A FORMALISM FOR FUZZY BUSINESS RULES

Vasile Mazilescu
vasile.mazilescu@ugal.ro
Dunarea de Jos University of Galati, Romania

The aim of this paper is to provide a formalism for fuzzy rule bases, included in our prototype system *FUZZY_ENTERPRISE*. This framework can be used in Distributed Knowledge Management Systems (DKMSs), real-time interdisciplinary decision making systems, that often require increasing technical support to high quality decisions in a timely manner. The language of the first-degree predicates facilitates the formulation of complex knowledge in a rigorous way, imposing appropriate reasoning techniques.

Keywords: Fuzzy Enterprise Framework, Knowledge Representation, Fuzzy Processing

1. Introduction

Decision Support Systems (DSS), as a kind of interactive computer-based information systems, help decision makers utilize data and models to solve mostly semi-structured or un-structured decision problems in practice. Intelligent DSS, along with knowledge-based decision analysis models and methods, incorporate well databases, model bases and intellectual resources of individuals or groups to improve the quality of complex decisions (Bergmann et al. 2009, Walzer et al. 2008, Mazilescu 2012). In the recent years, multi-criteria DSS, group DSS, and Web-based Customer Recommender Systems have had unimaginable developments and improvements in dealing with complex, uncertain, and un-structured decision problems under the support of computational intelligent technologies (Wagner et al. 2001). DKMS implies human resources management, ICT methods and AI techniques for which representing and processing of knowledge are key problems. The correct management of knowledge, organized in the distributed knowledge models and exploited by an Enterprise Control Support System (ECSS), can raise the competition strength of the enterprise only through the complete integration of the technological aspects within the human and organizational ones. A well defined knowledge structure, which is based on the inclusion of heterogeneous and distributed information which may be used by any member of the enterprise at any moment, represents the basis for a knowledge ontology. The ontologies are just the first step in the knowledge structuring and managing of distributed knowledge, implemented at the level of the different agents with which the enterprise may necessarily interact. The goal proposed for development within the project *FUZZY_ENTERPRISE_PR357* is to integrate the multiple sources of knowledge regarding products, services, financial resources, firms and associations in order to offer a rigid, unique, interactive resource process for enterprise leadership as well as other components such as distributed agents, which may integrate different knowledge models (Mazilescu 2010, Quian 1992).

Knowledge representation and exploitation within an expert system are rather conflicting characteristics, whereas the increase in knowledge representation power reduces system's efficiency and increases the difficulty of developing it. Many Artificial Intelligence problems are difficult to solve from the computational point of view. An observation which may help to reduce this complexity is that often these problems have the following property: inputs can be divided into two parts, of which, a part of these inputs is relatively constant long time, compared with the second part. In such situations, seems right to make some changes on the constant part, in order to reduce the time of obtaining the solution for the AI problem, if the second part varies, but is known at certain moments of time. Transformations made in advance are called pre-processing or knowledge compilation. Using variables in an expert system allows knowledge factorization. First order predicates language facilitates expressing complex knowledge rigorously, imposing appropriate reasoning techniques. Definition of certain propagation and inference procedures for real-time expert systems, involves the development of powerful reasoning mechanisms, as well as adapting the control algorithms to the state spaces, which are often very large. The FRCOM system is conceptually based on all the properties summarized above and consists of the compiled fuzzy rules base (management model) and the inference engine. To highlight the characteristics of this system, the following elements must be described: i) The formalism, specifying the types of knowledge supported by the system. Are presented, in order, the fuzzy knowledge syntax and the basic features of compiled linguistic models, system's parameters, the elementary fuzzy filtering, the compatibility of possibility distributions for generalized modus ponens inference scheme; ii) Compiler design, which includes the static structural discrimination component of the fuzzy state, the fuzzy unification as the basic element of the structure of compiled fuzzy knowledge (aimed at checking the consistency of fuzzy substitutions), the algorithm that generates the variables binding network. Network termination nodes correspond bijectively to the rules from the rules base; iii) Inference engine algorithm based on fuzzy logic, which includes techniques for reasoning in the presence of compiled imprecise knowledge. As a first step in the practical implementation of any fuzzy expert system, knowledge representation aims to describe the problem domain as a model that includes relational entities and symbols, according to an appropriate formalism. Section 2 is about general characteristics of our framework. We provide an extended framework for a compiled fuzzy rule base, included in our prototype system FRCOM. Section 3 points out concluding remarks.

2. The Fuzzy Formalism

The types of knowledge accepted by FRCOM system are: i) variables (symbols always preceded by '?', as ?x, ?y, which will occur only in rules); ii) atomic constants (numbers or strings); iii) possibility distributions or fuzzy constants (symbols always preceded by the character '*' and used to represent imprecision); iv) logical operators. Possibility distribution can take any form. This complexity can cause a number of difficulties for the application of possibility theory. In economic practice, when the variables are numeric, it appears that a trapezoidal possibility distribution on continuous referentials is well suited. It can be represented through four parameters (g, d, φ, δ). Trapezoidal form of possibility distributions is preserved in most of the inference and calculation operations. All the fuzzy constants used in knowledge representation and modelling, for the synthesis of fuzzy reasoning algorithms, are represented by trapezoidal possibility distributions, such as $g \le d$, φ, $\delta \ge 0$, called T-numbers. Fuzzy constants can occur both in facts and rules, and are always associated to fuzzy sets (T-numbers) through *constfaz* function. Within FRCOM we can equate the fuzzy set to a fuzzy constant. Undefined fuzzy constants are not allowed. A fuzzy constant has always a value corresponding to a continuous, trapezoidal and normalized fuzzy set. Using possibility distributions provides a unified framework for representing imprecision and uncertainty. Parameter ζ is used to measure fuzzy sets uncertainty $(0 \le \zeta \le 1)$. If a fuzzy set is uncertain, the parameter ζ must be defined in *constfaz* function through a list (uncertain ζ). We admit that a completely uncertain fuzzy set $(\zeta = 1)$ has no effect on system's behaviour. In contrast to facts, a reason is a structured list in which variables may occur. This signifies the presence of variables, atomic constants and of fuzzy constants within reasons' structure.

In addition, the reasons *(named also* motives*) may occur in both the conditional part and in rules' conclusion. Uncertainty is allowed in the conditional part and in the consequent of generalized modus ponens inference scheme, only if a particular linguistic model requires it. In most expert systems, the rules within the rules base are independent. Independence indicates clearly that no other rule interferes with the consequences inferred by any of the rules. Such rules are called normal rules. For fuzzy knowledge processing, the rules' independence hypothesis is not always valid. It is the case of incomplete rules which infer conclusions based on a collection of dependent rules. Appears obvious the syntax difference between the two types of rules, as follows:*

independent_rule:: = (R antecedent name → consequent)

where *name* is an atom indicating the name of the rule, *antecedent* is a conjunction of several reasons and predicates, *consequent* is a sequence of actions. Normal and incomplete rules' antecedent is a conjunction of reasons and predicates. During the stage, the reasons will be correlated with the facts stored in the facts base. In order to increase the knowledge representation capacity, are introduced *predicates* that appear as reasons in the left side of the rules. It is highlighted the presence, within the management model structure, of predicates F(a), which are flags that emulate human reasoning sequences, in order to achieve the management expert system's goal. The knowledge representation formalism for FRCOM system is:

antecedent :: = condition*
condition :: = reason | reason reason_index | predicate

reason_index	:: =< atomic constant >
reason	:: = expression in which are allowed the three types of data;
predicate	:: = (predicate_sym predicate_arg predicate_arg)
predicate_arg	:: = predicate | atomic constant | fuzzy constant | variable
predicate_sym	:: = {=*, *<, >*, *Π, *N}
consequent	:: = conclusion*
conclusion	:: = reason | reason reason_index | predicate | procedure

Predicates =*, *< and >* have binary values, while *Π, *N are fuzzy predicates. If fuzzy linguistic models are introduced in an expert system, it becomes more complicated due to considering the fuzzy processing at all system's levels, such as: fuzzy filtering, imprecise sets compatibility, fuzzy unification, calculation of the inferred conclusion together with the calculation of propagation for the parameters that manage imprecision, selection of strategies in which are naturally embedded also elements of factual knowledge imprecision. A major difficulty in the use of expert systems in DKMSs is the unpredictability of the rules' execution time. While a series of works have dealt with the problem of optimizing the response time of an expert system based on production rules, by developing more competitive algorithms or parallel hardware structures, the approaches to guaranteeing the response time using filtering algorithms, such as Rete algorithm (Shvartzshnaider et al. 2010). In the next paragraph we present the problems in the filtering stage of the inference engine's cycle, emphasizing the classic characteristics of a knowledge compiler.

Problem solving is the process which consists of searching within a search space of the problem and of knowledge search. We are concentrating on Rete search algorithm, adapted for fuzzy knowledge. We believe that the execution time in knowledge search is the basis for planning activities. Software and hardware improvement is a possible approach to obtaining certain response times. To obtain such response times, is necessary to elaborate a predictability mechanism for filtering time, to evaluate the response times. Also, has been shown the NP-completeness of Rete algorithm.

Execution time prediction is a distinct problem. There are three basic aspects of real-time problems: execution speed, ability to react after a certain time and sensitivity (system's ability to react to external interruptions during the reasoning process). Production systems are characterized by a rules base and a facts base, each rule being characterized by a left side and a right side. On the right, there are frequently three functions likely to be achieved: *add* an element in the facts base, *modify* or *delete* an existing fact. Such a system repeats, during its execution, the basic cycle. Most of the execution time of an expert system is used for the filtering stage, Rete algorithm aiming to improve the filtering stage time. The elements moving within the network are called chips. The network consists of different types of nodes, through which chips may pass, being stored in α-memories. Copies of chips that satisfy certain conditions are transmitted to successive nodes. The most used node is the one with two inputs. Chips arriving at the nodes with two inputs are compared with the chips stored in the opposite memory of chips. In this way, the chips can flow through the network until they reach the terminal nodes. If the antecedent of a rule is instantiated at some point by a set of chips, then the rule in question reaches the conflicts set. We are interested in the time in which a chip covers the Rete network, as the basic problem in assessing the problem solving time of an expert system based on compiled knowledge structure.

Expert systems' compiling technique is applied exactly in the filtering stage, aiming to improve the filtering efficiency by avoiding exhaustive browsing of the rule base. Working memory content represents the expert system's state x^{SEC} at a time $k \in T$. Classic filtering algorithm analysis illustrates the inefficiency of the inference engine's control cycle, which has to perform numerous unnecessary operations. It forgets the information acquired in previous cycles and executes the procedure of correlating all the rules' conditions with all the facts in the facts base. Noticing all these shortcomings, there have been proposed compilation techniques based on the fact that, in every control cycle, only the facts that change will occur in the set of conflicts. Changes in the facts base are very few compared with the number of control cycles executed by the inference engine. Knowledge compiling allows: determining the characteristics to be tested for each filter, so that only the facts with the same characteristics to be able to unify with the filter; organizing a global network that allows sharing common structures; maintaining, within the network, the information obtained from previous cycles and allowing the propagation of changed facts.

Improving condition-fact pattern-matching is based on the following observation: "There may be maintained some characteristics for the conditions from the conditional part of the rules (which refers to conditions' syntax). Using these characteristics we can develop a tree that will discriminate the conditions in small groups or individual conditions. With the occurrence of a fact, its characteristics are tested. If the fact has the same characteristics as the cause, it may be an instance of the condition. In this way, we determine a minimum set of conditions which are able to filter this fact". For example, for Rete compiler, the discrimination tree is a *test tree* which is constructed as follows: each condition in the rules base is compiled into a series of single-cause tests, considered as features related to the length of the elements (if the condition is a list) and of the condition's constants. The set of conditions is grouped into a network with a common root, called distribution node. The test tree can be generated incrementally. Knowing the tree corresponding to a set of causes $\{C_1, ..., C_{i-1}\}$, the testing sequence for the cause C_i will be a new path added to this tree, starting from the root or from a sub-tree. The test tree will have as much leaves as different conditions existing in the left side of all rules in the rules base. The test tree will be used to assess any changes in the facts base. Propagation of facts is thus easy to manage. The test tree, even if considerably improves the condition-fact filtering operation, has also some disadvantages: the data can traverse the network on multiple paths. The use of binary tests makes the same information to be extracted several times by the same fact. Given these disadvantages, we propose a *unification tree* structure*.*This tree, like the test tree, discriminates the conditions but in a more efficient way. Relative to the complexity of algorithms, we can mention: **i)** For *classical filtering algorithm* its value is **O**(|BF| |AB|); **ii)** The cost of *the test tree* is **O**(|BF|). In the worst case, there is no common condition among rules, or **O**(log|BF|) for the most favourable hypotheses; **iii)** For the *classical unification tree,* complexity is **O**(log k), where k is the number of different conditions within the rules base. There can be formulated also the problem of unification tree optimization, but it belongs to the class of NP-complete problems. In addition, the characteristics evaluation time and the frequency of events occurrence in the facts base are often unknown. *Improving the variable binding***.** This stage aims to determine the global instances of the rules. Internal and graphic representation of the rules is achieved through a *variables binding network,* which is developed taking into account the discrimination tree's leaves. It definitely reflects the relationship between the rules' conditions. The relationship between the conditions of the rules is seen in the relationship between conditions' variables. Let us examine the variables binding network for the rule R_1: If (A (B ?x) C) (B ?y ?x) (C D ?x ?y) then ... In this situation, the relationship between the conditions of R_1 is formulated through testing the equality of variable x? from the cause (A (B ?x) C) and (B ?y ?x). The α-memories serve to store the set of instances of reasons (5 facts), while β-memories are used to store a conjunction of partial instances, corresponding to the global antecedent of the rule. The network is built incrementally through iterative procedure, as follows: i) Choose two parts of the antecedent of a rule (reason or conjunction of reasons); ii) Build a *connecting node* to store the relationships needed to be checked by the two parties (their conjunction); iii) Create a β-memory, allowing storage of the instances that satisfy the tests present at node level, within the variables binding network structure. The procedure is repeated until is obtained an instance at the level of the global antecedent of a rule.

Generalized Modus Ponens Inference Scheme

A rule can be represented based on a conditional possibility distribution. Let a relationship between two variables X and Y respectively defined on the universes of discourse U and V, which is a restriction of possible values of Y when is assumed that the variable X takes certain values. It can be expressed through a function $\pi_{Y|X}: V \times U \rightarrow [0,1]$. This function is called conditional possibility distribution and is the degree of possibility that Y takes the value v, given that X takes the value u. If π_X is a possibility distribution which narrows a priori the possible values of X, we can calculate the possibility distribution $\pi_{X|Y}$ which narrows the possible values of the pair (u, v), through the combination of $\pi_{Y|X}$ and π_X. For $(\forall)$ $v \in V$ and $(\forall)$ $u \in U$, takes place the relationship $\pi_{X|Y}(u, v) = \min(\pi_X(u), \pi_{Y|X}(v, u))$. By projection, is obtained the possibility distribution of $\pi_Y(v)$, which narrows the possible values of variable Y, given by:

$$(\forall)v \in V, \pi_Y(v) = \sup_{u \in U} \pi_{X_Y}(u, v) = \sup_{u \in U} \min\left(\pi_X(u), \pi_{Y_{|X}}(v, u)\right)$$

In practice, one rule is not sufficient to fully describe a relationship, this requiring a collection of rules. If the relationship is represented by a single rule, then we get:

$$(\forall)v \in V, \mu_B(v) \geq \sup_{u \in U} \min\left(\pi_A(u), \pi_{Y_{|X}}(v, u)\right)$$

Inequality comes from the fact that is wanted "Y is B" when "Y is B", as soon as B′ is enclosed in B. There are several possible solutions, but we are essentially interested in the solution which is the largest (with respect to fuzzy sets inclusions), since this is the least restrictive (denoted $\pi'_{Y|X}(v, u)$), where:

$$\pi'_{X|Y}(v, u)=\begin{cases}0, \text{ if } \mu_A(u)\leq\mu_B(v)\\ \mu_B(v), \text{ else}\end{cases}$$

Generalized Modus Ponens inference scheme *features* are: **i)** the conclusion B′ obtained by applying it, cannot be more specific or more restrictive than the rule consequent B, i.e. $B \subseteq B'$ if A′ is normalized; **ii)** the more restrictive is A′, the more restrictive is the conclusion B′, i.e. for $A'' \subseteq A'$, it follows $B'' \subseteq B'$; **iii)** if there is no perfect filtering between the given fact A′ and the rule antecedent A, then occurs a level of indeterminacy θ. This level is a measure of the possibility that the value of X is outside the support of A, given that "X is A′", or a measure of the possibility that X is not A, given that "X is A′"; **iv)** in practice, the scheme is expressed as an operator which is represented at semantic level as a conditional possibility distribution. Introduced by Zadeh, the scheme has the following solution:

$$(\forall)v\in V,\ \mu_{B'}(v)=\sup_{u\in U}\min\left(\pi_{A'}(u),\pi'_{Y|X}\right)$$

where $\pi'_{Y|X}$ can be denoted by $\xrightarrow{G}$. In this case, this relationship can be re-written:

$$(\forall)v\in V,\ \mu_{B'}(v)=\sup_{u\in U}\min\left(\pi_{A'}(u),\pi_A(u)\xrightarrow{G}\pi_B(u)\right) \text{ or simply } B'=A \circ (A\xrightarrow{G}B)$$

Let the calculation relationship for $\mu_{B'}(u)$ be a function that transforms μ_B into $\mu_{B'}$, during which occur two phenomena: **i)** a global level of indeterminacy of B′, for the values which are outside the support of B; **ii)** calculating the core of the inferred conclusion B′, which is determined by enlarging the core of the set B.

Definition 1. The degree of indetermination in the conclusion inferred through generalized modus ponens scheme, is the degree of possibility that A′ is not included in the support of set A, and the complement to 1 of this degree is to what extent it is necessary that Y takes values in A. The degree of indeterminacy is noted with θ and is calculated as follows:

$$\theta = \sup\{\mu_{A'}(u)\},\ u\in\{u\in U, \mu_A(u)=0\}$$

Let $\mu_A = tp(g, d, \varphi, \delta)$ and $\mu_{A'} = tp(g', d', \varphi', \delta')$. We get:

$$\theta=\begin{cases}\max(\mu_{A'}(g-\varphi),(\mu_{A'}(d+\delta)), \text{ if } Ker(A')\cap Supp(A)\neq\emptyset\\ 1, \text{ else}\end{cases}$$

The degree of indetermination θ depends on the position of A′ with respect to A.

$$\theta_1=\mu_{A'}(g-\varphi),\ \theta_2=\mu_{A'}(d+\delta)$$

where $Supp(\mu_A) = (g-\varphi; d+\delta) \Rightarrow \theta = \max\{\theta_1, \theta_2\} = \theta_1$. We may see in figure 8 that if $g'\leq g-\varphi$ or $d'\geq d+\delta$, then θ = 1, i.e. conclusion B′ is completely uncertain. The degree of indeterminacy occurs whenever $Supp(A') \not\subseteq Supp(A)$. If $Supp(A) \supseteq Supp(A')$, then θ = 0. We may see the degree of indeterminacy exactly as the maximum value in *Supp*(A′) which is not included in *Supp*(A).

Definition 2. The core of the inferred conclusion B′, denoted K, is to determine for which interval in V, $\mu_{B'}(v) = 1$. According to relationship

$$\mu_{B'}(v)= 1 \Rightarrow \sup_{u\in U}\min(\pi_{A'}(u),\pi_A(u)\xrightarrow{G}\pi_B(v))=1\Rightarrow(\exists)\ u\in U, \text{ so that } (\pi_{A'}(u)=1) \text{ and } \pi_A(u)\leq\pi_B(v)$$

This is equivalent to the relation:

$$K = \inf(\mu_A(u) \mid u \in \{u \in U \mid \mu_{A'}(u) = 1\}\}$$

We can see that $Ker(\mu_{A'}(u)) = [g', d']$, and obtain $K_1 = \mu_A(g')$ and $K_2 = \mu_A(d')$, which implies $K = \min(K_1, K_2)$. It follows that $\mu_B(v) \geq K$. Parameter K varies as the parameter θ, in connection with the position of A′ against the position of A. If $Ker(\mu_{A'}(u)) \subset Ker(\mu_A(u))$, then K = 1.

For $g' \leq g-\varphi$ or $d' \geq d+\delta$, it follows that K = 0. It is noted that the parameter K is zero if and only if $\theta = 1$, and K > 0 if and only if $\theta < 1$. Knowing the value of K, we can determine the core Ker(B′) of $\mu_{B'}$, which is the K-cut of B. For $\mu_B = (g_B, d_B, \varphi_B, \delta_B)$ and $Ker(\mu_{B'}) = [g_{B'}, d_{B'}]$, the core and the support of the inferred conclusion $\mu_{B'}$ are calculated as follows:

$$g_{B'} = f^1_{B'}(g_B, \varphi_B, K),\ d_{B'} = f^2_{B'}(d_B, \delta_B, K),\ \varphi_{B'} = f^3_{B'}\left(g_{B'}, g_B, \varphi_B, \theta\right),\ \delta_{B'} = f^4_{B'}\left(\delta_B, d_B, d_{B'}, \theta\right)$$

Functions

$$f^1_{B'},\ f^2_{B'},\ f^3_{B'} \text{ and } f^4_{B'}$$

are similar to those presented in (Mazilescu 2012). The core of B′ is completely determined by the value of the parameter K.

For K = 1, we get Ker(B′) = Ker(B). Conversely, if 0<K<1, i.e. the core of A′ is not included in the core of A, but is included in A's support, then Ker (B′) ⊃ Ker(B), which means that the result inferred by a rule is less accurate than the rule's conclusion. This is due to the filtering between condition A and the fact A′, which is not precise. The conclusion obtained by generalized modus ponens always leads to Ker(B′) ⊃ Ker(B), i.e. the result inferred cannot be more precise than the rule's conclusion. The inferred conclusion $\mu_{B'}$ is determined finally with the relationship:

$$\mu_{B'} = \max\left((g_{B'}, d_{B'}, \varphi_{B'}, \delta_{B'}), \theta\right)$$

which shows that the basic element in the generalized modus ponens calculation is to evaluate parameters θ and K. If the rule's fact and condition are low compatible, the inference engine of FRCOM based on generalized modus ponens scheme induces an increase of the inferred conclusion's imprecision. These issues will arise in the case study for the fuzzy case of the flexible production system, when activating the meta-rules along with their chaining, demonstrating the effective use of all these theoretical results. Knowing the significance of the four parameters Π, N, θ, K, we may consider the problem of choosing the reasonable thresholds for the possibility and necessity measures, to determine the facts that do not filter at all the reasons. This choice must be made in close accordance with the values of θ and K parameters, knowing the links between these pairs of parameters. For the generalized modus ponens scheme, we consider that the associated T-numbers are: (*Constfaz* *m(tp g_m, d_m, φ_m, δ_m)), (*constfaz* *c(tp g_c, d_c, φ_c, δ_c)) and (*constfaz* *f(tp φ_f, d_f, φ_f, δ_f)); this scheme proves the following properties:

Proposition 1 a) K = 0 ⇔ θ = 1; **b)** K> 0 ⇔ θ <1; **c)** Conclusion *c′ inferred through generalized modus ponens is totally uncertain ($\mu_{*c'} = 1$) ⇔ θ = 1.

Choosing the right thresholds for measures Π and N is based on the next results.

Proposition 2 N(*m, *f) > 0 ⇔ θ <1. This derives from the observation that: θ < 1 ⇔ *Supp*(*m) ⊃ *Ker*(*f) ⇔ N(*m, *f) >.

We may notice the fulfilment of the condition which ensures the inequality N(*m, (f)> 0 (i.e. θ <1, equivalent with *Supp*(*m) ⊃ *Ker*(*f)) (Figure 1).

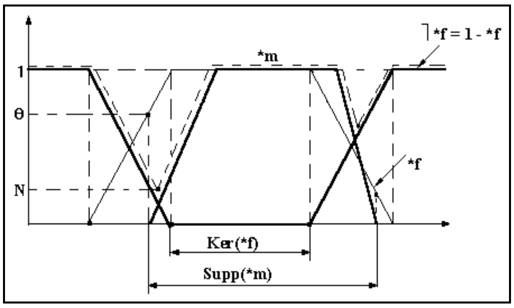

Figure 1. Graphical relationship between N and θ

For the necessity measure and the Generalized Modus Ponens inference scheme, are proved the following equivalent relations: **a)** N > 0 ⇔ θ < 1 ⇔ K > 0; **b)** N(*m, *f) = 0 ⇔ θ = 1; **c)** If N = 1, then θ = 0. Inverse property (θ = 0 ⇒ N = 1) is not always true. Outlining a demonstration for the direct, we get: N = 1 ⇔ *Ker*(*m) ⊇ *Supp*(*f) ⇒ *Supp* (*m) ⊇ *Supp* (*f) ⇔ θ = 0.

The above results show that the necessity measure and the degree of indeterminacy θ are closely linked. Also can be revealed a quantitative relation between θ and N. According to a previous observation, the necessity measure is defined by: N(*m, *f) = 1 – max Π(E_s, *f), Π(E_d, *f)), showing clearly that the terms Π(E_s, *f) and Π(E_d, *f) determine the value of necessity measure. If N(*m, *f) is known, the following two situations may be deducted: 1) Π(E_s, *f) = 1-N and Π(E_d, *f) ≤ 1-N or 2) Π(E_s, *f) ≤ 1-N and Π(E_d, *f) = 1-N.

First, we consider case 1), where Π(E_s, *f) ≥ Π(E_d, *f), understanding the fact that the value of possibility measure between the fuzzy set E_s and the fact (fuzzy) *f determines the necessity measure because:

$$N(*m, *f) = 1 - \mathbf{max}\ (\Pi(E_s, *f), \Pi(E_d, *f)) = 1\ \mathbf{max}(1-N, \Pi(E_d, *f) \leq 1-N)) = 1-(1-N) = N$$

It suffices to consider the left side, E_s, of the complement of ⌉*m. We have the situation when Π(E_s, *f) ≥ Π(E_d, *f), and the significance of Π(E_s, *f) is defined by:

$$\Pi(E_s, *f) = \sup \min \mu_{Es}(u_0), \mu_{*f}(u_0))$$

Following the above observations, we get:

$$a)\ 0 < \theta_d < \theta_s < 1, \Pi(E_s, *f) \geq \Pi(E_d, *f), \theta = \max(\theta_s, \theta_d) = \theta_s$$
$$b)\ 0 < \theta_d < \theta_s < 1, \Pi(E_s, *f) \leq \Pi(E_d, *f), \theta = \max(\theta_s, \theta_d) = \theta_d$$

For N(*m, *f) = 1 - max (Π(E_s, *f), Π(E_d, *f)), with max (E_s, E_d) = ⌉*m, we get:

$$a)\ \text{if } \Pi(E_s, *f) \geq \Pi(E_d, *f), \text{ then } \theta = \theta_s$$
$$b)\ \Pi(E_s, *f) \leq \Pi(E_d, *f), \text{ then } \theta = \theta_d$$

Based on these observations, we may conclude that the values of parameters θ and K can be considered as a type of filtering between a fact and the rule's antecedent, when applying generalized modus ponens scheme. Necessity N and the degree of indeterminacy play a similar role in evaluating the basic filtering degree. Thus, it becomes natural to select as threshold for the necessity measure in the filtering stage (for the inference engine that uses generalized modus ponens scheme) the value N > 0, as it ensures consistency between fuzzy filtering and generalized modus ponens inference scheme. Following the analysis in this paragraph, and correlating it with the values of possibility and necessity measures also used in the fuzzy variables binding stage, we settle as a criterion for fuzzy filtering within FRCOM, the simultaneous fulfilment of conditions:

$$\Pi(*m, *f) = 1 \text{ and } N(*m, *f) > 0$$

which ensure the preservation of normality for the possibility distributions derived from the logical inferences.

A Fuzzy Knowledge Model

Predicates ≥* and =* within the structure of fuzzy rules base allow imprecise comparison of the values of instant loads X1, ..., X6, for subsystems M1, ..., M6 respectively. In this case, the fuzzy management model becomes:

1. *If* ((≥*(X1 ?x)(X2 ?y))∧(≥*(X1 ?x)(X3 ?z)) ∧ (≥*(X1 ?x)(X4 ?v)) ∧ (≥* (X1 ?x)(X5 ?w)) ∧ (≥* (X1 ?x)(X6 ?ξ)) ∧ (¬(Fa1 ?v_1)) ∧ (¬(Fa3 ?v_3)) ∧ (¬(=*(X1 ?x)(X2 ?y))) ∧ (ge *n))
then ((x1b 1) ∧ (x3b 0) ∧ (x5b 0) ∧ (x7b 0) ∧ (x9b 0) ∧ (x10b 0) ∧ (x11b 0) ∧ (x2b 1))
2. *If* ((≥*(X1 ?x)(X2 ?y)) ∧ (≥*(X1 ?x)(X3 ?z)) ∧ (≥*(X1 ?x)(X4 ?v)) ∧ (≥*(X1 ?x)(X5 ?w)) ∧ (≥*(X1 ?x)(X6 ?ξ)) ∧ (¬(Fa2 ?v_2)) ∧ (¬(=*(X1 ?x)(X3 ?z))) ∧ (≥* (X1 ?x)(CFS *zero)) ∧ (ge *n))
then ((x1b 2) ∧ (x2b 0) ∧ (x4b 0) ∧ (x6b 0) ∧ (x7b 0) ∧ (x9b 0) ∧ (x10b 0) ∧ (x11b 0) ∧ (x3b 1))
3. *If* ((≥*(X2 ?y)(X1 ?x)) ∧ (≥*(X2 ?y)(X3 ?z)) ∧ (≥* (X2 ?y)(X4 ?v)) ∧ (≥* (X2 ?y)(X5 ?w)) ∧ (≥*(X2 ?y) (X6 ?ξ)) ∧ (¬(Fa3 ?v_3)) ∧ (¬(Fa1 ?v_1)) ∧ (¬(=*(X1 ?x) (X2 ?y))) ∧ (ge *n))
then ((x1b 3) ∧ (x3b 0) ∧ (x5b 0) ∧ (x7b 0) ∧ (x9b 0) ∧ (x10b 0) ∧ (x11b 0) ∧ (x4b 1))
4. *If* ((≥*(X3 ?z)(X1 ?x)) ∧ (≥* (X3 ?z)(X2 ?y)) ∧ (≥* (X3 ?z)(X4 ?v)) ∧ (≥* (X3 ?z)(X5 ?w)) ∧ (≥*(X3 ?z)(X6 ?ξ)) ∧ (¬(Fa4 ?v_4)) ∧ (¬(=*(X3 ?z)(X5 ?w))) ∧ (≥*(X1 ?x)(CFS *zero)) ∧ (ge *n))
then ((x1b 4) ∧ (x2b 0) ∧ (x4b 0) ∧ (x6b 0) ∧ (x7b 0) ∧ (x9b 0) ∧ (x10b 0) ∧ (x11b 0) ∧ (x5b 1))
5. *If* ((≥*(X3 ?z)(X1 ?x)) ∧ (≥* (X3 ?z)(X2 ?y)) ∧ (≥* (X3 ?z)(X4 ?v)) ∧ (≥* (X3 ?z)(X5 ?w)) ∧ (≥* (X3 ?z)(X6 ?ξ)) ∧ (¬(Fa5 ?v_5)) ∧ (=* (X1 ?x) (CFS *zero)) ∧ (ge *n))
then ((x1b 5) ∧ (x2b 0) ∧ (x3b 0)∧(x4b 0) ∧ (x5b 0) ∧ (x7b 0) ∧ (x9b 0) ∧ (x10b 0) ∧ (x11b 0) ∧ (x6b 1))

6. *If* ((≥* (X4 ?v) (X1 ?x)) ∧ (≥* (X4 ?v) (X2 ?y)) ∧ (≥* (X4 ?v) (X3 ?z)) ∧ (≥* (X4 ?v) (X5 ?w)) ∧ (≥* (X4 ?v) (X6 ?ξ)) ∧ (¬(Fa6 ?v_6)) ∧ (¬(=* (X4 ?v) (X2 ?y)))) ∧ (≥* (X1 ?x) (CFS *zero)) ∧ (ge *n))
then ((x1b 6) ∧ (x2b 0) ∧ (x3b 0) ∧ (x4b 0)∧(x5b 0) ∧ (x8b 0) ∧ (x9b 0) ∧ (x10b 0) ∧ (x11b 0) ∧ (x7b 1))
7. *If* ((≥* (X4 ?v)(X1 ?x)) ∧ (≥*(X4 ?v)(X2 ?y)) ∧ (≥* (X4 ?w)(X3 ?z)) ∧ (≥* (X4 ?v) (X5 ?w)) ∧ (≥* (X4 ?v)(X6 ?ξ)) ∧ (¬(Fa7 ?v_7)) ∧ (≥* (X1 ?x) (CFS *zero)) ∧ (ge *n))
then ((x1b 7) ∧ (x2b 0) ∧ (x3b 0) ∧ (x4b 0) ∧ (x5b 0) ∧ (x7b 0) ∧ (x9b 0) ∧ (x10b 0) ∧ (x11b 0) ∧ (x8b 1))
8. *If* ((≥*(X5 ?w) (X1 ?x)) ∧ (≥*(X5 ?w)(X2 ?y)) ∧ (≥*(X5 ?w) (X3 ?z)) ∧ (≥*(X5 ?w)(X4 ?v)) ∧ (≥*(X5 ?w)(X6 ?ξ)) ∧ (¬(Fa8 ?v_8)) ∧ (¬(Fa9 ?v_9)) ∧ (¬ (=* (X5 ?w) (X6 ?ξ))) ∧ (ge *n))
then ((x1b 8) ∧ (x2b 0) ∧ (x3b 0) ∧ (x4b 0) ∧ (x5b 0) ∧ (x7b 0) ∧ (x11b 0) ∧ (x9b 1))
9. *If* ((≥*(X6 ?ξ) (X1 ?x)) ∧ (≥* (X6 ?ξ) (X2 ?y)) ∧ (≥* (X6 ?ξ) (X3 ?z)) ∧ (≥* (X6 ?ξ) (X4 ?v)) ∧ (≥* (X6 ?ξ) (X5 ?w)) ∧ (¬(Fa9 ?v_9)) ∧ (¬(Fa8 ?v_8)) ∧ (¬ (=* (X6 ?ξ) (X4 ?v))) ∧ (ge *n))
then ((x1b 9) ∧ (x2b 0) ∧ (x3b 0) ∧ (x4b 0) ∧ (x5b 0) ∧ (x7b 0) ∧ (x11b 0) ∧ (x10b 1))
10. *If* ((≥* (X6 ?ξ) (X1 ?x)) ∧ (≥* (X6 ?ξ) (X2 ?y)) ∧ (≥* (X6 ?ξ) (X3 ?z)) ∧ (≥* (X6 ?ξ)(X4 ?v)) ∧ (≥*(X6 ?ξ) (X5 ?w)) ∧ (¬(Fa10 ?v_{10})) ∧ (¬(=*(X1 ?x) (X2 ?y)))) ∧ (≥*(X1 ?x)(CFS *zero)) ∧ (ge *n))
then ((x1b 10) ∧ (x2b 0) ∧ (x3b 0) ∧ (x4b 0) ∧ (x5b 0) ∧ (x7b 0) ∧ (x9b 0) ∧ (x10b 0) ∧ (x11b 1))
11. *If* ((=* (X1 ?x) (X2 ?y)) ∧ (=* (X1 ?x) (X3 ?z)) ∧ (=* (X1 ?x) (X4 ?v)) ∧ (=* (X1 ?x) (X5 ?w)) ∧ (=* (X1 ?x) (X6 ?ξ)) ∧ (ge *s)) *then* ((x1b 0) ∧ (sol ?s))
12. *If* (ge *n) ∧ (x12b 1) *then* ((x1b 0) ∧ (x2b 0) ∧ (x3b 0) ∧ (x4b 0) ∧ (x5b 0) ∧ (x6b 0) ∧ (x7b 0) ∧ (x8b 0) (x9b 0) ∧ (x10b 0)) ∧ Init(control module)
13. *If* (X1 ?x) ∧ (X2 ?y) ∧ (X3 ?z) ∧ (X4 ?v) ∧ (ge *n) ∧ (gep_1 *b) *then* (X1 X2 X3 X4 ?v_q) ∧ (gep_1 *b_1)
14. *If* (X1 X2 X3 X4 ?v_q) ∧ (ge *n) ∧ (gep_1 *b) ∧ (d_{56} *ma) *then* (gep_1 *b_1) ∧ Init(r8)
15. *If* (X1 X2 X3 X4 ?v_q) ∧ (ge *n) ∧ (gep_1 *b) ∧ (d_{35} *ma) *then* (gep_1 *b_1) ∧ Init(r4)
16. *If* (X1 ?x) ∧ (X2 ?y) ∧ (X4 ?v) ∧ (ge *n) ∧ (gep_2 *b) *then* (X1 X2 X4 ?v_q) ∧ (gep_2 *b_1)
17. *If* (X1 X2 X4 ?v_q) ∧ (ge *n) ∧ (gep_2 *b) ∧ (d_{13} *ma) *then* (gep_2 *b_1) ∧ Init(r2)
18. *If* (X3 ?z) ∧ (X4 ?v) ∧ (X5 ?w) ∧ (X6 ?ξ) ∧ (ge *n) ∧ (gep_3 *b) *then* (X3 X4 X5 X6 ?v_q) ∧ (gep_3 *b_1)
19. *If* (X3 X4 X5 X6 ?v_q) ∧ (ge *n) ∧ (gep_3 *b) ∧ (d_{21} *ma) *then* (gep_3 *b_1) ∧ Init(r3)
20. *If* (X1 ?x) ∧ (X3 ?z) ∧ (X4 ?v) ∧ (X5 ?w) ∧ (X6 ?ξ) ∧ (ge *n) ∧ (gep_4 *b) *then* (X1 X3 X4 X5 X6 ?v_q) ∧ (gep_4 *b_1)
21. *If* (X1 X3 X4 X5 X6 ?v_q) ∧ (ge *n) ∧ (gep_4 *b) ∧(d_{42} *ma) *then* (gep_4 *b_1) ∧ Init(r6)
22. *If* (X2 ?y)∧(X4 ?v)∧(X5 ?w)∧(X6 ?ξ) ∧ (ge *n) ∧ (gep_5 *b) *then* (X2 X4 X5 X6 ?v_q) ∧ (gep_5 *b_1)
23. *If* (X2 X4 X5 X6 ?v_q) ∧ (ge *n) ∧ (gep_5 *b) ∧ (d_{43} *ma) *then* (gep_5 *b_1) ∧ Init(r7)
24. *If* (X1 X2 X4 ?v_q) ∧ (ge *n) ∧ (gep_2 *b) ∧ (d_{56} *ma) *then* (gep_2 *b_1) ∧ Init(r8)
25. *If* (X3 X4 X5 X6 ?v_q) ∧ (ge *n) ∧ (gep_3 *b) ∧ (d_{13} *ma) *then* (gep_3 *b_1) ∧ Init(r2)

Rules 5 and 7 have priorities 0.3 in order to allocate the priority 0.1 to rules 13, 16, 18, 20, 22, 25, to meet the management strategy. The other rules (2 ,3, 4, 6, 8, 10, 14, 15, 17, 19, 21, 23, 24), except for rules 9 and 1 who had priority 0, will have priority 0.1. Intermediate fuzzy variables gep_1, gep_2, gep_3, gep_4 and gep_5 are degrees of partial balance between groups of subsystems (1, 2, 3, 4), (1, 2, 4), (3, 4, 5, 6), (1, 3, 4, 5, 6) and (2, 4, 5, 6). These variables were used to eliminate the unbalance situations obtained in the crisp case, these being the expert management meta-knowledge in the field. They depend essentially on the structure of flexible production system and are actually imprecise knowledge used in decision synthesis. For each inference cycle must calculate all the corresponding fuzzy values, and generate the updated facts and thus the current state of the management system. The fuzzy management model has the following properties: i) variable gep_1 appears in rules 14 and 15, and its value is recalculated after the filtering between its current fuzzy value and the reason (gep_1 *b) by applying the generalized modus ponens inference scheme. The same calculation method applies to other partial balance degrees; ii) for meta-rules 14, 16, 18, 20 and 22 fuzzy unification is applied differently, requiring the calculation of the resulting fuzzy value for the variable ?v_q from the consequent (according to the results and examples presented in paragraph 2.2.2); iii) variable ?v_q, together with variables gep_1, gep_2, gep_3, gep_4, gep_5, allows chaining the fuzzy rules while the management model is used by the RT-CFKMS inference engine; iv) within the structure of meta-rules 13-25 are present fuzzy variables d_{56}, d_{35}, d_{13}, d_{21}, d_{43}, d_{42}, representing the fuzzy instant differences between subsystem loads referenced by the corresponding indexes, in the appropriate order. The values of these variables can be T-numbers tp *ma(2 gep_j 1 1), j = 1 ... 5 and tp *mi(0 1 0 1). Fuzzy value *b is generated similarly.

3. Conclusions

Formal logic reasoning is embedded in rule-based production systems. Experts' operational knowledge is expressed as rules, each rule gathering information relative to its launching conditions and information on the resulting effects. Following the tests presented and the knowledge representation and exploitation structures within FRCOM, we may outline the following conclusions: i) the formalism chosen for knowledge representation is strong enough to support the representation of some types of knowledge underlying the management decisions synthesis. It has the advantage of factorizing the knowledge, which substantially reduced the size of fuzzy rules base. Also, this knowledge representation method is more appropriate to express some types of knowledge similar to those commonly used in the decision synthesis through natural language; ii) the management fuzzy model for the problem presented was developed incrementally, as was embedded in the model sufficient domain knowledge, resulting from the limitations observed for the crisp case. Is it possible to constantly adapt the management fuzzy model through simulations, often rather difficult; iii) the inferential subsystem based on fuzzy logic, designed and presented throughout this paper, solves the management situations correctly, both from the computational point of view and in terms of the semantics of the conclusions inferred through the chosen inference scheme; iv) modeling the economic problems and the expert system as systems with logical events allowed the qualitative analysis of management expert system.

References

1. *Bergmann, G., Rath, I., Varro, D. 2009. Parallelization of Graph Transformation Based on Incremental Pattern Matching, Proceedings of the Eighth International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2009), Electronic Communications of the EASST, Volume 18, ISSN 1863-2122.*
2. *Dubois, D., Prade, H. 1987. Théorie des possibilités, applications à la représentation des connaissances en informatique, Masson, 2-ième edition.*
3. *Eleftherakis, G. and Cowling, A.J. 2003. An agile formal development methodology, in Tigka K. and Kefalas P. (Eds.), Proceedings of the First South-East European Workshop on Formal Methods, Thessaloniki, Greece, 20 November 2003, Thessaloniki, pp. 119-137.*
4. *Fagiolo, G., Dosi, G., and Gabriele R. 2004. Matching, bargaining, and wage setting in an evolutionary model of labor market and output dynamics, Advances in Complex Systems, vol. 7, pp. 157-186.*
5. *Forgy, C.L. 1982. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence (19), pp. 17-37.*
6. *Galbraith, J. K. 1968. Le nouvel Etat Industriel: essai sur le systeme économique américain, Editions Galimard, Paris.*
7. *Galland S. & al, 2005. L'implication des experts dans un processus de prise de decision, Actes du Colloque ATELIS - Atelier d'Intelligence Stratégique. Poitiers, pp 61-71.*
8. *Jacobi, I., Radul A. 2010. A RESTful Messaging System for Asynchronous Distributed Processing, http://dig.csail.mit.edu/2010/Papers/WS-REST/wsrest2010.pdf (accessed 25th December, 2012).*
9. *Mazilescu, V. 2010. Characterizing an Extended Fuzzy Logic System with Temporal Attributes for Real-Time Expert Systems, The 10th WSEAS International Conference on Systems Theory and Scientific Computation (ISTASC'10), Taipei, TAIWAN, p.159-165.*
10. *Mazilescu, V. 2011. An Intelligent System for a Resource Allocation Problem based on Fuzzy Reasoning, Computer Technology and Application, David Publishing Company, From Knowledge to Wisdom, ISSN 1934-7332, p. 247-255.*
11. *Mazilescu, V. 2012. A Knowldege Management System Embedded in the New Semantic Technologies, Chapter 1, pp. 1-22, in New Research on Knowledge Management Technologies, edited by Huei-Tse Hou, INTECH, Janeza Trdine 9, 51000, Rijeka, Croatia, ISBN 978-953-51-0074-4.*
12. *Qian, D. 1992. Representation and Use of Imprecise Temporal Knowledge in Dynamic Systems, Fuzzy Sets and Systems, Vol. 50, no 1, Aug. 25, p. 59-77.*
13. *Reaz, A., Boutaba. R., 2007. Distributed Pattern Matching for P2P Systems, IEEE Journal on Selected Areas in Communications, http://bcr2.uwaterloo.ca/~rboutaba/Papers/Journals/JSAC-07.pdf (accesed 2th December, 2012).*
14. *Shvartzshnaider, Y., Ott, M., Levy. D. 2010. Publish/Subscribe on Top of DHT using Rete Algorithm. Lecture Notes in Computer Science, Vol. 6369/2010.*
15. *Wagner, T., Phelps, J., Qian, Y, Albert, E. and Beane, G. 2001. A Modified Architecture for Constructing Real-Time Information Gathering Agents, Agent Oriented Information Systems (AOIS), 2001.*
16. *Walzer, K., Breddin, T., and Groch, M. 2008. Relative temporal constraints in the Rete algorithm for complex event detection. In DEBS '08: Proceedings of the second international conference on Distributed event-based systems, pages 147-155, New York, NY, USA, 2008. ACM.*
17. *Walzer, K., Groth, M., Breddin, T. 2008. Time to Rescue - Supporting Temporal Reasoning in the Rete Algorithm for Complex Event Processing, Lecture Notes in Computer Science, Volume 5181/2008, pp. 635-642.*
18. *Westerhoff, F.H. 2004, Multi-assets Market Dynamics, Macroeconomics Dynamics vol. 8, pp. 596-616.*