Pseudo Rule-Driven Model for a Programmable T-norm

Enrique Frías-Martínez Julio Gutiérrez-Ríos Felipe Fernández-Hernández
Dpto. de Tecnología Fotónica, Universidad Politécnica de Madrid
28660 Boadilla del Monte, Madrid, Spain
e-mail: efrias@dtf.fi.upm.es

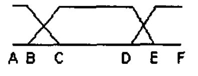
Abstract - **Rule-Driven processing has been proven to be a way of achieving high speed in fuzzy processing. Up to now, Rule-Driven architectures where designed to work with MIN or PROD as T-norm because they were the most commonly used in applications. This paper proposes a new Rule-Driven model valid for any T-norm (programmable T-norm) and any kind of membership functions (MF), provided the overlap factor is two, and they are a fuzzy partition. The architecture proposed can be implemented either by hardware or software.**

I. INTRODUCTION

The need to process Fuzzy Knowledge base systems with high speed resulted in the development of fuzzy hardware architectures. Two ways of achieving high speed are using paralelism and pipeline processing, concepts developed in computer architecture. Another option is to use Rule-Driven processing. In that case only the rules relevant to the final result are executed, which produces very high speed fuzzy processors [4], because the set of the relevant rules is only a small part of the complete Fuzzy Knowledge Base.
Ikeda [7] developed one of the first rule-driven architectures. Other Authors, like [1], [2], [3] and [6], have developed fuzzy rule-driven processors, but all of them were designed for a MIN T-norm. The design of a rule-driven architecture depends on the T-norm used by the system, because it (with the T-conorm if necessary) determines if a rule is relevant or not by calculating the degree of truth of the antecedent.

II. BASIC CONCEPTS

A Fuzzy System, *FS*, is defined as a vector, $FS=(T_n,T_c,M,OP,OA,D,R,MF,I,O)$, with T_n the T-norm, T_c the T-conorm, *M* the set of modifiers, *OP* the propagation operator, *OA* the aggregation operator, *D* the deffuzification algorithm, *R* the set of Rules , *MF* the membership functions defined as a fuzzy partition [Kli88] and overlapping of two, *I* the inputs and *O* the outputs of the system [Fri99]. For the rest of the paper we will consider a system with $Card(I)=2$.
For $I_i \in I$, the Activation Intervals of I_i (AI_i) are defined as the union of the set of intervals given by the supports of the membership functions defined in *I*. See *fig. 1.*



$AI_1=\{(A,B),(B,C),(C,D),(D,E),E,F)\}$

Fig. 1 Activation Intervals of I_1.

Given AI_i, $i=1...Card(I)$, where $Card(I)$ gives the cardinality of I, and *S* the N-dimensional input space given by $I_1,..,I_n$, *P* is defined as a partition of *S*, given by the cartesian product (*X*) of AI_i $i=1...Card(I)$,

$$P = \underset{i=1...Card(I)}{\mathrm{X}} AI_i$$

A rule R_i is defined as:

$$If\ I_1\ Is\ MF_{1i}\ and\ I_2\ is\ MF_{2j}\ Then\ O\ is\ MF_k \quad [2]$$

P_j is defined by $(IA_{1i'}, IA_{2j'}, ..., IA_{Nk'})$. It will be said that a rule $R_i \in R$ is included in a partition $P_j \in P$ if and only if:

$$Ri \subset Pj \Leftrightarrow \forall MF_{ij}\ of\ Ri\ verifies\ MF_{ij} \cap IA_{ij'} \neq \varnothing \quad [3]$$

A rule will be part of a partition if and only if every membership function of the rule is in the *AI* of the correspondent dimension of *Pj.*
CR is defined as the classification of the set of rules *R* in the partition *P* using [3].
The Activation Area of a rule R_i, in a partition P_j, $AA(R_i, P_j)$ is defined as the area of P_j, where the inputs of the system make the rule relevant to the output.
Given a rule R_i defined by MF_{1i} and MF_{2i}' as in [2], $R_i \in P_j$ and defining the inequalities $\{F_1,F_2,F_3\}$ as the regions that mark the area where T_n is sensible (with a value greater that 0), the Activation Area, $AA(R_i, P_j)$, is given by the union of the following areas:

$$F_1(X,Y) \geq 0$$
$$F_2(X,Y) \geq 0$$
$$F_3(X,Y) \geq 0$$

Where $X=MF_{1i,}$ *and* $Y=$ $MF_{2i'}$.
The Activation Area of a rule R_i in the input space defined by *I* , *S*, is the addition of the Activation Areas of R_i in all the partitions of P.

$$AA(R_i) = \Sigma AA(R_i,P_j)\ j=1..Card(P)$$

III. ACTIVATION AREA OF GENERIC T-NORM

The model presented in this paper is designed to work with a generic T-norm, so the only information available about the T-norm is the one given by its definition [8]:

$$T:[0,1] \times [0,1] \rightarrow [0,1]$$

Satisfying:

- $T(0,0) = 0;\ T(a,1) = T(1,a) = a$ (boundaries)
- $T(a,b) \leq T(c,d)$ Si $a \leq c$ y $b \leq d$ (monotony)
- $T(a,b) = T(b,a)$ (commutativity)
- $T(a,T(b,c)) = T(T(a,b),c)$ (asociativity)

With this information the general appearance of the *Activation Area* of a rule $Ri \in R$ in the input space can be sketched. The *Activation Area* (*AA*) of a rule is divided in two parts, AA_1 which represents the part of the *Activation Area* where one of the inputs is 1, and AA_M, which represents the part of the *Activation Area* where none of the inputs are 1, as seen in (3).

$$AA(R_i) = AA_1(R_i) + AA_M(R_i) \quad (3)$$
$$AA_1(R_i) = \{(x,y) \,/\, x=1 \vee y=1\}$$
$$AA_M(R_i) = \{(x,y) \,/\, x\neq 1 \wedge y\neq 1 \wedge T(x,y) > 0\}$$

The key of the representation is $T(a,1)=T(1,a)=a$, which produces the area marked with ① in *fig. 2*. This area is the part of the *Activation Area* given by AA_1. The striped area is where the T-norm is calculated for the rest of the cases, and in that case the representation will depend on the mathematical definition of the T-norm. This part of the *Activation Area* is the part given by AA_M. Only a part of the striped area will be AA_M. The maximum value of AA_M will be the complete striped area, and the minimum value 0, in that case *AA* will be the cross given by AA_1.

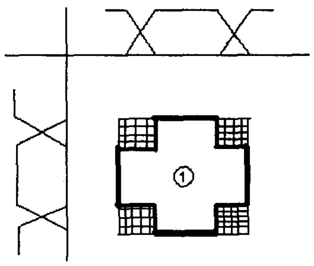


Fig.2 Activation Area for a generic T-norm.

Fig. 4 represents the *Activation Area* of the rules given in *Fig. 3* for a generic T-norm, this example will be used later in the model.

R_1 :	*If*	I_1	*Is* MF_{11}	*And*	I_2	*Is* MF_{21}	*Then*	*Z*	*Is*	S_1
R_2 :	*If*	I_1	*Is* MF_{12}	*And*	I_2	*Is* MF_{21}	*Then*	*Z*	*Is*	S_2
R_3 :	*If*	I_1	*Is* MF_{11}	*And*	I_2	*Is* MF_{22}	*Then*	*Z*	*Is*	S_3
R_4 :	*If*	I_1	*Is* MF_{12}	*And*	I_2	*Is* MF_{22}	*Then*	*Z*	*Is*	S_4

Fig, 3. Fuzzy Knowledge Base.

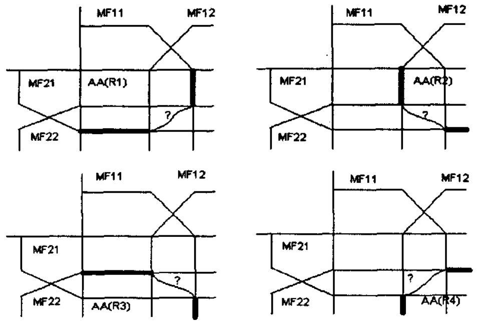


Fig. 4 Activation Areas of Fig.3

IV. SENSIBLE AREAS OF THE INPUT SPACE

The concept of *Sensible Area* (*SA*) is the key concept of the model. A *Sensible Area* of the input space, is defined as the area where a set of active rules is executed (4).

$$R=\{R_1,\ldots,R_k,R_{k+1},\ldots,R_n\}$$
$$SA\{R_1, \ldots R_k\} = \{(x,y) \,/\, (x,y) \in AA(R_1) \vee \ldots \vee$$
$$(x,y) \in AA(R_k) \wedge (x,y) \notin AA(R_{k+1}) \wedge (x,y) \notin AA(R_n)\} \quad (4)$$

Sensible Areas are obtained from the intersection of *Activation Areas. Fig. 5* represents the *Sensible Areas* of a system with a generic T-norm, and the set of active rules that define them, where $SA8=\{R_1, R_2, R_3, R_4\}$. A similar concept has already been calculated in the previous step, for calculating the *Activation Areas*. *CR*, the classification of *R* in *P*, contains very similar information, but with a different semantic. In this case the area given by *(x,y)* is associated to a set of rules, in *CR*, the rules are associated to a partition *P*.

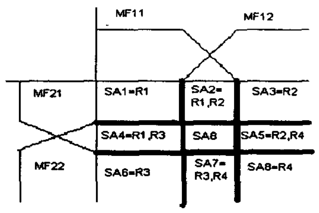


Fig. 5 *Sensible Areas* of the knowledge base.

The general appearance of the sensible areas of a two-input system with trapezoidal Membership Function and a programmable T-norm is given in *Fig. 6.* This partition is the same for any of the considered membership functions.

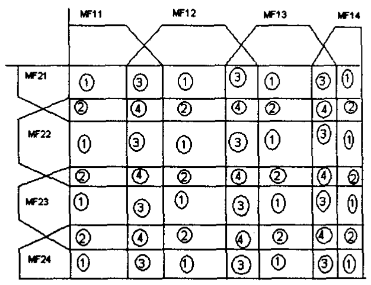


Fig. 6 Sensible Areas of the input space.

There are four different types of *Sensible Areas(SA)*:

- *SA* marked with ① have only one rule associated (the rule constructed with the related membership functions)
- *SA* marked with ②and ③ have two rules associated.
- *SA* marked with ④ have four rules associated.

The reason for naming the model Pseudo Rule-Driven comes from having four rules associated with ④. These four rules associated are not always relevant. As the model deals with a programmable T-norm, there is no way of identifying the active ones. Furthermore, in the case that the model had information about the T-norm, it could be that the mechanism for identifying the relevant rules would consume more time than executing the rules.

V. IDENTIFICATION OF THE SENSIBLE AREAS

In order to determine the relevant rules, a mechanism to map the inputs of the system to the *Sensible Areas* is needed. The solution proposed by the model consists of the equalization of the input membership functions (with overlap of two and partition of unity), so that the system works with a predetermined set of known Membership Functions. The equalization [5] is done by the function *N(x)* (5). This function is the same for any normalized Membership Functions.

$$N(x) = MF_{11}(x)*0+MF_{12}(x)*1+MF_{13}(x)*2+MF_{14}(x)*3 \quad (5)$$

The equalized membership functions are fuzzy partition triangles with overlap of two and range between *[0...Card(MF)]*. An example of the equalization done for an input with trapezoidal MF is given in *Fig. 7.*

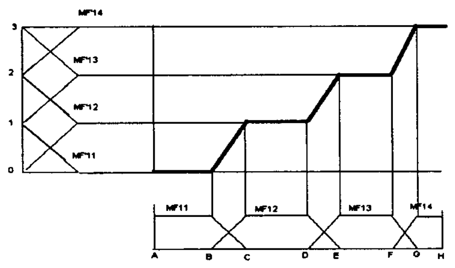


Fig.7 Example of equalization done by *N(x).*

As a consequence of the Membership Functions having been equalized, the input space of the system is equalized also. *Fig. 8* presents the equalization of the system presented in *fig. 6*. Basically, all the constant parts of the Membership Functions disappear, so only the *Sensible Areas* marked with ④ remain in the equalized input space.

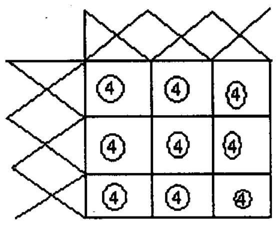


Fig. 8 Equalized input space.

Working in the equalized space makes identification of the *Sensible Areas* much easier. Defining the vector $I'=(I'_1,I'_2)$ as the equalization of the input vector $I=(I_1,I_2)$:

- If I' is in ①, I'_1 and I'_2 will be integer, and the associated rule will be given by (I'_1,I'_2).
- If I' is in ②, I'_1 is integer and I'_2 is non-integer. The set of two rules associated will be given by $(I'_1, integer(I'_2))$
- If I' is in ③, I'_1 is non-integer and I'_2 is integer. The set of two rules associated will be given by $(integer(I'_1),I'_2)$.
- If I' is in ④, both I'_1 and I'_2 are non-integers. The four rules associated will be given by $(integer(I'_1), integer(I'_2))$

Each one of the pointers to the active rules is not only identified by a pair of integers, but also by the kind of area where the non equalized input is in.

VI. ARCHITECTURE OF THE MODEL

The modules of the proposed architecture can be seen in *fig. 9*. The architecture presented refers only to the Rule-Driven system. Once the active rules have been identified, the fuzzy inference system will execute the rules. The Rule-Driven architecture proposed is independent from the architecture of the inference system, as seen in *Fig.10*

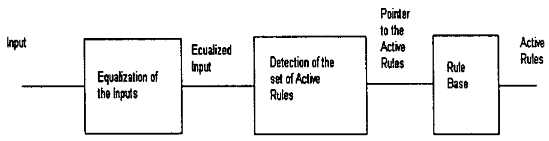

Input
Equalization of the Inputs
Ecualized Input
Detection of the set of Active Rules
Pointer to the Active Rules
Rule Base
Active Rules

Fig. 9 Architecture of the Rule-Driven model.

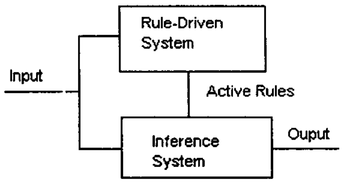


Fig. 10 Interface with the Inference System.

Once an input ***I*** has been equalized, the system will detect which of the conditions has been met. Once it is found, a list with pointers to the different active rules is obtained. From these pointers, the Active Rules are read from the Rule Base and executed.

The key of the Architecture is the "Detection of the Set of Active Rules" module. This module from the equalized input (I'_1,I'_2) produces a set of pointers to the relevant rules which are used as inputs to the Rule Base. The algorithm to the detect the set of pointers is given in *Fig. 11*

```
function ACTIVE_RULES (X'1,Y'1)
X'1,Y'1 : EQUALIZED_INPUTS
{

if ( X'1 is INTEGER)
        if  (Y'1 is INTEGER) return RULE_1[X'1,Y'1]
        else return RULE_2[X'1,integer(Y'1)]
else
        if  (Y'1 is INTEGER) return Rule_3[integer(X'1),Y'1]
        else return Rule_4[integer(X'1),integer(Y'1)]
}
```

Fig. 11 Algorithm of detection of *Sensible Areas*

The data structure needed for the algorithm is given in *Fig. 12.* All these structures are generated off-line.

```
#define X    NUMBER_OF_MF_OF_I1
#define Y    NUMBER_OF_MF_OF_I2
array RULE_1[X][Y] of RULE_POINTER
array RULE_2[X][Y-1] of array RULE_POINTER[2]
array RULE_3[X-1][Y-1] of array RULE_POINTER[2]
array RULE_4[X-1][Y-1] of array RULE_POINTER[4]
```

Fig.12 Data structures for Active Rule detection.

VII. RULE-DRIVEN ARCHITECTURE VS. NON RULE DRIVEN ARCHITECTURE

The comparative of both solutions is done in terms of memory needed and execution time.

Memory Comparison
A non Rule-Driven solution will need only the Rule Memory. A Rule-Driven solution, in addition to the Rule Memory, needs the structures of *Fig. 12.* The total amount of memory in bytes is given by:

$$\textit{Num. of Bytes} = XY + 2X(Y-1) + 2(X-1)Y + 4(X-1)(Y-1) \;\; * B \;\; \textit{bytes/pointer} \quad (6)$$

where X is the number of membership functions of I_1 and Y of I_2. In the example provided by *fig. 3*, $X=4$, $Y=4$, which gives a total of 100 pointers, and with 4 bytes/pointer gives a total of 400 bytes more than the non-Rule Driven solution. This result is without adding the memory needed to code the "Detection of the Set of Active Rules" module.

Execution-time Comparison
The execution time for a Non Rule-Drive 2-Input system with 4 MF per input is $16*K$, where K is the execution time of a Rule.
In the case of the Rule-Driven architecture proposed, given an equal probability for the 2 inputs of the system to have any value of the input space, and a typical set of membership functions given in *fig. 13*, the probability of the input I to active the rule set of each kind of Sensible Area is:

For ①: $P_1 = \dfrac{4a^2+4b^2+8ab}{(2a+2b+3c)^2}$

For ②: $P_2 = \dfrac{6bc+6ac}{(2a+2b+3c)^2}$

For ③: $P_3 = P_2$

For ④: $P_4 = \dfrac{9c^2}{(2a+2b+3c)^2}$

With these probabilities, the time of execution of the model is given by:

$$T = (P_1+2P_2+2P_3+4P_4)\,K \qquad (7)$$

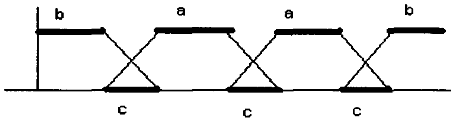


Fig.13 Typical structure of MF.

The amount of time saved for a two input system with four MF per input is given by :

$$\textit{Time Saved} = 100 - 6.25\,(P_1+2P_2+2P_3+4P_4) \quad (8)$$

For example in a system with $a=2$, $b=1.5$, $c=1$, the time saved by the rule-driven system will be near 90%. With these benefits, the increase of the memory needed by the rule-driven model is justified.

VIII. CONCLUSIONS AND FUTURE WORK.

This paper presents a Rule-Driven architecture for a programmable T-norm and any kind of membership functions provided an overlap factor of two and that they are defined as a fuzzy partition. The results obtained for a typical two-input fuzzy system produces a time savings of up to 90%. This makes the model proposed completely worth while. The architecture can be implemented both in hardware and in software.
This model is a first step towards designing a full programmable high speed fuzzy controller.

ACKNOWLEDGMENT

This paper is supported by grant CICYT-TIC96-1393-C06-01.

REFERENCES

[1] G. Ascia, V. Catania, , L. Vita, "Rule-Driven VLSI Fuzzy Processor", *IEEE Micro*, Vol. 16, No. 3, June 1996
[2] T. Chiueh, "Optimization of Fuzzy Logic Inference Architecture", *IEEE Computer*, May 1992
[3] D. Falchieri, "Very Fast VLSI Fuzzy Processor: 2 inputs 1 Output", *FuzzIEEE 1997*, Barcelona
[4] E. Frias, J. Gutierrez, F. Fernandez, "Modelo de Procesamiento Rule-Driven para Suistemas con T-norma Generica", *EUSFLAT-ESTYLF Joint Conference*, 1999
[5] Felipe Fernandez, "The FRISC Model", Technical Report, Dpto. Tecnologia Fotonica, unpublished, UPM, 1998
[6] A. Gabrielli, "Design of a VLSI very high speed reconfigurable digital fuzzy processor", *Proceedings of the ACM* 1995, Nashville
[7] H. Ikeda, "A fuzzy Inference Coprocessor Usign a Flexible Active-Rule-Driven Architecture", *Proc. IEEE Int. Conf. On Fuzzy Systems,* March 1992
[8] G.J. Klir, T.A. Folger, "Fuzzy Sets, Uncertainty and Information", Prentice Hall,1988