DIDIER DUBOIS and HENRI PRADE are CNRS research scientists at IRIT Institute of Paul Sabatier University, in Toulouse, France. RONALD R. YAGER is Director of the Machine Intelligence Institute at Iona College, in New Rochelle, New York.

Visit our Web site at: http://www.wiley.com/compbooks/
INFORMATION ENGINEERING

A Guided Tour of Applications

EDITED BY
Didier Dubois, Henri Prade, and Ronald R. Yager

WILEY COMPUTER PUBLISHING

JOHN WILEY & SONS, INC.
New York • Chichester • Brisbane • Toronto • Singapore • Weinheim
For future high-level control systems, FLS Automation is working with integration of various control and prediction techniques. In our opinion, the optimal overall control system is most likely to be a combination of the rule-based fuzzy logic approach, neural network techniques, and model-based systems, with the possibility of operator interference. In particular, neural nets seem to a promising technique for integrating the operator in a structured way as an active part of the automatic control system.

Figure 22.13 shows a diagram of an operator-activated control objective. The neural net has been trained to reveal key information about the actual process state, the so-called state indices in FUZZY II terminology. The operator may judge the same index variables, and a FUZZY II control objective has been defined to cope with the situation in which the neural net and the operator do not agree.

In this way, the operator has the possibility of interfering with the automatic control strategy. Operator interference is not necessary, but when he or she interferes, it is done in a well-defined way that has been decided through the definition of the special operator-activated control objective.

22.8. References

The metarules are of three different types:

**Rule-oriented metarules.** These metarules, as has been mentioned, are related to the applicability of the rules within a module. Depending on contextual facts, some rules are removed from the active rule set; for example, the rules that deduce some kind of virus disease depending on a blood analysis cannot be applied unless the results of the analysis are available.

**Module-oriented metarules.** These metarules refer to the global strategy of selecting subgoals. After the application of each module, the set of metarules modifies the strategy, adding or deleting modules from the list of objectives; for example, a very high certainty value about the conclusion of a bacterial disease will probably discard the modules concerning nonbacterial diseases.

**Strategy-oriented metarules.** When more than one strategy is generated, they are responsible for combining them into only one. They use domain knowledge and control knowledge (knowledge referring the conflicting strategies) in their premises.

**Facts.** Facts are the most elementary objects of the system. They are of four different types:

- **Boolean**
- **Fuzzy**—The value is a certainty degree, represented by a linguistic fuzzy number or a linguistic fuzzy label. [1.2]
- **Numerical**—The value of the fact is a number.
- **Subsets**—The value of a fact is a subset of a given set of possible values.

In rule definition it is necessary to perform type checking of the facts, as well as of the predicates. MILORD keeps track of the rules that deduce each fact and the rules they belong to, as part of the premise. This is necessary to perform forward and backward chaining in an efficient way.

### 23.1.2. KB Edition and Verification

This system has a rule-oriented editor that provides the functions necessary to develop the knowledge base (KB). The main features are:

- Basic commands to add, modify, or delete every object in the KB.
- Automatic prompting of the attributes of the objects, with syntactic control.
- Type checking for the typified attributes.
- Advanced commands to search, rename (with automatic changes of all pointers within the system), and copy objects.
The editor can insert new objects in the KB, automatically updating it. To provide flexibility, it is possible to use objects before defining them. The editor maintains all previous commands in an agenda. At the session closure, however, the original KB is connected with the original KB. If any operation has been made is connected with the original KB, the editor asks the user for a solution. If he or she cannot resolve the question, the editor restores the last correct situation for the object.

Logical verification requires detection of any KB situations that could cause improper behavior of the inference engine. These situations can be logical errors (contradictory rules or circular rules), missing knowledge (unreachable conclusions) or knowledge duplication (redundant rules, subsumed rules or unnecessary IF-conditions). In Boolean logic, all these situations are detected with a static rule analysis. [3] In fuzzy systems, further extensions are required. These are discussed next.

**Redundant rules.** Two rules are redundant if they are identical, ignoring the certainty values. For example, these rules are redundant (the values between parentheses represent the certainty values):

\[
\begin{align*}
(1) & \quad p \rightarrow q \\
(2) & \quad p \rightarrow (q)
\end{align*}
\]

These rules are not a logic problem in a Boolean system, but they are in a fuzzy system. The same information may be counted twice, with an erroneous certainty value in the conclusion.

**Subsumed rules.** A rule is subsumed by another if: (1) both have the same conclusion, and (2) both share a set of IF-conditions but one of them has additional IF-conditions. If the following example \( R1 \) is subsumed by \( R2 \):

\[
\begin{align*}
(3) & \quad R1: p \rightarrow q \\
(4) & \quad R2: p \rightarrow r
\end{align*}
\]

This is a legal construction. To avoid an erroneous combination of the certainty values, care must be taken not to fire \( R2 \) if \( R1 \) has been fired.

**Unnecessary IF-conditions.** An unnecessary IF-condition is contained in two rules if: (1) the conclusions are identical, (2) both rules have the same IF-conditions except one, and (3) that IF-condition is the same in both rules, but is affirmed in one and negated in the other. For example:

\[
\begin{align*}
(5) & \quad R1: p \rightarrow q \\
(6) & \quad R2: p \rightarrow \neg q
\end{align*}
\]

It is possible to eliminate this IF-condition in Boolean logic and collapse the two rules in one. In general, this is not possible in fuzzy logic. This situation must be detected to warn the user.

**Circular rules.** A set of rules is circular if they create an infinite loop in the backward engine. For example, the following set of rules is circular:

\[
\begin{align*}
(7) & \quad R1: p \rightarrow q \\
(8) & \quad R2: q \rightarrow r \\
(9) & \quad R3: r \rightarrow p
\end{align*}
\]

If the goal is \( p \), the backward engine tries to validate \( r \) (rule \( R3 \)). Next it tries to validate \( q \) (rule \( R2 \)) and then \( p \) (rule \( R1 \)), which returns it to the original goal. The problem is the same in fuzzy and Boolean logic.

**Contradictory rules.** In Boolean logic, two rules are contradictory if they have the same IF-conditions and opposite conclusions. For example:

\[
\begin{align*}
(10) & \quad R1: p \rightarrow q \\
(11) & \quad R2: p \rightarrow \neg q
\end{align*}
\]

In fuzzy logic, this is a legal construction. It is possible to have some degree of belief in some fact \( q \) and also a degree of belief in the opposite fact \( \neg q \). A contradiction may be considered when for an expert-defined threshold \( \pi \), we have:

\[
\text{certainty}(q) + \text{certainty}(-q) > \pi
\]

To detect such contradictions it is necessary to keep for every fact \( q \) the opposite fact \( \neg q \), and compute the certainty values of \( q \) and \( \neg q \), in every step of every deduction chain. In any case, the rules that are contradictory in a Boolean sense (without considering the certainty values) are displayed.

**Unreachable conditions.** If the conclusion of a rule is not a goal and it does not match any IF-condition of another rule, this conclusion is unreachable and this rule will not be fired. In fuzzy logic, a conclusion may match an IF-condition of another rule. However, the rule will not be fired if the rules chaining always propagates a certainty value lower than a given validity threshold (in MYCIN is 0.2, in MILORD is SLIGHTLY POSSIBLE). Therefore, this is also a situation of unreachable conclusion.

**Dead-end IF-conclusions and dead-end goals.** If a goal or an IF-condition does not match a conclusion of any rule, this is a dead-end goal or IF-condition. They will not be validated and there is a gap in the KB. In fuzzy logic another type of dead-end IF-condition is possible: it occurs when the rules chaining always stops because the certainty value is below the validity threshold.

Redundant rules, subsumed rules, unnecessary IF-conditions, circular rules, unreachable conclusions, and dead-end IF-conditions and goals (the two last in the Boolean case only), can be detected by a static rule
analysis. Problems associated with the propagation of the certainty values below the validity threshold (unreachable conclusions and dead-end IF-conditions) can be detected traversing the AND/OR deduction tree. Given a goal, an upper bound of the certainty value reachable for each possible deduction chain can be computed. This is done by applying the implication function to the certainty values of the rules in the deduction chain. If some deduction chain is cut because the upper bound falls below the mentioned validity threshold, the cutting point is a dead-end IF-condition, and the remainder of the chain becomes a set of unreachable conclusions (for this goal). If none of these rules is used in another deduction chain, it can never be fired.

23.1.3. Semantic Network Organization

There is an obvious hierarchical structure, (inclusion relationship), among the objects defined in MILORD. Each element has, in this sense, some inherited properties. A fact inherits the names of the rules it is related to. The rules and the facts inherit the name of the module they appear in, and so on.

This structure can be seen as a three-level semantic network. In each level objects have interrelations that express priority, subsumption, inclusion by property chains, specificity, and so on. These relations are obtained from a syntactic analysis of the Knowledge Base (KB) that is performed by the editor described previously. Together with this explicit structure, there are explicit relations among these objects given by the metarules. Let us see a set of examples of these relations.

In the following examples, a semantic network is established in such a way that it allows the system to fulfill the restrictions imposed by the knowledge base. In Figures 23.1 to 23.3 bold lines represent the relationships inferred from the information given in the definition of the KB.

The next example shows an inclusion via property chains. In this case the property type is “is a kind of”:

Figure 23.2 Subsumption of rules (all links are POS).

<table>
<thead>
<tr>
<th>Rule</th>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>R02013</td>
<td>1) Community-acquired pneumonia</td>
<td>[Quite possible] Pneumococcus</td>
</tr>
<tr>
<td>2) Frequent contacts with animals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) Bacterial disease</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R02014</td>
<td>1) Community-acquired pneumonia</td>
<td>[Very possible] Pneumococcus</td>
</tr>
<tr>
<td>2) Frequent contacts with birds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) Bacterial disease</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There is a relation “bird is a kind of animal” that makes the rule R02014 more specific than the rule R02013.

What follows is an example of subsumption among three rules:

<table>
<thead>
<tr>
<th>Rule</th>
<th>IF</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>R02006</td>
<td>1) Community-acquired pneumonia</td>
<td>[Moderately possible] Pneumococcus</td>
</tr>
<tr>
<td>2) (&gt; age 70)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3) Lives in an old people’s home</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4) Bacterial disease</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 23.3 Priority relationship.
Figure 23.3 shows a priority relationship between conditions of different rules.

23.1.4. Knowledge Translation from External to Internal Representation

In order to improve the system's performance, a translation of the external representation is made. The internal object representation is based on a framelike structure. Each object, fact, rule, or module, is represented by a structure that holds the properties, relations, and general information of each object. This representation makes explicit the implicit relations of the external representation. Moreover, some computation may be done; for example, the maximum reachable value of a fact, also some misspellings, as well as syntactic and semantic errors can be detected. Type checking is applied here in order to test the KB. Next, we can see a partial result of the compilation process over a rule and a fact:

The external aspect of a rule is:


The internal representation of a rule contains information that is not explicit in the external one, such as contextual and control information to perform a more efficient explanation of the reasoning strategies. The control slots EX-P-CUT and EX-O-CUT (see III-B), which are nil by default, retain information about the evaluation process.

The internal representation is implemented using the "defstruct" primitive of Common-Lisp. For example:

The internal representation contains information about context, cross-reference, control, value, and type of value. The contextual information is given by the location of the fact in the KB, its name, the KB to which it belongs, and the modules where it appears. Each fact has two lists that contain the names of the rules where it appears as part of the premise, and the rules where it appears as conclusion. This is a very useful way to perform cross-reference tables that are mainly used in the development stage. Each fact has a type and a set of legal values that are included in the representation of the facts, assuming by default that a fact is of a fuzzy type. Control information is related to the cuts the system may perform in the evaluation of a fact (see IV-B).

The internal representation of the fact LOBAR PATTERN is:

The internal representation of a rule contains information that is not explicit in the external one, such as contextual and control information to perform a more efficient explanation of the reasoning strategies. The control slots EX-P-CUT and EX-O-CUT (see III-B), which are nil by default, retain information about the evaluation process.

The internal representation is implemented using the "defstruct" primitive of Common-Lisp. For example:
23.2. MILORD Multilevel Architecture

When defining an application (a KB), two main aspects of the problem must be solved by a given shell: The representation of Domain Knowledge and of Control Knowledge. The second aspect is the main one in most AI applications. In this section the relations between them will be pointed out.

The architecture of MILORD presents a multilevel structure in both the domain representation and the control representation. There is a relation between these two representations as can be seen in Figure 23.4.

The Semantic Network acts over the facts, structuring their interdependencies and controlling mechanisms such as semantic subsumption between concepts (for example, if the fact AIDS is present in the context of a bacterial disease then the fact Alcoholic is not relevant and should not be considered).

The metarules hierarchy controls the application of their corresponding domain knowledge. Before moving down in the domain level hierarchy (from strategies to rules and facts), the corresponding control level is consulted.

This separation between different levels of control and of domain representation matches quite well various design processes in KB definition:

1. Allows a top-down design of the KB, because the decisions about domain knowledge and the corresponding control items are independent of later refinements. For example, in the case of strategy definitions and resolution of conflicts between strategies, you can first define strategies based only on domain knowledge, and later you can define how to solve the possible conflicts between them.

2. Allows defining different problem solving methods, due to the flexibility at the operational level. For example, you can define simple classification or heuristic classification processes. [4]

3. Allows defining different hierarchical structures: mixed hierarchies or pure hierarchies in the domain knowledge, and the corresponding control level that supervises the use of the different strategies.

However, this architecture is not suitable to define generative systems, that is, systems that can generate or build the solutions to the problem. MILORD presumes that all the solutions are known a priori.

This multilevel definition has also some interesting issues from the point of view of the knowledge engineer:

- **Extensibility**: adding new domain knowledge and new control knowledge is easy and safe.
- **Debugability**: to correct mistakes and validate the KB is easier.
- **Explainability**: to explain the behavior of the system is easier because upper levels in the hierarchy provide the correct justifications.

### 23.3. Uncertain Reasoning

The numerical approaches to the representation of uncertainty imply hypothesis of independence, mutual exclusiveness, and so on about the information they deal with. On the other hand, they oblige the expert and the user to be unrealistically precise and consistent in the assignment of such numerical values to rules and facts. Furthermore, these approaches are computationally expensive.

Two approaches, based on a linguistic characterization of uncertainty, have been developed in MILORD. The certainty values are linguistic terms defined by the expert. In the first approach, the internal representation of each term is a fuzzy number on the interval [0,1], following the work of Bonissone, [5] and allows to deal with uncertain facts and with rules whose uncertainty concerns the strength of the implication. The second approach is an extension of the first one that allows one to deal with rules containing linguistically expressed uncertainty that modifies the conditions and conclusions. Such type of uncertainty is internally represented by fuzzy labels. [6] For computational reasons, both representations are parametrized at the very beginning of each application design. In our application to pneumonia diagnosis, we have first been using the first approach, and later, we have switched to the second approach according to the preference of our medical expert.

MILORD has also been parametrized in order to perform different calculi of uncertainty operating on the expert defined term set of linguistic certainty values.
23.3.1. The Calculus of Uncertainty

It can be shown [7] that Triangular norms (T-norms) and Triangular conorms (T-conorms) are the most general families of two-place functions from \([0,1] \times [0,1]\) onto \([0,1]\) that respectively satisfy the requirements of conjunction and disjunction operators.

A T-norm \(T(p,q)\) performs a conjunction operator, on the degrees of certainty of two or more conditions in the same premise, satisfying the following properties:

\[
T(0,0) = 0 \\
T(0, p) = T(p, 0) = p \\
T(p, q) = T(q, p) \\
T(p, q) \leq T(r, s) \text{ if } p \leq r \text{ and } q \leq s \\
T(p, T(q, r)) = T(T(p, q), r)
\]

A T-conorm \(S(p,q)\) computes the degree of certainty of a conclusion derived from two or more rules. It is a disjunction operator satisfying the following properties:

\[
S(1,1) = 1 \\
S(0, p) = S(p, 0) = p \\
S(p, q) = S(q, p) \\
S(p, q) \leq S(r, s) \text{ if } p \leq r \text{ and } q \leq s \\
S(p, S(q, r)) = S(S(p, q), r)
\]

For suitable negation operators \(N(x)\), T-norms and T-conorms are dual in the sense of De Morgan laws.[8]

An implication function \(I(p,q)\) gives the certainty degree of a rule as a function of the certainty degree of the premise and the certainty degree of the conclusion. The more usual types of implication functions are:

1. S-Implications, defined by \(I_s(p,q) = S(N(p),q)\). \(S\) being a T-conorm and \(N\) a negation operator.

2. R-Implications, defined by \(I_r(p,q) = \text{SUP}\{c \in [0,1] \mid T(p,c) \leq q\}\). Being \(T\) a T-norm.

A "Modus Ponens-generating function" (m.p.g.f.),[9] for an implication function \(I\), propagates a certainty value to the conclusion from the certainty values of the premise and the rule:

1. For an \(S\)-Implication generated by a T-conorm \(S\), a m.p.g.f. can be defined by \(m_s(p,r) = \text{INF}\{c \in [0,1] \mid S(N(p),c) \geq r\}\).

2. For an \(R\)-Implication generated by a T-norm \(T\), \(T\) itself is a m.p.g.f.

Some usual pairs of dual T-norms and T-conorms are:

\[
T_0(x,y) = \begin{cases} 
\min(x,y), & \text{if } \max(x,y) = 1 \\
0, & \text{otherwise}
\end{cases} \quad S_0(x,y) = \begin{cases} 
\min(x,y), & \text{if } \max(x,y) = 1 \\
0, & \text{otherwise}
\end{cases}
\]

\[
T_1(x,y) = \max(0,x+y-1) \quad S_1(x,y) = \min(1,x+y)
\]

\[
T_{1,5}(x,y) = \frac{xy}{2-(x+y-xy)} \quad S_{1,5}(x,y) = \frac{x+y}{1+xy}
\]

\[
T_2(x,y) = xy \quad S_2(x,y) = x+y-xy
\]

\[
T_{2,5}(x,y) = \frac{xy}{x+y-xy} \quad S_{2,5}(x,y) = \frac{x+y-2xy}{1-xy}
\]

\[
T_3(x,y) = \min(x,y) \quad S_3(x,y) = \max(x,y)
\]

It can be shown that they are ordered as follows:

\[
T_0 \leq T_1 \leq T_{1,5} \leq T_2 \leq T_{2,5} \leq T_3
\]

\[
S_0 \leq S_{1,5} \leq S_2 \leq S_{2,5} \leq S_3
\]

Some usual Implication functions and its m.p.g.f. are:

\[
I_0(x,y) = I_0(x,y) = \min(1-x+y,1) \quad m_0 = T_1
\]

\[
I_1(x,y) = 1-x+xy \quad m_1(x,y) = \begin{cases} 
\max(x+y-1,0), & \text{if } x \neq 0 \\
0, & \text{otherwise}
\end{cases}
\]

\[
I_2(x,y) = \frac{1}{x} \quad \text{otherwise} \quad m_2(x,y) = T_0
\]

\[
I_3(x,y) = \max(1-x,y) \quad m_3(x,y) = \begin{cases} 
0, & \text{if } y \leq 1-x \\
y, & \text{otherwise}
\end{cases}
\]

\[
I_0(x,y) = \begin{cases} 
1, & \text{if } x \leq y \\
y, & \text{otherwise}
\end{cases} \quad m_0 = T_0
\]

In MILORD we have specially considered the pairs \((T_1,S_1), (T_2,S_2),\) and \((T_3,S_3)\) following the experimental results obtained by Bonissone [10] which consisted in applying nine T-norms to three different term sets. Bonissone analyzed the sensitivity of each operator with respect to the granularity (number of elements) in the term sets and concluded that only the T-norms \(T_1, T_2,\) and \(T_3\) generated sufficiently different results for term sets that do not have more than nine elements. On the other hand, according to the results of Miller [11] concerning the span of absolute judgment, it is unlikely that any expert or user would consistently qualify uncertainty using more than nine different terms.
23.3.2. The Linguistic Certainty Values

MIORLD allows the expert to define the term set of linguistic certainty values which constitutes the verbal scale that he or she and the users will use to express their degree of confidence in the rules and facts respectively. Recent psychological studies have shown the feasibility of such verbal scales: "...A verbal scale of probability expressions is a compromise between people's resistance to the use of numbers and the necessity to have a common numerical scale" [12]; "...people asked to give numerical estimations on a common-day situation err most of the time and in a nonconsistent way. Furthermore, they are unable to appreciate their judgments imprecision (errors are by far bigger than the maximum error accepted as possible by the subjects themselves). Nevertheless, judgments embodied in linguistic descriptors appear consistent in this same situation ..." [13].

23.3.3. The Management of Uncertainty with Fuzzy Numbers

Each linguistic value is represented internally by a fuzzy interval (fuzzy number) that is, the membership function of a fuzzy set on the real line, or, more precisely, on the truth space represented by the interval [0,1]. These membership functions can be interpreted as the meanings of the terms in the term set. The conjunction and disjunction operators applied to these functions will produce another membership function as a result that will have to be matched to a term in the term set, in order to keep the term set closed. This can be done by a linguistic approximation process that will be described later (see Bonissone [5] for an extensive study of the linguistic approximation process).

23.3.3.1. The Term Set for PNEUMONIA and Its Representation.

Although the expert can define its own term set together with its internal representation, MIORLD provides a default term set obtained by means of a survey conducted among several hundred medical professionals in the Barcelona area. A similar study was also performed [15] although the goal there was not to use the term set in an expert system. The default term set is:

(IMPOSSIBLE, ALMOST-IMPOSSIBLE, SLIGHTLY-
POSSIBLE, MODERATELY-POSSIBLE, POSSIBLE,
QUITE-POSSIBLE, VERY-POSSIBLE, ALMOST-SURE,
SURE)

Each term Tj is represented by a membership function \( \mu(x) \), for \( x \) in the interval [0,1].

Each membership function is represented by four parameters \( T_j = (a_b, c_d, d) \), corresponding to the weighted interval in Figure 23.5.

The nine-element term set has the following representation resulted from the conducted survey:

<table>
<thead>
<tr>
<th>Term</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPOSSIBLE</td>
<td>( (0,0,0,0) )</td>
</tr>
<tr>
<td>ALMOST-IMPOSSIBLE</td>
<td>( (0,0,0.05,0.08) )</td>
</tr>
<tr>
<td>SLIGHTLY POSSIBLE</td>
<td>( (0.05,0.07,0.14,0.17) )</td>
</tr>
<tr>
<td>MODERATELY POSSIBLE</td>
<td>( (0.10,0.15,0.35,0.45) )</td>
</tr>
<tr>
<td>POSSIBLE</td>
<td>( (0.25,0.35,0.55,0.65) )</td>
</tr>
<tr>
<td>QUITE POSSIBLE</td>
<td>( (0.45,0.55,0.75,0.85) )</td>
</tr>
<tr>
<td>VERY POSSIBLE</td>
<td>( (0.65,0.75,1,1) )</td>
</tr>
<tr>
<td>ALMOST SURE</td>
<td>( (0.95,0.98,1,1) )</td>
</tr>
<tr>
<td>SURE</td>
<td>( (1,1,1,1) )</td>
</tr>
</tbody>
</table>

corresponding to following functions in Figure 23.6.

In order to be able to evaluate the T-norms \( T_1, T_2, T_3 \) and the T-conorms \( S_1, S_2, S_3 \) on the elements of the term set, we apply the following formulæ according to the arithmetic rules on fuzzy numbers [1]:

Given two fuzzy intervals \( T = (a, b, c, d) \) and \( T' = (a', b', c', d') \), we have:

- \( T + T' = (a + a', b + b', c + c', d + d') \)
- \( T - T' = (a - a', b - b', c - b', d - d') \)
- \( T \times T' = (aa', bb', cc', dd') \)
- \( \min(T, T') = (\min(a, a'), \min(b, b'), \min(c, c'), \min(d, d')) \)
- \( \max(T, T') = (\max(a, a'), \max(b, b'), \max(c, c'), \max(d, d')) \)

23.3.3.2. The Linguistic Approximation. A linguistic approximation process is performed in order to find a term (linguistic value) in the term set whose "meaning" (membership function) is the closest (according to a given metric) to the "meaning" (membership function) of the result of the conjunction or disjunction operation performed on any two linguistic values of the term set. This allows maintenance of closed operations for

![Figure 23.5 Parametric representation of membership functions](image-url)
any T-norm and T-conorm. The problem is, therefore, that of computing a distance between two trapezoidal membership functions. In order to do so, we have adopted for the sake of simplicity a solution consisting in the computation of a weighted Euclidean distance of two relevant features of the functions: the first moment and the area under the function. Figure 23.7 shows the results obtained with the selected T-norm $T_2$ on the term set of Figure 23.6.

23.3.4. The Management of Uncertainty Using Fuzzy Linguistic Labels

As it has been mentioned, this is the approach that has been finally preferred by our expert.

23.3.4.1. Introduction. This is an alternative approach that allows to combine imprecision and uncertainty. Let us consider a proposition such as ‘X is A’ where X is a variable that takes its values from a fuzzy subset A of a Universe of Discourse U. Then X induces a possibility distribution [2] on A:

\[
T_2
\]

\[
\begin{array}{c}
\text{Impossible} \\
\text{Almost impossible} \\
\text{Slightly possible} \\
\text{Possible} \\
\text{Moderately possible} \\
\text{Quite possible} \\
\text{Very possible} \\
\text{Almost sure} \\
\text{Sure}
\end{array}
\]

Figure 23.7 AND matrix computed using $T_2$.

\[
P_0(x) = \mu_A(x), \text{ for all } x \in U
\]

where $\mu_A$ stands for the membership function of the fuzzy subset A.

From now on, we will identify A with the possibility distribution $P_0$ on A. We can define a fuzzy label [6] by means of a function $t : [0,1] \rightarrow [0,1]$ such that the proposition ‘(X is A)’ is $t$-would be equivalent to ‘X is A’", where $A'' = t \circ A$, in the sense of the usual functions composition. See, for example, Figure 23.8.

From this point of view, a fuzzy label can be understood as a linguistic modifier of the fuzzy subset A. But on the other hand, given A and A”, we can interpret t as the compatibility of A with A”, that is, the fuzzy label we have to apply to A to get A”. The compatibility is defined as follows:

\[
t(x) = \begin{cases} 
\sup \{A''(u) : u \in A^{-1}(x)\}, & \text{if } A^{-1}(x) \neq \emptyset \\
0, & \text{otherwise}
\end{cases}
\]

23.3.4.2. Connective Operators between Fuzzy Labels. In order to propagate and combine uncertainty, it is necessary to define the following operators.

In what follows, A will stand for a fuzzy subset of a universe $U$, and B a fuzzy subset of a universe $V$.

1. Composition. If we apply a linguistic modifier $t_2$ to a fact that is already labeled by $t_1$, what we are doing is to apply consecutively two functions to a possibility distribution; in other words, we are modifying the fact by the composition of two labels:

\[
((X \text{ is } A) \text{ is } t_2) \text{ is } t_2 = (X \text{ is } A) \text{ is } t, \text{ being } t = t_2 \circ t_1
\]

2. Compatibility. The compatibility of ‘(X is A)’ is $t_1$’ with respect to ‘(X is A)’ is $t_2’’ is $t_1 \circ t_2$ if $t_1 \circ t_2$ is the least fuzzy label (w.r.t. the pointwise order) such that:

Figure 23.8 Modification of a fuzzy set by a fuzzy label.
in the sense that \( t_{1,2} \circ t_1 \geq t_2 \), where \( t_{1,2} \) is defined as:
\[
t_{1,2}(x) = \text{Sup}( t_1(y), t_2(y)) \mid x = y, \text{for all } x \in [0,1].
\]

3. **Conjunction.** A composed proposition of the kind "\((X \text{ is } A) \text{ AND } (Y \text{ is } B)\)" has an associated two-dimensional possibility distribution, \( \Pi_t(u,v) = T(A(u), B(v)) \), \( T \) being the \( T \)-norm conjunction operator. In the same way, if the proposition is "\((X \text{ is } A) \text{ AND } (Y \text{ is } B)\)" the possibility distribution will be \( \Pi_t(u,v) = T(A'(u), B'(v)) \), \( A' = t_1 \circ A \) and \( B' = t_2 \circ B \). Then, for a given \( T \)-norm \( T \), we define \( t_1 \land_T t_2 \) as a label that transforms \( \Pi_t \) into \( \Pi_{t_1} \), as follows:

\[
[(X \text{ is } A) \text{ is } t_1] \text{ AND } [(Y \text{ is } B) \text{ is } t_2] = [(X \text{ is } A) \text{ AND } (Y \text{ is } B)] \text{ is } t_1 \land_T t_2
\]
where \( t_1 \land_T t_2(z) = \text{Inf}(T(t_1(x), t_2(y)), \mid z = T(x,y)) \), for all \( z \in [0,1] \).

4. **Disjunction.** A composed proposition of the kind "\((X \text{ is } A) \text{ OR } (Y \text{ is } B)\)" has an associated two-dimensional possibility distribution \( \Pi_t(u,v) = S(A(u), B(v)) \), \( S \) being the \( T \)-conorm disjunction operator. In the same way, if the proposition is "\((X \text{ is } A) \text{ OR } (Y \text{ is } B)\)" the possibility distribution will be \( \Pi_t(u,v) = S(A'(u), B'(v)) \), \( A' = t_1 \circ A \) and \( B' = t_2 \circ B \). Then, for a given \( T \)-conorm \( S \), we define \( t_1 \lor_T t_2 \) as a label that transforms \( \Pi_t \) into \( \Pi_{t_1} \), as follows:

\[
[(X \text{ is } A) \text{ is } t_1] \text{ OR } [(Y \text{ is } B) \text{ is } t_2] = [(X \text{ is } A) \text{ OR } (Y \text{ is } B)] \text{ is } t_1 \lor_T t_2
\]
where \( t_1 \lor_T t_2(z) = \text{Inf}(S(t_1(x), t_2(y)), \mid z = S(x,y)) \), for all \( z \in [0,1] \).

5. **Negation.** Given a proposition "\((X \text{ is } A) \text{ is } t\)", we define the negation of \( A \) as a label \( -t \) that transforms the possibility distribution \( \neg A \) into \( -t \circ A \), that is:

\[
-((X \text{ is } A) \text{ is } t) = (X \text{ is } \neg A) \text{ is } \neg t
\]
where \( \neg A \) stands for the standard complement of \( A \) that is:

\[
\mu_{\neg A}(u) = 1 - \mu_A(u),
\]
and \( \neg t = n \circ t \circ n \), with \( n(x) = 1 - x \).

With these definitions, notice that, if \( T \) and \( S \) are dual, the following De Morgan’s equality can be easily verified:

\[
t_1 \land_T t_2 = -((\neg t_1) \lor_T (\neg t_2))
\]

6. **Inference (Modus Ponens).** Following the “modified” Compositional Rule of Inference, [16] the basic scheme of inference is:

\[
\begin{align*}
\text{IF } (X \text{ is } A) & \text{ THEN } (Y \text{ is } B) \text{ is } t_1 \\
(X \text{ is } A) & \text{ is } t_2 \\
(Y \text{ is } B) & \text{ is } t_3 = \text{INFER}(t_1, t_2)
\end{align*}
\]

where \( t_3(y) = \text{ Sup}( m_i(t_2(x), t_1(I(x,y))) \mid x \in [0,1] ) \), for all \( y \in [0,1] \).

Here \( I(x,y) \) is the Implication function chosen to represent the conditional statement, and \( m_1 \) is its corresponding “modus ponens generating function.”

This formulation can be generalized if the conditions and the conclusions contain linguistically expressed uncertainty modifiers. In such a case, the inference process needs three steps:

\[
\begin{align*}
\text{IF } (X \text{ is } A) & \text{ is } t_1 \text{ THEN } (Y \text{ is } B) \text{ is } t_3 \\
(X \text{ is } A) & \text{ is } t_4 \\
(Y \text{ is } B) & \text{ is } t_5 = \text{INFER}(t_1, t_4) \circ t_3
\end{align*}
\]

that is:

1. computing the compatibility of \( t_1 \) with \( t_4 \) we get \( t_{1,4} \)
2. applying the Compositional Rule of Inference \( t \circ t_{1,4} \) and \( t_3 \), we get \( \text{INFER}(t_{1,4}, t_3) \)
3. and, finally, the composition of \( \text{INFER}(t_{1,4}, t_3) \) with \( t_5 \) gives \( t_5 \)

23.3.4.3. Selection of a Family of Labels. Because of the arbitrary ways to modify a possibility distribution, it is necessary to work on a restricted family of fuzzy labels. The labels we have considered have a two-parameter representation \( t = (a,b) \), where \( a = 0 \) or \( b = 1 \), leading to the two following types of labels:

![Figure 23.9](image)

These labels perform two kinds of transformations (see Figure 23.9).

\( A' \) is a less restrictive fuzzy subset than \( A \), while \( A'' \) is a more restrictive one. Therefore, modifications of Type I can be viewed as a linguistic way to emphasize the fact of having less confidence on the premise \( X \) is \( A' \), whereas modifications of Type II emphasize more confidence.

Furthermore, a set of certainty degrees can be established according to the different modifications of \( A \) (see Figure 23.10). Thus, an implicit order relation "\( s_t \)" is defined: if \( t_1 = (a_1, b_1) \) and \( t_2 = (a_2, b_2) \) then: \( t_1 \leq s_t t_2 \) if \( a_1 + b_1 \leq a_2 + b_2 \) (i.e., \( t_1 \leq s_t t_2 \) if \( t_1(x) \geq s_t t_2(x) \), for all \( x \in [0,1] \).

Finally, notice that this set of labels actually applies over values with a possibility degree between 0 and 1 (increasing it or decreasing it), and has no effect on values that are totally possible.
23.3.4.4. The Connective Operations over the Family of Labels and its Approximation. The results of the connective operations defined in section 23.4.2 are shown in Table 23.1 for the family of labels mentioned in the last section with the two-parameter representation. In some cases the result of such operations will be a label not belonging to our family. Therefore, a first approximation process is performed in order to keep the family closed under the connective operations.

23.3.4.5. Selection of the Term Set and its Linguistic Approximation. MILORD works with two term sets of linguistic certainty values. One, given by the expert, is used externally, and the other one is automatically generated from the first and used transparently by the user. The reason of this is explained later in this section.

Each linguistic certainty value, as it has been explained in section 23.4.3 is internally represented by a fuzzy label, which can be understood as the meaning of the terms in the term sets.

The connective operators applied to these linguistic certainty values will produce another certainty value, that will be matched to a term in
one of the two term sets, depending on the operator used as it will be described later.

In the expert term set $E$ of PNEUMON-I-A (see section 23.4) there are nine linguistic values with the following parametric representation:

\[
\begin{align*}
& t_0: \text{UNKNOWN} & (0,0) \\
& t_1: \text{VERY-LITTLE-POSSIBLE} & (0,0.05) \\
& t_2: \text{SLIGHTLY-POSSIBLE} & (0,0.1) \\
& t_3: \text{MODERATELY-POSSIBLE} & (0,0.65) \\
& t_4: \text{POSSIBLE} & (0,1) \\
& t_5: \text{QUITE-POSSIBLE} & (0.35,1) \\
& t_6: \text{VERY-POSSIBLE} & (0.9,1) \\
& t_7: \text{ALMOST-SURE} & (0.95,1) \\
& t_8: \text{SURE} & (1,1)
\end{align*}
\]

These labels are applicable to facts and rules. If no label is specified, MILORD assumes the default value ALMOST-SURE ($t_7$), instead of SURE ($t_8$) because $t_8$ has a behavior different from the others: it cannot be modified by composition (e.g., $t \cdot t = t_8$ for any $t$), and it is totally incompatible with anything else (e.g., compatibility of $t_8$ with $t$ is $t_8$ for every $t \neq t_8$). Therefore, $t_8$ and, by duality, $t_7$ are reserved for nonfuzzy reasoning.

As mentioned, after every logical operation, an approximation process is performed to keep the expert term set closed. In this process, the extreme labels ($t_0$ and $t_8$) are not considered. For example, the following conjunction operation for three different $T$-norms:

\[
\text{(slightly possible }[[0,0.1]]) \wedge_T \text{(moderately possible }[[0,0.65]]) \rightarrow \\
gives \\
\begin{cases}
(0,0), & \text{if } T = T_1 \\
(0,0.65), & \text{if } T = T_2 \\
(0,0.1), & \text{if } T = T_3
\end{cases}
\]

Then, the linguistic approximation process gives:

very-little-possible (for $0,0$)
very-little-possible (for $0,0.65$), and
slightly-possible (for $0,0.1$)

The linguistic certainty label used by the expert to qualify a rule is internally substituted by its compatibility with the default value (ALMOST SURE). The reason is that the basic inference process:

\[
\text{IF } [(X \text{ is A}) \text{ is } t_i] \text{ THEN } [(Y \text{ is B}) \text{ is } t_j] \text{ is } t_k
\]
(X \text{ is A}) \text{ is } t_i

actually works only with compatibility labels: the first one is the matching degree of the premise ($t_i$); the second one, the certainty value of the rule ($t_j$), is in fact a satisfaction degree; and the last one (INFER($t_i$, $t_j$, $t_k$)), is the result of combining the first two, and has to be interpreted as the matching degree propagated to the conclusion. This degree will be composed with $t_k$ in order to obtain the final certainty degree of the conclusion.

Thus, the labels produced by a compatibility or inference process do not really have the same linguistic interpretation as the labels of the expert term set $E$. Hence, it is meaningless to match them. Moreover, the term set is not large enough to avoid approximation errors. For example, the compatibilities of $t_8$ with $t_8$, $t_8$, $t_8$, and $t_8$ would all be linguistically approximated to $t_8$; in the same way, the compatibilities of $t_8$ with $t_8$, $t_8$, $t_8$, and $t_8$ would all be approximated to $t_8$. Because of this, MILORD generates an extended term set $E^*$ that is only internally used and transparent to the user.

In MILORD the default extended term set contains the following 15 terms:

\[
\begin{align*}
& t_1^* = t_1 \\
& t_2^* = t_2 \\
& t_3^* = t_3 \\
& t_4^* = t_4 \\
& t_5^* = t_5 \\
& t_6^* = t_6 \\
& t_7^* = t_7 \\
& t_8^* = t_8 \\
\end{align*}
\]

And the domain and image of each operator is:

\[
\begin{align*}
\text{OR} &: E \times E \rightarrow E \\
\text{COMPATIBILITY} &: E \times E \rightarrow E^* \\
\text{AND} &: E^* \times E^* \rightarrow E^* \\
\text{INFERENCED} &: E^* \times E^* \rightarrow E^* \\
\text{COMPOSITION} &: E^* \times E \rightarrow E \\
\text{NEGATION} &: E^* \rightarrow E^*
\end{align*}
\]

Let us see an example of the inference process:

RO8004 IF 1. Community acquired pneumonia is almost sure
2. Bacterial disease is possible
3. (No aspiration) is very possible

THEN [Possible]
Enterobacteria is quite possible

Observed facts:

1. Community acquired pneumonia is very possible
2. Bacterial disease is almost sure
Inference steps:

(a) Compatibility between (1) and (1') gives: moderately possible
(b) Compatibility between (2) and (2') gives: almost sure
(c) Negation of (3'): (no aspiration) is (not slightly possible) i.e., very possible
(d) Compatibility between (3) and (c) gives: possible
(e) [(a) and (b)] and (d) gives: moderately possible
(f) Inference in (e) and the rule value (possible) gives: moderately possible
(g) Composition between (f) and the conclusion label (quite possible) gives: possible

23.3.5. Nonmonotonic Reasoning

Let us suppose that at a given moment, the inference engine has deduced a fact 'Y is B' with a certainty fuzzy label $t_y$, and later a new fact 'X is C' provides evidence against 'Y is B'. Then, a mechanism to modify (decrease) the certainty of 'Y is B' is needed. This can be performed introducing a new rule, whose conclusion is not a fact but its certainty expressed by a fuzzy label, that is:

If (X is C) Then ((certainty-of Y is B) is $r^*$)

being $r^*$ a fuzzy label of Type 1 (e.g., $t^*(x) \geq r$).

After applying this rule, the new fuzzy label associated to 'Y is B' will be $t_y^* = t_y * t_b$. Where $t_y^*$ is smaller than $t_y$ because $r^*$ is of type I. But, what is the expected behavior of this type of nonmonotonic rule? It is expected that the less certain is 'X is C', the less the certainty of 'Y is B' will decrease. Taking into account the behavior of the Modus Ponens Inference Rule (23.4), the only way to achieve this is increasing the certainty of 'Y is B' instead of decreasing the certainty of 'Y is B'. Then, the internal formulation of these rules is the following:

[If (X is C) Then ((certainty-of Y is B) is $r^*$)] is $t_b$

So, if we have the fact 'X is C' with a fuzzy label $t_b$, then the inference mechanism will infer:

(\text{certainty-of } Y \text{ is } B) \text{ is } r^{**}

where, $r^{**} = (\text{INFER}(t_b, t_y)) o (-r^*)$

and, finally we obtain:

(Y is B) is $t_y^{**} = (-r^{**}) o t_y$

Remark. $r^*$ is a term of the expert defined term set, but, again, it is strictly associated by its compatibility with the default value (almost sure) and matched against a term of the extended term set. In section 23.4 we give an example of nonmonotonicity.

23.4. Inference Engines

23.4.1. Tabular Representation of Logical Operations

We have pointed out in the preceding sections some advantages of representing the uncertainty by a set of linguistic values. In particular we have seen that this allows precomputing of all the logical operations off-line and storing of the results in matrices. Then, at execution time, the computation of an operation is reduced to accessing a matrix. For example, to apply a rule that contains $n$ conditions would require, in the worst case, $3n + 2$ additive operations, $n + 1$ division operations, and $2n + 1$ comparisons to compute the certainty of the conclusion from the certainty of the conditions and of the rule itself, if the computation is performed on-line, whereas with our approach only $n + 1$ matrix accesses are necessary.

This method has another interesting advantage from the implementation point of view: to perform an access to a matrix there is no need for extra memory space, whereas a function call and the use of local variables needed in other approaches do need extra memory space. This point is very important, in terms of computing time, in languages like LISP that have a dynamic management of memory and produce extra calls to the garbage collector.

The tabulation of the logical operations gives an enormous flexibility to the system. A change in the logical connectives is performed just by changing the matrices. Hence, it is easy for the expert to experiment with different reasoning schemes during the development of an application. The only problem with this approach is that the linguistic approximation introduces an error in the propagation of uncertainty through the rules. This error can be important in a long reasoning chain and a compensation mechanism has to be found. In the application to pneumonia diagnosis, the longest chain so far encountered contains only six rules and the accumulated error was small. Figure 23.11 shows an example of such matrices, in the case of the OR connective, and with the linguistic term set of the pneumonia application.

23.4.2. The Lookahead Technique in the Backward Engine Based on a Generalized Alpha-Beta Pruning Technique

To be accepted, any hypothesis has to be verified with a linguistic certainty value greater than a threshold defined by the expert. This allows application of a lookahead technique in the backward reasoning process, whose advantage is to guide the selection of questions and to minimize the number of hypotheses to be verified.
reach. In the evaluation of the premises, the system calculates again the minimum value that each one of them must reach in order to get over the threshold. Then this value is passed back as the threshold for the conclusions of the rules above $R_i$. This is a recursive procedure. When the process arrives to a non-derivable fact and asks the user for a value, it already knows the minimal value that this fact should have. If the value given by the user is smaller, the process stops (the search tree is cut) and the system backtracks to another rule concluding the hypothesis under evaluation. If there are no more rules, it backtracks to another hypothesis.

In the procedure explained previously, all the computations are, again, performed off-line just once, and the results stored in matrices that represent a kind of inverse operator of the AND, OR, and INFERENCE operators. Figure 23.12 is the inverse operator matrix of the T-norm $T_2$.

The cuts produced by this method are recorded as set values in the internal representation of facts and rules. This is very useful for explanation purposes. There are four possible cuts in MILORD:

1. The system is evaluating a deducible fact and detects that it will not reach the threshold. A report is recorded as the value of the slot EX-C-CUT in the rule representation of that fact. This report states which rule was the cause of the cut and which rules were not yet applied. This cut is propagated forward.

2. The system is evaluating a rule and the conditions do not reach the threshold. The report is recorded as the value of the slot EX-P-CUT in the rule representation frame. This cut is propagated forward.

3. The system is evaluating a rule whose conditions have reached the lowest linguistic certainty value (that cannot be increased, due to the characteristics of the T-norms). A report is recorded as the value of the slot EX-O-CUT in the rule representation frame. This cut is not always propagated.

Figure 23.11 OR matrices computed from the PNEUMON-IA term set.

Traversing the AND/OR search tree from the hypotheses to the non-derivable facts depends on the heuristic selection of rules and conditions in the rules. At each selection step, MILORD checks whether the linguistic values obtained until that moment are sufficiently high to allow the hypothesis to reach the acceptance value.

To do that, the lookahead process starts computing the maximum reachable value (MRV) of the hypothesis, assuming that all the unknown non-derivable conditions have the highest linguistic certainty value in the ordered set of linguistic values defined by the expert. The known non-derivable facts assume their real values. The MRV is a function of the premises that are known at each moment. Given an MRV the system selects a rule $R_i$ and calculates the minimum quantum of certainty that this rule must propagate in order to reach the threshold. Then, this quantum is passed back to its premises and constitutes the threshold they must
4. The system detects that a fact will never reach the threshold because it is greater than the MRV. This cut is recorded as the value of the slot EX-I-CUT in the fact representation frame, and it is propagated forward. This information is very useful during the design of the knowledge base.

23.4.3. Cooperation between Engines

In MILORD, the two engines, forward and backward, can cooperate during the reasoning process. The forward engine is applied to the rule sets that are driven by the conditions, such as metarules or treatment modules. The backward engine is applied to the rule sets that are driven by the goals, such as diagnostic modules. However, in some cases, the two engines may cooperate to perform concrete tasks. When the forward engine reaches a conclusion with a certainty value near the acceptance threshold value, it calls the backward engine in order to try to increase this certainty value over the threshold, by considering other rule paths that conclude the same hypothesis. Furthermore, when the amount of information furnished by the user in the evaluation of a concrete case is large, the backward engine can call the forward engine in order to perform a first inference step that could clarify the possible diagnoses. Also, when new information is voluntarily introduced by the user, a call to the forward engine is performed in order to immediately take into account this new information.

This cooperation is not presently specified in the KB definition. It is internal to MILORD, although it will be soon possible to specify it at definition time using metarules.

23.5. PNEUMON-I A: An Application in Medicine

This application concerns the diagnosis and treatment of pneumonia diseases. This application, as many others in medicine, has a lot of complex and interesting characteristics that serve as a benchmark for expert systems environments. Among them nonmonotonic reasoning, uncertain, and incomplete information can be pointed out.

These problems have several solutions in MILORD. Uncertain reasoning may be accomplished by the methods described in 23.2.3 and 22.2.4. Nonmonotonic reasoning is performed by the inheritance mechanism of the frame oriented representation and by the mechanism described in 23.2.5. The present implementation includes the methodology described in 23.2.3.

Let us see an example of nonmonotonic reasoning. We focus our attention on a smoker, bronchitic, 62-year-old patient with a community-acquired pneumonia, and with grampositive coccus in pairs. First of all, let us assume that the system deduces that it is very possible that the disease has a bacterial origin and only slightly possible that it has an atypical origin. Afterwards, a control metarule deduces that, for a patient with these characteristics, the system should first try to validate the hypothesis: Streptococcus pneumoniae (a kind of bacteria). Now, suppose that there are only two applicable rules in this case:

\[ R02003 \text{ IF} \quad \text{1. Community-acquired pneumonia} \]
\[ \text{2. Respiratory chronic disease} \]
\[ \text{3. Bacterial disease} \]
\[ \text{THEN} \quad \text{[Possible]} \]
\[ \text{Streptococcus pneumoniae} \]

\[ R02026 \text{ IF} \quad \text{1. Community-acquired pneumonia} \]
\[ \text{2. Pleuresy} \]
\[ \text{3. Grampositive coccus in pairs or strings} \]
\[ \text{THEN} \quad \text{[very possible]} \]
\[ \text{Streptococcus pneumoniae} \]

After applying these two rules, it deduces that it is very possible that the bacteria is Streptococcus pneumoniae. Two days later, we obtain an X-ray and observe that cavitations are present, which is evidence against streptococcus. This implies that our certainty of Streptococcus pneumo-

niae was too high and, therefore, it must be decreased. In this case, the following nonmonotonic rule is applied:

\[ R02034 \text{ IF} \quad \text{1) Streptococcus Pneumoniae is quite possible} \]
\[ \text{2) X-ray-cavitations} \]
\[ \text{THEN} \quad \text{[Sure]} \]
\[ \text{The certainty of Streptococcus pneumonia is moderately possible} \]

This rule forces the certainty of Streptococcus pneumoniae to go down according to the certainty degree propagated by the rule as described in 23.2.5. The final result is that the certainty value of Streptococcus pneu-
noniae is slightly possible.

When a diagnosis or a treatment fails it has to be modified. The modification has to take into account the new information related to the failure and the previous diagnosis and treatment. The modification is monitored by metarules whose conditions are relevant to the wrong diagnosis or treatment.

At any time, the user may input new relevant information that, depending on a set of metarules, may trigger a change in the search process, which consists in adding modules to or deleting modules from the list of objectives.

The use of control metarules for rules and modules has led us to design a hierarchical structure of the knowledge base. The Knowledge Base of
PNEUMON-IA has three different contexts: DIAGNOSIS, TREATMENT, and COMPLICATIONS, supervised by a module containing module oriented metarules (see Figure 23.13). The supervising module allows different interactions, depending on what the user is interested in. For example, a user who already knows the diagnosis will only be interested in the treatment. At present, the diagnosis context, almost completed for community acquired pneumonia, contains ten modules: a metarule module and nine rule modules that cluster rules concluding the same germ or family of germs. The treatment context is also in progress, while the complications context will be soon started.

23.6. Conclusion

We have described some aspects of the MILORD system and in particular its architecture and its management of uncertainty. The most relevant features of our approach is the representation of uncertainty by means of expert-defined linguistic statements.

The main advantage of this approach is that once the linguistic values have been defined by the expert, the system computes and stores the matrices corresponding to the different conjunction, disjunction, and implication operations on all the pairs of terms in the term set. When MILORD is run on a particular application, the propagation and combination of uncertainty is performed by simply accessing these precomputed matrices.

This approach also allows management of nonmonotonic reasoning using the uncertain reasoning framework. Furthermore, the frame-based representation allows one to deal with incomplete information through inheritance mechanisms.

The hierarchical knowledge base structure allows expression of the knowledge with rules containing a small number of conditions, increasing in this way the efficiency of the system.

The application to the diagnosis and treatment of pneumonia presently contains around 1000 rules and 350 facts covering more than 95 percent of the community-acquired pneumonia. In the near future, we are planning to cover also hospital-acquired pneumonia as well as post-treatment complications.

Other applications using MILORD include VLSI design [17] and Automatic Control [18]. Future extensions of MILORD will concern the management of imprecise information and an interface with a knowledge acquisition system, based on personal construct psychology, under development in our group. MILORD is implemented in VaxLisp DEC VAX machines.

23.7. References