Compilation algorithms like ops5 for fuzzy knowledge models

Senior lect. Vasile MAZILESCU, PhD., Senior lect. Cristian GEORGESCU, PhD.
"Dunarea de Jos" University, Galati

This paper focuses on the problem of decreasing the response time of a fuzzy rule-based system. Typical examples include control software for monitoring, safety-critical and risky economic and industrial applications. We give some algorithms for off-line computing the compiled fuzzy model, namely the ***VLN****. To have a better idea of the applicability of the* ***VLN*** *algorithm A2, we present a* ***CFKBS*** *for a flexible production system (****FPS****).*

Keywords: *fuzzy knowledge, compiled model, OPS5, variable linking network.*

1 Introduction

The handling of the incertitude and of the imprecise knowledge represents a main feature of a Fuzzy Knowledge Based System (**FKBS**), which includes all the inexact knowledge whose results address to the human decedent. If the knowledge model $\mathbf{M}_K$ is inefficient as a dimension, complexity, and operation time, then the application of the compilation technique becomes useful for the $\mathbf{M}_K$ model, like OPS5, used in our Compiled Fuzzy Knowledge Based System (**CFKBS**), which is the subject of the foregoing work.

Many problems of the Artificial Intelligence (**AI)** are difficult to solve from a computational point of view [2]. A remark which may lead to the decrease of this complexity expresses the fact that these problems have the following property: the input can be divided into two parts: a part of these inputs $\mathbf{M}_K$ is long time relative constant in relation to the other part **F.** Under these circumstances, the achievement of particular off –line transformations upon the $\mathbf{M}_k$ constant part becomes meaningful to diminish the time for the getting of problem solution, if the second part varies and it is known at certain intervals [1]. The performed transformations in advance are called pre-processing or the knowledge compilation. The language of first-order predicates facilitates the expression of the complex knowledge in a very rigorous way implying adequate thinking techniques. The definition of some propagation and interference procedures for **CFKBS** of real time implies the elaboration of powerful reasoning mechanisms as well as the adjustment of the control algorithms within the state spaces, which are often higher.

The system **CFKBS** has as a conceptual base all the features shown within the work [6] and it consists of: **i)** *So-called formalism,* which specifies the types of knowledge accepted by the system. There are integrated the syntax of the fuzzy knowledge and the basic features of the compiled linguistic models, the system parameters, as well as the elementary fuzzy filtering, the distributions compatibility of the possibility for the diagram of generalized modus ponens inference; **ii***) The compiler conception* that includes the static structural discrimination component of the fuzzy condition, the fuzzy unification (having as purpose the control of the fuzzy substitutes consistence), the generation algorithm of variable linking network (**VLN**). The terminal knots of the **VLN** correspond to the bijective rules from $\mathbf{M_g}$**; iii**) The algorithm of the interference motor/instrument that has the thinking methods.

The hereby work aims to present the elaborated algorithms for **VLN** generation, which are an integral part of the compiled structure of fuzzy models made by us, and applied for incorporated knowledge in different **IA** systems of planning, diagnosis, classification type in mono or multi-agent structures [5, 7].

Within the Section 2 the **CFKBS** formalism is briefly presented, section 3 deals with the proposed and tested algorithms for **VLN** generation, the section 4 is a synthesis of the main conclusions and of the possible research directions.

2. The Features of the CFKBS formalism

The achievement of the efficient inference algorithms in a real-time **CFKBS** implies a corresponding analysis of the knowledge type and of the knowledge meaning from the structure of the involved models. The next shown elements are mainly focuses on the logical aspects concerning the fuzzy inference with fewer emphases on the semantics of the knowledge models based on fuzzy rules. From this point of view, the implication and the multivalent extensions can accurately express the semantics problem of the fuzzy rules rarely investigated in the literature. There are suggested three types of fuzzy rules "if … then" as per the work [3], which are to be presented: i) *Rules for the certitude qualification.* These rules are under the form "the $u \in A$, the more certain $v \in B$, which may be translated by the relation $(\forall)u$, $\mu_A(u) \leq g_t(B)$, where $g_t(B)$ evaluates the certainty degree of the enunciation $v \in B$, when $x=u$. The function g_t can be any measure of necessity, possibility or probability type; ii) *Gradual rules* (for truth qualification) are under the form "the $u \in A$, the $v=f(u) \in B$" i.e. there is f: $Supp(A) \rightarrow Supp(B)$, so that $f(A) \subseteq B$. This condition can be rewritten under the form $(\forall)u \in A$, $\mu_A(u) \leq \mu_B(f(u))$, which is the relation suggested for the fuzzy function definition [3]. This last relation can be relaxed through the replacement of the function f with the fuzzy relation R, and thus we have the inequality $(\forall u$, v, $T(\mu_A(u), \mu_R(u,v)) \leq \mu_B(v)$ where T is a triangular norm. This way can be modeled sentences as "the $u \in A$, and u is in relation with v, the $v \in B$". Under such a circumstance, the truth degree of the antecedent limits the truth degree of the rule consequent.

iii) *Qualification rule of the possibility* are under the form "the $u \in A$, the more possible $v \in B$ is" that represents a partial description of the relation R between u and v. In this case, there takes place the inclusion $AxB \subseteq R$, which implies $\mu_R(u,v) \geq \min(\mu_A(u), \mu_B(v))$. Such a rule is used for the fuzzy control.

The semantic interpretation of the fuzzy rules is important to select some ϕ- operators in accordance with the rule significance. In case of the gradual rules, the Yager's principle of minimum specificity is sufficient to get the distributions of involved possibilities in this rule. For example, the R relation from the definition of the gradual rule is a fuzzy relational equation with the μ_R unknown[4]. The principle of minimum specificity application leads to the definition of the distribution of the possibility $\mu_{x|y}(u,v)$, which expresses the rules semantics as maximum solution for the relation $\mu_A(u) \leq \mu_B(f(u))$, thus $\mu_{x|y}(u,v) = \sup\{\mu|? T(\mu_A(u,\mu) \leq \mu_B(v), ?u,v\}$. This result endows the R – implications with representation semantics of the gradual rules. The minimum specificity principle is not sufficient to solve the above – mentioned inequalities, mainly when B is fuzzy. This last problem implies the use of qualification α-uncertainties applied to B. The foregoing aspects impose possible field of application of the fuzzy rules types in relation to their semantics for different types of thinking: uncertain/doubtful, interpolate, through analogy. In the **CFKBS** prototype case, the interferential subsystem based on fuzzy logic uses the generalized modus ponens interference diagram. The argument based on knowledge, which are represented under the form of some possible distributions, uses the similarity notion defined as the distance complement [6].

The compilation technique applies exactly to the pattern-matching level whose aim is to improve the filtering efficiency through the haustive looking into the M_K rules base. The content of the working memory

represents the **CFKBS** state at a certain given moment k∈T. The analysis of the classic filtering algorithm allows the inefficiency evidence of the inference engine control cycle that has to perform many futile operations. It forgets the achieved information of the previous cycles and performs the procedure to set into agreement of all the rules conditions with all the facts from the fact base. Realizing all these lacks, compilation techniques have been suggested based on the fact that with each control cycle only the facts subject to change will interfere in the conflict set. The changes from the fact base **F** are very few with regard to the number of control cycles performed by the inference engine.
The knowledge compilation allows: i) the settlement for each filter of the testing features as only the facts with the same features to have the possibility of unification with the filter; ii) the organization of a global network to allow the share of the common structures; iii) the keeping of the information achieved from the previous cycles within the network and the admission of the modified facts propagation.
The accepted knowledge by the **CFKBS** system are: i) variables (symbols always preceded by the symbol "?, as ?x, ?y, which will appear only in the rules); ii) atomic constants (numbers or series of characters); iii) fuzzy constants or possibilities distributions (symbols always preceded by the character "*" and used for the uncertainty representation); iv) logic operators. The fuzzy constant can occur both in facts and in rules and they always associate to the fuzzy multitudes (T-numbers) by means of *constfaz* function. The knowledge representation formalism in **CFKBS** system is the following:

*Antecedent:: = condition **
Condition::= motive½motive index_ motive ½ predicate½
index _ motive::= < atomic constant >
motive::= expression in which the three types of data are allowed;
predicate::=(predicate _sym predicate _arg predicate _arg)
predicate_arg::= predicate½ atomic constant ½fuzzy constant½variable
predicate _sym :: = {=, *<, >*, *? , *N}*
*Consequent = conclusion **
Conclusion::= motive ½motive index _ motive ½predicate½procedure

The predicates =*, *<, >* have binary values while the predicated *Π, *N are fuzzy. If there are introduced the fuzzy linguistic models within an expert system, this system becomes more complicated because of the taking into consideration of the fuzzy processing at all the system levels of the type: the fuzzy filtering/pattern-matching, the compatibility of the fuzzy sets, the fuzzy unification, the calculus of the inferred conclusion together with the calculation of the parameters propagation, which manage the uncertainty, the selection strategies in which they are naturally included and imprecise elements of the factual knowledge.
A major obstacle of the knowledge based systems use in real time applications is represented by the impossibility to predict the rules operation time. *The improvement of the condition-fact pattern-matching* is based on the following remark: "There can be stored certain features for the present conditions within the conditional part of the rules (those referring to the condition syntax). Using these features a tree can be built to discriminate the conditions into small groups or the individual conditions. Together with the fact occurrence, its features are tested. If the fact has the same features as the cause, it is possible that to be a condition instance. In this way, a minimum multitude of conditions that have the possibility to filter this fact can be established". The test tree in this manner built will be used to evaluate all the changes from the **F** fact base. The fact propagation is easily to administrate. The tests tree even though considerably ameliorates/ improves the condition- fact filtering operation, has also got some disadvan-

tages: the data can pass through the network on different paths. The use of binary tests makes the same information to be several times selected for the very same fact. Taking into account these disadvantages, a *unifying tree* structure has been suggested. This tree, as the test tree, discriminates more efficiently the conditions. Referring to the evaluation of the algorithmic complexity we can mention: i) For the *classic filtering algorithm* this gets the value O ($|F|.|M_k|$); ii) the test tree cost is O ($|F|$). In the most unfavorable case, there is no common conditions among the rules or O ($\log|F|$) in case of the most unfavorable hypothesis; iii) For the classic unification tree, the complexity is **O (log k),** where **k** represents the number of the distinct conditions from the rule base. The problem of the unification tree optimization can be also issued, but this belongs to the problem class of the **NP-** complete. Additionally, the evaluation time of the features and the occurrence frequency of the facts within the fact base are often unknown.

VLN has as aim to determine the global instances of the rules. The inner and graphical representation of the rules is achieved by a **VLN,** which is built by taking into consideration the discrimination tree leaves. It categorically reflects the relations among the rules conditions. The relations among the variables present in the conditions translate the relation among the rules conditions. The network is incrementally built by an iterative procedure, thus it: **i**) chooses two parts from the antecedent of a rule (motive or motive conjunction); **ii**) builds *a linking knot* that memorizes the necessary relations to be verified by the two parts (their conjunction); **iii**) creates a β-memory that allows the storage of the instances satisfying the present tests at the knots level from the **VLN** structure. The procedure is repeated till an instance at the global antecedent of a rule is achieved.

If many works have dealt with the problem of the answering time optimization for an expert system based on production rules, by means of the elaboration of more advanced algorithms, or by some parallel hardware structures, there are few attempts concerning the guarantee of the answering time by using specific filtering algorithms of Rete type.

Within the next section, we will present problems occurring during the filtering step of the interference motor cycle, underlying the classical features of a knowledge compiler, as well as the basic elements that improve the operation of a linguistic model fuzzy compiled by the **VLN** generation. Thus, we want to achieve the *predictability* of the filtering time for the Rete algorithm, adjusted for the fuzzy knowledge case, as this type of algorithms are used for the systems based on knowledge of real time [1,7].

3. The algorithms for the VLN generation

The elementary pattern-matching of the condition-fact fuzzy represents the first stage of the inference engine operation of the **CFKBS** system. The second stage is the variables linking to determine the consistent instances at the global level of any afferent antecedent of a fuzzy rule, starting with the instance multitude of the individual motives. The operation of the variables linking is expressed as follows: for a R rule with the conditions set **CND** (R)=$\{C_1,\ldots, C_K\}$, we must determine the set $\{(\sigma_1,F_1), (\sigma_2,F_2), ..., (\sigma_k,F_k)\}$ for **CND**(R), thus $(\sigma_j,F_j) \subset$**IC**(C_j) and $F_j=\sigma_j\cdot C_j$ (**IC**(C_j) represents the instance set of the C_j condition. If the terms associated to the common variable present within the substitutions are identical, then the substitutions $\sigma_1,\ldots, \sigma_k$ are consistent in the classic case. The composition of the consistent substitutions is written under the form: $\sigma=\sigma_1\bullet\sigma_2\bullet\ldots\bullet\sigma_k$ and includes all the distinct variables of the substitutions. During the variables linking stage, as all the possible instances of the rules must be settled, a problem of combinatory explosion can occur. The linking stage cost of the variables for the R rule is written by τ and can be evaluated as follows: $\tau=\prod_{k=1,n}$

$|IC(C_i)| \leq |F|^k$. For a $\mathbf{M}_K$ knowledge model expressed under the form of a rule can take place: $\tau_{MK} \leq |\mathbf{M}_K|\mathbf{x}|F|^k$. In the end the total cost of the classic algorithm is achieved and it is inferior to the value: $T\mathbf{x}K\mathbf{x}|F|\mathbf{x}\ |\mathbf{M}_K|\mathbf{x}|\mathbf{M}_K|\mathbf{x}|F|^k$ according to the shown model, the variable linking stage is more expensive than the condition–fact filtering stage.

3.1 The elementary VLN algorithm

By means of the fuzzy unification tree, the inference engine permits the fast association for all the facts (inclusive the new shaped ones) of motives, which these can filter. **VLN** is the second part of the compiler. It constitutes from the fuzzy unification tree leaves (**AUF**) shown in the [6] work and it gives the possibility to perform the linkings between the motives and the consistence control of the variables shown in the rules antecedent. The goal of this stage is to get the sets of $\mathbf{MC}_k$ whose elements are instances of the rules. From the classic filtering algorithm analysis of the interference motor results that a current problem in relation with the variables linking can be the inefficiency caused by the combinatory explosion.

We apply **VNL** in fuzzy case, having in mind the avoidance of the combinatory problems within the linking knots. We take into consideration an R simple rule: (A?x?y)(B?x ?y)→…) which involves two motives and the compilation network of the figure 1 fits to it. The **VNL** consists of linking knots and needles. The linking knots represent the location where the causes linking (the motives) takes place, the consistent instances are generated and stored by the fuzzy values compatibility control of the variables. A linking knot can have two inputs and one or several outputs (if the knot is split) and it serves for the consisted instances propagation.

If a variable occurs in the both inputs of the linking knot, then the associated values of this variable must be checked to provide the instance consistence. The performing tests are recorded within the linking knot from the figure 1 by test?x and test?y.

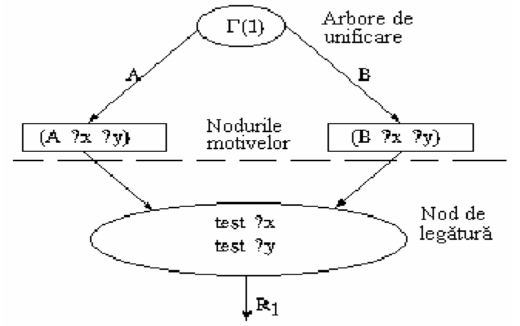

Fig.3 1. The linking knot from the VLN structure

To analyze the constituting algorithm of **VLN,** in the beginning we will present the afferent algorithm for one simple rule. Let be R a rule, **CND(R)**=$\{C_1,\ldots,C_k\}$, each condition C_i, i=1,…k can involve more variables $V(C_i)=(V(i,1),\ldots, V(i,j)$. The algorithm for the network generation is incrementally achieved by the selection of the most efficient linking knot from the combinatory instances limitation point of view. Next, we will designate through the antecedent part either a single condition of the rule, or a conjunction of condition, which are present in the rule antecedent. Each linking knot corresponds to an antecedent part. If P is a list of antecedent parts, then the generating algorithm of **VLN** for a single condition is the following one:

The algorithm A1. (**VLN** *for a single rule)*

The initialization: LS ¬ ($C_{1,\ldots,}C_k$)

Two parts of the rule antecedent presented within the list LS are chosen, which have a maximum number of common variables. Let be p' and p" the two parts of the antecedent selected from the list LS. Then, V(p')=(v'(1),…,v'(t)); V(p")=(v"(1),…, v"(u)) takes place. The selection may be guided by the following heuristics:

p' and p" have common variables within the maximum number. Let be C the list of common variables, C=V(p')Ç V(p")¹Æ;

the individual conditions and those with fewer facts are privileged.

A new linking knot is established, which represents the conjunction between p' and

p". This conjunction marked with p becomes a new antecedent part of the rule. Within this new knot all the variables from C must be tested. For p takes place: V(p)= V(p')È V(p")
p' and p" are eliminated from the LS list and p is added to the list.
If (êLSï ¹ 1) then pas 2, thus
Stop
The **VLN** structure is established by the selection stage, which has a heuristic character. Under a real situation there are lots of rules and the simplest method of **VLN** generation is the repetition of the foregoing algorithm. This method introduces invariably the redundancies within the variables linking network due to the difficult administration of the common structures' distribution of the rules (to minimize the knots within the network).

3.2 The general VLN algorithm
The important aspects that occur at this level are: i) VNL is not the single structural point of view (even if there is a single rule) and the efficient topology settlement needs difficult simulations; ii) within a VNL can exist structures which must be divided in common and such a thing modifies the outputs number of the network knots. Additionally, the efficiency of the network depends on the distribution of the facts within the fuzzy unifying tree knots that are difficult to evaluate and for this reason, always the facts are considered to be uniformly distributed.
The **VNL** generation for the R_i i= 1,…,n n>1, applying the above-mentioned algorithm, will introduce into the network the redundant knots due to the disregard of the common structures. An efficient algorithm in this case must solve in a convenient way the common structures distribution. The **VNL** building takes place incrementally. Each cycle of the generation algorithm of the network implies the adding of a linking knot in its structure. There are preferred the knots that can be distributed by several rules. The terminal knots of **VNL** correspond bijectivelly to the rules of the rule base. The second stage of the algorithm is meant to form the network efficiency. The linking of fuzzy variables resembles to the classic case. The important differences occur for the fuzzy unification and the fuzzy substitutions consistency.
Example. We examine in this example two **VLN** for a rule, as: (R (A ?x) (B ?y) (*C ?y) ? ...) (figure 2 - a and b).

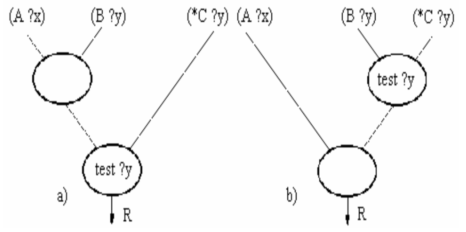

Fig.2. The VNL topologies for the rule R

The A2 Algorithm. (*VNL for a multitude of fuzzy rules)*
Let be **LCD***={C(1),…,C(k)} the list of distinctive conditions achieved as a border of the unification tree. Each C(i), i=1,…k condition is associated to two lists: V(C(i)) and NR(C(i)) where V(C(i)) represents the list with all the afferent variables to the C(i) condition and NR(C(i)) represents the list with the rule name according to which the condition C(i) can be determined. V(C(i))=V{v(i,1),...,v(i,j)}, NR(C(i))={R_i(1), ..., R_i (j)}.* **PANT***=the set of the antecedent parts.*
***1**. (PANT ¬ LCD)*
***2**. Select a pair of antecedent parts from the PANT list marked as pant' and pant". Let be R_LIST=NR(pant')ÇNR(pant"). Marking by LVC the common variable list between pant' and pant" we get:*
LVC= V(pant')Ç V (pant").
***3**. If |R_LIST|=0 then **Stop,** else go to 4*
***4**. Create a new linking knot.*
***4.1**. If êR__LISTï=1 (there is a single rule that contains the pant' and pant" conjunction), then the variables tests between pant' and pant" are recorded within the new linking knot further used for the building of the associated linking tree. The new linking knot will have a single output, marked pant. **pant** represents a new antecedent part and it will be associated to the*

NR and V list such: NR(pant)=êR__LISTï, V9pant)=V(pant')È V(pant")

***4.2** If êR__LISTï ³2, then the new created linking knot will be shared by at least two rules. There can be underlined all the variables tests from pant' and pant" for each rule from R__LIST.*

***5.** Classify the rules from R_LIST in several groups as per the differences of the tests (the rules are divided in an equal number of groups by the distinct tests). Let be g(i) such a group, i=1,…,n_g. Establish a (pant(i) output for the new linking knot and all the variables tests of the g(i) group are marked with the name of the (pant(i) output. It attaches the (pant(i) output to the NR(pant(i) lists=the rules of the g(i) group and V(pant(i)=V(pant')È V(pant").*

6. *Change the PANT list such that:*

- *Add pant or pant (i) to the PANT list*
- *For pant' and pant" selected at step 2:*

NR(pant')=NR (pant') - R_LIST and NR(pant")=NR (pant") – R_LIST

7. *If* |NR(pant')=0 *eliminate pant' from PANT;*

If |NR(pant")=0 *eliminate pant" from PANT;*

8. *Repeat (2)*

Stop

4. Conclusions

The problems like **FPS** [6] can be represented by a pair of series sets S={<x,y>}, in which each x series from this structure represents the $\mathbf{M}_k$ fixed part, and y represents the variable part **F**. The term of *fixed part* is different from the accepted meaning in the computational complexity analysis of the *constant*. In case of such a problem compilation, the fixed part is represented by that part from the problem input which are known before any on-line processing (for example the model made up of 25 rules). The main goal of the compilation is to solve difficult **NP**-problems within a poly-nominal time applying the preprocessing of some part from the input of the problem.

These remarks concerning the compilation of the guiding models justify the integration into the **CFKBS** structure of the fuzzy knowledge compiler, to which **VLN** belongs together the fuzzy unification tree, the inference engine, etc., with all its conceptual difficulties and its limits related to complexity aspects. In the future, it is desirable to apply such a preprocessing approach of the knowledge models in different other distributed **IA** systems such the multi-agent system for the assisted learning in which the classifying agent will work on the principles of the diagnosis systems. Thus, I have presented a series of results in the work [7].

References

1. Bouaud J., 1992 - *TREE: une strategie de jointure heuristique pour un algoritme de filtrage,* Revue d'Intelligence Artificielle, Vol. 6, n^o 4, 1992, Hermes, Paris, p. 457 - 493

2. Dubois D., Lang J., Prade H., 1992 - *Inconsistency in possibilistic knowledge bases: To live with it or not live with it,* Fuzzy Logic for the Management of Uncertainty, Edited by Lotfi Zadeh and Janusz Kacprzyk, Wiley Professional Computing, p. 335-351

3. Gottwald S., 1993 - *Fuzzy Sets and Fuzzy Logic/Foundations of Application - from a Mathematical Point of View,* Vieweg, ISBN 3-528-05311-9

4. Lange D.B., 1998 – *Mobile agents: environments, technologies, and applications.* In Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, 11-14. The Practical application Company

5. Mazilescu V., 1998 - *Fuzzy real time expert control systems. Characteristics and semantic of fuzzy compiled models,* Applications of Artificial Intelligence in Industrial Automation (Tempus Intensive Course Postrints), Ed. Academica Publishing, may 22-27, ISBN 973-97816-8-3, p. 26-1/26-15

6. Mazilescu V., 2002 – *The Predictability in a Multiagent System,* International Conference on Economic Cybernetics, October 01-06, Department of Economic Cybernetics, CSIE Faculty, Academy of Economic Studies, Bucharest, Romania