X1[i] -> v1[i] = (in1[i] XOR in2[i]);
    c2{i}: X2[i] -> out[i] = (v1[i] XOR inc[i]);
    c3{i}: A2[i] -> v2[i] = (v1[i] AND inc[i]);
    c4{i}: A1[i] -> v3[i] = (in1[i] AND in2[i]);
    c5{i}: O1[i] -> carry[i] = (v2[i] OR v3[i]);
    c6{i |i<#i}: inc[i+1]=carry[i];
    obs1{i}: in1[i]=0  "all first input bits are 0";
    obs2{i}: in2[i]=0  "all second input bits are 0";
    obs3: inc[1]=0     "the first input carry bit is 0";
    obs4{i}: out[i]=if(i<#i,0,1) "the last output is 1";
    obs5: carry[#i]=0  "the last carry bit is 0";
  PARAMETER CONTRADICTION:=0; TAUTOLOGY:=1;
  QUERY
    q1: ABEL_SP(TAUTOLOGY);
  QUERY
    q2: ABEL_DQS(CONTRADICTION);
END
```

Note that using the index mechanism of LPL, the model description is very short. Also, in that way it is very easy to model adders of any size as described in [2]. Currently, we have been working with n-bit ripple carry adders up to 256 input bits.

The Result

The set of arguments explaining the observations for the knowledge base is returned by the first query. Here, the most probable explanations are those 5 explanations consisting only of a single component. In contrast, it is much less probable, although not excluded, that more than one component is faulty at the same time.

```
/*-- query q1: --*/
ABEL_SP(1)=(
 ~x1[2] AND ~x1[3] AND ~o1[1] OR
 ~x1[2] AND ~x1[3] AND ~a1[1] OR
 ~x1[2] AND ~x1[3] AND ~a2[1] OR
 ~x1[3] AND ~o1[2] OR
 ~x1[3] AND ~a1[2] OR
 ~x1[3] AND ~a2[2] OR
 ~x1[4] OR
 ~x2[4] OR
 ~a2[3] OR
 ~a1[3] OR
 ˜o1[3])

/*-- query q2: --*/
ABEL_DQS(0)=0.892595952
```

5.9 A 16-Bit Ripple Carry Adder (II) (adder-16b)

The Description

The second version of an n-bit ripple carry adder is almost the same as the first one presented previously. The difference is the way the signal wires for the two binary n-bit input numbers and the signal wires for the carry bits are connected to the n binary one-bit adders. Figure 11 shows the kind of binary one-bit adder used in this second version.

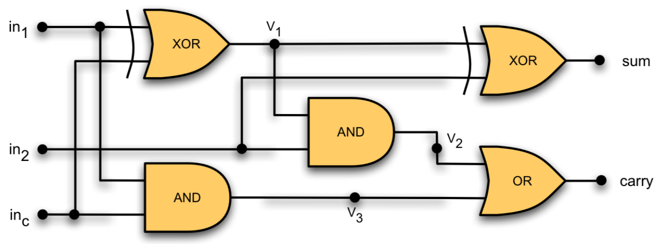


Figure 11: Version II of a binary one-bit adder

Again, we suppose that both 4-bit input numbers are equal to 0, whereas the 4-bit output number is equal to 2^3, which means that the last of the binary output bits is 1 instead of 1.

The Model

The modeling in LPL is as follows:

```
MODEL N_BIT_ADDER "A Ripple Carry Adder (II)";
  SET i := / 1:4 /;
  BINARY VARIABLE
    in1{i}    "first input bits";
    in2{i}    "second input bits";
    out{i}    "output bits";
    inc{i}    "input carry bits";
    v1{i}     "local variables";
    v2{i}     "local variables";
    v3{i}     "local variables";
    carry{i}  "output carry bits";
  ASSUMPTION BINARY VARIABLE
    a1{i} PROB 0.99  "first and-gates (ok? yes/no)";
    a2{i} PROB 0.99  "second and-gates (ok? yes/no)";
    x1{i} PROB 0.98  "first xor-gates (ok? yes/no)";
    x2{i} PROB 0.98  "second xor-gates (ok? yes/no)";
    o1{i} PROB 0.95  "or-gates (ok? yes/no)";
  CONSTRAINT
    c1{i}: x1[i] -> v1[i] = (in1[i] XOR inc[i]);
    c2{i}: x2[i] -> out[i] = (v1[i] XOR in2[i]);
    c3{i}: a2[i] -> v2[i] = (v1[i] AND in2[i]);
    c4{i}: a1[i] -> v3[i] = (in1[i] AND inc[i]);
    c5{i}: o1[i] -> carry[i] = (v2[i] OR v3[i]);
    c6{i |i<#i}: inc[i+1]=carry[i];
    obs1{i}: in1[i]=0  "all first input bits are 0";
    obs2{i}: in2[i]=0  "all second input bits are 0";
    obs3: inc[1]=0     "the first input carry bit is 0";
    obs4{i}: out[i]=if(i<#i,0,1) "the last output is 1";
    obs5: carry[#i]=0  "the very last carry bit is 0";
  PARAMETER CONTRADICTION:=0; TAUTOLOGY:=1;
  QUERY q1: ABEL_SP(TAUTOLOGY);
  QUERY q2: ABEL_DQS(CONTRADICTION);
END
```

The Result

The execution of the above model gives the following result:

```
/*-- query q1: --*/
ABEL_SP(1)=(
 ~x1[4] OR
 ~x2[4] OR
 ~a2[3] OR
 ~a1[3] OR
 ~o1[3])

/*-- query q2: --*/
ABEL_DQS(0)=0.893860547
```

Note that even if this second version of an n-bit ripple carry adder is very similar to the first version and even if we have the same observations, the computed explanations are different. Here, we have 5 arguments, whereas in the previous version we had some additional arguments.

6 Causal Modeling and Uncertain Systems

Causal 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*”. Generally, causality can be seen as “any natural ordering in which knowledge of an event influences opinion concerning another event. This influence can be logical, physical, temporal, or simply conceptual” [21].

Causal relations are often uncertain in the sense that the corresponding if-then-rules only hold when certain additional conditions or circumstances are present. The concept of Bayesian networks is an approach to uncertain reasoning which is based on the idea of such uncertain causal relations. The following subsections show that different types of uncertain causal systems can just as well be treated by probabilistic argumentation systems and can be modeled in LPL.

6.1 Medical Diagnostic I (meddiag1)

The Description

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 introduced in [21] in order to illustrate the use of Bayesian networks [27]. Here, the same example is treated by assumption-based reasoning.

The doctor's medical knowledge is useful to find the causes of the patient's symptoms. The knowledge is summarized as follows (see also [21]):

> 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 12 this small piece of fictitious medical knowledge is shown as a causal network. Causal relations are described by uncertain implications and 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.

All causal relations of Figure 12 are uncertain. Therefore, for each of the eight causal relations an assumption is needed. In addition, another five assumptions are used for all other (unknown) causes. With the thirteen assumptions $a_1, \ldots, a_{13}$ the fictitious medical knowledge is modeled as shown below.

$$\begin{aligned}
&(1) \quad && v \wedge a_1 \rightarrow t, \;\; a_2 \rightarrow t, \;\; t \rightarrow (v \wedge a_1) \vee a_2; \\
&(2) \quad && s \wedge a_3 \rightarrow \ell, \;\; a_4 \rightarrow \ell, \;\; \ell \rightarrow (s \wedge a_3) \vee a_4; \\
&(3) \quad && s \wedge a_5 \rightarrow b, \;\; a_6 \rightarrow b, \;\; b \rightarrow (s \wedge a_5) \vee a_6; \\
&(4) \quad && t \wedge a_7 \rightarrow x, \;\; \ell \wedge a_8 \rightarrow x, \;\; a_9 \rightarrow x, \\
& && x \rightarrow (t \wedge a_7) \vee (\ell \wedge a_8) \vee a_9; \\
&(5) \quad && t \wedge a_{10} \rightarrow d, \;\; \ell \wedge a_{11} \rightarrow d, \;\; b \wedge a_{12} \rightarrow d, \;\; a_{13} \rightarrow d \\
& && d \rightarrow (t \wedge a_{10}) \vee (\ell \wedge a_{11}) \vee (b \wedge a_{12}) \vee a_{13}.
\end{aligned}$$

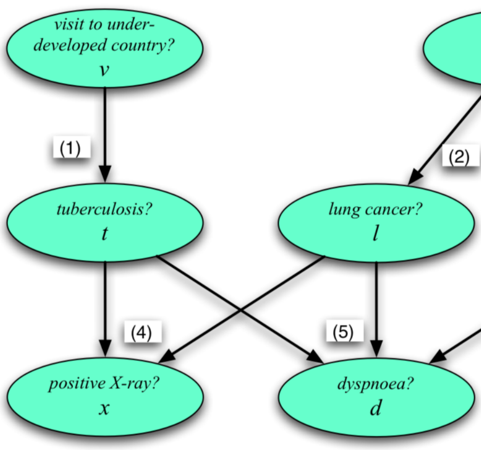


Figure 12: Causal network for the example "medical diagnostic I"

Concerning the probabilities, we assume the following values:

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}
0.1	0.01	0.2	0.01	0.3	0.1	0.9	0.8	0.1	0.9	0.8	0.7	0.1

The Model

Note that we discuss in the following how this medical knowledge can be modeled. For that purpose, the model is developed in several steps and is given as a sequence of blocks. Clearly, all these blocks have to be adjoined in order to obtain the entire model which can be executed.

First of all, the seven binary variables and the thirteen binary assumptions have to be declared:

```
MODEL MedDiag "Medical Diagnostic I";
  BINARY VARIABLE
    visit        "patient visited an underdeveloped country";
    tuberculosis "patient has tuberculosis";
    x_ray        "patient with a positive x-ray";
    lung_cancer  "patient has lung cancer";
    smoker       "patient is a smoker";
    bronchitis   "patient has bronchitis";
    dyspnoea     "patient is suffering from dyspnoea";
  ASSUMPTION BINARY VARIABLE
    a01 PROB 0.1  "reasons why visit leads to tuberculosis";
    a02 PROB 0.01 "other reasons for tuberculosis";
    a03 PROB 0.2  "reasons why smoking leads to lung cancer";
    a04 PROB 0.01 "other reasons for lung cancer";
    a05 PROB 0.3  "reasons why smoking leads to bronchitis";
    a06 PROB 0.1  "other reasons for bronchitis";
    a07 PROB 0.9  "lung cancer can lead to positive x-ray";
    a08 PROB 0.8  "tuberculosis can lead to positive x-ray";
    a09 PROB 0.1  "other reasons for a positive x-ray";
    a10 PROB 0.9  "tuberculosis can lead to dyspnoea";
    a11 PROB 0.8  "lung cancer can lead to dyspnoea";
    a12 PROB 0.7  "reasons why bronchitis leads to dyspnoea";
    a13 PROB 0.1  "other reasons for dyspnoea";
...
```

The second step then consists in formulating the constraints according to the causal relations of Figure 12. This gives the following:

```
...
  CONSTRAINT
    c01: visit AND a01 -> tuberculosis;
    c02: a02 -> tuberculosis;
    c03: tuberculosis -> (visit AND a01) OR a02;
    c04: smoker AND a03 -> lung_cancer;
    c05: a04 -> lung_cancer;
    c06: lung_cancer -> (smoker AND a03) OR a04;
    c07: smoker AND a05 -> bronchitis;
    c08: a06 -> bronchitis;
    c09: bronchitis -> (smoker AND a05) OR a06;
    c10: lung_cancer AND a07 -> x_ray;
    c11: tuberculosis AND a08 -> x_ray;
    c12: a09 -> x_ray;
    c13: x_ray -> (lung_cancer AND a07) OR
                  (tuberculosis AND a08) OR a09;
    c14: tuberculosis AND a10 -> dyspnoea;
    c15: lung_cancer AND a11 -> dyspnoea;
    c16: bronchitis AND a12 -> dyspnoea;
    c17: a13 -> dyspnoea;
    c18: dyspnoea -> (tuberculosis AND a10) OR
                     (lung_cancer AND a11) OR
                     (bronchitis AND a12) OR a13;
...
```

LPL does not really support an interactive way to build incrementally the knowledge base. However, using the keyword FREEZE, we may later switch individual constraints on and off. This is what we use below:

```
...
    obs1: dyspnoea;
    obs2 FREEZE: visit;
    obs3 FREEZE: smoker;
    obs4 FREEZE: ~ x_ray;
...
```

That the patient is suffering from dyspnoea is for the moment the only observation which is part of the knowledge base. At this point, the doctor can compute the degree of support for the three diseases:

```
...
  QUERY
    q1: ABEL_DSP(tuberculosis), ABEL_DSP(bronchitis),
        ABEL_DSP(lung_cancer);
...
```

The following numerical results are obtained:

```
/*-- query q1: --*/
ABEL_DSP(lung_cancer)=0.269643985 ,
ABEL_DSP(bronchitis)=0.485954645 ,
ABEL_DSP(tuberculosis)=0.118283626
```

None of the three hypotheses are strongly supported. This is a sign that additional observations are required. From a discussion with the patient the doctor learns that the patient recently visited an under-developed country:

```
...
  UNFREEZE obs2;    -- visit to under-developed country
  QUERY
    q2: ABEL_DSP(tuberculosis), ABEL_DSP(bronchitis),
        ABEL_DSP(lung_cancer);
...
```

This additional fact increases the degree of support for tuberculosis:

```
/*-- query q2: --*/
ABEL_DSP(lung_cancer)=0.269643985 ,
ABEL_DSP(bronchitis)=0.485954645 ,
ABEL_DSP(tuberculosis)=0.206455263
```

However, it is still not sufficient for the doctor to make a decision. After the observation that the patient is a smoker, more support is obtained for lung cancer and bronchitis:

```
...
  UNFREEZE obs3;    -- the patient is a smoker
  QUERY
    q3: ABEL_DSP(tuberculosis), ABEL_DSP(bronchitis),
        ABEL_DSP(lung_cancer);
...
```

Here are the corresponding numerical results:

```
/*-- query q3: --*/
ABEL_DSP(lung_cancer)=0.366978272 ,
ABEL_DSP(bronchitis)=0.591431336 ,
ABEL_DSP(tuberculosis)=0.206455263
```

The doctor is not yet satisfied because no decision is possible between lung cancer and bronchitis. So an X-ray chest test might give further hints. This test turns out to be negative:

```
...
  UNFREEZE obs4;    -- the X-ray chest test is negative
  QUERY
    q4: ABEL_DSP(tuberculosis), ABEL_DSP(bronchitis),
        ABEL_DSP(lung_cancer);
END
```

The following numerical results are obtained:

```
/*-- query q4: --*/
ABEL_DSP(lung_cancer)=0.061671312 ,
ABEL_DSP(bronchitis)=0.760124606 ,
ABEL_DSP(tuberculosis)=0.061870288
```

Now, the doctor is satisfied because a very clear diagnosis can be stated. Bronchitis has a quite strong degree of support, whereas lung cancer and tuberculosis are very incredible diagnoses. Figure 13 summarizes the process of finding the diagnosis.

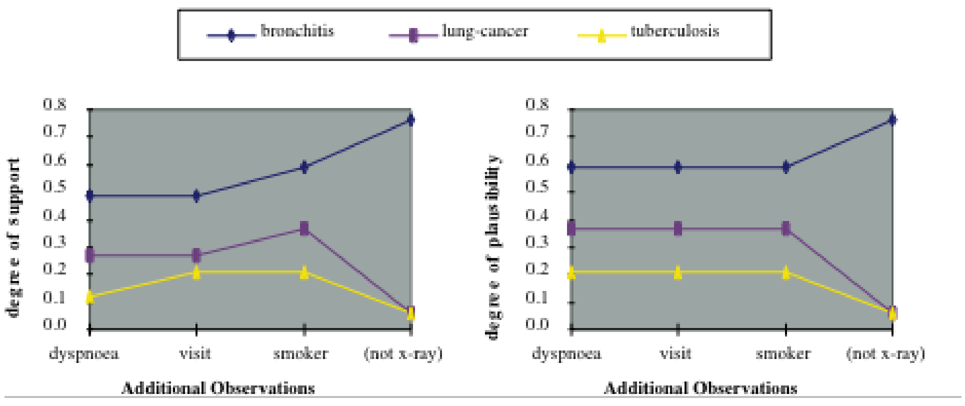


Figure 13: Degrees of support and possibility

Note that the doctor is only able to exclude lung cancer and tuberculosis with the utmost probability after observing `~x-ray`. This last observation decreases the degrees of support of lung cancer and tuberculosis significantly. This shows that assumption-based reasoning is non-monotone in the sense that additional information can decrease the belief or credibility of a hypothesis.

6.2 A Markov System (markov)

The Description

General Markov systems are a well-known mathematical concept for describing dynamic patterns of influence for stochastical variables. Here, only discrete Markov systems are considered where time is treated as a sequence of discrete points. Formally, a **discrete Markov system** is defined by a set of states $S = \{S_0, S_1, \ldots, S_n\}$, and by probabilities p_{ij}, $0 \leq i, j \leq n$, that state S_j is directly accessible from state S_i. Often, a specific state $S_I \in S$ is declared as **initial state**. Furthermore, two conditions must hold for the probabilities:

$$0 \leq p_{ij} \leq 1,$$
$$\sum_{j=0}^{n} p_{ij} = 1.$$

Figure 14 shows an example of a discrete Markov system. It corresponds to the example described below. States are represented by circles and non-zero probabilities $p_{ij} > 0$ by directed arrows from S_i to S_j. The corresponding probabilities p_{ij} are attached to the arrows.

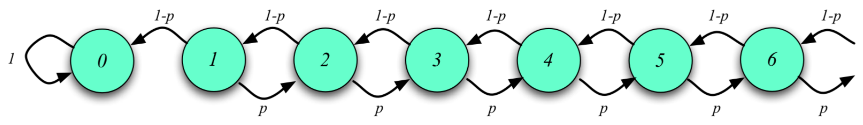


Figure 14: Graph of a Markov system

The initial state S_I in Figure 14 is state 2. Starting from the initial state, a random walk through the graph is performed. If a random variable X_t is used to denote the actual state at time $t = 0, 1, \ldots$, then for $t = 0$ we have

$$\begin{aligned} P(X_0 = S_I) &= 1, \\ P(X_0 = S_i) &= 0, \quad \text{for all } S_i \neq S_I. \end{aligned}$$

If all the probabilities $P(X_t = S_i)$ are known for an arbitrary point in time t, then the probability distribution for the random variable X_{t+1} is given by:

$$P(X_{t+1} = S_j) = \sum_{i=0}^{n} \left(P(X_t = S_i) \cdot p_{ij} \right).$$

Model

Now, let us investigate a concrete example of a discrete Markov system. Suppose that Peter is gambling with an initial amount of m Dollars. At each step, he plays with 1 Dollar. In each game, the probability of winning is the same, say p. When Peter has lost all his money, he stops gambling.

Suppose that Peter's initial amount is 2 Dollars. This situation is shown in Figure 14. Note that state 0 can be considered as the final state, in which Peter has lost all his money. We suppose a probability $p = 0.4$.

The story can be encoded in LPL as follows:

```
MODEL MARKOV "A Markov System";
  SET
    i := / 1:7 /;
    j := / 1:9 /;
  INTEGER VARIABLE
    x{i}  "state at time i";
  ASSUMPTION BINARY VARIABLE
    win{i} PROB 0.4  "probability to win";
  CONSTRAINT
    c1{i|i<#i}: (x[i]>0) -> (win[i] -> (x[i+1] = x[i]+1)) AND
                           (~win[i] -> (x[i+1] = x[i]-1));
    c2{i|i<#i}: (x[i]=0) -> (x[i+1] = 0);
    obs1: x[1] = 2;
...
```

Note that the example is limited to 6 games only. The first implication describes the normal situation where Peter is winning or loosing. The second implication ensures that Peter stops gambling when he has lost all his money. The variables `x[0]` to `x[7]` represent Peter's amount of money at the corresponding point in time. The initial amount `x[0]` is 2 Dollars.

Now, the interesting question is Peter's amount of money after he has played six times. The probability distribution for the variable `x[6]` is obtained by the following query:

```
...
  QUERY
    q1: ABEL_DSP(x[#i] = 0),
        {j} ABEL_DSP(x[#i] = j);
...
```

The following result is obtained:

```
/*-- query q1: --*/
ABEL_DSP(x[7]=9)=0 ,
ABEL_DSP(x[7]=8)=0.004096 ,
ABEL_DSP(x[7]=7)=0 ,
ABEL_DSP(x[7]=6)=0.036864 ,
ABEL_DSP(x[7]=5)=0 ,
ABEL_DSP(x[7]=4)=0.129024 ,
ABEL_DSP(x[7]=3)=0 ,
ABEL_DSP(x[7]=2)=0.193536 ,
ABEL_DSP(x[7]=1)=0 ,
ABEL_DSP(x[7]=0)=0.63648
```

The degree of support that Peter ends up with no money is obtained by the query below. Clearly, this probability is increasing with the number of games:

```
...
  QUERY
    q2: {i} ABEL_DSP(x[i] = 0);
END
```

The following result is obtained:

```
/*-- query q2: --*/
ABEL_DSP(x[7]=0)=0.63648 ,
ABEL_DSP(x[6]=0)=0.5328 ,
ABEL_DSP(x[5]=0)=0.5328 ,
ABEL_DSP(x[4]=0)=0.36 ,
ABEL_DSP(x[3]=0)=0.36 ,
ABEL_DSP(x[2]=0)=0 ,
ABEL_DSP(x[1]=0)=0
```

A good advice for Peter is not to gamble at all, because the probability to end up with no money tends towards 1.

7 Evidential Reasoning

Judging hypotheses is often a process of collecting evidence. In most cases, many sources of evidence are present. The problem then is to combine the evidence obtained from the different sources. Note that evidence can be both confirmative and contradictory. The process of combining evidence from different sources is called **evidential reasoning**. This section discusses different examples of evidential reasoning.

7.1 Web of Trust (trust)

The Description

The increasing importance of computer networks requires secure communication techniques. For that purpose, messages are encrypted. One way of encoding messages is to use **public key** algorithms like RSA [29]. The idea behind public key systems is to define a pair of keys. One key is public, and the other is secret. Everyone can encrypt a message with the public key. Then, the message can only be decrypted with the secret key. The problem using these systems is to get the public key of the receiver. If an other key than the real public key of the receiver is used to encrypt the message, then the owner of the wrong public key can read the message. Therefore, the sender is interested in the authenticity of the public key.

A solution is the **Public Key Infrastructure** (PKI) presented in [25]. The problem is as mentioned above: The sender (called Alice in the rest of the text) has a public key of the receiver (called Bob), but she does not know whether the public key is authentic. Alice also has some public keys of other people, but she is not sure whether they are all authentic or not. Furthermore, she trusts only some of the people. Suppose that Alice also gets certificates and recommendations of the other people she knows. Certificates concern the authenticity of other public keys, and recommendations influence the trust in other people. Based on this information, she wants to decide whether the public key of Bob is authentic or not.

The authenticity of a public key is always relative to a possible sender. This is noted as a predicate $Aut_{A,X}$, where X is the expected owner of the public key, and A is the point of view from which the public key is authentic or not. Another predicate $Cert_{X,Y}$ says that X confirms the authenticity of Y. With certificates, another problem arises: Why should the receiver A of the certificate trust X? For that purpose, a predicate $Trust_{A,X}$ is introduced into the PKI. Finally, the predicate $Rec_{X,Y}$ represents a request of X to trust Y.

The following two rules express the relation between authenticity, certificate, trust, and recommendation. The first rule tells us that a certificate from X for Y implies the authenticity of Y if (1) X is authentic and (2) A trusts X. The second rule is analogous for the relation between recommendation and trust.

$$\begin{aligned}(Aut_{A,X} \wedge Trust_{A,X}) &\rightarrow (Cert_{X,Y} \rightarrow Aut_{A,Y}),\\ (Aut_{A,X} \wedge Trust_{A,X}) &\rightarrow (Rec_{X,Y} \rightarrow Trust_{A,Y}).\end{aligned}$$

Since trust is not a question of yes or no, it should be graded as a value in $[0, 1]$. The graduation of the trust influences all the other values. The problem then is to calculate the authenticity of Bob's public key, which is also a value in $[0, 1]$.

The problem of computing authenticity is illustrated for the simple situation shown in Figure 15. Alice wants to send a confidential message to Bob, but she is not sure about the authenticity of Bob's public key. However, she trusts Dan and Conny, whose public keys are known (but possibly not authentic). Dan and Conny give her both the same public key for Bob. Furthermore, Conny certificates the public key of Dan.

The Model

In order to model the trust and authenticity network shown in Figure 15, the variables for the trust and authenticity of each person, including Alice, have to be defined first:

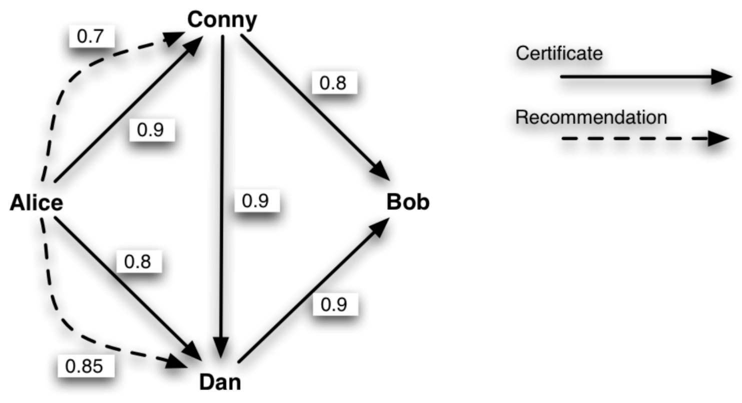


Figure 15: A trust and authenticity network

```
MODEL WebOfTrust "Web of Trust";
  BINARY VARIABLE
    Aa  "Authenticity of Alice (view of Alice)";
    Ta  "Trust of Alice in herself";
    Ac  "Authenticity of Conny (view of Alice)";
    Tc  "Trust of Alice in Conny";
    Ad  "Authenticity of Dan (view of Alice)";
    Td  "Trust of Alice in Dan";
    Ab  "Authenticity of Bob (view of Alice)";
    Tb  "Trust of Alice in Bob";
...
```

Second, the recommendations and certificates are defined as binary assumptions:

```
...
  ASSUMPTION BINARY VARIABLE
    Cac PROB 0.9  "Alice confirms the authenticity of Conny";
    Rac PROB 0.7  "Alice requests to trust Conny";
    Cad PROB 0.8  "Alice confirms the authenticity of Dan";
    Rad PROB 0.85 "Alice requests to trust Dan";
    Ccd PROB 0.9  "Conny confirms the authenticity of Dan";
    Ccb PROB 0.8  "Conny confirms the authenticity of Bob";
    Cdb PROB 0.9  "Dan confirms the authenticity of Bob";
...
```

The relations between the variables can then be written as follows:

```
...
  CONSTRAINT
    c1: Aa AND Ta AND Cac -> Ac;
    c2: Aa AND Ta AND Rac -> Tc;
    c3: Aa AND Ta AND Cad -> Ad;
    c4: Aa AND Ta AND Rad -> Td;
    c5: Ac AND Tc AND Ccd -> Ad;
    c6: Ac AND Tc AND Ccb -> Ab;
    c7: Ad AND Td AND Cdb -> Ab;
...
```

Because Alice knows and trusts herself, the two variables `Aa` and `Ta` are always true:

```
...
    obs1: Aa;
    obs2: Ta;
...
```

Now, Alice's view of the situation is defined and the computation of the support for the authenticity of Bob's public key can begin:

```
...
  QUERY
    q1: ABEL_SP(Ab), ABEL_DSP(Ab);
END
```

The following result is obtained:

```
/*-- query q1: --*/
ABEL_DSP(Ab)=0.8249022 ,
ABEL_SP(Ab)=(
 Cad AND Rad AND Cdb OR
 Cac AND Rac AND Rad AND Ccd AND Cdb OR
 Cac AND Rac AND Ccb)
```

This result tells us that there are 3 possibilities for proving the authenticity of Bob's public key. The corresponding quantitative judgment of the situation is given by the result of the numerical query.

7.2 Information Retrieval (retrieval)

The Description

The information retrieval problem consists in selecting, from a collection of documents, the relevant documents according to a given query. The selection of the documents should

(1) contain all relevant documents and

(2) contain no irrelevant documents.

Satisfying just one of the above criteria is very simple. The first criterion, for example, can be completely satisfied by selecting the entire collection of documents. Similarly, the second criterion is completely satisfied when no documents are returned. Therefore, the information retrieval problem consists in satisfying both criteria at the same time.

As an example consider the collection of documents shown in Figure 16. The collection consists of books about geography and biology. The lower part of the picture shows the relations between the books, the middle part the relations between the terms appearing in the books, and the upper part represents possible queries. For each query, the most relevant books must be selected.

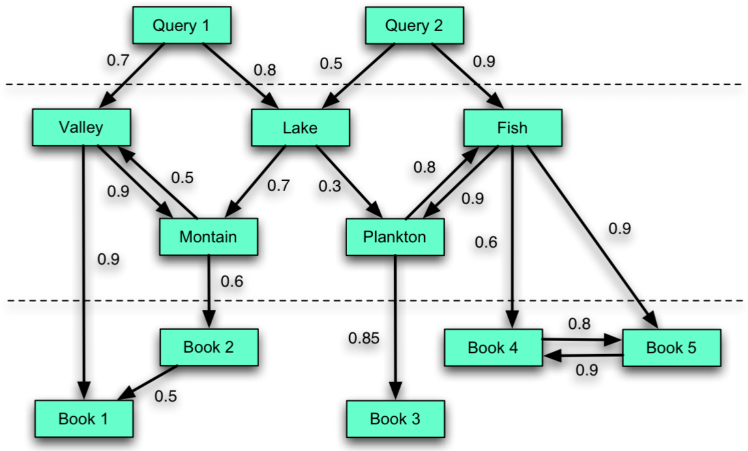


Figure 16: A simple literature database

The Model

The information retrieval problem can be solved by a corresponding LPL model. First, the nodes appearing in the graph of the database are defined as binary variables.

```
MODEL InfRetrieval "Information Retrieval";
  BINARY VARIABLE
    query1     "possible query (yes/no)";
    query2     "possible query (yes/no)";
    valley     "possible term";
    lake       "possible term";
    mountain   "possible term";
    plankton   "possible term";
    fish       "possible term";
    book1      "document of the collection";
    book2      "document of the collection";
    book3      "document of the collection";
    book4      "document of the collection";
    book5      "document of the collection";
...
```

The links between the different nodes of the graph in Figure 16 are of different strength. From the point of view of probabilistic argumentation systems we say that the links are uncertain. Therefore, assumptions are defined for every link with corresponding probabilities as indicated in Figure 16:

```
...
  ASSUMPTION BINARY VARIABLE
    query1_valley  PROB 0.7  "link between query and terms";
    query1_lake    PROB 0.8  "link between query and terms";
    query2_lake    PROB 0.5  "link between query and terms";
    query2_fish    PROB 0.9  "link between query and terms";
    valley_mountain PROB 0.9 "link between terms";
    lake_mountain   PROB 0.7 "link between terms";
    lake_plankton   PROB 0.3 "link between terms";
    mountain_valley PROB 0.5 "link between terms";
    plankton_fish   PROB 0.8 "link between terms";
    fish_plankton   PROB 0.9 "link between terms";
    valley_book1   PROB 0.9  "link between terms and books";
    mountain_book2 PROB 0.9  "link between terms and books";
    plankton_book3 PROB 0.85 "link between terms and books";
    fish_book4     PROB 0.6  "link between terms and books";
    fish_book5     PROB 0.9  "link between terms and books";
    book2_book1 PROB 0.5     "link between books";
    book4_book5 PROB 0.8     "link between books";
    book5_book4 PROB 0.9     "link between books";
...
```

The uncertain relations between the nodes of the graph can now be expressed by corresponding rules:

```
...
  CONSTRAINT
    c01: query1 AND query1_valley -> valley;
    c02: query1 AND query1_lake -> lake;
    c03: query2 AND query2_lake -> lake;
    c04: query2 AND query2_fish -> fish;
    c05: valley AND valley_mountain -> mountain;
    c06: lake AND lake_mountain -> mountain;
    c07: lake AND lake_plankton -> plankton;
    c08: mountain AND mountain_valley -> valley;
    c09: plankton AND plankton_fish -> fish;
    c10: fish AND fish_plankton -> plankton;
    c11: valley AND valley_book1 -> book1;
    c12: mountain AND mountain_book2 -> book2;
    c13: plankton AND plankton_book3 -> book3;
    c14: fish AND fish_book4 -> book4;
    c15: fish AND fish_book5 -> book5;
    c16: book2 AND book2_book1 -> book1;
    c17: book4 AND book4_book5 -> book5;
    c18: book5 AND book5_book4 -> book4;
...
```

To obtain a measure for selecting the most relevant books of the first query, let us compute the following degrees of support:

```
...
  QUERY
    q1: ABEL_DSP(query1 -> book1),
        ABEL_DSP(query1 -> book2),
        ABEL_DSP(query1 -> book3),
        ABEL_DSP(query1 -> book4),
        ABEL_DSP(query1 -> book5);
...
```

This gives the following results:

```
/*-- query q1: --*/
ABEL_DSP(query1->book5)=0.182016 ,
ABEL_DSP(query1->book4)=0.177408 ,
ABEL_DSP(query1->book3)=0.204 ,
ABEL_DSP(query1->book2)=0.75348 ,
ABEL_DSP(query1->book1)=0.777294
```

The first two books are obviously the most relevant documents in the collection. In a similar way we can compute the results for the second query:

```
...
  QUERY
    q2: ABEL_DSP(query2 -> book1),
        ABEL_DSP(query2 -> book2),
        ABEL_DSP(query2 -> book3),
        ABEL_DSP(query2 -> book4),
        ABEL_DSP(query2 -> book5);
END
```

This gives the following results:

```
/*-- query q2: --*/
```

```
ABEL_DSP(query2->book5)=0.864576 ,
ABEL_DSP(query2->book4)=0.842688 ,
ABEL_DSP(query2->book3)=0.712725 ,
ABEL_DSP(query2->book2)=0.315 ,
ABEL_DSP(query2->book1)=0.244125
```

Now, the situation has changed: the first two books are rather irrelevant, while the other three books seem to fit well for the second query.

8 Gaussian Linear Models

Gaussian linear models are a generalization of classical linear regression models known from various statistical domains. For doing assumption-based reasoning, the usual independence and full rank requirements are not needed.

Gaussian linear models represent uncertain linear relationships between variables. These relationships are expressed by equations, which stand for a measurement or an observation, and each of these equations is disturbed by an error term.

More precisely, there is a set of real-valued variables $\Theta = (\Theta_1, \ldots, \Theta_n)$, which span the space $\mathbb{R}^n$, called the *frame of discernment*.

An equation has then the form $\sum_{j=1}^{n} a_j \Theta_j + \Omega = z$, $a_j, z \in \mathbb{R}$, where the value of the real-valued disturbance variable Ω is unknown, but assumed to be normally distributed according to $\mathcal{N}(0, \sigma^2)$. Given the disturbance were $\Omega = \omega$, $\omega \in \mathbb{R}$, and assuming that there is a $j \in \{1, \ldots, n\}$ such that $a_j \neq 0$, every such equation says that the true parameter $\theta^* = (\theta_1^*, \ldots, \theta_n^*)$ has to be in the $(n-1)$-dimensional linear manifold in $\mathbb{R}^n$ defined by the equation. For different assumptions on Ω, these linear manifolds are parallel.

Similarly, when there are several measurements, these linear constraints

$$\sum_{j=1}^{n} a_{ij} \Theta_j + \Omega_i \quad = \quad z_i, \qquad i = 1, \ldots, m, \quad a_{ij}, z_i \in \mathbb{R}$$

have to be satisfied altogether. Then for some assumed value $\omega \in \mathbb{R}^m$ for the disturbance variable $\Omega = (\Omega_1, \ldots, \Omega_m)$, two cases have to be distinguished: For some assumptions, there are no possible values of Θ, whereas for all the other assumptions the possible values are on parallel linear manifolds of some dimension $k \leq m, n$.

The principle of assumption-based reasoning is then to extract the assumptions which do not lead to an empty set of possible values of Θ, the other assumptions are excluded as *inadmissible*. The distribution of the assumptions has then to be conditioned to this event induced into the sample space $\mathbb{R}^m$ of the assumption variable. For the technicalities of this inference refer to [26].

The following examples are modeled in LPL and can be accessed on-line under

```
http://diufpc03.unifr.ch/lpl/gauss.html
```

8.1 Wholesale Price Estimation

This example considers a small causal model for estimating the wholesale price of a car [27].

In this model, there are observations of quantities that influence this wholesale price (like production cost and marketing cost) and quantities that are influenced by the wholesale price (like dealer asking prices). Besides, each observation has an associated Gaussian random term simulating the variation that estimation and profits can have. Then, inferences are made on the wholesale price, i.e. what is the wholesale price of the car, given costs or/and final selling prices asked by dealers.

More precisely, the wholesale price is influenced by the production cost, the marketing cost, and the industry profit. On the other hand, the wholesale price influences the asking prices on the market; the wholesale price is thus estimated on the basis of two dealers' asking prices. In summary, the following variables are used:

Main variable to be inferred on

X**:**	Wholesale price

Quantities influencing the wholesale price

U_1**:**	Production cost
U_2**:**	Marketing cost
U_3**:**	Industry profit

Quantities influenced by the wholesale price

Y_1**:**	Dealer-1 asking price
Y_2**:**	Dealer-2 asking price

Furthermore, there is information about how some of these quantities can be computed, namely the production and marketing costs and the dealer asking prices:

- there are two independent experts' estimations for both the production cost and the marketing cost;
- the mean profit of each dealer over the past few years and its variance are known;
- there is a known mean of the industry profit.

In summary, the following observations will be used:

Estimates by experts for the production cost

I_1**:** Expert 1

I_2**:** Expert 2

Estimates by experts for the marketing cost

J_1**:** Expert 1

J_2**:** Expert 2

Dealers mean profit over past years

Z_1**:** Dealer-1

Z_2**:** Dealer-2

Some of the quantities defined above may have a certain degree of error or imprecision. Their degree of reliability is measured by attributing a Gaussian random variable with each estimation. This induces then the following system of equations:

$$\left\{\begin{array}{rcl} W & = & U_1 + U_2 + U_3 + \Omega_W \\ Y_1 & = & X + Z_1 + \Omega_{Y_1} \\ Y_2 & = & X + Z_2 + \Omega_{Y_2} \\ I_1 & = & U_1 + \Omega_{I_1} \\ I_2 & = & U_1 + \Omega_{I_2} \\ J_1 & = & U_2 + \Omega_{J_1} \\ J_2 & = & U_2 + \Omega_{J_2}, \end{array}\right. \tag{1}$$

where the assumption variables are distributed normally with zero mean and variance as following:

Variable	Ω_W	Ω_{I_1}	Ω_{I_2}	Ω_{J_1}	Ω_{J_2}	Ω_X	Ω_{Y_1}	Ω_{Y_2}
Variance	σ^2_W	$\sigma^2_{I_1}$	$\sigma^2_{I_2}$	$\sigma^2_{J_1}$	$\sigma^2_{J_2}$	σ^2_X	$\sigma^2_{Y_1}$	$\sigma^2_{Y_2}$

This is illustrated in Figure 17 by a directed acyclic graph where a variable in a node is the sum of all the variables in the in-going nodes. In LPL, this can be modeled as follows:

```
MODEL WholesalePriceEstimation;
  SET vendors              "the vendors asked";
      experts              "the experts asked";
  PARAMETER
    I{experts} "production cost estimation";
    J{experts} "marketing cost estimation";
    Z{vendors} "mean profit";
```

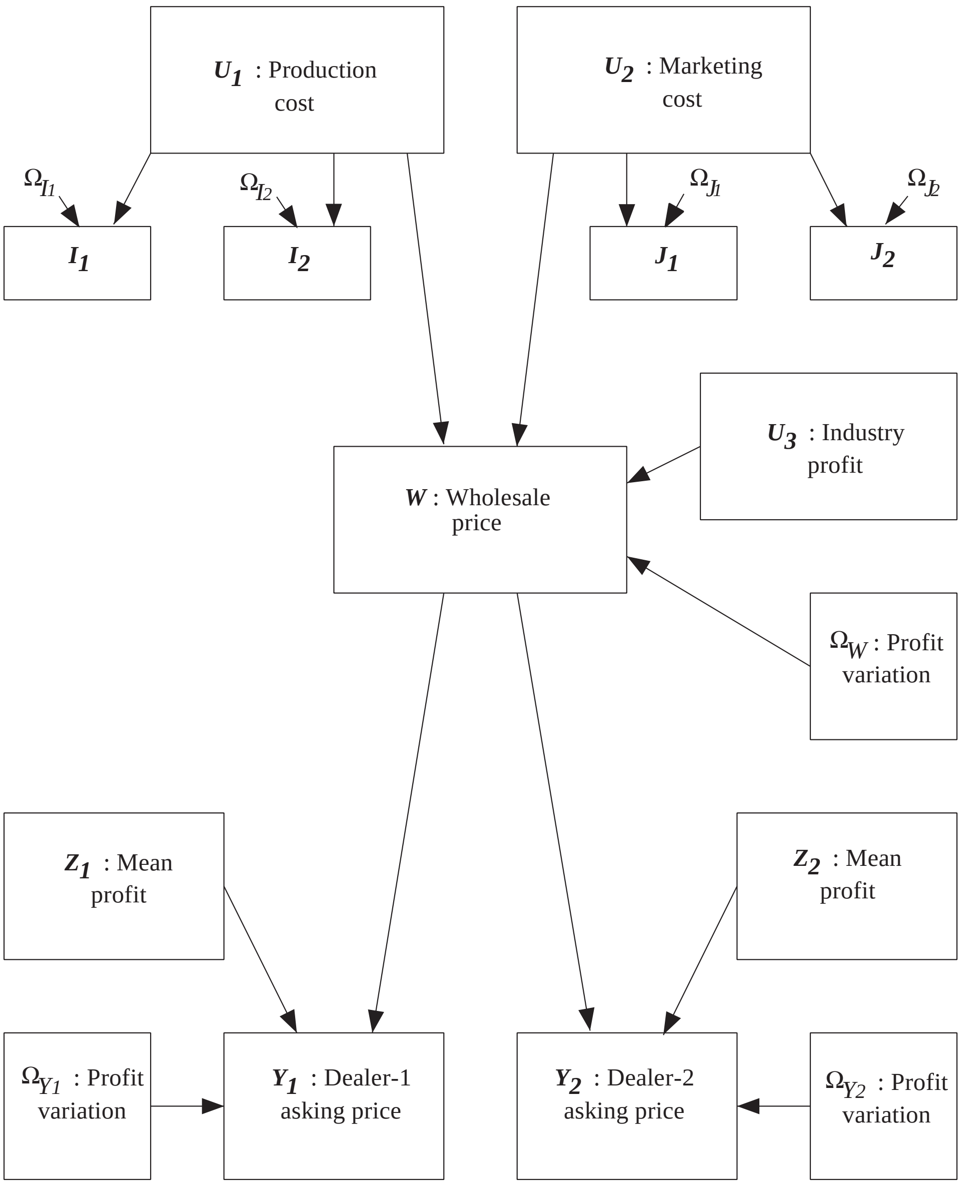


Figure 17: A causal model used for estimating the wholesale price X of a car.

```
    Y{vendors} "asking price";
    U3 "manufacturer profit estimation";
  VARIABLE
    W  "the price to be estimated";
    U1 "production cost";
    U2 "marketing cost";
  ASSUMPTION REAL
    o_i{experts} "production cost estimation variance";
    o_j{experts} "marketing cost estimation variance";
    o_y{vendors} "vendor profit variation";
    o            "manufacturer profit variation";
  CONSTRAINT
    U1estimation{experts}: I = U1 + o_i;
    U2estimation{experts}: J = U2 + o_j;
    PriceEstimation{vendors}: Y = W + Z + o_y;
    Main: W = U1 + U2 + U3 + o;
  MODEL DATA myData "sample data from the book";
  BEGIN
    vendors := /1:2/;
    experts := /1:2/;
    I{experts} :=   [ 5000, 6500 ];
    o_i{experts} := [ 40000, 90000 ];
    J{experts} :=   [ 500, 600 ];
    o_j{experts}:=   [ 2500, 400 ];
    Y{vendors} :=   [ 8000, 10000 ];
    o_y{vendors} := [ 90000, 1000000 ];
    Z{vendors} :=   [ 1000, 1000 ];
    U3 := 1000;
    o := 90000;
END
```

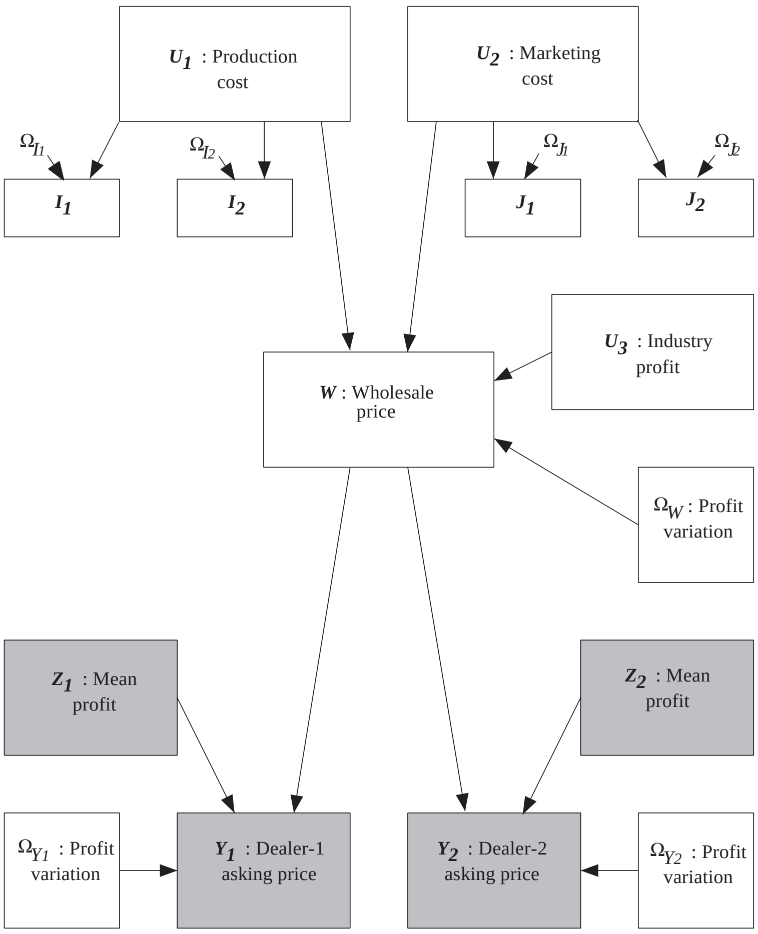
The model described here will be used to infer on the wholesale price W (i.e. the value of W will be estimated in the light of different observations) in three ways: a *diagnostic estimation*, using information observed on variables influencing the wholesale price W; a *predictive estimation*, using information observed on variables which are influenced by the wholesale price W; *combining diagnostic and predictive estimates*, using all the available information.

8.1.1 Diagnostic Estimation

In this first case, there are observations of the variables which are influenced by the wholesale price, thus the name diagnostic. Figure 18 shows the values of the observed variables and the graphical representation where the nodes of the variables which have been observed are shaded. The query that computes the diagnostic estimation of the variable W can be formulated in LPL in the following way:

Variable	Value
Y_1	8000 \$
Y_2	10000 \$
Z_1	1000 \$
Z_2	1000 \$
σ_{Y_1}	1000 \$
σ_{Y_2}	300 \$

(a) Data in the diagnostic problem



(b) Graphical representation

Figure 18: Diagnostic estimation

```
...
  QUERY diagnostic SUBJECT TO PriceEstimation: W ;
...
```

Since the first dealer is less shaky in varying the asking price, the first estimation should be given more importance in the combined estimation. Intuitively, the price estimate should be between the two asking prices, but closer to the value of the second dealer. What about the reliability of this estimation? Since the first dealer is much less reliable than the second, the reliability of the combined estimate cannot be much greater than that of the second dealer. More technically, the Gaussian linear system corresponding to the diagnostic case is

$$\left\{ \begin{array}{rcl} W + \Omega_{Y_1} & = & Y_1 - Z_1 \\ W + \Omega_{Y_2} & = & Y_2 - Z_2. \end{array} \right. \tag{2}$$

Then, the first equation can be subtracted from the second to form the equivalent system

$$\left\{ \begin{array}{rcl} W + \Omega_{Y_1} & = & Y_1 - Z_1 \\ \Omega_{Y_2} - \Omega_{Y_1} & = & (Y_2 - Z_2) - (Y_1 - Z_1). \end{array} \right. \tag{3}$$

This corresponds to a transformation of (2) by the regular matrix

$$B = \left[\begin{array}{cc} 1 & 0 \\ -1 & 1 \end{array} \right].$$

Let

$$\Omega_Y = \left[\begin{array}{c} \Omega_{Y_1} \\ \Omega_{Y_2} \end{array} \right] \sim \mathcal{N}(0, \Sigma_Y), \qquad \Sigma = \left[\begin{array}{cc} \sigma^2_{Y_1} & 0 \\ 0 & \sigma^2_{Y_2} \end{array} \right].$$

Then the transformed disturbance variable $\Xi = B\Omega_Y$ is distributed normally according to $\mathcal{N}(0, B\Sigma B')$. However, since the second component Ξ_2 of Ξ is constant, the distribution can be conditioned to this event. This yields an estimated price

$$\mu(W; Y_1, Y_2, Z_1, Z_2) = \Sigma(W; Y_1, Y_2, Z_1, Z_2) \left(\frac{Y_1 - Z_1}{\sigma^2_{Y_1}} + \frac{Y_2 - Z_2}{\sigma^2_{Y_2}} \right) \tag{4}$$

with

$$\Sigma(W; Y_1, Y_2, Z_1, Z_2) = \left(\frac{1}{\sigma^2_{Y_1}} + \frac{1}{\sigma^2_{Y_2}} \right)^{-1} = \frac{\sigma^2_{Y_1}\sigma^2_{Y_2}}{\sigma^2_{Y_1} + \sigma^2_{Y_2}}. \tag{5}$$

For the given numbers, the LPL output for the diagnostic problem is:

```
+W +o_y[1] = 7165.137614679

var(o_y[1]): 82568.807339449
```

8.1.2 Predictive Estimation

Here we observe the values of the variables which influence the wholesale price, thus the name predictive. Figure 19 shows the values of the observed variables in a graphical representation where the nodes of the variables which have been observed are shaded. The query that computes the predictive estimation of the variable W can be formulated in LPL in the following way:

```
...
  QUERY predictive
    SUBJECT TO U1estimation, U2estimation, Main:
    W;
...
```

This problem is analysed similarly, so the respective estimates for U_1 and U_2 are in this case

$$\mu(U_1; I_1, I_2) = \Sigma(U_1; I_1, I_2)\left(\frac{I_1}{\sigma^2_{I_1}} + \frac{I_2}{\sigma^2_{I_2}}\right), \tag{6}$$

$$\mu(U_2; J_1, J_2) = \Sigma(U_2; J_1, J_2)\left(\frac{J_1}{\sigma^2_{J_1}} + \frac{J_2}{\sigma^2_{J_2}}\right) \tag{7}$$

with

$$\Sigma(U_1; I_1, I_2) = \frac{\sigma^2_{I_1}\sigma^2_{I_2}}{\sigma^2_{I_1} + \sigma^2_{I_2}}, \tag{8}$$

$$\Sigma(U_2; J_1, J_2) = \frac{\sigma^2_{J_1}\sigma^2_{J_2}}{\sigma^2_{J_1} + \sigma^2_{J_2}}. \tag{9}$$

These results can then be used to deal with the Gaussian linear system

$$\left\{\begin{array}{rcl} W - U_1 - U_2 + \Omega_W & = & U_3 \\ U_1 + \Xi_{I_1} & = & \mu(U_1; I_1, I_2) \\ U_2 + \Xi_{U_2} & = & \mu(U_2; J_1, J_2), \end{array}\right. \tag{10}$$

where $\Omega_W \sim \mathcal{N}(O, \sigma^2_W)$, $\Xi_{U_1} \sim \mathcal{N}(O, \Sigma(U_1; I_1, I_2)$ and $\Xi_{U_2} \sim \mathcal{N}(O, \Sigma(U_2; I_1, I_2)$. The result of the predictive estimation is then

$$\mu(W; I_1, I_2, J_1, J_2, U_3) = \mu(U_1; I_1, I_2) + \mu(U_2; J_1, J_2) + U_3, \tag{11}$$

$$\Sigma(W; I_1, I_2, J_1, J_2, U_3) = \Sigma(U_1; I_1, I_2) + \Sigma(U_2; J_1, J_2) + \sigma^2_W. \tag{12}$$

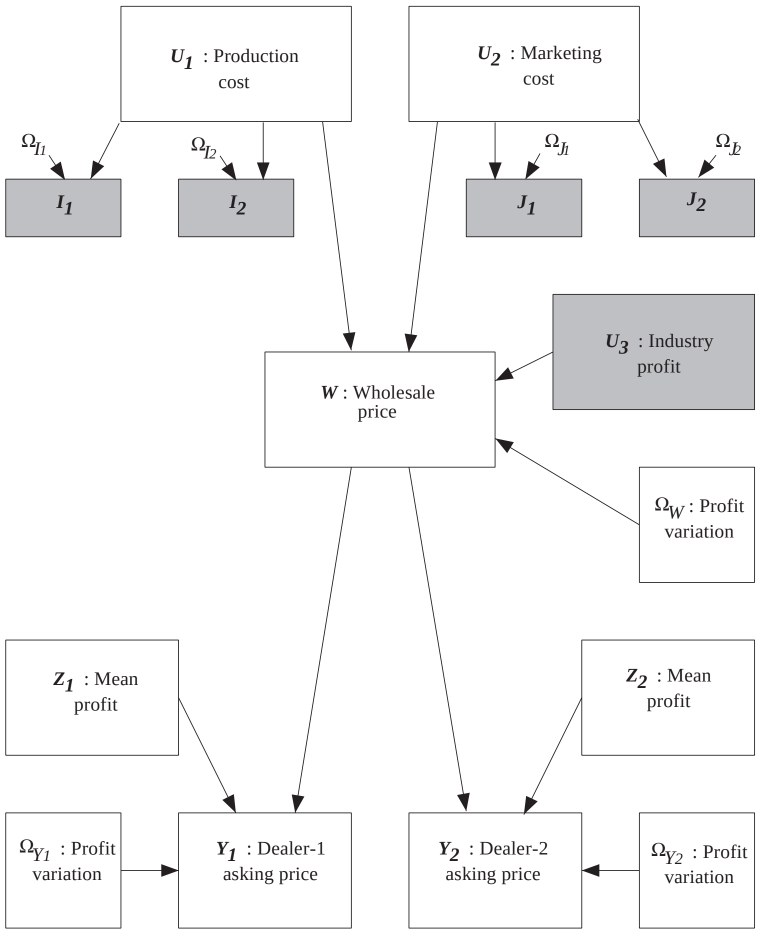
For the given numbers, the LPL output for the predictive problem is:

```
+1 W +o_i[1] = 7047.74535809

var(o_i[1]): 118037.135278515
```

Variable	Value	Variance	Value
U_3	1000 \$	σ^2_W	300 \$
I_1	5000 \$	$\sigma^2_{I_1}$	200 \$
I_2	6500 \$	$\sigma^2_{I_2}$	300 \$
J_1	500 \$	$\sigma^2_{J_1}$	50 \$
J_2	600 \$	$\sigma^2_{J_2}$	20 \$

(a) Data in the predictive estimation problem



(b) Graphical representation

Figure 19: Predictive estimation

8.1.3 Combined Diagnostic and Predictive Estimation

Here, the values of all the variables are observed, of those which influence the wholesale price, as well as of the ones influenced by it. Figure 20 shows a graphical representation where the nodes of the variables which have been observed are shaded. Then, inference can be either made from the whole model or by combining the two submodels, and both methods yield the same result. The former case can be handled by LPL by the following query:

```
...
  QUERY predictiveAndDiagnostic: W;
...
```

In terms of the intermediate results, the combined estimate is

$$\begin{aligned}&\mu(W;Y_1,Y_2,Z_1,Z_2,I_1,I_2,J_1,J_2)\\ =\;&\Sigma(W;Y_1,Y_2,Z_1,Z_2,I_1,I_2,J_1,J_2)\\ &\left(\frac{\mu(W;Y_1,Y_2,Z_1,Z_2)}{\Sigma(W;Y_1,Y_2,Z_1,Z_2)}+\frac{\mu(W;I_1,I_2,J_1,J_2,U_3)}{\Sigma(W;I_1,I_2,J_1,J_2,U_3)}\right)\end{aligned} \tag{13}$$

with

$$\begin{aligned}&\Sigma(W;Y_1,Y_2,Z_1,Z_2,I_1,I_2,J_1,J_2)\\ =\;&\frac{\Sigma(W;Y_1,Y_2,Z_1,Z_2)\Sigma(W;I_1,I_2,J_1,J_2,U_3)}{\Sigma(W;Y_1,Y_2,Z_1,Z_2)+\Sigma(W;I_1,I_2,J_1,J_2,U_3)}\end{aligned} \tag{14}$$

The LPL output looks as follows:

```
-1 W +o_i[1] = -7116.819312185

var(o_i[1]): 48583.73263783
```

8.2 Portfolio Estimation

This model describes a portfolio estimation based on the expected performance of the stocks it is composed of [24].

A financial asset can be characterised by a mean return and a Gaussian error term. A portfolio is a linear combination of asset variables. Then for a given portfolio composition, one can infer on the expected return and on the reliability of this estimation. [24] analyses this problem in terms of linear belief functions. Here, assumption-based reasoning is applied, which leads to the same results. In particular, since the model is Gaussian linear, this leads to a Gaussian hint [26].

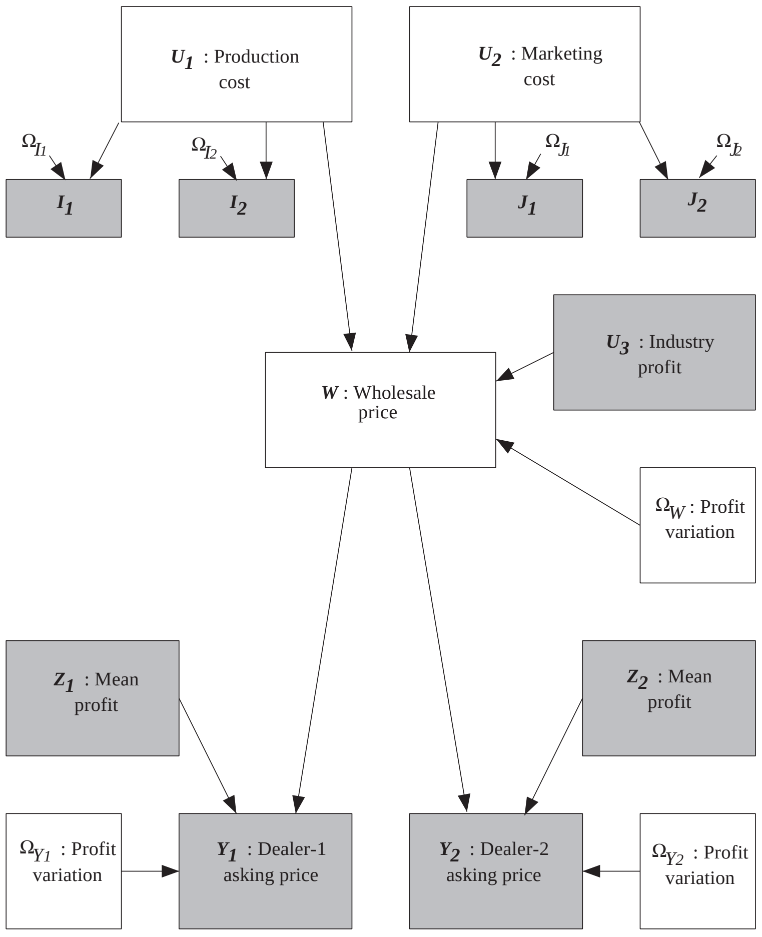


Figure 20: Combined diagnostic and predictive estimation

i	α_i	β_{iG}	β_{iM}	σ_i
1	0.03	0.60	0.40	0.08
2	0.03	0.45	0.25	0.04
3	0.03	0.50	0.30	0.05

i	μ_i	σ_i
G	5%	2%
M	2%	8%

Table 4: Sample data for multiple regression

A portfolio is evaluated using a multifactor regression model for stocks i and factors k,

$$r_i \quad = \quad \alpha + \sum_k \beta_{ik} F_k + \epsilon_i \tag{15}$$

where r_i is the return on stock i, β_{ik} the responsiveness of the stock i to factor k, and the $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ are stochastically independent random components. Furthermore, information available for the individual factors F_k can be given by $F_k = \mu_{F_k} + \Omega_k, \Omega_k \sim \mathcal{N}(0, \sigma_{F_k}^2)$ or just $F_k = \mu_{F_k}$.

Consider the following model from [24]: Given a portfolio consisting of three gold mining stocks S_1, S_2 and S_3. Each stock S_i ($i = 1, \ldots, 3$) is given by a mean α_i and is assumed to be influenced by three factors: the forecast of the change of the market return M, the forecast of the price of gold G, and by a firm specific unknown term $F_i \sim \mathcal{N}(0, \sigma_i^2)$. The modeled percentage of change of the gold price is denoted by μ_G with a tolerance $F_G \sim \mathcal{N}(0, \sigma_G^2)$. Similarly, the relative change of the stock market return is given by μ_M with a tolerance $F_M \sim \mathcal{N}(0, \sigma_M^2)$. The responsiveness of stock i to the gold price is given by β_{iG} and to the stock market return by β_{iM}. This induces the following model:

$$\begin{cases} G &= \mu_G + F_G \\ M &= \mu_M + F_M \\ S_i &= \alpha_i + \beta_{iG} G + \beta_{iM} M + F_i, \qquad i = 1, \ldots, 3. \end{cases} \tag{16}$$

Consider the following scenario from [24]: A central bank is selling a large amount of gold. Based on historical data or personal experience, it can be expected that this transaction negatively impacts the gold price by $\mu_G = 5\%$ on the average. However, the actual rate of change could vary with standard deviation $\sigma_G = 2\%$. Assuming China is joining the WTO, one may speculate that this could boost the stock market by $\mu_M = 10\%$ on the average with a wide spread $\sigma_M = 8\%$. The data of this regression model is summarized in Table 4. Applying assumption-based reasoning leads to the results summarized in Table 5.

$i \setminus j$	μ_i	S_1	S_2	S_3	P
S_1	0.0400	0.0076	0.0007	0.0009	0.0021
S_2	0.0325		0.0021	0.0006	0.0017
S_3	0.0350			0.0032	0.0009
P	0.0343				0.0017

Table 5: The result of the portfolio estimation

```
MODEL PortfolioEstimation "Portfolio Estimation";
   SET  stock;
        factor;
   PARAMETER
     meanReturn{stock};
     responsiveness{stock,factor};
     portfolio{stock};
     meanImpact{factor};

   VARIABLE
     impact{factor};
     stockReturn{stock};
     P;

  ASSUMPTION REAL
     eS{stock};
     eI{factor};

   CONSTRAINT
     I{factor | meanImpact[factor] <> 0}:
         impact + eI = meanImpact;
     S{stock}:  stockReturn =  meanReturn +
          SUM{factor} responsiveness * impact + eS;
     Portfolio: P = sum{stock} portfolio * stockReturn;

   QUERY belP: {stock} stockReturn, P;


   MODEL DATA myData "sample data from the paper";
   BEGIN
     stock :=  / S1, S2, S3 /;
     factor := / G, M /;
     meanReturn{stock} :=  [ 0.03, 0.03, 0.03 ];
     responsiveness{stock,factor} := /
        :   G     M    :
```

```
        S1  0.60  0.40
        S2  0.45  0.25
        S3  0.50  0.30 /;
     eS{stock} := [ 0.0064, 0.0016, 0.0025 ];
     eI{factor} := [ 0.0004,  0.0064 ];
     meanImpact{factor} := [ -0.05, 0.10 ];
     portfolio{stock} := [ 0.2 0.7 0.1 ];

   END
END
```

9 Conclusion and Outlook

The purpose of the present text is to illustrate the wide range of possible applications of probabilistic argumentation systems. The basic idea behind this approach is the concept of reasoning with symbolic and numerical arguments. In fact, this idea turns out to be appropriate and flexible enough for a number of different problems in the domain of reasoning under uncertainty. A corresponding modeling and query language called ABEL has been developed. By extending LPL, the Linear Programming Language invented and developed by T. Hürlimann at the University of Fribourg, the descriptive power of the index mechanism of LPL can be used for easy formulation of these problems from the domain of reasoning under uncertainty. Of course, this is only a first step, there is still a lot of work to do in order to improve the language to the users' needs.

References

[1] B. Anrig. Best next measurements in the framework of assumption-based reasoning. Working paper, Institute of Informatics, University of Fribourg, 1998.

[2] B. Anrig, R. Bissig, R. Haenni, J. Kohlas, and N. Lehmann. Probabilistic Argumentation Systems: Introduction to Assumption-based Modeling with Abel. Technical Report 99-1, Institute of Informatics, University of Fribourg, 1999.

[3] R.E. Barlow and R. Proschan. *Statistical Theory of Reliability and Life Testing*. New York, 1975.

[4] Frank Beichelt. *Zuverlässigkeits- und Instandhaltungstheorie*. B.G. Teubner, Stuttgart, 1993.

[5] R. Davis. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence*, 24:347–410, 1984.

[6] J. De Kleer and B.C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[7] R. Haenni, J. Kohlas, and N. Lehmann. Probabilistic argumentation systems. In J. Kohlas and S. Moral, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 5: Algorithms for Uncertainty and Defeasible Reasoning*, pages 221–287. Kluwer, Dordrecht, 2000.

[8] R. Haenni and N. Lehmann. Belief function propagation based on Shenoy's Fusion Algorithm. In *IPMU'00, Proceedings of the 8th international conference, Madrid, Spain*, pages 1928–1931, 2000.

[9] R. Haenni and N. Lehmann. Implementing belief function computations. *International Journal of Intelligent Systems*, 18(1):31–49, 2003.

[10] R. Haenni and N. Lehmann. Probabilistic argumentation systems: a new perspective on the Dempster-Shafer theory. *International Journal of Intelligent Systems*, 18(1):93–106, 2003.

[11] Tony Hürlimann. Index sets in modeling languages. *Annals of Operations Research*, 99(1–4):427–442, December 2000.

[12] Tony Hürlimann. Reference manual of lpl, version 4.50. Technical report, Department of Informatics, University of Fribourg, 2004.

[13] J. Kohlas. *Zuverlässigkeit und Verfügbarkeit*. Teubner, 1987.

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

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

[16] J. Kohlas, B. Anrig, R. Haenni, and P.A. Monney. Model-based diagnostics and probabilistic assumption-based reasoning. *Artificial Intelligence*, 104:71–106, 1998.

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

[18] J. Kohlas, R. Haenni, and S. Moral. Propositional information systems. *Journal of Logic and Computation*, 9 (5):651–681, 1999.

[19] 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.

[20] J. Kohlas and P.A. Monney. Statistical information and assumption-based inference: Continuous models. Technical Report 04-08, Department of Informatics, University of Fribourg, 2004.

[21] 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.

[22] N. Lehmann. *Argumentation System and Belief Functions*. PhD thesis, Department of Informatics, University of Fribourg, 2001.

[23] N. Lehmann and R. Haenni. An alternative to outward propagation for dempster-shafer belief functions. In A. Hunter and S. Parsons, editors, *European Conference ECSQARU'99, London*, pages 256–267. Lecture Notes in Artificial Intelligence, 1999.

[24] Liping Liu, Catherine Shenoy, and Prakash P. Shenoy. Knowledge representation and integration for portfolio evaluation using linear belief functions. 2003.

[25] U.M. Maurer. Modeling a public-key infrastructure. In *Proc. of the European Symposium on Research in Computer Security ESORICS'96*. Lecture Notes in Computer Science, 1996.

[26] P.-A. Monney. *A Mathematical Theory of Arguments for Statistical Inference*. Contributions to Statistics. Physica-Verlag, 2003.

[27] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[28] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[29] L. R. Rivest, A. Shamir, and M. Adlemann. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[30] D.F. Simonaitus, R.T. Anderson, and M.P. Kaye. Reliability evaluation of a heart assist system. In *Proc. of the 1972 Annual Reliability and Maintainability Symposium, San Francisco*, 1972.