Management of Statements in Frame Interpreter of CC_SHELL

Wojciech CHOLEWA, Ernest CZOGAŁA
Technical University, Gliwice, Poland

Abstract

CC_SHELL is an expert system shell that can handle several kinds of uncertain data and knowledge simultaneously. The processing of such information is based on the fuzzy logic, possibility theory, as well as frames for statement representation. The shell is composed of the fuzzy production system, frame interpreter and tools for programmers. In this paper we shall focus on the application of frame interpreter for the management of statements in CC_SHELL.

Keywords: expert system shell, approximate statements, frame, object oriented programming.

1. Introduction

It has been assumed that a knowledge base is a collection of statements describing relationships between entities of the real world as well as abstract concepts. Such statements are variously called sentences, clauses, formulas and most often facts. Of course from the direct use of the notion *fact* there may result a lot of misinterpretations, because the particular statements are not independent, real existing facts - they are only some belief that something has happened or has been done.

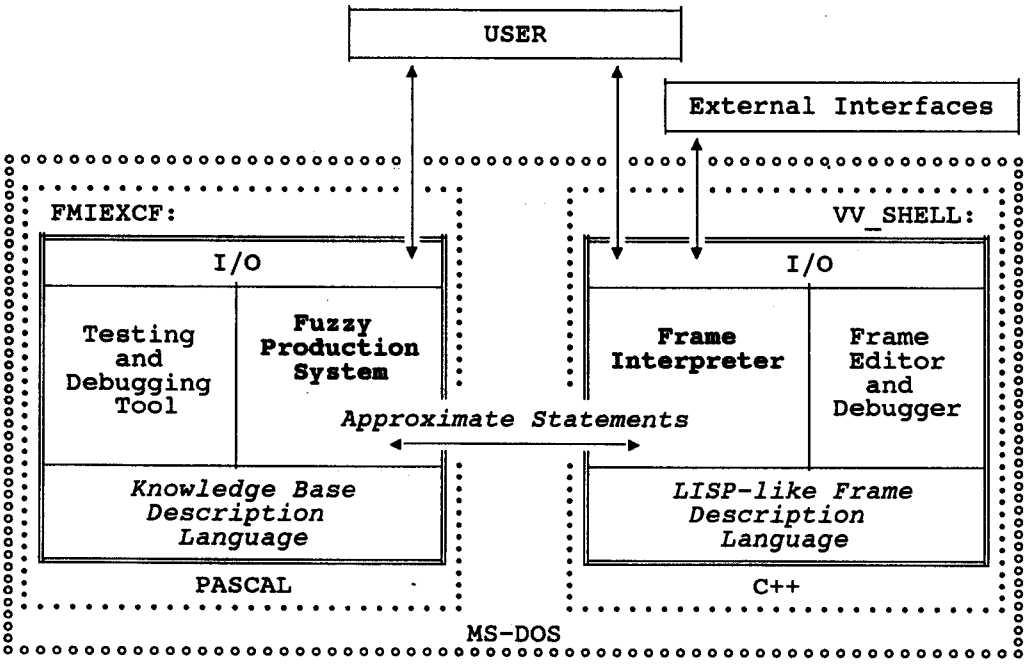

Fig.1. The general structure of CC_SHELL.

The aim of this paper is to show, how statements are managed in an expert system shell. The general structure (Fig.1) and features of CC_SHELL have been discussed in a separate paper.

2. Statements and frames

A common way to represent statements is an object-attribute-value triple used, eg., in a pattern expert system MYCIN [9] dedicated to medical applications. Attributes are general characteristics of properties possessed by objects. The value specifies the particular nature of a property in a given situation. Of course, the sets of such triples are flat (they contain no underlying structure) and the maintenance of great sets is strongly difficult. A dozen of different idea of structuring the knowledge base has been proposed and implemented. Very important is the idea of frames. It is an example of object-oriented-programming, which makes it possible to generalize, classify and generate abstractions.

The notion of frames was introduced by Minsky [8] (see also [1] and [7]). His basic goal was concerned with the designing of a data-base containing encyclopedic knowledge, needed in commonsense reasoning. Frames offer high computational efficiency. They are an interesting tool for the designing of interfaces with users and with external sources of data (eg. measuring devices).

2.1. Frames in CC_SHELL

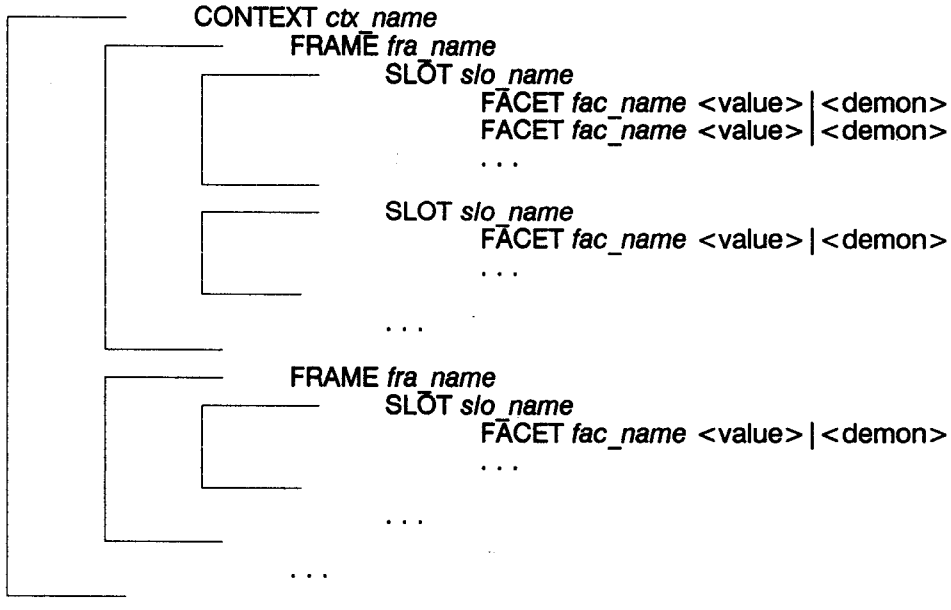

Fig.2. Elements of a frame.

A frame is a description of a real or an abstract object. It is a special form of data and code structure. A frame contains slots representing attributes of the object. Slots may be interpreted as special representations of statements. Slots contain facets connected with values, default values and/or procedures (called demons) by which the values may be obtained (see Fig.2). It is important to point out that each facet can contain values or demons. Such an inclusion of demons in frames joins procedural and declarative representations.

Examples of demons included in frame interpreter of CC_SHELL [4]:

- frame processing demons
  `(COPY fac), (DEL fac), (FIX fac val), (GET slo), (SET val fac),`
  `(VIEW slo), (VIEW_FIX slo), . . .`
- task arranging demons
  `(EXIT errlev), (GOAL val . . .), (IF cond val1 val2),`
  `(RUN_GOAL ctx), (WHILE cond val), . . .`

- list processing demons
  `(HEAD lst), (SEL lst pos), (TAIL lst), . . .`
- uncertainty processing demons
  `(X_AGR sa sb), (X_AND sa sb), (X_IMP sa sb), (X_NOT sa sb), (X_OR sa sb), . . .`
- user interface demons
  `(CONFIRM txt), (DISPLAY val1 val2 ...), (PROMPT typ val1 val2), . . .`

All the elements of frames are identified by their names. The names ought to be locally unique. It means, eg., that all slots in a given frame ought to posses individual, different names. Global uniqueness of names is not required, i.e. we can set the same name for slots in different frames. Such an assumption results in a polymorphism - the names are shared and their meaning depends on the given context.

2.1.1. Inheritance

Frames may be arranged in hierarchical structures (see Fig.3, Fig.4) which make it possible to develop and process the idea about classes without being disturbed by details of any particular object. Such structures are given by links called AKO (a kind of) between superframes (parent frames) and subframes (derived frames). In CC_SHELL such links are listed as follows:

```
FRAME SubFrame_1                                          /* see Fig.3 */
   SLOT ako
      value = (Frame)

FRAME Frame                                               /* see Fig.4 */
   SLOT ako
      value = (SuperFrame_1 SuperFrame_2 SuperFrame_3)
```

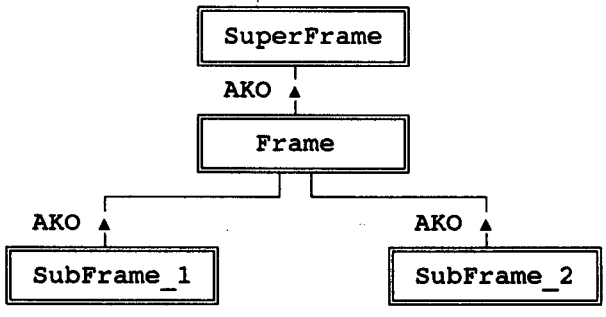

Fig.3. Simple hierarchies (there exists at most only one superframe for each frame).

Searching for a slot value for a frame is the *basic task* in frame systems. Special properties are assigned to the facets: *value*, *if_needed*, *if_added* and *if_removed*. The slot value is assigned to the facet *value*. When we are looking for the value of the slot, the content of the facet *value* is returned. If such a facet is missing, the facet *if_needed* points to the value or to a demon returning the value. Slots that are not present in a frame are inherited from superframes. Searching returns alternatively:

- a list containing values of the same slots in all superframes,
- only the first value is found; this value can override values in other superframes.

Inheritance is the most important feature of frames, which makes it possible to eliminate a redundancy of data and to handle exceptions. It can also be used to generate reasonable default data or assumptions in the case of incomplete information. Special facets, such as *if_added* and *if_needed* may be applied for forward and backward chaining, respectively.
Simple hierarchy results in a tree structure of frames (see Fig.3). For such structures a search path is

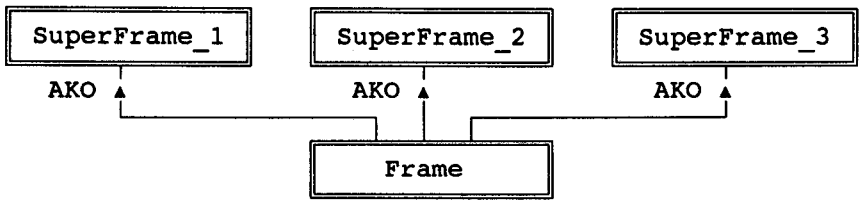

Fig.4. Multiple inheritance (each frame can posses a few superframes).

given by the AKO links, starting from the selected node upwards and search time grows, at worst, linearly with the number of nodes. More advanced multiple inheritance results in a directed acyclic graph (see Fig.4), for which the strategies for *depth first* or *breadth first* searching ought to be applied, where

- the results depend strongly on the arrangement of superframes,
- search time grows, at worst, exponentially with the number of nodes,
- redundant searches may be expected (for depth-first search).

2.1.2. Encapsulation

Encapsulation is a property postulated by object-oriented-programming. It states that data structures and procedures which are to manipulate the data ought to be coupled together and isolated to some degree from direct access by other procedures. The frames enable us to control the degree of encapsulation. This is achieved by the use of demons (making no difference between data and code) and by the following two possibilities for the pointing of a slot:

- slot may be pointed out by its name,
- slot may be pointed out by its qualified name, i.e. by the pair composed of the name of the slot and the name of a frame to which this slot belongs.

In the first case we obtain an access to the slot (for which we are looking) in the current frame. In the second case access is obtained to a particular slot in a given frame. This allows us to use some slots as global and some slots as local (private) ones. In both cases all the assumptions about the inheritance are valued.

2.2. Rules

Mathematical logic was one of the first formalism that was proposed as a representation of knowledge. The inference in logic is connected with the notion of implication. If p and q are given statements, then the implication $p \rightarrow q$ is true when q is true or p is false (see Tab.I). Implication $p \rightarrow q$ is often written as follows

if p then q (1)

Tab.I. Definition of implication; T=true, F=false.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

To provide an appropriate explanation facility, the knowledge base ought to contain the rules in an extended form

if p then q because *explanation* (2)

Classical logic uses two basic patterns for the deduction in propositional calculus:

- modus ponens (rule of detachment) - for statements p and q, if the rule $p \rightarrow q$ is true and p

is true, then q is true, too; we can express this in the following form

$$\begin{array}{ll} p \rightarrow q & \text{rule} \\ \underline{p} & \underline{\text{premise}} \\ q & \text{conclusion} \end{array} \qquad (3)$$

- modus tollens (rule of contrapositive) - for statements p and q, if the rule $p \rightarrow q$ is true and q is false, then p is false; this can be written down in the following form

$$\begin{array}{ll} p \rightarrow q & \text{rule} \\ \underline{\neg q} & \underline{\text{premise}} \\ \neg p & \text{conclusion} \end{array} \qquad (4)$$

If we know only that p is false (or respectively that q is true) we are not able to arrange a reliable inference about the logical value of q (or respectively p). A lot of misunderstandings is connected with the interpretation of the implication. It is necessary to point out that the implication $p \rightarrow q$ doesn't state that q follows from p; eg. both implications

$$(2>1) \rightarrow (3>0) \quad \text{and} \quad (2<1) \rightarrow (3>1)$$

are true (compare Tab.I). Moreover it is important that we can draw reasonable conclusions only when the implication is true, because from the assumption that the implication $p \rightarrow q$ is false it follows only that p has to be true and q has to be false. From the last remark it follows that the knowledge base ought not to contain the rules in the form

It is not true that if p then q.

The conditional part (left-hand side) of a rule may include the composed statements. In pure logic it is enough to consider only two kinds of such a composition, namely the conjunction ($p = p_1$ AND p_2) and disjunction ($p = p_1$ OR p_2). In each real application of expert systems we can easily obtain a situation where the logical value of a statement results independently from several statements (eg. opinions of different experts). Although it seems to be a case of redundancy in expert systems, we are not allowed to reduce this kind of overloading, because designing the knowledge base we don't know which sources of information will be available for the user. In such circumstances the conjunction and disjunction are irrelevant operators and we have to introduce the next one called aggregate ($p = p_1$ AGG p_2). The properties of such an operator have been discussed eg. in [2].

The rules contained in the knowledge base can be driven forward (activated) from those statements that we know to be true towards unknown statements or they can be driven backward from the statements (hypothesis) that we wish to establish to the statements necessary to prove their truth. It is important to make a clear distinction between forward/backward chaining of rules and forward/backward solving strategies used by the expert system (forward/backward reasoning). In addition to distinguishing between forward/backward chaining and reasoning, we also need to distinguish between depth-first and breadth-first search of rules in a knowledge base.

It seems today that experts can express most of their problem-solving techniques as a set of condition-action rules. Rules "if *premise* then *conclusion*" can be very easily extended into production rules "if *condition* then *action*". Such rules provide an extremely powerful model of human thought and allow us to represent the knowledge about how to carry out our reasoning.

3. Approximate statements

Knowledge base results from the experiences of experts. It is not given in a rigorous form and we often have to deal with rules which are true in most (but not all) cases. It means that statements and rules in such applications are often uncertain and/or imprecise. Approximate statements can be represented in a lot of different ways, where certain rules and certain statements can always be taken into account as special cases of an approximate one. Several ad hoc approaches, empirical and theoretically based, to represent approximate statements and rules are known (see eg. [6], [10], [9]).

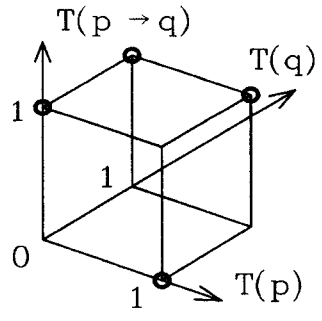

Fig.5. Two-valued implication.

The simplest approach is the direct application of the probability theory and Bayesian standard model. A modification of the probability theory results in the *truth values* T(*s*) from the range [0,1] (or [-1,1]), assigned to each of the statements. They are interpreted as an extension of two logical values NO=0 and YES=1, onto the ordered set [0,1] of real numbers. Particular implementations differ mainly in the interpretation of the value T(*s*)=0, which can point out statements that are false or which can point out only statements for which we haven't got any source of information that they are true. The last case doesn't mean that there exist any reasons to interpret such statements as false.

For reasoning by means of truth value we need an extension of implications. The two-valued implication defined in Tab.I is shown in Fig.5. In order to extend the definition into a definition of continuous implication, which is necessary to deal with truth values $T(s) \in [0,1]$, we have to define a surface (a function) spanned on the points shown in Fig.5. Some examples are (see Fig.6):

$$T(p \to q) = \begin{cases} 1 & \text{if } T(p) \le T(q) \\ T(q) & \text{otherwise} \end{cases} \tag{5}$$

$$T(p \to q) = \max(1 - T(p), T(q)) \tag{6}$$

$$T(p \to q) = \min(1 - T(p) + T(q), 1) \tag{7}$$

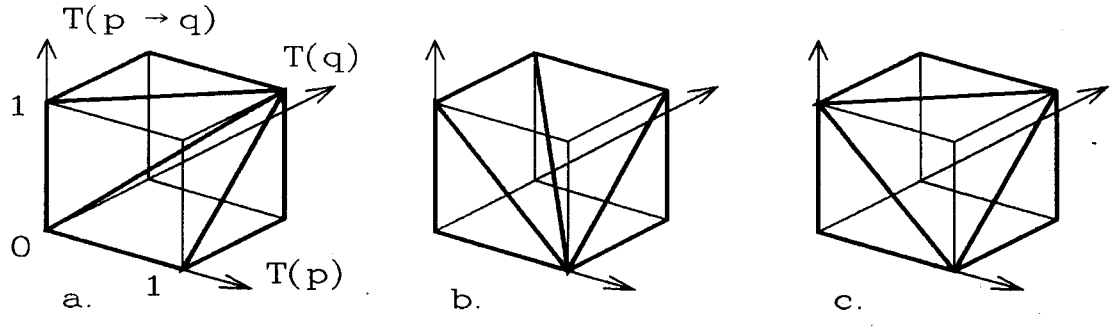

Fig.6. Examples of implication: a) (5), b) (6), c) (7).

On the basis of such implications we can generalize the modus ponens and modus tollens (see eg. [6]).

3.1. Belief and disbelief

The first well-known expert system (designed for medical applications about 1972) which uses uncertain statements and uncertain rules is MYCIN [9]. For a given evidence e (reference statement) and hypothesis h (statement to be evaluated) MYCIN introduces the following three measures:

- measure MB(h,e) of belief in the hypothesis h,
- measure MD(h,e) of disbelief in the hypothesis h,
- certainty factor CF(h,e) considered as a truth value of the hypothesis h, and indicating a predominance of confirming (positive value) or predominance of opposing (negative value) evidence.

Measures of belief and disbelief are interpreted as some kinds of conditional probability, where

$$\mathrm{MB}(\neg h,e) = \mathrm{MD}(h,e) \tag{8}$$

$$\mathrm{CF}(h,e) = \mathrm{MB}(h,e) - \mathrm{MD}(h,e) \tag{9}$$

MYCIN uses the following formulas

$$\begin{aligned} \mathrm{MD}(h_1 \wedge h_2,e) &= \max\left(\mathrm{MD}(h_1,e), \mathrm{MD}(h_2,e)\right) \\ \mathrm{MB}(h_1 \wedge h_2,e) &= \min\left(\mathrm{MB}(h_1,e), \mathrm{MB}(h_2,e)\right) \\ \mathrm{MD}(h_1 \vee h_2,e) &= \min\left(\mathrm{MD}(h_1,e), \mathrm{MD}(h_2,e)\right) \\ \mathrm{MB}(h_1 \vee h_2,e) &= \max\left(\mathrm{MB}(h_1,e), \mathrm{MB}(h_2,e)\right) \end{aligned} \tag{10}$$

Most interesting is that MB(h,e) and MD(h,e) allow us to take into account separately all the premises *pro* and *contra* and we can distinguish between two cases:

- we haven't got any sources of information about the logical value of the hypothesis,
- known premises *pro* are compensated by known premises contra.

We are not able to make such distinctions when we use only a single value (eg. truth value) assigned to the statement, because the value $T(s)=0.5$ maps both these cases.

3.2. Possibility and necessity

An interesting modification of reasoning patterns was obtained by means of modal logic. The notions of possibility and necessity form a concept for the measures of possibility $\Pi(s)$ and necessity $N(s)$, assigned to statements. Leaving out a rigorous explanation the values of these measures may be interpreted as boundaries of a hypothetical range for the unknown truth value

$$0 \le N(s) \le T(s) \le \Pi(s) \le 1 \tag{11}$$

By means of $N(s)$ and $\Pi(s)$ we can distinguish the case of compensated premises *pro* and *contra* $N(s)=\Pi(s)=0.5$ from the case with a lack of information $N(s)=0$ and $\Pi(s)=1$. Some extensions of modus ponens and modus tollens were proposed (see [6]):

$$\frac{N(p \rightarrow q) \geq a \quad N(p) \geq b}{N(q) \geq \min(a,b)} \tag{12}$$

$$\frac{N(p \rightarrow q) \geq a \quad \Pi(q) \leq b}{\Pi(p) \leq \max(1-a,b)} \tag{13}$$

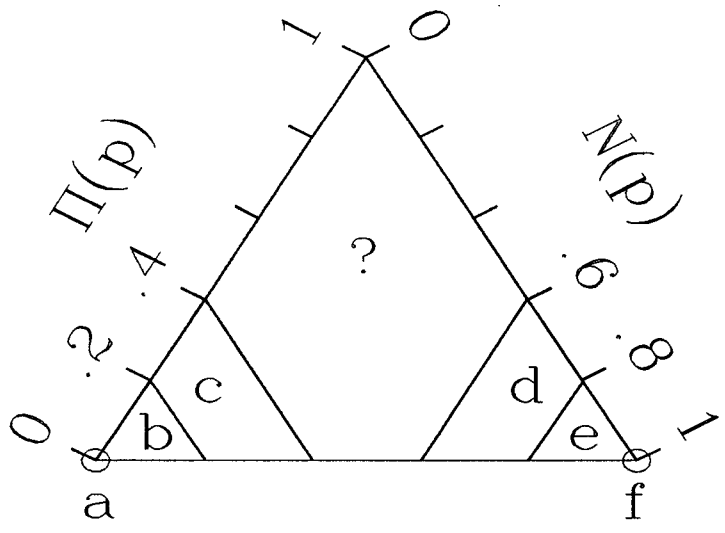

Fig.7. Plot of an approximate statement: a) certainly NO, b) perhaps NO, c) maybe NO, d) maybe YES, e) perhaps YES, f) certainly YES.

The result of reasoning obtained with the use of possibility and necessity may be mapped on the diagram in the form of a triangle (see Fig.7). Such a diagram allows us to introduce linguistic descriptions for the selected pairs of values (N,Π). This simplifies the dialogue with users because verbal descriptions of certainty are more user-friendly than numbers.

3.3. Bilateral implication

The idea of bilateral implication results from technical diagnostics, where a lot of examples of dependencies between the state of an object and features of diagnostic signals exists. They can be interpreted as the following ordered relations:

from the known state follows *a special property of the signal*

Of course we are not able to use such a relation directly for robust reasoning about the state of the object on the basis of known properties of signals (it is the main task in technical diagnostics) because it is highly possible that the same special property of a signal follows from some other state of the object. Moreover we often have no reason to assume that the discussed relation is a causal relation. The simplest way to represent correctly all cross-dependencies is to write the relations in such a form that both modus ponens and modus tollens may be applied together. This can be done by means of bilateral implication [3]. The bilateral implication $p \leftrightarrow q$ is simply a pair of both underlaying implications $p \rightarrow q$ and $q \rightarrow p$. The implication has to be symmetric for modus ponens and tollens because the result of reasoning ought to be independent from particular forms of basic statements:

- statement 1: *object X has the property A*,

- statement 2: *object X has the property* $\neg A$,

and questions sent to the user.

The notion of necessity and possibility and the concept of bilateral implication may be used together. It results in the following joint (generalized) modus ponens and tollens

$$\frac{\begin{array}{l} T(p \rightarrow q) \\ T(q \rightarrow p) \\ N(p), \Pi(p) \text{ such that } N(p) \le T(p) \le \Pi(p) \end{array}}{N(q) \le T(q) \le \Pi(q)} \tag{14}$$

where, eg., for Łukasiewicz's implication (7) we have

$$\begin{aligned} N(q) &= \max(0, N(p) + T(p \rightarrow q) - 1) \le N(p) \\ \Pi(q) &= \min(1, \Pi(p) - T(q \rightarrow p) + 1) \ge \Pi(p) \end{aligned} \tag{15}$$

The process of reasoning can be mapped on the diagram Fig.7, as a sequence of vectors going upstairs. It is easy to see that the certainty of the conclusion can not be better than the certainty of the premise. In order to improve this situation, that means to obtain conclusions that are more certain than the premises, we ought to have some set of independent rules resulting in the same conclusion obtained as an aggregate of partial conclusions.

3.4. Handling of uncertainty in the frame interpreter of CC_SHELL

The value of each statement in the frame interpreter of CC_SHELL is given by three numbers (*n p v*), where *n* - necessity, *p* - possibility, *v* - degree of importance. It has been assumed that:

$$(n\ p) = (n\ p\ 1.0)$$
$$(n) = (n\ 1.0\ 1.0)$$
$$(\ ) = (0.0\ 1.0\ 1.0)$$

and

$$\text{YES} = (1.0\ 1.0\ 1.0) \quad \text{and} \quad \text{NO} = (0.0\ 0.0\ 1.0)$$

The following set of operators is included (for the given statements s_1, s_2) [4]:

- X_AND
  $n = \min(n_1, n_2)$
  $p = \min(p_1, p_2)$
  $v = \max(v_1, v_2)$
- X_OR
  $n = \max(n_1, n_2)$
  $p = \max(p_1, p_2)$
  $v = \min(v_1, v_2)$
- X_IMP
  $n = \min(n_1, n_2)$
  $p = \max(p_1, p_2)$
  $v = v_1 * v_2$
- X_AGR
  $n = (n_1 * v_1 + n_2 * v_2) / (v_1 + v_2)$
  $p = (p_1 * v_1 + p_2 * v_2) / (v_1 + v_2)$
  $v = v_1 + v_2$

- X_NOT

$$n = 1 - n_1$$
$$p = 1 - p_1$$
$$v = v_1$$

4. Acknowledgment

Both authors want to express their gratitude to the Alexander von Humboldt Foundation for their constant support.

5. References

[1] BARTLETT F.C.: *Remembering.* The University Press, Cambridge 1932.

[2] CHOLEWA W.: *Aggregation of fuzzy opinions - an axiomatic approach.* Fuzzy Sets and Systems, 17, 1985, p.249-258.

[3] CHOLEWA W.: *Reciprocal fuzzy implication.* First Joint IFSA-EC and EURO-WG Workshop on Progress in Fuzzy Sets. Abstracts. Warszawa 1986, p.19.

[4] CHOLEWA W. et al: *VV_SHELL User's Guide and Reference Manual* (in Polish). Report G-547/RMT-4/91, Technical University, Gliwice 1991.

[5] CZOGAŁA E. et al: *Approximate Reasoning in Knowledge-Based Systems* (in Polish), Report RRI.14/T/15/52/90, Technical University, Gliwice 1990.

[6] DUBOIS D., PRADE H.: *Possibility theory - An approach to computerized processing of uncertainty.* Plenum Press, New York 1988.

[7] GOFFMAN E.: *Frame Analysis.* Harper & Row, New York 1974.

[8] MINSKY M.: *A Framework for Representing Knowledge.* [in:] Computers and Thought. [ed.:] WINSTON P.H.; McGraw-Hill, New York 1975, p.211-277.

[9] SHORTLIFFE E.H.: *Computer-based medical consultation MYCIN.* Elsevier, New York 1976.

[10] ZADEH L.A.: *The role of fuzzy logic in the management of uncertainty in expert systems.* Esevier Science Publishers 1983.