A system for handling relational dependencies in approximate reasoning

Philippe Chatalic, Didier Dubois, Henri Prade

Third
INTERNATIONAL
EXPERT
SYSTEMS
Conference
London
2 – 4 June 1987

Sponsored by
Expert Systems
The International Journal of Knowledge Engineering

Organised by Learned Information (Europe) Ltd
Publishers of *Expert Systems Journal*,
Expert Systems/AI In Business, Online Review,
Electronic and Optical Publishing Review,
The Electronic Library, Monitor and
Information World Review

Learned Information Oxford and New Jersey

A SYSTEM FOR HANDLING RELATIONAL DEPENDENCIES IN APPROXIMATE REASONING

P. Chatalic, D. Dubois, H. Prade,
Université Paul Sabatier, France

Keywords : *deductive reasoning ; dependency graph ; evaluation tree ; uncertainty.*

Abstract : The graph of dependencies (or relationships) between numerical or logical variables which can be built from the expert knowledge available in a given domain (generally expressed under the form of rules) may be extremely intricate. It is usually far from a tree-like structure and exhibits cycles. In order to make the best out of the expert knowledge in a given situation it may be dangerous to use the knowledge in a too granular way. A procedure for building an evaluation tree from the expert knowledge is proposed. It is worth noticing that this tree is determined by the variables, which are to be evaluated with respect to a user query. Moreover, due to logical or functional dependencies, some variables must be conjointly evaluated by means of subsets of rules which cannot be processed separately in order to completely take advantage of the expert knowledge. This method is implemented in the MIRACLE system which is running on a micro-computer. Besides, it is capable of dealing with uncertainty. The uncertainty is encoded under the form of Shafer belief functions or possibility measures. However the approach is not specific of a particular treatment of uncertainty.

1 INTRODUCTION

Traditionally, the inference engine of an expert system processes rules in a rigidly oriented manner, by selecting a set of rules enabled by matching their condition parts with available facts, triggering these rules one at time so as to produce new facts (or delete old ones). Results obtained from independent rules must be combined, especially when uncertainty is present. New facts may be used to trigger other rules and so on. This methodology is rather efficient because rules are oriented, and because reasoning reduces to a totally distributed process. However there are cases when no results are found where some should be ; and a totally distributed technique is not very well adapted to the treatment of uncertainty.

Contrastedly with such local strategies, global approaches have been proposed which turn inference into a mathematical programming problem (Nilsson, 1986, Zadeh, 1979), or into a large-scale combination problem in which all pieces of information are processed once (Chatalic et al, 1986). These methods theoretically give exact results but are computationally untractable. The idea is then to decompose a knowledge base into several sub-bases where only a global reasoning technique applies. These sub-bases form the nodes of a network where information flows, computations being locally performed on the nodes. This type of idea has been developed by Pearl (1986) for Bayesian networks, and Shafer et al (1986) using belief functions for the modelling of uncertainty.

In this paper we propose a technique which decomposes a knowledge base into locally independent sub-bases, by analyzing the dependency graph between elemen-

tary or complex facts. A query-driven reasoning technique is described, which turns the dependency graph into a rooted evaluation tree of local computations which proceed from the leaves to the root. The evaluation tree is independent of the treatment of uncertainty involved in the local computations.

2 FROM A KNOWLEDGE BASE TO A DEPENDENCY GRAPH

One of the advantages of classical logic-based knowledge representation techniques is to model any piece of information in the same format. However classical logic does not account for the imperfection of expert knowledge. Here we try to have a knowledge representation setting which remains in the spirit of propositional calculus, while providing room for imprecision and uncertainty.

2.1 Knowledge representation conventions

The basic primitive which expresses an elementary fact is of the form of a list (x A) which means "The value of x is in A", where x is an ill-located object in some set X and A is a subset of X. X is a referential set which can be a numerical scale (size, temperature, ...) or a finite set of alternatives (e.g. {blue, yellow, ...} for a color). x is called a variable, and stands for the attribute of a given object (e.g. the size of John ..., the color of car 1). It can be a Boolean variable expressing an event which occurs or not (then X = {true, false}). Our concept of variable is closer to that of a random variable rather than a variable in predicate calculus. A acts as a constraint limiting the values of x.

The piece of information (x A) is said to be precise if and only if A is a singleton of X, otherwise it is said to be imprecise (i.e. x is not completely specified). A negative fact "not (x,A)" is equivalent to $(x,\bar{A})$ where $\bar{A}$ is the complement of A (in X), and is thus captured by this technique. Hence (x,A) is equivalent to a literal in propositional logic, A standing for a predicate, and x for a constant.

A composite fact is made up of elementary facts which are related by means of standard connectives of logic. A composite fact can be expressed under the generic format (x,A) where x is a composite variable $(x_1,...,x_n)$ and A is a set-theoretic combination of elementary constraints $A_1,...,A_n$ expressing a relation between the variables $x_1,...,x_n$. For instance a composite fact of the form (x A) and (y B) is equivalent to ((xy)(A×B)) where × expresses a Cartesian product, while (x A) or (y B) translates into ((x,y),A+B) with $\overline{A+B} = A \times B$. A production rule if $(x_1 A_1)$ and ... and $(x_n A_n)$ then (y B), is interpreted in the tradition of logic as a composite fact (not $(x_1 A_1)$ or not $(x_2 A_2)$ ... or not $(x_n A_n)$ or (y B)), which translates into $((x_1...x_n y),\bar{A}_1+\bar{A}_2+...+\bar{A}_n+B)$. However rules of the form if $(x_1 A_1)$ and ... and $(x_n A_n)$ then $(y_1$ B) or ... or $(y_n B_n)$ are also allowed. Generally, any composite fact can be put under a disjunctive form $((x_1,x_2...x_n)A_1+A_2+...+A_n)$ or can be decomposed into a conjunction of disjunctive facts, which are treated separately. A disjunctive fact can be viewed as a relation between variables.

An uncertain fact is of the form (F U) where F, called the content, is an elementary or a disjunctive fact, and U is a list which expresses the uncertainty of the fact. It can be a degree of probability, or a pair (belief, plausibility) in the sense of Shafer (1976) or a pair (necessity, possibility) as in possibility theory (Zadeh (1978), Dubois-Prade (1985)). Note that F cannot involve conjunctions of more elementary facts because, under uncertainty, the decomposition of conjunctions is no longer allowed. Uncertain facts must be viewed as conjectures on particular objects rather than general default rules about classes of objects, as in default logic (Reiter, 1980).

2.2 The dependency graph

These knowledge representation principles require a set of variables to be exhibited, together with constraints restricting their ranges (elementary facts) and relations linking them (disjunctive facts). Hence the knowledge base can be viewed as a network whose vertices are the facts and the variables and edges link the variables to the facts involving them. More formally let $\mathcal{V} = \{x_1 ... x_n\}$ be the set of variables, $\mathcal{E} = \{E_1 ... E_p\}$ be the set of elementary facts (involving only one variable, and $\mathcal{D} = \{D_1 ... D_q\}$ be the set of disjunctive facts (expressing rules). The dependency graph is $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, a non-directed graph whose vertices form the set $\mathcal{N} = \mathcal{V} \cup \mathcal{E} \cup \mathcal{D}$, and edges are of the form (x_i,E_i) if and only if $E_i = (x_i,A_i)$ (x_i,D_i) if and only if x_i appears in D_i.

An example of dependency graph is given below (without the elementary facts)

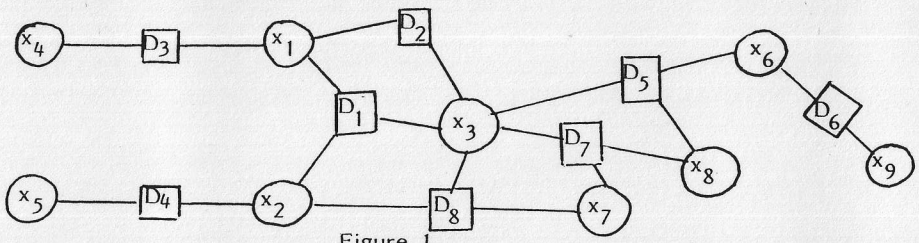

Figure 1

It is important to notice that the non-directedness of the graph enables pieces of information to be propagated in various directions, contrary to usual production systems , which cannot deal with modus tollens-like inferences. This type of graph is a particular case of relational graphs studied by Shafer et al (1986) and Kong (1986), but where relations can be decomposed as a disjunction of elementary constraints on single variables.

3 REASONING ON THE DEPENDENCY GRAPH

Querying a knowledge base is viewed as asking for the value of a variable x, or checking for a given value of this variable. It always comes down to evaluating the value of the concerned variable. This evaluation is obtained by a conjunctive combination of all constraints acting on the variable. These constraints are of two kinds :

- Elementary facts pertaining to x, i.e. facts of the form (x,A) ; it is called a direct justification of x.
- Indirect constraints induced from other variables which relate to x via disjunctions, provided that these variables have been evaluated in turn. Any disjunction D where x is involved is thus called an indirect justification of x. The influence of D on x is computed by projecting the relation, obtained by combining D and the justifications of the other variables, on the reference set of x.

3.1 Problems in evaluating

Direct justifications clearly act on variables independently of one another. However indirect justifications are often mutually dependent, as soon as the corresponding disjunctive facts involve common variables. In that case, indirect justifications cannot be processed separately. Let us consider two examples :

1st example : Consider the two rules and the following fact :
D_1 : if (x,A) then (y,B)
D_2 : if (x,C) then (y,D)
E_1 : (x,A∪C) (where ∪ expresses union)

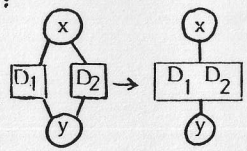

Fig. 2

Combining E_1 and D_1 separately clearly produces no information about y nor combining E_1 with D_2. In terms of production systems, no rule can be triggered by E_1. Contrastedly, combining the three items leads to the expected conclusion (y B∪D) since $B \cup D = \text{Proj}_Y((\bar{A}+B) \cap (\bar{C}+D) \cap (A \cup C))$ where ∩ expresses intersection and Proj_Y a projection on Y. The necessary fusion of D_1 and D_2 is symbolized by the transformation of the dependency graph in fig. 2.

2nd example : It is dual of the first one, namely
D_1 : if (x,$\overline{A}$) and (y,$\overline{B}$) then (z,C)
D_2 : if (x,A) and (y,$\overline{B}$) then (t,D)
E_1 : (z,$\overline{C}$), E_2 : (x,A).
Suppose we wish to evaluate t. We must evaluate x and y due to D_2. But evaluating x requires y and z be evaluated and evaluating y requires x and z be evaluated. The way out of this tricky situation is to conjointly evaluate x and y from the knowledge of z, and then use conjointly this information to compute t using D_2. The reader can check that from E_1 and E_2 we can come up with the conclusion (t,D). Such an inference is generally not possible with usual production systems. Note that the dependency graph is modified to express that x and y must be conjointly evaluated when the query is about t. The directedness of the new graph expresses an ordering on the reasoning steps (from z to t).

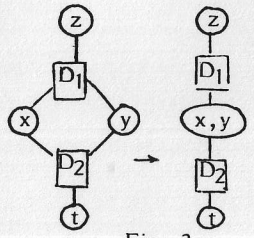

Fig. 3

Such situations will be found with a higher level of complexity in general dependency graphs, depending on the number of paths relating the variables. Especially any cycle in the dependency graph creates evaluation problems and prevents a completely distributed procedure for evaluation. Contrastedly, if the dependency graph is a tree then computing justifications in a separate way is allowed. This result becomes clear by formalizing a concept of logical dependency.

3.2 Notion of logical dependency

Clearly, if two variables x and y belong to two disjoint parts of the dependency graph then constraining x does not influence the value of y and conversely. Similarly two disjunctive facts belonging to disjoint parts of $(\mathcal{N},\mathcal{A})$ influence distinct sets of variables. More generally two vertices $N \in \mathcal{N}$ and $N' \in \mathcal{N}$ will be called independent if and only if there is no path between N and N'.

A weaker notion is that of conditional independence, namely N and N' will be said to be independent given N" if and only if all paths between N and N' contain N". In other words deleting N" breaks the dependency graph in (at least) two disjoint parts. It means that the only dependency between N and N' goes through N". Hence, if N" is the very variable to be evaluated, and N and N' are variables, the influence of N" on these variables is not considered, and they can be separately evaluated. Similarly, if N and N' are disjunctions, they need not be fused because they only relate through variable N". As a consequence if $\mathcal{G}$ is a tree, no fusion of disjunctive facts is necessary, and variables can be separately evaluated. These notions of conditional independence and absolute independence are purely logical and are very close to the notions developed by Shafer et al (1986) in qualitative Markov trees ; but they are weaker than probabilistic independence concepts as used by Pearl (1986).

4 BUILDING AN EVALUATION TREE

As mentioned earlier, evaluating a variable comes down to combining direct justifications with indirect ones. To do it, we organize the dependency graph in a directed way, all edges pointing towards the query variable x corresponding to a vertex N_Q. From N_Q we reach the disjunctive facts involving x, say $\mathcal{D}(x)$, and from $D \in \mathcal{D}(x)$ we reach the new variables $\mathcal{V}(D)$ distinct from x, but contained in $\mathcal{D}(x)$, and then we start again from each variable $y \in \bigcup_{D \in \mathcal{D}(x)} \mathcal{V}(D)$, until the whole dependency graph is scanned. We must take care of not considering a variable or a disjunction encountered in previous steps. As such this procedure cannot work properly in all cases.

4.1 Case when G is a tree

When $\mathcal{G}$ is a tree, if z_1 and z_2 are two newly encountered variables in $\mathcal{V}(D_1)$ and $\mathcal{V}(D_2)$ respectively, we are sure that $z_1 \neq z_2$. Indeed D_1 and D_2 are conditionally independent with respect to x. Similarly if D_1 and D_2 are two newly encountered disjunctions they only share at most one variable, and then it is y such that $\{D_1,D_2\} \subseteq \mathcal{D}(y)$. Hence the dependency graph is naturally arranged in a directed tree with root x, and the evaluation of x can be achieved by locally propagating the constraints from the leaves to the root. Namely if $y_1 \ldots y_k$ are leaves neighboring D, D is combined with direct justifications of $y_1 \ldots y_k$ and the result is projected to the universe of its (unique) father y. This corresponds to triggering a rule in a production system. This process is repeated until the query variable is reached, and evaluated by combination of so obtained indirect justifications and the direct justification of the query variable. Note that the leaves of the evaluation tree only contain variables such that $\mathcal{D}(y) = \emptyset$.

4.2 Case when G is not a tree

There are two reasons why the above procedure will not work

- One is when $\mathcal{V}(D_1) \cap \mathcal{V}(D_2) \neq \emptyset$. It means that D_1 and D_2 are not logically independent and should be simultaneously used. They are then put into the same sub-base. If $D_1 \in \mathcal{D}(y_1)$, $D_2 \in \mathcal{D}(y_2)$, $y_1 \neq y_2$, the dependency between D_1 and D_2 is carried over to y_1 and y_2 that must be simultaneously evaluated and are thus kept in the same group. The consequences of the detected dependency are thus propagated backwards, up to the common father in the tree under construction.
- the other reason is dual. It is when y_1 and y_2 are such that $\exists D \in \mathcal{D}(y_1) \cap \mathcal{D}(y_2)$. It implies that the two variables y_1 and y_2 are dependent and must be simultaneously evaluated. The consequences of the detected dependency are similarly propagated backwards.

At the end we come up with a directed tree of variables and disjunctive facts whose root is x, the query variable, and nodes are groups of disjunctive facts or groups of dependent variables. Moreover the father of a group of variables is a group of disjunctive facts whose father is a group of variables. The knowledge base is thus decomposed into independent sub-bases of dependent disjunctive facts, arranged in an evaluation tree. The input variables to a sub-base are its sons in the tree, and the output is the father, i.e. a single group of variables. Leaves of the evaluation tree are either single variables, or disjunctive facts, each being the son of a group of several variables.

4.3 Example

Consider the eight following rules :

1. A person with a well-organized mind and who has received a good education, speaks easily
2. A person with an intricate mind does not speak easily
3. A person with an intricate mind has a weak will
4. A person who behaves roughly has not received any education
5. A rich person who spends his/her time making speeches has a good chance to become a M.P.
6. A person who is not rich does not smoke cigars
7. A person who spends his/her time making speeches and who has no serious chances to become a M.P., speaks uneasily
8. A person who spends his/her time making speeches and speaks easily has received a good education.

These rules can be expressed as disjunctive facts with the following variables :

variable x	scale X
1. mind	{well-organized (WO), intricate (I)}
2. education	{no (NE), limited (LE), good (GE)}
3. speaks	{uneasily (U), easily (E)}
4. will	{strong (S), weak (W)}
5. behaviour	{rough (RO), correct (CO), nice (N)}
6. wealth	{rich (RI), average (A), poor (PO)}
7. makes speeches	{yes (Y), no (NS)}

8. chance of becoming a MP {good (GMP), no (NMP)}
9. smoker {no (NS), cigars (CI), other (OT)}

For instance rule 1 translates into if (x_1 WO) and (x_2 GE) then (x_3 E), i.e. ((x_1 x_2 x_3) $\overline{WO} \vee \overline{GE} \vee E$). The dependency graph corresponding to the 8 rules is the one of figure 1.

Suppose we want to know if some person has a chance to become a MP. The evaluation tree is obtained as follows (SB_i is short for sub-base i ; GV_i for group of variables no i)

$\mathcal{D}(x_8) = \{D_5, D_7\}$; $\mathcal{V}(D_5) = \{x_3, x_6\}$; $\mathcal{V}(D_7) = \{x_3, x_7\}$. Hence D_5 and D_7 are dependent and grouped. $SB1 = \{D_5, D_7\}$. $\mathcal{D}(x_3) = \{D_8, D_1, D_2\}$; $\mathcal{D}(x_6) = \{D_6\}$. $\mathcal{D}(x_7) = \{D_8\}$. $\mathcal{D}(x_3) \cap \mathcal{D}(x_7) = \{D_8\}$, hence $GV1 = \{x_3, x_7\}$, $GV2 = \{x_6\}$. $\mathcal{V}(D_1) = \{x_1, x_2\}$; $\mathcal{V}(D_2) = \{x_1\}$; $\mathcal{V}(D_8) = \{x_2\}$; $\mathcal{V}(D_6) = \{x_9\}$. Hence D_2 and D_8 must belong to the same sub-base as D_1. $SB2 = \{D_1, D_2, D_8\}$, son of GV1. $SB3 = \{D_6\}$. $\mathcal{D}(x_1) = \{D_3\}$, $\mathcal{D}(x_2) = \{D_4\}$, $\mathcal{D}(x_9) = \emptyset$, hence $\{x_9\}$ is a leaf. $\{x_4\}$ and $\{x_5\}$ will be leaves to. So we get the following evaluation tree :

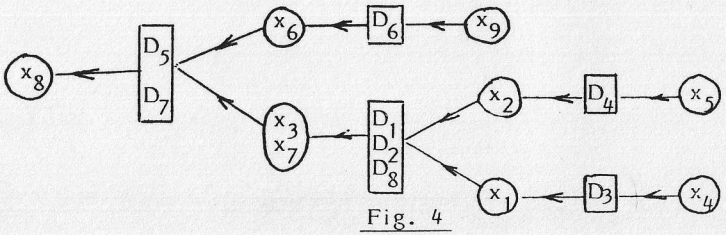

Fig. 4

5 THE EVALUATION PROCESS

Given the evaluation tree, the evaluation of the query variable proceeds as follows : leaves of the evaluation tree are the first to be evaluated by means of a combination of their direct justifications with the disjunctive facts they are sons of. The result is then projected on the universes of the higher level variables and so on until the query variable is reached. Calculations pertaining to sub-bases at a given level can be performed independently in a parallel manner.

More precisely let GV be a group of variables whose son is a sub-base SB with inputs GV_1 ... GV_k. If GV_j is a leaf then $GV_j = \{x_j\}$. The following data are available.

- justifications of variables in groups GV_j, j = 1,k obtained from calculations in lower levels of the tree. If $GV_j = \{x_{j_1} \ldots x_{j_i}\}$ then the direct justification is a relation R_j on $X_{j_1} \times \ldots \times X_{j_i}$.
- disjunctive facts linking the input variables to the output variables. They are D_1 ... D_n.
- direct justifications E_ℓ of all variables contained in $\cup_j GV_j$.

Let X be the space encompassing input and output variables, and X_0 the space of the variables in GV. The calculation of the justification of GV consists in performing the joint of $\{R_j | j = 1,k\}$, $\{E_\ell | x_\ell \in GV_j$, for all j$\}$ and $\{D_i | i = 1,n\}$ in X, and projecting the result on X_0. What is obtained is a relation R on X_0. Note that R is always expressed as a disjunction of elementary facts because R_j and D_j are too. This remark makes the actual computation simple, since it can be expressed as a symbolic calculation in the spirit of the resolution principle.

The above scheme must be slightly modified when the leaves of the tree contain disjunctions. In that case a terminal branch of the evaluation tree is of the form :

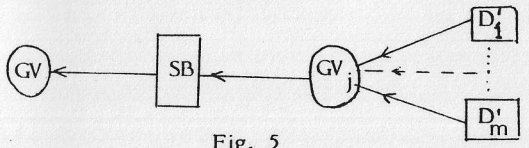

Fig. 5

When justifying GV, D'_1 ... D'_m must be included in the computation and combined with the direct justifications of variables in GV_j.

When the available data is insufficient to justify variables in a group GV, we get a relation $R = X_0$ itself, which provides no information. This branch can be pruned as being sterile.

Example (continued)

Assume we have the following elementary facts about some individual

(x_4 S) (x_2 GE) (x_7 Y)

- Combine (x_4 S) with $D_3 = \overline{I} \vee W$ and get (x_1 $\overline{I}$) = (x_1 WO)
- Delete branches containing x_6, x_9, x_5, which bring no information
- Combine (x_1 WO), (x_2 GE) with $D_1 = \overline{WO} \vee \overline{GE} \vee \overline{U}$, $D_2 = \overline{I} \vee U$, $D_8 = \overline{Y} \vee U \vee GE$, and get ($x_3$, $\overline{U}$)
- Combine (x_3 $\overline{U}$), (x_7 Y) with $D_5 = (\overline{RI} \vee \overline{Y} \vee GMP)$ and $D_7 = \overline{Y} \vee GMP \vee U$ and get (x_8 GMP) from which we conclude that the person has a good chance of becoming a M.P. □

The evaluation tree is not changing when the dependency graph is invariant. Hence it is interesting to compile this structure and store it. It is compiled in a recursive manner as a set of independent justifications connected to compiled sub-trees. The main advantage of this technique is that we take into account the fact that two evaluation trees pertaining to different queries may have some part in common. Hence the new evaluation tree must not be built from nothing ; branches of already existing trees can be used again.

6 DEALING WITH UNCERTAINTY

An uncertain fact of the form ((x A)δ) is interpreted according to Shafer (1976) evidence theory as follows (Chatalic et al, 1986) : δ is a pair $(\alpha,\beta) \in [0,1]^2$ where $\alpha \leq \beta$, α is interpreted as a degree of certainty of A, i.e. $\alpha = Bel(A)$ and $\beta = Pl(A) = 1\text{-}Bel(\bar{A})$ is a degree of plausibility of A. Bel and Pl are set-functions on X deriving from a basic probability assignment. m, such that $Bel(A) = \Sigma\{m(B) \mid B \subseteq A\}$, $\Sigma_B m(B) = 1$, and m(B) is the probability that the uncertain fact ((x A)δ) means (x B) exactly. The expert is supposed to provide the numbers α and β with the following conventions $\alpha = 1$ means (x,A) is certain ; $\alpha = 0$, $\beta = 1$ means total ignorance ; $\alpha = \beta = 0.5$ means (x,A) is true 50 % of the time ; $\beta = 0$ means $(x,\bar{A})$ is certain.

The pair (α,β) is translated into a basic probability assignment m according to the principle of minimum specificity (Dubois, Prade, 1986) which basically says that the weights m(B) should be allocated to the least specific subsets of X. Namely ((x A) (α β)) is translated into $m(A) = \alpha$, $m(\bar{A}) = 1\text{-}\beta$, $m(X) = \beta\text{-}\alpha$. Indeed it means that α reflects the positive evidence about A, β the negative evidence, and β-α the amount of ignorance. This representation encompasses the case of possibilistic information (when $\max(\beta, 1\text{-}\alpha) = 1$, the set-function Pl is a possibility measure in the sense of Zadeh (1978)), as well as probabilistic information (when $\alpha = \beta$, i.e. there is no imprecision about the degree of uncertainty). In that case, the conjunctive operation performed to combine information in the sub-bases, is Dempster rule of combination (Shafer, 1976 ; Gordon and Shortliffe, 1985). When the uncertainty is purely possibilistic, the minimum rule of fuzzy set theory can be used instead of Dempster rule, in order to preserve the possibilistic nature of uncertainty through all the deduction process. In order not to hide conflicts in the knowledge

base, we do not use the normalized version of Dempster rule. On the contrary, we maintain an evaluation of the amount of conflict which is the weight $m(\emptyset)$ bearing on the empty set after the combination process. Indeed, we consider that when $m(\emptyset)$ becomes close to 1, the meaningfulness of the results obtained so far becomes dubious. More details on the treatment of uncertainty in the actual system, called MIRACLE, is to be found in (Chatalic et al, 1986 , Chatalic, 1986).

CONCLUSION

The MIRACLE system (Chatalic, 1986) implementing these ideas is presently working on a micro-computer, and has been applied to surveillance problems involving around 50 rules and 25 variables. The inference mechanism it proposes is seemingly more powerful than standard expert systems inference engines because it allows for modus tollens-like reasoning, and rules with disjunctions of facts in their conclusion part. Its basic inference step is similar to the resolution principle applied to propositional clauses involving unary predicates. However it is able to deal with uncertainty in a rigorous manner, within a general modeling framework encompassing evidence theory, possibility and probability theories. Moreover, it automatically turns the knowledge base into a set of precompiled evaluation trees, so as to carry out the deduction steps in the most parallel fashion, in small independent sub-bases. Of course the efficiency of this technique is directly related to the level of intricacy of the knowledge base. It is especially efficient for sparse dependency graphs. The technique of the evaluation tree can be viewed as a generalization of the backward chaining method.

REFERENCES

Chatalic P. Raisonnement déductif en présence de connaissances imprécises et incertaines : un système basé sur la théorie de Dempster-Shafer. Thèse de Doctorat, Université Paul Sabatier, Toulouse, 1986.

Chatalic P., Dubois D., Prade H. An approach to approximate reasoning based on Dempster rule of combination. Proc. 8th IASTED Int. Symposium on Robotics and Artificial Intelligence, Toulouse, France, June 18-20, 1986, 333-343.

Dubois D., Prade H. Théorie des Possibilités. Applications à la Représentation des Connaissances en Informatique, Masson, Paris 1985.

Dubois D., Prade H. The principle of minimum specificity as a basis for evidential reasoning. Proc. Int. Conference on Information Processing, and Management of Uncertainty in Knowledge-based Systems, Paris, France, June 30-July 4, 1986, pp. 40-43.

Gordon J., Shortliffe E. A method for managing evidential reasoning in a hierarchical hypothesis space. Artificial Intelligence, 26, 323-357, 1985.

Kong A. Multivariate belief functions and graphical models. Doctoral Dissertation, Department of Statistics, Harvard University, 1986.

Nilsson N. Probabilistic logic. Artificial Intelligence, 28, 71-88, 1986.

Pearl J. Fusion, propagation and structuring in Bayesian networks. Artificial Intelligence, 29, 241-288, 1986.

Reiter R. A logic for default reasoning. Artificial Intelligence, 13, 81-132, 1980.

Shafer G. A mathematical theory of evidence. Princeton University Press, Princeton N.J., 1976.

Shafer G., Shenoy P.P., Mellouli K. Propagating belief functions in qualitative Markov trees. Working paper n° 186, School of Business, University of Kansas, Lawrence, 1986.

Zadeh L.A. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems, 1, 3-28, 1978.

Zadeh L.A. A theory of approximate reasoning, Machine Intelligence, Vol. 9 (J.E. Hayes, D. Michie, L.I. Mikulich, Eds.), Wiley N.Y. 1979, 149-194.