Visual design support in dynamic probabilistic networks for driver modelling

Axel Vogler[1]**, Patrick Rammelt**[1]**, Jörg Herbers**[1] **and Dietmar Neumerkel**[1]

Abstract. Understanding inference in probabilistic networks is an important point in the design phase. Their causal structure and locally defined parameters are intuitive to human experts. The global system induced by the local parameters can lead to results not intended by the human expert. Comprehending the behaviour of dynamic probabilistic networks (DPN) for tuning the model is a time consuming task. Therefore this paper introduces tools supporting the design phase. The application of these tools is shown by means of a DPN for human driver modelling.

1 INTRODUCTION

Probabilistic expert systems are a common tool for modelling under uncertainty. One of their main advantages is the intuitive knowledge representation with meaningful structure and parameters. Domain experts can design a model by directly transferring their specific knowledge to causal links and local conditional probability distributions. While probabilistic networks can be constructed by human experts, learning algorithms are available for estimating parameters and structure from data.

These properties favour the application for human behaviour modelling. Former publications deal with driving action recognition or autonomous driving control, e.g. [1]. The driver modelling application considered in this paper has a different purpose. The intention using dynamic probabilistic models is to recognise in which kind of car the driver feels comfortable. It is investigated whether he feels safe or driving with the specific car is tiring or uncomfortable.

Applications of probabilistic networks often involve the inference from any subset of possible observations to particular variables of interest. Even in small networks, inference from a single observation can lead to surprising outcomes. In particular, if inference does not follow the direction of the network links, the results often do not correspond with human expectations. This is caused by the fact that only the locally modelled relations are intuitive to the expert. In a complex global model, however, the local relations induce global relations that may change with each new piece of evidence. This can lead to conclusions surprising to a user having his local dependency models in mind.

Explaining the result of inference in probabilistic networks is necessary to speed up the design process and to gain acceptance of the final application. Moreover, tools are needed to supply hints where to find relevant parameters for a wrong behaviour of the network.

Understanding the inference series of a dynamic system is even more complex than the conclusion of a static network. The huge amount of information must be suitably reduced. The aim of this work is to help the user to understand and debug specified networks with respect to some hypotheses or test data. It is not intended to change parameters of the network automatically, but to support an expert to validate his knowledge versus a given network. The user should be supported in comprehending how a specific time series leads to a specific inference result.

Available techniques concerning explanation in probabilistic networks include conflict analysis and sensitivity analysis. [2] introduces a conflict measure considering how well an observed case is covered by the network. Tasks in the field of sensitivity analysis examine subsets of total or possible evidence. Questions that might be posed are: what is the crucial set or the minimal sufficient set of findings regarding a hypothesis.

In the design phase, relevant sets found by a sensitivity analysis might not fit to the knowledge of the expert. When debugging probabilistic networks tools are needed not only declaring which variables have high impact but also explaining why another do not. The unexpected behaviour of a network may be induced by more than one design problem. There might be missing edges leading to missing dependency properties of the model. Another source are poorly declared conditional probabilities. There might be just one wrong user input that can even cancel the dependency statement implied by a connecting edge in the graph. This effect is called parametric cancellation.

The next chapter introduces shortly DPNs. Chapter three explains the proposed approach followed by the driver modelling application. The last chapter describes in detail how the required values can be calculated.

2 DYNAMIC PROBABILISTIC NETWORKS

A (static) probabilistic network is defined by a DAG G (directed acyclic graph) consisting of a set of nodes $\boldsymbol{N}$ and directed edges $\boldsymbol{E}$ between these nodes. Further the conditional probability for each node given its parents is needed (for each node state given each parent configuration). The independence relations encoded by the graph allow the joint probability of all variables to be calculated as follows:

$$p(\boldsymbol{N}) = \prod_{V \in \boldsymbol{N}} p(V \mid \mathrm{pa}(V)) \tag{1}$$

A dynamic probabilistic network is a replication of a static network for every time slice. Apart from the intra-slice edges there

[1] DaimlerChrysler AG, Research and Technology, Intelligent Systems, Alt-Moabit 96a, 10559 Berlin, Germany
email: axel.vogler@daimlerchrysler.com

exist inter-slice edges (*temporal edges*) between subsequent time slices, which are the same for any time slice. Only networks with the first-order Markov property are considered. The parents of a node in time slice *t* must be members of time slice *t* or *t-1*. This ensures the time *t+1* slice to be independent of time slices *1,…,t-1* given time slice *t* (Markov property). The conditional probabilities attributed to the temporal edges are assumed to be constant at all times.

In this paper dynamic probabilistic networks are considered as endlessly running systems. The inference at a time slice *t* takes into account only evidence observed so far. This task is also known as *filtering*, while the inference including future observations is called *(backward) smoothing*. The need for an inference result at any time step does not allow the inclusion of future information.

For the first time slice a simplified structure is considered. It is assumed that priori probabilities exist for all nodes connected by temporal edges.

3 PRINCIPLE

Modelling human action is a quite uncertain domain. In general, first approaches of dynamic probabilistic models will not immediately fit real world data. Consequently tools are needed to understand how a resulting series of inferences was achieved. As a motivation for this work a look at a very simple dynamic probabilistic network is helpful. A simple HMM (Hidden Markov Model) like the model shown in figure 1 has only one hidden and one observed node. The hidden node is parent of the observed node in the same time slice and its copy in the next time slice.

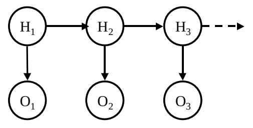

Figure 1. Simple dynamic probabilistic network

The task might be to comprehend how an observed time series leads to wrong inference results. In this simple model it is useful to visualise whether the observations O_t have too strong impact on the hidden node H_t. The hidden node state would be directly determined by the observation at any time step. In the opposite, a too strong temporal influence could prevent the hidden node from changing its state.

This work is based on [3] and [4] who presented an approach to visualise the flow of evidence between the variables along the edges of a static network. First a single hypothesis variable is established. This means the analysis is directed to explain how a single variable is affected at any time step by impacts of evidence.

It is common to start designing a probabilistic network by drawing causal edges between the nodes in the model. Therefore it is straightforward to explain the inference result by showing the strength of the influences along the edges of the network. Starting from the observed nodes, the evidence flow is visualised along the edges up to the hypothesis node. This allows to recognise the type and strength of influence originated by observed nodes. Conflicts and confirmation in different pieces of evidence can be realised at nodes where they combine. Moreover, the blocking and weakening along the graph can be traced.

In a dynamic model the procedure can be applied by unrolling the network over the time and visualising evidence flows along the edges of the graph is possible only for a very short time period. To support a user the influences reaching a node at all times from the different directions will be combined into a chart attached to the nodes. Together with the probabilities over the time this allows to identify time slices with high influence. Further conflicts between the different sources of influences can be recognised.

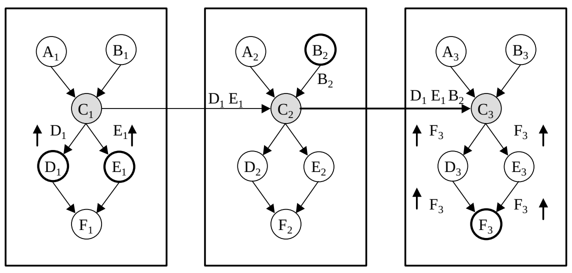

Figure 2. Sample edge d-connecting sets

To explain how the hypothesis node C_t (see figure 2) is affected by evidence (thicker lines) the influences along different edges will be visualised. Evidence flows not affecting the hypothesis node are not shown in the graph. E.g. the influence of D_1 and E_1 reaching A_1 over the hypothesis node C_1 is not considered.

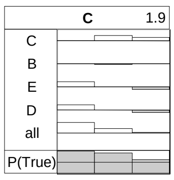

Figure 3. Example node chart

Charts are attached to the nodes indicating the influences reaching the node from different directions over time (starting on the left). The above figure shows the chart of the hypothesis node (C_t) for the example in figure 2. It can be realised that in the first time slice strong positive (above the line) influences from *E* and *D* is apparent. This effect keeps on influencing the node *C* over the time. Furthermore the instantiation of *B* in the second time slice has nearly no consequences on *C*. In the last time slice there is a negative (below the line) joint effect of *D* and *E* on *C*. These two nodes became dependent by observing their common child *F*. Joint effects are shown grey shaded. The maximum value of the weight of evidence (see section 5.3) is indicated in the upper right corner of the box.

4 DRIVER MODELLING APPLICATION

The example used in this paper to show network visualisation is a driver modelling application. A test group was asked for their preference on two different cars. The car handling was experienced by means of a driving simulator.

The driving simulator consists of a dome with a complete car mounted inside. In front of the car a 180 degrees image projection system shows the traffic scene in front of the car. By means of hydraulic actuators the whole dome is accelerated and moved according to the car's driving dynamics. The properties of the different cars were supplied by different models of the car dynamics.

The testers' task was to drive on a highway and to carry out a number of overtaking manoeuvres with each of the two cars. After the experiment the participants were asked for their preference concerning the two cars. The example network in this paper models the final preference as a function of physical parameters which were recorded during the tests. The network presented is not intended to solve the driver modelling task. However, it illustrates the benefits of visualisation tools in order to design dynamic networks.

The example network models the driver's feeling of comfort while handling the car. It is based on analysis of overtaking manoeuvres. The idea behind the model is that a content driver shows a different behaviour in handling the car compared to a driver having problems in controlling the dynamics of the car. A test driver feeling comfortable with the car tends to execute precise or smooth manoeuvres. A driver stressed by the car's behaviour might need more steering activity to position the car in a new lane.

It should be remarked that the whole driver action highly depends on the driver's experience and personal driving style. Furthermore the manoeuvres of a specific driver have a high variety. For that reason a human driver model should allow for extreme values in other directions.

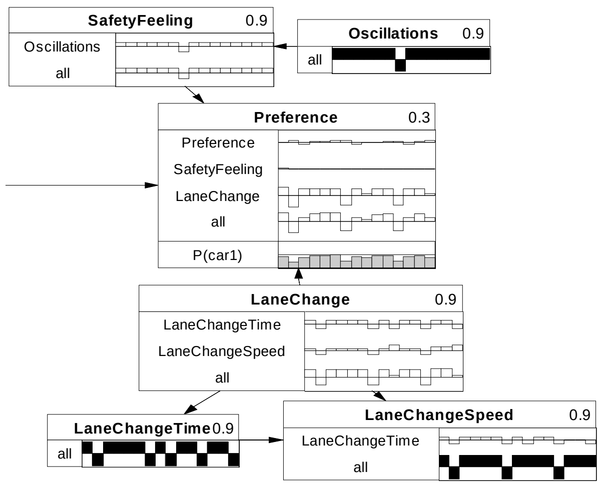

Figure 4. DPN incorrectly predicting driver's preference of car 1

While a wider group of drivers are modelled correctly by the first designed network, the network did not correctly predict the preference of some drivers. In figure 4 the probability of the hypothesis node *Preference* (of car one) oscillates up and down. This is caused by the changing impacts from node *LaneChange*, which nearly corresponds with the whole impact *Preference* receives. The whole impact is shown in the node box marked with "*all*". The *SafetyFeeling* and the "temporal edge" have no impact. Most observations favour correctly the hypothesis (driver prefers car 1). Instantiated nodes are marked with black boxes. A box over the dividing line is a positive observation, while a box below the line is negative.

From these hints the networks parameters were adapted. The conditional probabilities attached to the temporal edge were strengthened to keep the node from changing its state too fast. Figure 5 shows a higher impact from *LaneChange* on *Preference* in the first time steps. In later time steps the negative impact is quite low. This is achieved by adapting the prior probability and the conditional probability attached to the temporal edge.

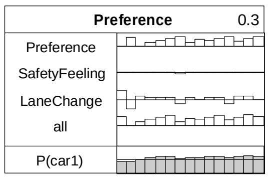

Figure 5. DPN correctly predicting driver's preference of car 1 (the remaining are unchanged, see figure 4)

5 ALGORITHMIC DETAILS

Most inference algorithm for static probabilistic networks perform computations efficiently in a secondary structure called junction or join tree (e.g. [5]). These structures for efficient inference can be constructed in advance. For exact inference in dynamic networks an online junction tree construction for each time window could be performed (e.g. [6]). Unfortunately, this approach is quite time consuming.

In [7] it is shown that the junction tree of an unrolled DPN has a unique backbone and a cyclic structure. Due to this fact, large junction trees are constructed in advance. The parts of the junction tree corresponding to the first and last time slice differ. Because of the cyclic structure spare time slices can be cut and the edges of the junction tree redirected to the part of the tree representing the last time slice.

When considering an "endlessly" running system, like a driver model, the junction trees can not be constructed in advance. Another important point is the need for an inference result at any time step. This does not allow to take into account future evidence. The inference task is reduced compared to a full forward backward calculation.

5.1 Time sliced junction tree

Following the ideas in [7] a generic junction tree is constructed usable for every time slice except the first. The first time slice needs some special care because of the missing nodes.

The generic time slice junction tree is constructed from the nodes of the first two time slices. First the interface nodes (nodes with children in the next time slice) are fully connected. These connections would be filled in by eliminating the future time slice. This is shown in [6] following from properties in [8]. Next, all nodes of the time slice are eliminated - children before their parents. The junction tree construction starts with the clique resulting from the first node elimination. Every clique is connected to a clique eliminated later. The clique is selected as the neighbour which contains all member nodes except the node which created the clique by its elimination.

This results in a junction tree preserving as much as possible from the structure of the original graph. Depending on the structure of the graph the junction tree is larger compared to the HUGIN-approach in [5]. The junction tree can be further condensed by contracting cliques containing all member nodes of other cliques. An appropriate algorithm can be found in [7].

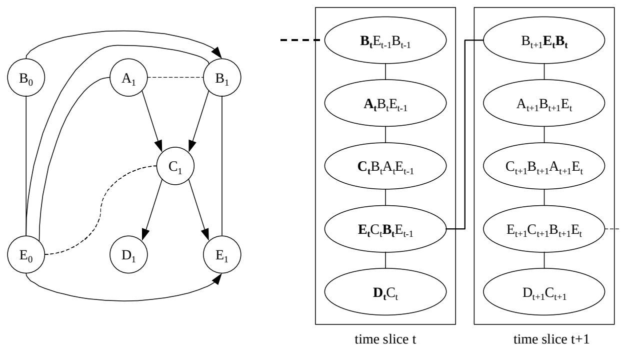

Figure 6. Example network and its junction tree. The elimination order is D_t, E_t, C_t, A_t, B_t. Dashed lines are moral edges, solid lines are created by triangulation.

Inference in a single time slice is divided into three phases. The first phase is a *distribute-evidence* phase starting from the last time slice forward interface clique without evidence. This is needed to achieve a-priori probabilities for the current time slice. Further it is used for evidence retraction.

For inference and weights of evidence calculation two phases of *lazy propagation* with evidence entered are carried out. A detailed description of lazy propagation can be found in [9]. In the collect-evidence phase the last time slice forward interface clique collects messages from all cliques belonging to the current time slice. In the *distribute-evidence* phase messages from this clique are sent to all cliques in the current time slice.

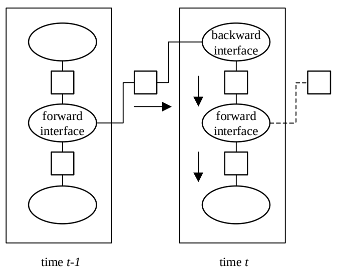

Figure 7. Distribute-evidence in a generic time slice junction tree

The aim of the lazy propagation algorithm is to maintain multiplicative decomposition of potentials and to postpone their combination. Opposite to the HUGIN propagation, the cliques are not initialised with joint potentials over their member nodes. The conditional probability tables of each node are assigned to the clique created by the node's elimination. A further important point is that like in the *Shafer-Shenoy-Architecture* (e.g. [10]) messages sent from one clique to the other are not considered when generating the opposite direction message.

5.2 Absorption

The creation and integration of a message from a clique C_i to another clique C_j is called absorption (C_j absorbs from C_i). In order to build a message first all potentials obtained from neighbouring separators are collected except for the message-receiving separator. When marginalising down to the separator potential, first *barren* nodes are eliminated. A variable is said to be barren if it is not an evidence variable and has only barren descendants. These variables do not affect the posterior probability of other variables. When marginalised first their probability sums to one.

Evidence nodes are instantiated before combination. Evidence on a node is only entered by reducing the potential dimension if the observed node is not member of the receiving clique. For later analysis evidence nodes and combined conditional probabilities are noted with every potential.

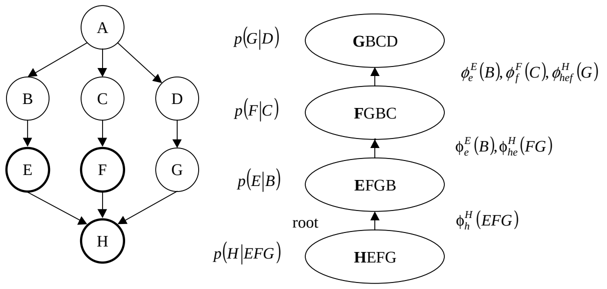

Figure 8. Example part of a collect-evidence phase

Together with the structure of the junction tree this propagation method allows for tracing the combination and decomposition of potentials. Barren variables do not influence other nodes and their potentials are not propagated through the junction tree. If a node with multiple parents is observed (like H in figure 8) an instantiated potential over its parents is created. The figure only shows the first part of the collect evidence phase. On the left of the junction tree the conditional probabilities attached to the cliques are displayed and on the right the potentials sent. Messages received from a neighbour clique are not considered when sending a message. Potentials are marked with their combined conditional probabilities as superscript and evidence nodes as subscripts. The instantiation of E at the next clique reduces the dimension of the message sent by H and creates a new potential over B. The same holds for the instantiation of F at the following clique. These three effects can be decomposed reaching C. If F were not observed the conditional probability of F and the potential over F and G would have been combined. Then only two effects – one over B and a joint effect over C and D – could be decomposed reaching C.

5.3 Weight of evidence

The most basic definition needed is a measure of influence. Using weights of evidence for measuring influence is motivated by a remark in [11]. It is mentioned that in analogy to the decibel in acoustics a relation between units perceptible to human judgement and weights of evidence exists. The weight of evidence will be used for measuring the impact of evidence reaching a node along its edges. This allows to recognise the origin and the kind of effects arriving from different directions. The weight of evidence influencing a node X over an edge e is calculated by

$$W(X : E_e) = \log \frac{P(E_e | X = x_{\text{supp}})}{P(E_e | X = x_{\neg\text{supp}})} \qquad (2)$$

E_e is the evidence reaching node X over edge e. The impact is measured according to the state x_{supp} of node X supporting the hypothesis H. It is important to consider the hypothesis in every edge flow. This allows a user to distinguish positive and negative impacts for the node of interest all over the graph. Further it supports recognition of conflicting evidence regarding the hypothesis at a first glance. Different to others approaches no explicit graph traversing is necessary to inspect evidences reaching a node along the individual edges. Favoured by the structure of the junction tree the influencing evidence can be analysed in the clique created by the node elimination. Graph traversal is just used to exclude flows not reaching the hypothesis node.

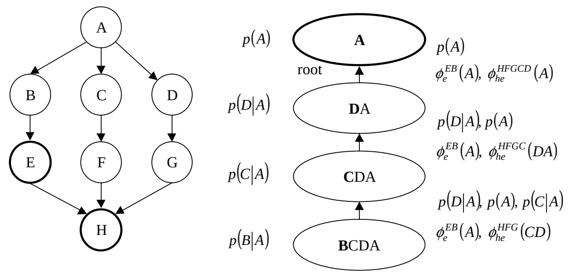
A
B
C
D
E
F
G
H
$p(A)$
A
$p(A)$
root
$\phi_e^{EB}(A),\ \phi_{he}^{HFGCD}(A)$
$p(D|A)$
DA
$p(D|A), p(A)$
$\phi_e^{EB}(A),\ \phi_{he}^{HFGC}(DA)$
$p(C|A)$
CDA
$p(D|A), p(A), p(C|A)$
$\phi_e^{EB}(A),\ \phi_{he}^{HFG}(CD)$
$p(B|A)$
BCDA

Figure 9. Example part of a junction tree with messages

To calculate the flow to a node *X* from one of its children the potentials sent to its home clique are analysed. The home clique of *X* is the clique created by the elimination of *X*. First, potentials are selected based on a combination of the child's conditional probability (superscript tags) with other potentials. By the construction of the junction tree the child is not a member of its parent's clique. It was eliminated prior to its parent and therefore it is no member of any subsequent clique. Next, all potentials sent to the clique are added containing nodes that are members of already selected potentials. The conditional probability of *X* given its parents located at the clique is appended together with the prior probability of the parents not included in one of the potentials before. These potentials are combined and marginalised down to *X* to get the joint probability of *X* and the evidence the child is involved in. From this joint probability it is straightforward to calculate the conditional probability needed to compute weights of evidence. The weight of evidence originating from parents are calculated analogously. Instead of inspecting the conditional probabilities combined into a potential (superscript tags) its current member nodes are considered.

Figure 9 shows a probabilistic network with a part of its uncondensed junction tree and the messages of lazy propagation. The upper messages were created by distribute-evidence and the messages below by collect-evidence. The missing part of the junction tree can be found in figure 8. When applying the above algorithm two influences on node *A* will be identified – the impact of instantiating *E* reaching *A* over *B* and a joint effect over *C* and *D* originating from the instantiation of *H.*

$$\begin{aligned} \phi\left(A, e_{\bar{A}}\right) &= \phi_e^{EB}(A)\cdot p(A) \\ \phi\left(A, e_{\bar{C}}\right) = \phi\left(A, e_{\bar{D}}\right) &= \phi_{he}^{HFGCD}(A)\cdot p(A) \end{aligned} \tag{3}$$

It is important to note that effects on a node cannot always be decomposed and assigned to individual variables, e.g. the effect of two children with an instantiated common child on their parent. The method described above automatically returns an existing joint or an individual effect. The kind of influence can be easily recognised and marked with a different colour or grey shading from the received potentials at the clique.

6 CONCLUSIONS

Visualising the effects reaching a node along its different edges in a graph can help to understand how a variable of interest is influenced over the time. It allows for recognising the type and strength of influence originated by observed nodes. Moreover, blocking and weakening along the graph can be traced. Strong temporal edges determining the state of other nodes over multiple time steps can be identified as well as weak time links allowing a switching behaviour of the affected nodes.

The propagation algorithm described supports the access to partial evidence necessary for the explanation tools without the generation of additional messages or variable propagation. Future research will be directed to a more detailed analysis of the enlarged junction tree depending on the structure of the graph compared to the savings gained by neglecting d-separated evidence and barren nodes. Another point of interest is the use of approximate methods. For visualisation purpose a user needs to discover differences in influences rather than exact values. Here a wide range of algorithms is available. Another direction would be to exploit the knowledge of low influence values for faster approximate inference.

References

[1] J. Forbes, N. Oza, R. Parr and S. Russell: *Feasibility study of fully automated traffic using decision-theoretic control*, California PATH Research Report UCB-ITS-PRR-97-18, Institute of Transportation Studies, University of California, Berkeley, 1997

[2] F.V. Jensen, S.H. Aldenryd, K.B. Jensen: *Sensitivity analysis in Bayesian networks*, Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Froidevaux, C. & Kohlas, J. (eds.), Springer Lecture Notes in Artificial intelligence 946, Springer Verlag, Berlin, 243-250, 1995

[3] D. Madigan, K. Mosurski, R.G. Almond: *Graphical explanation in belief networks*, Journal of Computational and Graphical Statistics, **6**, 2, 160-181, 1997

[4] A. Vogler: *Visual design support for probabilistic network application,* in Proceedings of the seventh international Workshop on Artificial Intelligence and Statistics (Uncertainty 99), Morgan Kaufmann, 309-314, 1999

[5] F.V. Jensen, S.L Lauritzen., K.G. Oleson: *Bayesian updating in causal probabilistic networks by local computations*, Computational Statistics Quarterly 4, 269-282, 1990

[6] U. Kjærulff: *A computional scheme for reasoning in dynamic bayesian networks,* in Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, California, 121-129, 1992

[7] G. Zweig: *Speech Recognition with dynamic Bayesian networks,* PhD dissertation, University of California, Berkeley, 1998

[8] D.J. Rose, R.E. Tarjan, G.S. Lueker: *Algorithmic aspects of vertex elimation on graph,* SIAM Journal on Computing, **5**, 266-283, 1976

[9] A.L. Madsen, F.V. Jensen: *Lazy Propagation: A Junction Tree Inference Algorithm based on Lazy Evaluation,* Artificial Intelligence, 113 (1-2), 203-245, 1999.

[10] P. Shenoy, J. Kohlas: *Computation in Valuation Algebras*, in and S. Moral (eds.), Handbook of Defeasible Reasoning and Uncertainty Management Systems, Kluwer Academic Press, in press, 2000

[11] I.J. Good: *Weights of evidence: a brief survey,* Bayesian Statistics 2: Proceedings of the Second Valencia International Meeting, 1983, J.M. Bernado, M.H. DeGroot, D.V. Lindley and A.F.M. Smith (eds.). New York: North Holland, 249-269, 1985