Original paper

Soft Computing 7 (2002) 65 – 71 © Springer-Verlag 2002
DOI 10.1007/s00500-002-0175-9

Design of a Lukasiewicz rule-driven fuzzy processor

E. Frías-Martínez

Abstract Rule-driven processing is a proven way of achieving high-speed in fuzzy processing. Up to now, rule-driven architectures where designed to work with minimum or product as T-norm. Nevertheless, a Lukasiewicz T-norm is typically used with the compositional rule of inference in expert systems applications that are based on a fuzzy inference engine. This paper presents a rule-driven processing architecture for systems with a Lukasiewicz T-norm and partition of unity membership functions with an overlap factor of two.

Keywords Fuzzy processing, Rule-driven processing, T-norm, Lukasiewicz

1 Introduction

The need to process fuzzy knowledge base systems with high speed resulted in the development of fuzzy hardware architectures. Rule-driven processing is a technique used to obtain high-speed fuzzy processors, because it allows only relevant rules to be executed and this set of relevant rules comprise only a small part of the complete fuzzy knowledge base. Ikeda [6] developed one of the first rule-driven architectures. Other authors, like [1–5], have developed fuzzy rule-driven processors but, all of them, were designed for minimum or product as T-norm. The design of a rule-driven architecture depends on the T-norm used by the system, because it determines if a rule is relevant by calculating the degree of truth of the antecedent.

Lukasiewicz T-norm is used with the compositional rule of inference in the case that it is important that the system verifies:

$$A \cap \mathrm{NEG}(A) = 0 \quad (1)$$

$$A \cup \mathrm{NEG}(A) = 1 \quad (2)$$

This is true for a Lukasiewicz T-norm with the negation operator defined as:

$$\mathrm{NEG}(A) = 1 - A \quad (3)$$

These characteristics are typically used in knowledge-based systems and expert systems applications. Since these systems usually have a big number of rules, a rule-driven system will improve its efficiency. Although some implementations have been done [8, 9], no rule-driven architecture for Lukasiewicz has been developed. In this paper, a rule-driven architecture for high speed processing of systems with a Lukasiewicz T-norm (4) is proposed.

$$T_n = \max(0, (a + b - 1)) \quad (4)$$

2 Justification of a rule-driven architecture

$\mathrm{FS} = (T_n, T_c, P, A, D, R, \mathrm{MF}, \mathrm{MF}_o, I, O)$ is a vector that defines a MISO fuzzy system with T_n the T-norm, T_c the T-conorm, P the propagation operator or implication function [7], A the aggregation operator [7], D the defuzzification algorithm, R the set of rules, MF the membership functions defined on the inputs, MF_o the membership functions defined on the output, $I = (I_1, \ldots, I_n)$ the input and $O = (O_1)$ the output of the system. The system's only restriction is that the membership functions defined on the inputs, MF, are normalized, with an overlap factor of two and partition of unity (the sum of all of them for each input equals one).

Given $m = \mathrm{Card}(R),\ n = \mathrm{Card}(I)$, and G the overlap factor of the input membership functions, R_a, the set of active rules, verifies that:

$$R_a = n^G \ll m \quad (5)$$

3 Lukasiewicz rule-driven model

The proposed model is designed to obtain the set of active rules for Lukasiewicz T-norm from an input. Figure 1 presents the graphic of a Lukasiewicz T-norm.

3.1 Mathematical definition of the model

The model, in order to obtain the set of active rules, is based on some definitions. The classification of rules (CR) is defined in a way, that, for a given input, it gives the maximum set of possible active rules. From the classification of rules, the concept of sensible area is introduced. A sensible area is an association between an area of the input space of the system and a set of rules. This association allows to identify the set of active rules from an input.

3.1.1 Maximum set of active rules

To obtain the maximum set of active rules for an input, a classification criteria for the rules is needed. This classi-

E. Frías-Martínez (✉)
Dpto. de Tecnología Fotónica,
E.T.S.I. de Telecomunicación, U.P.M. 28040,
Ciudad Universitaria s/n, Madrid, Spain
e-mail: efrias@tfo.upm.es

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

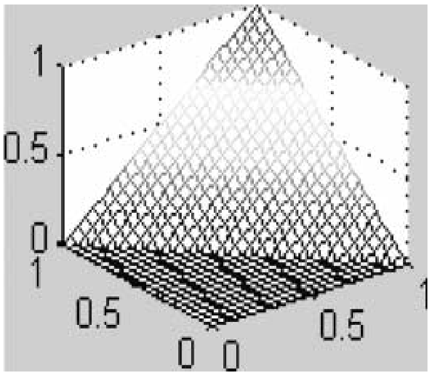

Fig. 1. Graph or the Lukasiewicz T-norm

fication criteria will associate each rule with a cell of the input space of the systems. Each one of this cells will be defined by a partition of the system. In order to obtain this partition, the concept of activation interval is introduced.

For each $I_i \in I$, the activation intervals of I_i, AI_i, are defined as the union of the set of intervals, left-closed and right-open, except the last one which is also right-closed, given by the extreme points of the kernel of each one of the membership functions defined in I_i. Figure 2 presents an example of activation intervals.

Formally, being $p = \mathrm{Card}(\mathrm{MF}_i)$, the activation intervals of I_i can be obtained as:

$$\mathrm{AI}_i = \cup \mathrm{Kernel_e}(\mathrm{MF}_{i,m})) \quad \text{with } m = 1, \ldots, p \tag{6}$$

where $\mathrm{Kernel_e}$ is a function that obtains the extreme points that define a kernel.

The set of activation intervals of the system are used to define a partition of the system. Given $\mathrm{AI}_i,\ i = 1, \ldots, \mathrm{Card}(I)$, and S the n-dimensional input space given by $(I_1, \ldots, I_n)$, P is defined as a partition of S given by the cartesian product $(\times)$ of $\mathrm{AI}_i,\ i = 1, \ldots, \mathrm{Card}(I)$:

$$P = \times \mathrm{AI}_i,\ i = 1, \ldots, \mathrm{Card}(I) \tag{7}$$

Once a partition has been defined, the model defines a criteria to classify the rules. Given $(\mathrm{AI}_{1,p'}, \ldots, \mathrm{AI}_{n,k'})$ the cell P_j of P, and the rule R_m defined as:

$$\begin{aligned} R_m:&\ \text{If } I_1 \text{ is } \mathrm{MF}_{1,p} \text{and} \ldots \text{and} \\ &\quad I_n \text{ is } \mathrm{MF}_{n,k} \text{ Then Consequence} \end{aligned} \tag{8}$$

It will be said that R_m is included in the cell P_j if and only if:

$$\begin{aligned} R_m \in P_j &\Leftrightarrow \forall(i,l,h) \\ &= (1,p,p'), \ldots, (n,k,k') \text{ verifies } \mathrm{MF}_{i,l} \cap \mathrm{AI}_{i,h} \neq \emptyset \end{aligned} \tag{9}$$

A rule R_m will be included in a cell P_j if and only if every membership function $\mathrm{MF}_{i,l}$ of the rule has a non empty intersection with the correspondent dimension $\mathrm{AI}_{i,h}$ that defines the cell P_j.

The classification of rules (CR) is defined as the classification of the set of rules R in P using (9). CR is expressed as a vector containing the active rules for each cell of P:

$$\begin{aligned} \mathrm{CR} &= (\mathrm{CR}_1, \ldots, \mathrm{CR}_k) \quad \text{with } k = \mathrm{Card}(P), \\ \mathrm{CR}_i &= (R_{k'}, \ldots, R_{j'}) \quad \text{with } i = 1, \ldots, k \end{aligned} \tag{10}$$

Example Given $\mathrm{FS} = (T_n, T_c, P, A, D, R, \mathrm{MF}, \mathrm{MF_o}, I, O)$ a fuzzy system with $n = 2$, $R = (R_1, R_2, R_3, R_4)$:

$$R_1\text{: If } I_1 \text{ is } A_{1,1} \text{ and } I_2 \text{ is } A_{2,1} \text{ Then } Z \text{ is } Z_1 \tag{11}$$

$$R_2\text{: If } I_1 \text{ is } A_{1,1} \text{ and } I_2 \text{ is } A_{2,2} \text{ Then } Z \text{ is } Z_2 \tag{12}$$

$$R_3\text{: If } I_1 \text{ is } A_{1,2} \text{ and } I_2 \text{ is } A_{2,1} \text{ Then } Z \text{ is } Z_3 \tag{13}$$

$$R_4\text{: If } I_1 \text{ is } A_{1,2} \text{ and } I_2 \text{ is } A_{2,2} \text{ Then } Z \text{ is } Z_4 \tag{14}$$

The membership functions are $(A_{1,1}, A_{1,2})$ and $(A_{2,1}, A_{2,2})$, defined in I_1 and I_2 respectively. The activation intervals will be (see Fig. 3):

$$\mathrm{AI}_1 = ([0, A), [A, B), [B, C]) \tag{15}$$

$$\mathrm{AI}_2 = ([0, A'), [A', B'), [B', C']) \tag{16}$$

The partition P of S will be (see Fig. 3):

$$\begin{aligned} P &= (P_1, P_2, \ldots, P_9) \\ &= (([0, A), [0, A')), ([0, A), [A', B')), ([0, A), [B', C']), \ldots) \end{aligned} \tag{17}$$

The classification of rules (CR) will be (see Fig. 3):

$$\begin{aligned} \mathrm{CR} = ((R_1), (R_1, R_2), (R_2), (R_1, R_3), \\ (R_1, R_2, R_3, R_4), (R_2, R_4), \ldots) \end{aligned} \tag{18}$$

The classification of rules (CR) shows for a Lukasiewicz T-norm the maximum number of active rules that can be executed for an input I included in a cell P_k.

3.1.2 Sensible areas

The Lukasiewicz rule-driven model is based on sensible areas. A sensible area k, SA_k, is defined by the region of the cell P_k and the set of associated rules CR_k, the maximum set of active rules.

$$\mathrm{SA}_k = \{P_k, \mathrm{CR}_k\} \tag{19}$$

The general appearance of the sensible areas of a two-input system with trapezoidal membership function is given in Fig. 4. This partition is the same for any type of partition of unity membership functions with an overlap factor of two. There are four different types of sensible areas:

- sensible areas marked with 1 are defined by the kernel of two membership functions, and have only one rule associated.

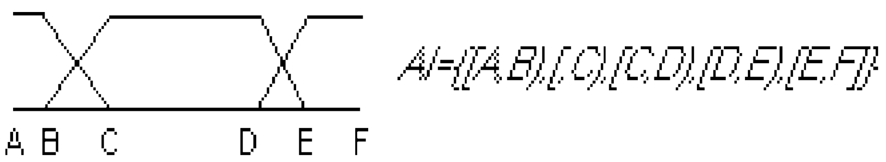

Fig. 2. Activation intervals (AI)

- sensible areas marked with 2 and 3 are defined by one kernel and by one transition between 0 and 1 of the membership functions, and have two rules associated.
- sensible areas marked with 4 are defined by two transitions between 0 and 1 of the membership functions, and have four rules associated.

Because $T_n(a, 1) = a$, if the input is included in a sensible area of type 1, 2 or 3 the rules associated will be the set of active rules. In case the input is included in a sensible area of type 4 the set of active rules will be a subset of the associated rules because none of the membership functions equals one. In a system with a Lukasiewicz T-norm, if none of the membership functions is one, there is no guarantee that the rule will be active. In case the input is in a sensible area of type 4, the set of active rules will have to be determined.

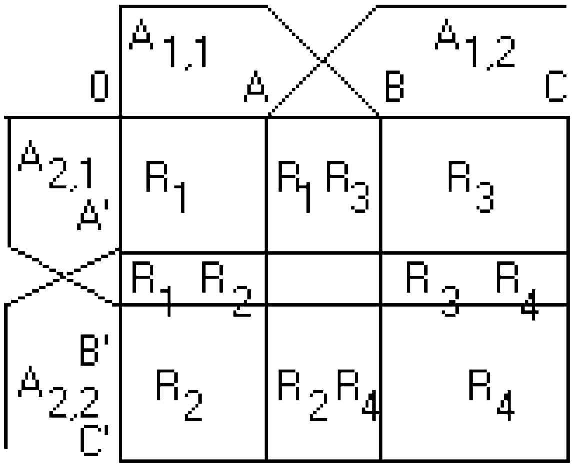

Fig. 3. AI_1, AI_2, P and CR

In order to determine efficiently the set of active rules, a mechanism to obtain this set from the input of the system is needed.

3.2 Architecture of the model

The architecture proposed is divided in two parts, the off-line processing, which adapts the representation of the system to achieve high speed, and the on-line architecture, which implements the results of the off-line processing to run the system.

3.2.1 Off-line processing

The objective is to compile the system to identify the set of associated rules with high speed. The off-line processing is divided in two steps:

- Equalization of the membership functions and of the sensible areas.
- Identification of the set of active rules.

3.2.1.1 Equalization of the membership functions and the sensible areas

The equalization of the membership functions of the system allows to obtain in an efficient way the sensible area in which an input I is included, and to identify the set of active rules. The equalization is done with the function N.

For each dimension $i = 1, \ldots, n$ the equalization function $N_i(I_i)$, is defined as:

$$N_i(I_i) = \mathrm{MF}_{i,1}(I_i) \cdot 0 + \mathrm{MF}_{i,2}(I_i) \cdot 1 + \cdots + \mathrm{MF}_{i,k}(I_i) \cdot (k-1), \quad \text{with } k = \mathrm{Card}(\mathrm{MF}_i) \qquad (20)$$

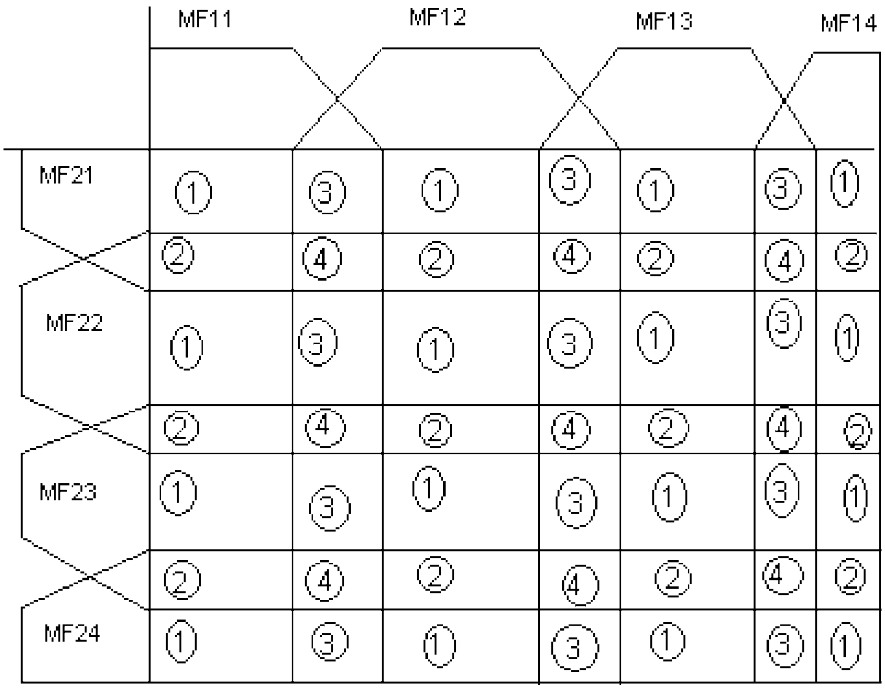

Fig. 4. Sensible areas of the input

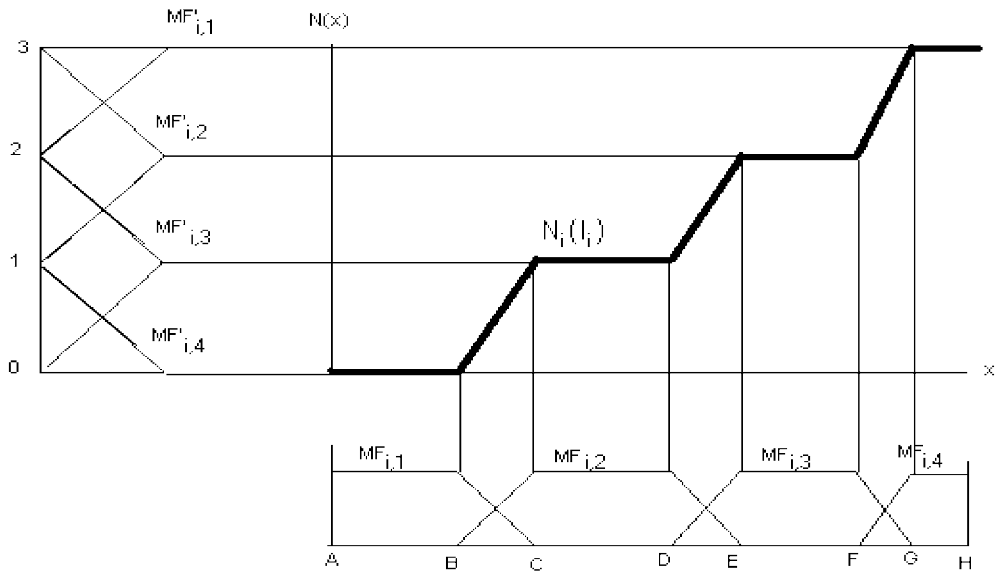

Fig. 5. Example of equalization done by $N_i(I_i)$

The membership functions obtained from applying the equalization function N_i are triangles with an overlap factor of two and partition of unity. The domain of each equalized dimension is $[0, \ldots, \mathrm{Card}(\mathrm{MF}_i) - 1]$, with $\mathrm{Card}(\mathrm{MF}_i)$ the number of membership functions defined in I_i. The set of equalization functions has been designed to eliminate the constant parts of the membership functions of the original system, and to be sensible to the variations. Figure 5 presents an example of the equalization done by $N_i(I_i)$.

Due to the equalization of the membership functions, the sensible areas are also equalized. Figure 6 presents the equalization of the sensible areas and membership functions given in Fig. 4. Basically, all the constant parts of the membership functions disappear, so only the sensible areas of type 4 remain in the equalized input space.

3.2.1.2
Identification of the set of active rules

Working in the equalized space makes the identification of the sensible areas easier. Defining $I' = (I'_1, I'_2)$ as the equalized input of $I = (I_1, I_2)$:

- If I'_1 and I'_2 are integers, I is included in a sensible area of type 1, and the rule associated will be identified by (I'_1, I'_2). The value of the T-norm will be 1.
- If I'_1 is integer and I'_2 is non-integer, I is included in a sensible area of type 2. The set of two rules associated will be identified by $(I'_1, \mathrm{integer}(I'_2))$. The T-norm will be fraction (I'_2) for one rule and *1*-fraction (I'_2) for the other.
- If I'_1 is non-integer and I'_2 is integer, I is included in a sensible area of type 3. The set of two rules associated will be identified by $(\mathrm{integer}(I'_1), I'_2)$. The T-norm will be fraction (I'_1) for one rule and *1*- fraction (I'_1) for the other.
- If both I'_1 and I'_2 are non-integers, I is included in a sensible area of type 4. This case is identified by $(\mathrm{integer}(I'_1)$, $\mathrm{integer}(I'_2))$. In that case the Lukasiewicz T-norm has to be calculated.

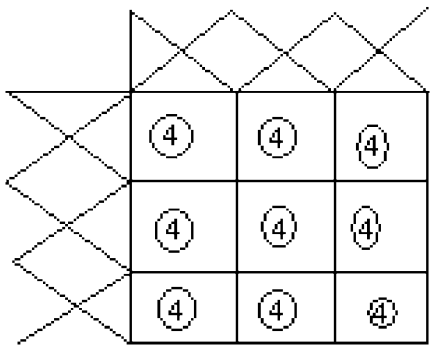

Fig. 6. Equalized input space

In case the input is included in a sensible area of type 4, the set of active rules has to be obtained. This set will be a subset of the associated rules of the sensible area.

Each one of the sensible areas of type 4 can be divided in four triangles (Fig. 7). This division, given by the definition of the Lukasiewicz T-norm (1), allows to obtain the set of active rules for a given input. Each one of those triangles will have two active rules associated.

In the equalized input space, the set of triangles of the sensible areas of type 4 will be seen as diamonds (Fig. 7). It has to be determined in which of this diamonds is the input, to obtain the two active rules associated. For that, each one of those sensible areas will have 4 points (C_1, C_2, C_3, C_4) associated (Fig. 7) which will represent the center of the four diamonds involved. In order to determine in which of the diamonds is the input, (21) has to be verified, where (I'_1, I'_2) is the equalized input, and $(C_{i,1}, C_{i,2})$ the center of the diamond. (21) is a ρ_1 metric [7] defined as a sphere or ball, $\rho_1((x,y),(1,1)) \leq 1$.

$$|I'_1 - C_{i,1}| + |I'_2 - C_{i,2}| \leq 1 \qquad (21)$$

3.2.2
On-line architecture

The architecture of the on-line system can be seen in Fig. 8. The rule-driven system will identify and pass the set of active rules to the inference system. The inference system will execute only this set of rules and not the whole rule base.

The rule-driven system proposed, Fig. 9, is independent from the architecture of the inference system, and is based on the off-line processing defined in the previous point.

The rule-driven system is divided in three blocks, the equalization of the inputs block, the detection of the set of active rules block, and the rule base.

In the equalization of the inputs block, from a given input $I = (I_1, \ldots, I_n)$, the equalized input $I' = (I'_1, \ldots, I'_n)$ is obtained applying the set of Equalization Functions $N_i(I_i)$, where $I'_i = N_i(I_i)$ with $i = 1, \ldots, n$. The set of functions $N_i(I_i)$ typically will be stored in a memory to obtain high efficiency.

In the detection of the set of active rules block, from I', a set of pointers to the active rules is obtained. Given X the number of membership functions of I_1, Y then number of membership functions of I_2, and ND the number of

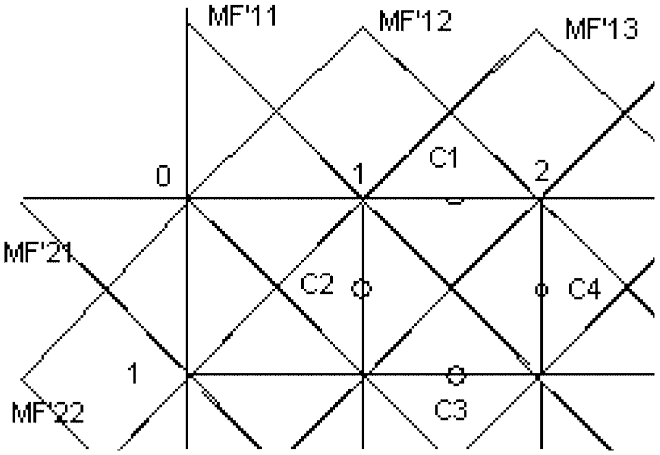

Fig. 7. Identification of the active rules in sensible areas of type 4

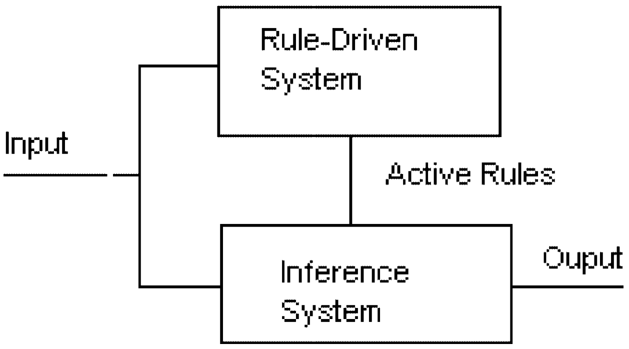

Fig. 8. Architecture of the fuzzy system

diamonds for each sensible area of type 4, the algorithm that implements the detection of the set of active rules is presented in Fig. 10.

Although the model has been presented for a two-dimensional system, it is easily extended to n-dimensional systems. In that case the model will produce n equalization functions $N_i(I_i)$ and 2^n different types of sensible areas.

4 Rule-driven architecture vs. non rule-driven architecture

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

4.1 Memory comparison

A non rule-driven solution will need the rule memory. A rule-driven solution, apart from the rule memory, will need the information on the sensible areas and the equalization functions. The amount of bytes needed to store the information of the sensible areas, M_1, is given by (22), with X the number of membership functions of I_1, Y of I_2, ND the number of diamonds involved (four in a two-dimension system), B the number of bytes per pointer and C the number of bytes per real.

$$M = (XY + 2X(Y-1) + 2(X-1)Y + 2(X-1)(Y-1)\mathrm{ND})B + 2XY \cdot \mathrm{ND} \cdot C \qquad (22)$$

The amount of memory M_2 needed to store the equalization functions $N_i(I_i)$, given n the dimension of the system and p the number of bits of the digital conversion, is:

$$M_2 = n2^p(p/8) \qquad (23)$$

The total amount of extra memory M_t needed by the rule-driven system for a Lukasiewicz T-norm is:

$$M_t = M_1 + M_2 \qquad (24)$$

4.2 Execution-time comparative

The execution time of a non rule-driven two-input system with 4 membership functions per input (as in Fig. 11) is $16K$, being K the execution time of a rule.

In the case of the rule-driven architecture proposed, given an equal probability of the input of the system to have any value of the input space, and a typical set of membership functions, as given in Fig. 11, the probability of the input I to be included in each type of sensible area, P_1 for the sensible areas of type 1 (25), P_2 for type 2 (26), P_3 for type 3 (27) and P_4 for type 4 (28), is:

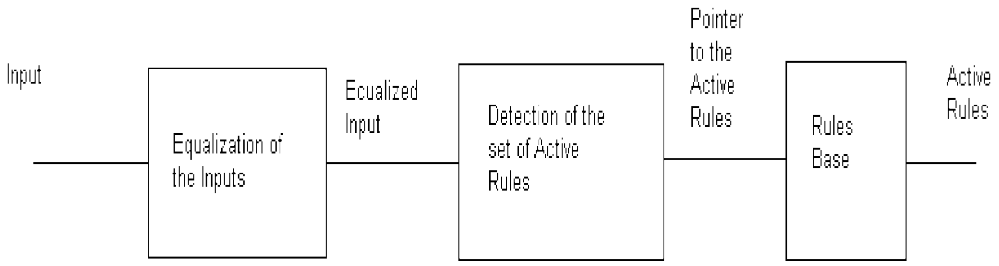
Input
Equalization of the Inputs
Ecualized Input
Detection of the set of Active Rules
Pointer to the Active Rules
Rules Base
Active Rules

Fig. 9. Architecture of the rule-driven system

Input:
I'
=(I'$_1$,I'$_2$) ; Equalization of I=(I$_1$,I$_2$)
Rules_1[X][Y] ; Pointer to the rule of each Sensible Area of type 1.
Rules_2[X][Y-1][2] ; Pointers to the rules of each Sensible Area of type 2.
Rules_3[X-1][Y-1][2] ; Pointers to the rules of each Sensible Area of type 3.
Rules_4[X-1][Y-1][ND][2] ; Pointers to the rules of each Sensible Area of type 4.
Diamond[X][Y][ND][2]; Contains the center of each diamond of the system.

Output:
R$_a$; Set of Active Rules

Algorithm for the Detection of Active Rules:

```
If I'1 is Integer
        If I'2 is Integer
                Ra=Rules_1[I'1][I'2]
        Else
                Ra=Rules_2[I'1][integer(I'2)]
Else
        If I'2 is Integer
                Ra=Rules_3[integer(I'1)][I'2]
        Else
                Ra=OBTAIN_ACTIVE_DIAMOND(I'1,I'2)
End

function OBTAIN_ACTIVE_DIAMOND (I'1,I'2)

    for ( CONT = 1; CONT <= ND ; CONT++)
        if (IS_IN_DIAMOND (DIAMOND[integer(I'1), integer(I'2),CONT],I'1,I'2))
        return RULE_4(integer(I'1),integer(I'2),CONT)
end_function

function IS_IN_DIAMOND( C1,C2, I'1,I'2)
C1,C2: CENTER_OF_DIAMOND
  {     return (abs(I'1-C1)+abs(I'2-C2) < 1)    }
```

Fig. 10. Algorithm for detection of active rules

$$P_1 = \frac{4a^2 + 4b^2 + 8ab}{(2a + 2b + 3c)^2} \qquad (25)$$

$$P_2 = \frac{6bc + 6ac}{(2a + 2b + 3c)^2} \qquad (26)$$

$$P_3 = P_2 \qquad (27)$$

$$P_4 = \frac{9c^2}{(2a + 2b + 3c)^2} \qquad (28)$$

With this probabilities, the execution time is given in (29), where R is the execution time needed to obtain in which diamond is I'.

$$T = (P_1 + 2P_2 + 2P_3)K + P_4(2K + R) \qquad (29)$$

R can be calculated, given an equal probability for the diamonds of the equalized input space, as given in (30), where D is the execution time of the expression (21).

$$R = 0.25D + 0.5D + 0.75D + 0.75D = 2.25D \qquad (30)$$

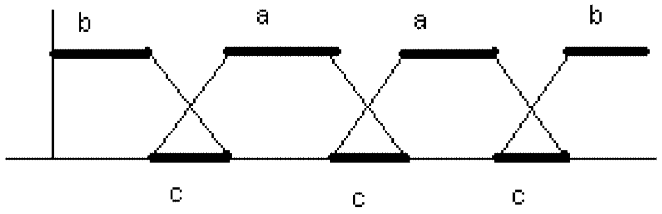

Fig. 11. Typical structure of a set of membership function

The time percentage saved, TS, for a two input system with four membership functions per input is given by (31).

$$TS = 100 - 6.25(P_1 + 2P_2 + 2P_3 + 2P_4) - 14.0625\frac{D}{K}P_4 \qquad (31)$$

For example, a system with $a = 2,\ b = 1.5,\ c = 1$, the amount of time saved by the rule-driven system will be 90%. With these benefits, the increase of memory needed by a rule-driven model, typically 5k in a two-dimensional system, is plenty justified.

5 Conclusions

This paper presents a rule-driven architecture for Lukasiewicz systems working with partition of unity membership functions with an overlap factor of two, which goes 90% faster than a non rule-driven architecture. The architecture can be implemented both in hardware and in software. This result is a first step towards the design of a high speed Lukasiewicz fuzzy processor.

The best part of ideas exposed in this paper can be also applied to the design of a rule-driven architecture for a generic T-norm, and for a n-dimensional system.

References

1. **Ascia G, Catania V, Vita L** (1996) Rule-driven VLSI fuzzy processor, IEEE Micro **16**: 62–74
2. **Chiueh T** (1992) Optimization of fuzzy logic inference architecture, IEEE Computer **25**(5): 67–71
3. **Falchieri D, Gabrielli A, Gandolfi E, Masseti M** (1997) Very Fast VLSI Fuzzy Processor: 2 Inputs 1 Output. Proceedings of the 6th International Conference on Fuzzy Systems, Barcelona, pp. 625–629
4. **Gabrielli A, Gandolfi E, Masseti M, Spotti M** (1995) Architecture of a 50 MFIPS Fuzzy Processor and the related 1.0 mm VLSI CMOS circuits. Proceedings of the Fourth International Conference on Microelectronics for Neural Network and Fuzzy System, Turin, Italy, pp. 125–133
5. **Hung T** (1996) VLSI Design of a Fuzzy Logic Controller, PhD Thesis, Texas A & M University, College Station, TX, USA
6. **Ikeda H** (1992) A fuzzy inference coprocessor using a flexible active-rule driven architecture. Proceedings of the 1st IEEE International Conference on Fuzzy Systems, San Diego, CA, pp. 135–142
7. **Klir G, Folger T** (1995) Fuzzy Sets and Fuzzy Logic, Theory and Applications, Prentice Hall, UK
8. **de Salvador L, Bernad P** (1998) Experimental Luckasiewicz Inference System (in Spanish). Proceedings of the VIII Spanish Congress of Fuzzy Logic & Technology, ESTYLF 99, Pamplona, pp. 327–332
9. **de Salvador L, Bernad P, Madroño D** (1999) High Performance Fuzzy ASIC. Proceedings of the 1999 EEUSFLAT-ESTYLF Joint Conference, Palma, Spain, pp. 307–310