Fuzzy Dynamic Discrimination Algorithms for Distributed Knowledge Management Systems

Vasile MAZILESCU
Dunarea de Jos University of Galati
vasile.mazilescu@ugal.ro

Abstract
A reduction of the algorithmic complexity of the fuzzy inference engine has the following property: the inputs (the fuzzy rules and the fuzzy facts) can be divided in two parts, one being relatively constant for a long a time (the fuzzy rule or the knowledge model) when it is compared to the second part (the fuzzy facts) for every inference cycle. The occurrence of certain transformations over the constant part makes sense, in order to decrease the solution procurement time, in the case that the second part varies, but it is known at certain moments in time. The transformations attained in advance are called pre-processing or knowledge compilation. The use of variables in a Business Rule Management System knowledge representation allows factorising knowledge, like in classical knowledge based systems. The language of the first-degree predicates facilitates the formulation of complex knowledge in a rigorous way, imposing appropriate reasoning techniques. It is, thus, necessary to define the description method of fuzzy knowledge, to justify the knowledge exploiting efficiency when the compiling technique is used, to present the inference engine and highlight the functional features of the pattern matching and the state space processes. This paper presents the main results of our project PR356 for designing a compiler for fuzzy knowledge, like Rete compiler, that comprises two main components: a static fuzzy discrimination structure (Fuzzy Unification Tree) and the Fuzzy Variables Linking Network. There are also presented the features of the elementary pattern matching process that is based on the compiled structure of fuzzy knowledge. We developed fuzzy discrimination algorithms for Distributed Knowledge Management Systems (DKMSs). The implementations have been elaborated in a prototype system FRCOM (Fuzzy Rule COMpiler).

Keywords: Fuzzy Unification Tree, Dynamic Discrimination of Fuzzy Sets, DKMS, FRCOM

JEL codes: C63, C88, D83

1. Introduction

One efficient pattern matching algorithm for implementing production rule systems is Rete algorithm, designed by L. Forgy of Carnegie Mellon University [5]. This algorithm has become the conceptual basis for a large number of Expert Systems, including here CLIPS, Jess, JBoss Rules. A Rete-based expert system transforms a rule base in a network of nodes, where each node corresponds to a pattern occurring in the left-hand-side (the condition part) of a rule (except the root). The path from the root node to a leaf node defines a complete rule left-hand-side. Each node has a memory of facts which satisfy that pattern. This structure is essentially a generalized Trie [12,13,14,16]. The Rete algorithm is designed to sacrifice memory for increased speed. In very large expert systems, however, the original Rete algorithm tends to run into memory consumption problems. Other algorithms, both novel and Rete-based, have since been designed which require less memory. Forgy developed Rete II as a successor of the Rete algorithm. Rete II is more performant for more complex problems and is officially implemented in CLIPS/R2. Rete II can be characterized by two areas of improvement; specific optimizations relating to the general performance of the Rete network (including the use of hashed memories in order to increase performance with larger sets of data), and the inclusion of a backward chaining algorithm tailored to run on top of the Rete network. Backward chaining alone can account for

the most extreme changes in benchmarks relating to Rete vs. Rete II. The Rete III algorithm is now implemented as part of the Blaze Advisor Rule Server [17,18]. Business Rules Management Systems, or BRMS, are enterprise product suites designed to help companies react more quickly to changes in business policy. A BRMS accomplishes this by providing tools that isolate *business rules* from traditional application code. Business rules are definitions of a company's business policy, typically stated in the form of if-then-else statements. In the pure BRMS space, the two most visible commercial offerings are from ILOG (JRules) and Fair Isaac (Blaze Advisor). The Blaze Advisor rules repository and his engine allow multiple business systems to utilize the same common set of business rules. The Blaze Advisor rule engine can quickly sort through large sets of rules to find the proper ones to apply. This way for customers is given a good view of your business. Likewise, employees are given a consistent view of customers—helping you make consistent business decisions; reduce errors and improve efficiencies. These are comprehensive product suites that have matured significantly over time (some may remember Blaze Advisor's heritage as Neuron Data). At the core of their rules engine offerings, these products feature forward chaining inference engines based on the Rete II algorithm. Rete III is the most advanced commercially-available inference engine, benchmarking more than 300% faster than its competitors, making it the ideal solution for organizations managing large, complex business rule sets, where high performance is critical. Generally Rete III simply incorporates more optimizations [12].

Business Rules is a powerful technology for representing a business policy. It opens the chance to react fast on changes in the market. For example, new pricing strategies can be implemented without a time consuming re-programming of a sales software. The knowledge expressed by rules can be easily understood by non-computer experts. Business experts therefore are able to handle Business Rules directly and therefore make changes even faster. For this, the construction of a Global Semantic Graph (GSG) to support future information- and collaboration-centric applications and services is a very important subject. The GSG is a publish/subscribe (pub/sub) based architecture that supports publication of t-uples and subscriptions with standing graph queries. An implementation of an efficient pattern matching algorithm such as Rete on top of a distributed environment might serve as a possible substrate for GSG's pub/sub facility [13]. Knowledge description and exploitation within a Business Rule Management System (BRMS) are somehow conflicting characteristics, since the increase of the representation power of knowledge diminishes the efficiency of the system and increases the difficulty of carrying it out. Many challenges in the BRMSs field are difficult to solve from a computational point of view [6,7,14].

The aim of this paper is to provide the analysis of the first inference engine pattern matching stage for a compiled rule base, included in our prototype system FRCOM. An amendment that may lead to a reduction of this complexity is the fact that these challenges often have the following property: the inputs can be devised in two parts, one being relatively constant for a long a time when it is compared to the second part. In order to process the fuzzy sets inside the linking tree, we may use a Dynamic Discrimination technique of the Fuzzy Sets based on the I-interval (DDFS-I). This tree will be noted DDMF-I and will be taken into account in order to improve the consistency of the fuzzy facts.The compilation algorithm describes how the Rules in the Production Memory to generate an efficient discrimination network. Maintaining in your business rules consistency and customer interactions can help to maintain a good rate of customer satisfaction and acquisition, helping to build revenues and profits. An implementation of an efficient pattern matching algorithm such as Rete on top of a distributed environment might serve for the new generation Internet and for the future Semantic Technologies [1,3,17]. Distributed pattern matching can be used to solve problems like wildcard searching (for file-sharing P2P systems), partial service description matching (for service discovery systems) etc. Such systems use a hierarchy of indexing peers for disseminating advertised patterns [16].

Section 2 is a short motivation of the impact and the actual interest for Rete algorithm, especially for DKMSs., Section 3 presents the main conceptual aspects of the two stages for fuzzy pattern matching, using Dynamic Discrimination technique of the Fuzzy Sets based on the I-interval, that comprises two main components: the static fuzzy discrimination structure (Fuzzy Unification Tree) and the Fuzzy Variables Linking Network. The processing of the fuzzy knowledge for computational point of view is

based on the possibility theory [4]. In addition, this section contains elementary implementation aspects. Finally, in section 4 are presented concluding remarks.

2. The Impact of the Rete Algorithm for DKMSs

The successful leadership of any organization which is in constant interaction with a multitude of external heterogeneous agents requires an increasing amount of knowledge, to integrate multiple sources of knowledge based on an Enterprise Control Support System (ECSS) for Economic Growth [15]. Thus, the management of distributed knowledge has become a critical factor for enterprise management, because: management structures are based on decisions taken at every organizational level, carrying out the production demands knowledge integrated into the different departments of the enterprise and the rapid and continual changes of the market impose accurate modifications (adaptations) in real time on clients and strategies [9,10,18]. DKMSs are real-time interdisciplinary decision making systems. Decision making becomes more complicated and difficult in today's rapid changed decision environment than ever before. Decision makers often require increasing technical support to high quality decisions in a timely manner. 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 [2]. 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. DKMS implies human resources management, ICT methods and AI techniques for which representing and processing of knowledge are ends all onto themselves. 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 ECSS purpose proposed for development within the project PR356 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 ontologies. The management structures of enterprises, companies or governmental agencies elaborate different sequences of plans with a purpose to fulfill their objectives. At the level of the whole organization, the managers analyze the internal as well as external information of the organization; they refine the different management plans and take certain decisions. Afterwards, the managers continue to supervise should the results of the decisions have been achieved and to what extent. Seeing as organizations work on certain hierarchical structures, the decisions of the managers (as acting leadership) are based on their implication in productive activities. Thus, the leadership attribute at the level of managers is fundamentally based on rational activities of planning the internal and external resources of the organization, having as a direct significance for ECSS, the following types of Artificial Intelligence: resource planning (human, soft, materials), classification and prediction, control (low level operative leadership). With a view towards the development of this ECSS, we will specify the effective execution of the leadership function of the proposed system, in the context of a type of organization. The characteristics of the current leadership of enterprises are visibly oriented towards the correct management of large quantities of knowledge and complex operations with a high rate of change. Therefore, the prediction given by the data accumulated in the past becomes difficult towards the accomplishing of activities distributed on a large scale.

Recently, incremental graph pattern matching approaches have become an important subject in the graph transformation community [8]. The basic idea is to improve the execution time of the time-

consuming pattern matching phase by additional memory consumption. Essentially, the (partial) matches of the left-hand side (LHS) of graph transformation rules are stored explicitly, and these match sets are updated incrementally in accordance with elementary model changes.

Distributed computing problems have been solved by partitioning data into chunks able to be handled by commodity hardware. Such partitioning is not possible in cases where there are a high number of dependencies or high dimensionality, as in reasoning and expert systems. This renders such problems less tractable for distributed systems. By partitioning the algorithm, rather than the data, we can achieve a more general application of distributed computing [11].

Keyword searching is one of the essential functionalities offered by any peer-to-peer file-sharing system. A Centralized file system, as present in any traditional operating system, permits more sophisticated search operations involving wildcards and partial keywords. Enabling existing P2P file sharing systems with wildcard search capability will allow users to perform more flexible and powerful searches. Besides inexact keyword matching, many problems like partial service description matching for service discovery systems, data record pre-scanning for distributed database systems, molecular fingerprint matching in a distributed environment, etc., can be mapped to and efficiently solved using Distributed Pattern Matching [16].

3. Conceptual and Implementation Aspects for FRCOM

The Knowledge Systems based on production rules have in their structure the inference engine that cyclically carries out the following stages, each being characterised as it follows:

1) *The Pattern matching Stage.* The premises of the rules are filtered with the current facts, with the $e_{r_k}^{SE}$ reference input and the $e_{p_k}^{SE}$ process outputs, being carried out at the current moment the following result: $MC_k=\{R_i:\{R_i, e_{u_k}^{e}\} \subset g^{SE}(\mathbf{x}_k^{SE})\}$.

In the case of the FRCOM system, this stage uses the knowledge compiled structure. Since the current system is characterised by factorised fuzzy knowledge, the rule base is greatly diminished, and this leads to the improvement of the pattern matching stage time.

2) *The Selection Stage* includes either the conflict solving strategies or the heuristic inference strategies. A rule or instance is selected in order to be activated. We may mention as possible conflict solving strategies: i) Refraction. All the rules within MC_k that have been previously activated are eliminated from the MC_k. If the activation of a rule affects the pattern matching data corresponding to the antecedents of other rules, these rules are allowed in order to solve conflicts; ii) Age. This uses priorities in order to activate the rules based on the age of the information from the fact base that filters the premises of each rule, according to the pattern matching criterion allowed within the knowledge system. Age represents the activations number of a rule; iii) the degree of difference between rules. The rule that best filters the current situation is activated. We may use various measurements in order to evaluate the inference degree, measurements that must take into account the intrinsic imprecise character of the knowledge included within the management model of the process; iv) Priority schemes. There are priorities assigned a priori to the rules by the field expert and by the knowledge engineer, at the same time with the synthesis of the management model. These values can be either constant or combined with the current imprecision of the premises of the rule, and it results a priority that modifies dynamically. The rule with the greatest current priority is thus activated; v) Arbitrarily. This activates a rule from the stochastic conflicts set.

3) *The Activation Stage*. The actions within the consequent of the selected rule are executed, the x^{SE} FRCOM system state is updated and the $e_{o_k}^{SE} = f_{e_k}^{SE}(e_{uk})$ output size is generated, so that the $R_i \in \Re$ global premises of the rule (instance) is evaluated as satisfying the $\Pi=1$ and $N > 0$ conditions within the FRCOM system. There is further synthesised the conclusion obtained by the inference engine that is either offered to the user or synthesised as a command inside the process.

3.1 Conceptual Aspects

At the end of the elementary fuzzy pattern matching stage, if the pattern matching degree satisfies the chosen threshold and if there is a σ consistent substitution, then the pattern matching process is a success. The fuzzy condition – fact pattern matching process is the first stage part of the overall behaviour of the inference engine, able to take into consideration the knowledge's imprecision. Each instance of a fuzzy reason is associated to a σ fuzzy substitute and to the Π, N, θ, K parameters [4,14]. The second stage in the pattern matching process on a global scale of the fuzzy rules is the fuzzy linking of variables. This conducts the fuzzy unification whose main aim is to verify the consistency of fuzzy substitutes, for which we have already presented a series of theoretical results. Using the tests present in the linking nodes, we can build a dynamic tree that allows adding or suppressing facts. Within each test node of this tree, the values of the variables are tested. If two facts go on the same path, then it is possible that the two facts are consistent. We may use this tree in order to avoid combination challenges. This tree is called linking tree and it is associated to the linking nodes. It presents difficulties for the discrimination of the fuzzy sets within its linking nodes, since certain parts (leaves) of the tree may contain multiple fuzzy facts. That is why the efficiency of this solution decreases, being similar to the use of the unification tree in order to discriminate the fuzzy motives. The main inefficiency factor is related to the disorder of the fuzzy facts in the leaves of the tree. In order to improve the situation, we may use the characteristics of the fuzzy sets in order to sort out facts. This approach was used in order to adapt the unification tree for the processing of fuzzy motives. The major difference that appears between the two situations is the fact that the unification tree is a static tree, i.e. the discriminator motives do not change, whereas the linking tree is dynamic, the discriminator facts being updated during the functioning period of the inference engine.

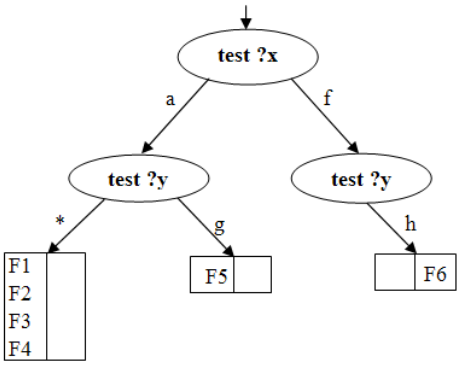

Figure 1. *A Linking Tree*

The linking process of the fuzzy variables consists of the fuzzy unification and the spreading of the Π, N, θ, K parameters evaluated on a global scale of the antecedent of the rule. We will further insist on the spreading process of the parameters obtained at the end of the pattern matching stage, in the consequent of the rules. The Π possibility and N necessity degrees represent the extent to which a rule is satisfied in the current state inside the fact base. During the selection stage, the system selects the rule that satisfies best these conditions in order to trigger it, and the θ and K parameters serve to apply the Generalised Modus Ponens inference scheme. In the conclusion part of a rule there may be multiple motives (some may be added, others may be deleted, once the rule has been triggered). Before the execution of these actions, we must instantiate all the motives from the right director in which there are three types of data as well as in the conditional director.

The parameters obtained during the pattern matching stage interfere with only two types of data. The linking tree of variables is not a static tree, since we may constantly add or remove facts. The spreading of facts within the linking tree is elementary. We exemplify the adding of the F={s(A d h), σ=(d/?x, h/?y) } fact. This enters at the root and, in the same time tests the value of the ?x variable. This value is normally indicated in the substitution associated to F for which ?x = d. The fact then passes on the branch tagged with a symbol corresponding to the value of the variable in question. In the following node, the value of the ?y variable is tested in the same way, and F will pass on the branch tagged with

the h symbol. At the end, the F fact is to be found at the level of a leaf within the linking tree and the spreading process thus ends. This moment we must obtain the possible instances, but this does not ensure the fact that these instances are also consistent for the fuzzy case. The instances that satisfy the consistency condition allow the in question rule to be added to the conflicts set. The linking tree is efficient since a fact always follows a unique path. Its depth depends on the number of different tests and its complexity is constant. The linking tree encounters difficulties regarding the discrimination of the fuzzy sets inside its linking nodes, since certain leaves may contain multiple facts.

Example 1. Take the situation in which the F_1 = (A a *b), F_2= (A a *c), F_3 = (A a *d), F_4 = (A a *e), F_5 = (A a g), F_6 = (B f h) facts occur, for which the present current fuzzy constants correspond to the following fuzzy sets: (*constfaz* *b(tp 10 18 12)), (*constfaz* *c (tp 7 12 1 1)), (*constfaz* *d (tp 5 10 1 2)), (*constfaz* *e (tp 16 20 2 2)). We have drawn the corresponding linking tree in Figure 2. We see that the leaf that contains the four fuzzy facts is overloaded. Adding the F_7 = (B a *e) fact by the right input, leads to its memorising in the corresponding field. This fuzzy fact is combined with all the other facts and we must verify the consistency of all combinations. The procedure is obviously inefficient from an algorithmic point of view, the main factor being the disorder of the fuzzy facts.

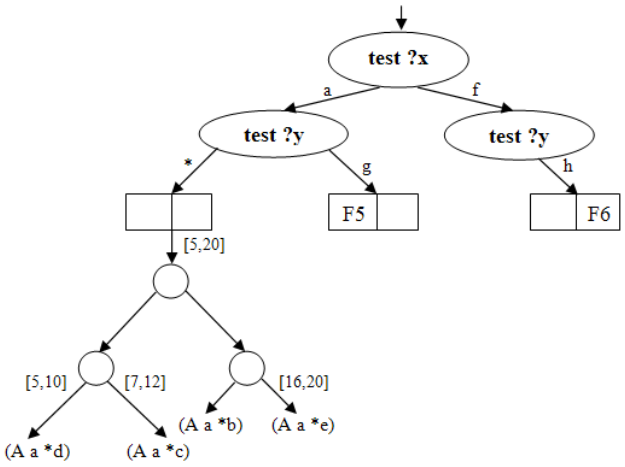

Figure 2. *The DDMF-I Tree*

In order to improve the situation we may use the characteristics of the fuzzy sets. This approach was used in order to adapt the unification tree to the processing of fuzzy motives. The major difference is the fact that the unification tree is a static tree, i.e. the discriminating motives do not change, whereas the linking tree is dynamic. Take M= {*m_1,...,*m_n}, a list of fuzzy sets that must be updated, where *m_i=(g_i d_i φ_i δ_i). Assuming that the inferior margins are used, we firstly obtain a M'={*m(1),..., *m(n)} list.

Using the superior margins we obtain a new list noted M"={*m' (1),...,*m'(n)}. A new *m fuzzy fact may be rapidly inserted within the M' or M" lists.

The problem that we lay down and that must be solved at present is: „for a *e fuzzy set defined by the four parameters *e = (g_e d_e φ_e δ_e), which are the fuzzy sets in M that are compatible to *e ?". The compatibility between two fuzzy sets has already been analysed, and the estimation of this compatibility depends on the chosen relationship. The compatibility algorithm presents also a series of disadvantages, even though it has the advantage of simplicity and these are: the registration of the two M' and M" lists, if a new fact reaches the level of a leaf it has to undergo twice the algorithm of insertion within the list, and it has to make an intersection operation in order to determine the consistent facts with the new fact.

In order to process the fuzzy sets inside the linking tree, we may use a dynamic discrimination technique of the fuzzy sets (DDMF), based on the I-interval. This tree will be noted DDMF-I and will be taken into account in order to improve the consistency of the fuzzy facts. Everything that follows is

based on the $\widetilde{R}$ relation and on the Π possibility measure as a criterion for the compatibility of fuzzy sets.

There are defined $I(M_s)$ and respectively $I(M_d)$ minimum intervals for the M_s' and M_d' sub-lists. The engendering of the DDMF-I tree is simple and its complexity is not a optimum one. Figure 3 highlights a DDMF-I tree in the fuzzy case, which is more efficient than the tree in Figure 2. This tree will be used for the fuzzy unification. The (B a *e) fact will be stored in the right-hand directory, and afterwards we will move on to the searching stage of the consistent instances, by spreading the new fact inside the DDMF-I tree.

For the parameters $\varepsilon = 1$ and $\eta = 0$ corresponding to the $\widetilde{R}$ relation, we notice that the composition core of the $\widetilde{R}$ relation with *e intersected with the determined interval after the spreading of all facts inside the DDMF-I tree leads to a non-void result. This means that there may exist facts inside the DDMF-I tree that may be compatible with the new fact. The {(A a *c), (B a *e)}, {(A a *d), (B a *e)}, {(A a *b), (B a *e)} and {(A a *e), (B a *e)} instances are consistent (Figure 2). In conclusion, the linking nodes represent an important cell in the linking network, and they have a binary and dynamic tree structure, useful for the improvement of the efficiency of the fuzzy inference engine, corresponding to the FRCOM system.
From a practical point of view we need to solve the spreading problem of overall parameters that are obtained during the fuzzy condition-fact pattern matching stage and the linking of fuzzy variables over the conclusion of the rule. Firstly, we will take the stock of the parameters obtained during the execution of the pattern matching stage of the control cycle of the fuzzy inference engine. Take a R rule that has K motives, i.e. $Cond(R) = \{C_1,...,C_k\}$.

After the condition-fact fuzzy pattern matching stage, each C_i condition filters a F_i fact and we thus obtain a σ_i substitute so that $F_i \cong \sigma_i \cdot Ci$, as well as the four parameters $\Pi(C_i,F_i)$, $N(C_i,F_i)$, $\theta(C_i,F_i)$, $K(C_i,F_i)$. On the other hand, if there are S different variables in the conditional part of the R rule, i.e. $V(Cond(R)) = \{?v_1,...,?v_s\}$, then for each $?v_j$ variable (which may occur several times within the conditional part) we may obtain as a result of the fuzzy substitutions a $^*t_{v(j)}$ fuzzy set as an associated value of the $?v_j$ variable, as well as the $\Pi_{v(j)}$ and $N_{v(j)}$ measures. The spreading of on the level of the antecedent of the R rule is defined as it follows:

$$\Pi = \mathbf{min}\ (\Pi(C_i, F_i), \Pi_{v(j)}) \quad (1 \le i \le k, 1 \le j \le s) \qquad (1a)$$
$$N = \mathbf{min}\ (N(C_i, F_i), N_{v(j)}) \quad (1 \le i \le k, 1 \le j \le s) \qquad (1b)$$
$$\theta = \mathbf{max}\ \theta(C_i, F_i) \quad (1 \le i \le k) \qquad (1c)$$
$$K = \mathbf{min}\ K(C_i, F_i) \quad (1 \le i \le k) \qquad (1d)$$

For $\Pi = 1$ and $N > 0$ it is considered that the rule satisfies the pattern matching condition. During the selection stage, the system selects the most appropriate rule to be triggered with the help of these two parameters and also of other selection parameters. The θ and K parameters are used in order to apply the generalised modus ponens scheme. Take a $?v_j$ variable from the conclusion of the $\Re$ rule. The value of this variable is obtained by the reunion between the $^*t_{v(j)}$ value (the only fuzzy value obtained after the composition of the fuzzy values that substitute the occurrences of the $?v_j$ variables from the antecedent) and the 1-N uncertainty degree. It thus results that the $?v_j$ variable takes the **max** value $(^*t_{v(j)}, 1\text{-}N)$.

Take a *Q fuzzy constant from the consequent of the rule with (*constfaz* *Q(tp $g_q,d_q,\varphi_q,\delta_q$)). Once the Generalised Modus Ponens inference has been applied and using the calculus relations for the θ and K parameters, we obtain the following expression for *Q': *Q'=**max** $((g'_q, d'_q, \varphi'_q, \delta'_q), \theta, 1\text{-}N)$. The values of the $g'_q, d'_q, \varphi'_q, \delta'_q$ parameters are $g'_q = g_q - \varphi_q(1-K)$, $d'_q = d_q + \delta_q(1-K)$, $\varphi'_q = (g'_q - g_q + \varphi_q)/(1-\theta)$, $\delta'_q = (d_q + \delta_q - d'_q)/(1-\theta)$. The *Q' degree of imprecision depends on the θ non-determination induced by the generalised modus ponens scheme and by the 1-N uncertainty degree.

Example 2. Take the (R: (*D *H ?x) (B ?x) → *add* (C *E ?x)) rule, with: (*constfaz* *E (tp 10 20 2 5)), (*constfaz* *D (tp 28 35 2 2)), (*constfaz* *H (tp 35 45 5 5)).
There are two distinct reasons within the antecedent: C_1= (*D *H ?x) and C_2 = (B ?x). We assume that there are the F_1= (*d_1 *h_1 *w) and F_2= (B *r) facts, with (*constfaz* *d_1 (tp 28 35 2 2)), *(constfaz* *h_1 (tp 43 47 2 2)), (*constfaz* *w (tp 10 12 1 1)), (*constfaz* *r (tp 11 13 1 1)).
The *d_1 fuzzy constant filters *D while *h_1 filters *H. The corresponding parameters are Π(*D, *d_1) =1, N(*D,*d_1)=0.5, θ(*D,*d_1)=0, K(*D,*d_1) =1, Π(*H, *h_1) = 1; N(*H, *h_1) = 0.428; θ(*H, *h_1) = 0; K(*H, *h_1) =0.6.
On the level of the overall C_1 motive of the rule it results that Π(C_i,F_i) = 1, N(C_i,F_i) = 0.428, θ(C_i,F_i) = 0, K(C_i,F_i) = 0.6, with σ_1 = {*w | ?x}.
For C_2 and F_2 the only result is the σ_2 ={*r | ?x} substitution. The fuzzy unification allows to check up the consistency of the fuzzy substitutes, where the variables are associated to some fuzzy constants according to the σ_1 şi σ_2 substitutes presented in Figure 3.
The consistency of the σ_1 and $\sigma_2$fuzzy substitutes must be verified, as well as the synthesis of the *t_q value from the consequent that substitutes the ?v_q variable according to the {*w|?x,*r|?x} →{*t_q|?v_q} relation.

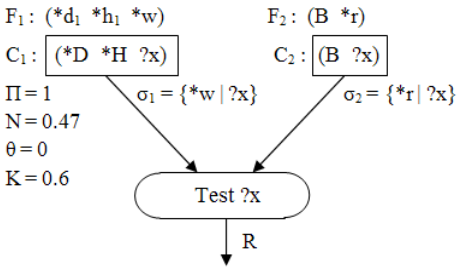

Figure 3. A Fuzzy Unification Example

Using the σ_1 and σ_2 fuzzy substitutes, the value of the ?x variable is determined from the conclusion of the R rule. For ε=1, η=0, it results that Π(r,*w **x***r)=1, N(r,*w **x***r)=0.5, and the two *w and *r fuzzy sets are not compatible (according to Figure 4 a, b).

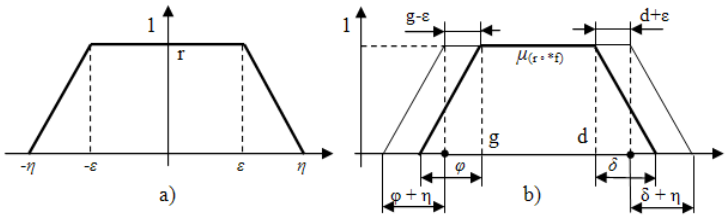

Figure 4. *The representation of the* $\mu_{(r\circ *t)}$ *set*

Take the *t_q fuzzy constant associated to the ?x variable from the consequent, with: *t_q=(r∘*w)∩(r∘*r)=(10 13 1 1), and r∘*w=(g_w-ε, d_w+ε , φ_w, δ_w)=(9 13 1 1) şi r∘*r=(10 14 1 1). For the *E fuzzy constant from the consequent, if we know that θ = 0 and K = 0.6, we obtain: $g_{q'}$= g_q - φ_q (1- K)=10-2·(1- 0.6) = 9.2, $d_{q'}$ = d_q+ δ_q·(1- K) =20 +5·(1-0.6)=22, *E'=(9.2 22 1.2 3). The uncertainty degree of the rule is 1- N = 0.572. The fact inferred by the activation of the R rule will thus be (C *E_q *t_q) with (*constfaz* *E_q (tp 9.2 22 1.2 3) (*uncertain* 0.572)) and (*constfaz* *t_q(tp 10 13 1 1)) (*uncertain* 0.572)).

3.2 Implementation Aspects for the Compiled Structure

The result of the compilation is represented by the compiled structure of the rule base that is saved in the following order: the number of fuzzy constants from the list of fuzzy constants of the knowledge base and all the values of the fuzzy constants orderly, the number of atomic constants and all the values of the atomic constants orderly, the number of logic constants and all the values of the logic constants orderly, the number of variables, the number of motives and all the motives orderly, the number of rules (for each rule we look for the corresponding motives and we save only the number of the corresponding motive), the number of facts (for each fact we look for the motives and we save only the

number of the corresponding motive), the number of the linking nodes of the linking network of variables, all the values of the linking nodes orderly, the number of linking node-type outputs, all the values of the linking node-type outputs orderly (for each linking node-type output we save the number of outputs and then the list of numbers corresponding to the linking nodes), we cover the fuzzy discrimination tree and all of its leaves are numbered, the number of leaves that belong to the fuzzy discrimination tree with all the values of the leaves of the fuzzy discrimination tree orderly (we cover the fuzzy discrimination tree and all of its nodes are numbered), the number of nodes of the fuzzy discrimination tree with all the values of the nodes of the fuzzy discrimination tree orderly, the number of leaves of the gross unification tree with the values of the leaves of the gross unification tree orderly (we cover the gross unification tree and all of its nodes are numbered), the number of nodes of the gross unification tree with all the values of the gross unification tree orderly, the number of discrimination criteria and their values, the number of discrimination values criteria and their values, the number of distinct motives. It is necessary to save the knowledge base, then to inversely save the compiled structure for all the elements that are part of it to already have saved the deliveries.

The launching process in execution of the inference engine starts with the initiation of the partitions from the linking nodes of the compiled structure, as it follows: **i**) we call on the „partition_initiation" method for each linking node; **ii**) we call on the "initiate_ fact_stocking_structure" method for each partition that is to be found in the "partition_list" that divides the corresponding linking node. If the number of variables from the „tests" object that belongs to the „variable_sets" class is zero, we initiate the fact_stocking_structure pointer with a pattern matching_facts_buffer, otherwise the void-type pointer is initiated with a linking_tree. The exploitation of the compiled structure assumes the introduction of the current state of the $\mathbf{x}^{SEC}$ expert management system on the structure, and the occurrence of some actions as it follows: **1**) the border of the compiled structure is initiated with the help of the "border_init" method (recursive function) from the "knowledge_exploiting_structure" class. The current fact covers the unification tree until it reaches its leaves; **2**) the conflict set is generated with the help of the "generate _conflict_set" method that belongs to the "knowledge_exploiting_structure" class. This method takes over all the facts from the fact base and sends them within the network with the help of the "send_fact_within_network" method. Since we have a fact that stands for a parameter, this method covers the unification tree according to the list of values of the fact, i.e. it realises the structural discrimination between the criteria values list inside the fact and the discrimination criteria inside the nodes from the unification tree. If the fact reaches the leaf (i.e. it is compatible from a structural point of view with a leaf of the tree), then a number of local objects that belong to the "mobile_exploiting_ structure" and "pattern matching_fact" classes is assigned. We then add to an object of the "pattern matching_facts_ conjunction" type nestled within the "mobile_exploiting_structure" the fact that covered the unification tree with the help of the "sublists_pattern matching" method, which in turn depends on another method, called "pattern matching" (both methods belong to the "mobile_exploiting_structure" class).

The pattern matching process between the fact that reached the leaf of the unification tree and the motive inside the leaf takes place. If the pattern matching process is a success, then that fact is added to the corresponding motive inside the leaf. We further use the "pattern matching" method, i.e. we verify whether the fact filters the motive inside the leaf.

If the fact filters the motive, we assign an object of the "mobile_exploiting_structures_list" type, and after that we add the previously assigned mobile structure to this list. Next, we enter a case with two different situations: add_facts_in_the_network and delete_facts_from_the_network. For each output towards a linking node we add facts inside the network, as it follows: we find the node using the address where the object of the "linking_nodes_outoputs" class and we use the "add_facts_in_the_network" method that belongs to the "linking_node" class. This method has as a parameter the address of the "mobile_exploiting_structures_list" object previously assigned. We next cover the list of all partitions that share that precise linking node. We initiate the "mobile_exploiting_structures_list" object with the same object used as a parameter, and we then use the "add_facts" methods within the linking tree inside the partition. A partition has nestled inside it a "linking_nodes_type_list" list. The "add_facts_in_ the_network" method is recursive and we use this

inside it for all the linking node type outputs, outputs that belong to the current partition from the current node. We thus cover and add facts inside the network with the help of "mobile_exploiting_structure" objects. We initiate the address to the "mobile_exploiting_ structures_list" object from inside the partition with the same address that comes as a parameter, and we then initiate a local parameter of the integer type with what gives back the "ret_ mobile _structure_source "that belongs to the "partition" class.

This method allows the admission within a repetitive structure that lasts until all list of pattern matching facts that are assigned as nestled objects of the "pattern matching_facts_conjunction" type inside the "mobile_exploiting_structure" object used as a parameter is covered, and we then enter a repetitive structure that lasts until we reach a number of cycles equal to the parameter that is called "the_index_of_the_first_right_hand_motive" that belongs to the "conditions_conjunction" object. Inside de two repetitive structures previously presented we verify the compatibility between each pattern matching fact and each motive that is to be found inside an object of the "motives_conjunction" class that is nestled in a "conditions_conjunction" object. If there is a motive which is compatible with a pattern matching fact, we turn left, otherwise we turn right. We go back to the "add_facts" method that belongs to the partition-type object. It is the case of the parameter that takes the right- or left-hand value. We initiate the substitutes list inside the "mobile_exploiting_structure" that came as a parameter. The "facts_stocking_structure_type" variable takes two values: pattern matching_facts_buffer and linking_tree.

On the pattern matching_facts_buffer branch we enter a repetitive structure that uses the "add_pattern matching_facts" method of the "pattern matching_facts_buffer" object for each cycle. This method has as parameters each mobile exploiting structure from the "mobile_exploiting_structures_list" object and it allows the assignment of multiple local data of pointer type to "pattern matching_facts_list", "conflicts_set", "mobile_exploiting_ structure_list", "rule" etc objects. We take the addresses of the "left_hand_facts" respectively "right_hand_facts" objects inside variables of the same type previously declared. We then enter a case after the "owner_type" variable which is given as a member of the "pattern matching_facts_buffer" object with two branches: partition and linking tree. On the partition branch we initiate the following local data: "conflicts_set", "mobile_structures", "number_of_outputs_from_the_partition", "current_rule" (rule-type object). We do the same thing on the linking tree branch. We next take the address of the pattern matching facts list from inside the mobile structure that came as a parameter and we assign the local variable of the same type previously declared. We then enter a repetitive structure where the "number_of_outputs_ from_the_partition" variable that belongs to the "partition" object is tested. If this is positive we apply certain methods that enrich the mobile exploiting structure received as a parameter. If it is zero, then we add the current rule from the partition to the conflicts set. In this case, we do not apply the fuzzy unification, but the classical one (the variables are instantiated by atomic_constants). On the linking_tree branch we use the „add_facts_inside_the_tree" method that belongs to the „linking tree" object. This allows the admission within a repetitive structure that takes all the „mobile_exploiting_structures" that belong to the current partition corresponding to the linking tree and the „add_facts_inside_tree_node" recursive function is used from the structure of the linking tree and the add facts objects. The „current_test_index" parameter indicates the level reached inside the tree. For the root the level is 1.

If there is no output we create a new output and either a corresponding node or a new leaf. The output from the current node can be of a leaf_linking tree or a node_linking tree type. The leaf-type output when we reach the „current_test_index" on the last variable from „variables_set" which is to be found in the „partition" that has the actual linking tree. The function is used recursively until the output from a node is of the leaf_linking tree type. When we reach the leaf of this tree we use the „add_facts_in the _tree_leaf" method that belongs to the „linking_tree" object. This method has inside it two cases after the atomic_constant and the fuzzy_constant.

On the dynamic_discrimination_trees_buffer branch we use: the ***„Add_facts_in_the_tree***" method that belongs to the „dynamic_discrimination_trees_buffer" object. This discrimination trees buffer has only two trees of this kind. On one of these branches we use the ***„Add_facts_to_the_tree***" method of the

ADDMF object (dynamic discrimination tree of fuzzy sets). This method assigns a „leaf_dynamic_discrimination_tree" object that is further initiated with the pattern matching fact from the mobile structure received as a parameter, with the corresponding substitute and parameters. We mention that the ADDMF has two „***Add_facts_in_the_tree***" methods: one is used when the tree is void and we create one leaf only, the other one is used when the tree is non-void. On each of the branches we use the „***Enrich_mobile_exploiting_structure***" method. This method makes the composition of the fuzzy substitutes and the calculation of the consistency measures that appear when the substitutes are created. With the help of this method we also verify whether this partition has outputs towards other nodes. Otherwise, we add to the conflicts set the instance from the current partition, and in an affirmative case we take the pattern matching facts that belong to the leaf from the other tree and we add them to the mobile exploiting structure. The state of the expert management system is saved only if the old front has changed its structure. The exploiting structure of knowledge represents the compiled structure of knowledge. This class contains building and exploiting functions regarding the compiled structure. Due to the form of the structure, the majority of the necessary functions are defined as member functions. The structure contains the following elements: the root of the unification tree (the input point within the compiled structure inside the memory), a list of discrimination criteria (instantiations of the objects), a list of values of the criteria (instantiations of the values), a table of characteristics used for the building of the linking network of the variables, a pointer at the knowledge base of the system, a list of linking nodes (nodes of the linking network of the variables) and a pointer at the conflicts set of the system. The public methods of this class are:

- ***Init_pointer_at_the_knowledge_base*** (initiate the pointer at the knowledge base),
- ***Gross_unification_tree_generation*** (generate the gross unification tree without the fuzzy discrimination trees and without the conditions conjunctions inside the leaves of the tree),
- ***Refine_unification_tree*** (generate within the created gross unification tree, the fuzzy discrimination trees and replace the distinct motives with the conjunction of conditions necessary to create the linking network of the variables),
- ***Build_linking_network*** (build the linking network of the variables),
- ***Generate_set_of_conflicts*** (generate the set of conflicts, i.e. the consistent instances of the rules. This function contains the pattern matching stage),
- ***Destroy_knowledge_exploiting_structure*** (destroy the compiled structure, including the characteristics table).

4. Conclusions

Distributed Knowledge Management Systems or BRMSs are developed in order to frequently operate in a dynamic environment. Knowledge, in this case, is time-variable and the knowledge system needs to be able to generate either hypotheses or conclusions. That is why it is necessary to verify the fact that this type of systems can guarantee a series of performances for which they have been created, at an appropriate level. The efforts and contributions from this paper refer mostly to the analysis of the model based on a compiled fuzzy structure (especially, at the level of the inferential subsystem) and its integration within a BRMS. We mention that our prototype is similar to a planner, since they can predict a number of states within the evolution of the process, on a future (finite) time horizon, as result of its own inferential results. The static properties of the fuzzy knowledge model influence considerably the dynamics of the inference engine. For example, this type of model does not have to be synthesised so that there are real states or situations which are not to be filtered by any rule, i.e. for which the knowledge based system will not react (an aspect that was solved by user interventions). That is why it is difficult enough to verify the model through simulation, especially for the real cases in which the number of rules can be considerably. The analyses of the inference process remains to be evaluated even after the static properties have been verified, which cannot detect all its functional insufficiencies. These aspects cause significant problems regarding the analysis and the design of a distributed knowledge management system for real-time applications, able to satisfy the planning aims. We exemplified on a reduced model, made up of 25 fuzzy rules, the behaviour of our prototype, respecting the formalisms and the analysed performances (as a real-time planning system).

References

1. Berners-Lee T., J. Hendler O. Lassila, 2001- Semantic Web, A New Form Of Web Content That Is Meaningful To Computers Will Unleash A Revolution Of New Possibilities, *Scientific American*, Vol. 248, p. 34-43
2. Bergeron, Bryan P. (2003) - *Essentials of knowledge management*, John Wiley & Sons
3. Dieng, R. and Corby, O., Faron-Zucker, C., (2004) - Querying the Semantic Web with Corese Search Engine. *Proc. of the 16th European Conference on Artifcial Intelligence*, 651-672
4. Dubois, D., Prade, H. (1987) - *Théorie des possibilités, applications à la représentation des connaissances en informatique*. Masson, 2-ième edition.
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. Fredric Landqvist, Kalevi Pessi (2004) – *Agent Action: Business cases with individualised information services in a Business Intelligence context*, Proceedings of the 37th Hawaii International Conference on System Sciences, University of Gothenburg
7. Fu-ren Lin, Shyh-ming Lin, Po-win Hsueh (2004) – *Dynamic Business Process Formation by Integrating Simulated and Physical Agent Systems*, Proceedings of the 37th Hawaii International Conference on System Sciences, Department of Information Management
8. Gabor Bergmann, Istvan Rath, Daniel Varro (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
9. Galbraith J.K., (1968) - *Le nouvel Etat Industriel: essai sur le systeme économique américain*, Editions Galimard, Paris
10. Galland S. & al, (2005) - *L'implication des experts dans un processus de prise de decision*, publicată în Actes du Colloque ATELIS ATELier d'Intelligence Stratégique. Poitiers, pp 61-71
11. Ian Jacobi, Alexey Radul (2010) – *A RESTful Messaging System for Asynchronous Distributed Processing*, http://dig.csail.mit.edu/2010/Papers/WS-REST/wsrest2010.pdf
12. Karen Walzer, Mattias Groth, Tino Breddin. *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
13. Karen Walzer, Tino Breddin, and Matthias Groch. *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.
14. Mazilescu V., 2009 - *A Real Time Control System based on a Fuzzy Compiled Knowledge Base*, Proceedings of The 13th WSEAS International Conference on COMPUTERS, CSCC Multiconference, Rodos Island, Greece, July 23-25, 2009, Conference track: Artificial Intelligence. Computational Intelligence, pp. 459-464, ID: 620-473, ISBN 978-960-474-099-4, ISSN 1790-5109, http://www.wseas.us/elibrary/conferences/2009/rodos/COMPUTERS/COMPUTERS70.pdf, http://portal.acm.org/citation.cfm?id=1627695.1627780&coll=GUIDE&dl=GUIDE&CFID=70595288&CFTOKEN= 74537819 (Association for Computing Machinery – USA)
15. Mazilescu V., 2010 – *An Enterprise Control Support System for Economic Growth*, The 10th WSEAS International Conference on Applied Informatics and Communication (AIC'10), Taipei, TAIWAN, August 20-22, Conference ID 647-307, p. 365-370
16. Reaz Ahmed, Raouf Boutaba, (2007) – *Distributed Pattern Matching for P2P Systems*, IEEE Journal on Selected Areas in Communications, http://bcr2.uwaterloo.ca/~rboutaba/Papers/Journals/JSAC-07.pdf
17. Yan Shvartzshnaider, Maximilian Ott, David Levy. *Publish/Subscribe on Top of DHT using Rete Algorithm*. Lecture Notes in Computer Science, 2010, Vol. 6369/2010 (Future Internet)
18. Wagner, T., Phelps, J.; Qian, Y, Albert, E. and Beane, G. *A Modified Architecture for Constructing Real-Time Information Gathering Agents*, Agent Oriented Information Systems (AOIS), 2001.