Fuzzy logic controllers

Digital fuzzy logic controllers

Doru Todinca

Department of Computers and Information Technology
UPT

Outline

Soures

This lecture contains text, figures, formulae, etc, taken (and adapted) from the final year project of Ana-Maria Badulescu [Bad99], which have been taken from [Pat96] and from [Chi92] (for the simplified FLC)

In this lecture the fuzzy sets will be denoted only with capital letters, without using the tilde symbol (˜) i.e. A, A', B, B', etc, instead of $\tilde{A}, \tilde{A}', \tilde{B}, \tilde{B}'$

Outline

Outline

The general scheme of the fuzzy control

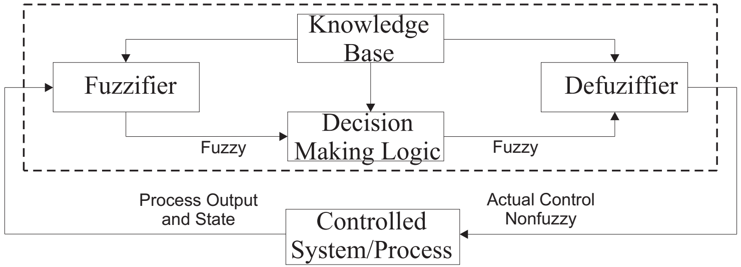


Figure 1: The general scheme of the fuzzy control

The general scheme of the fuzzy control

- The general scheme of the fuzzy control (fig 1) contains the FLC and the controlled system/process
- The rectangle with dashed line represents the fuzzy logic controller (FLC = Fuzzy Logic Controller)
- The outputs, and possible the states, of the controlled system, are read by the controller (by the FLC in this case) and used by the controller in order to determine the *actual control* values
- Any control system works in this way, the only difference being that the FLC is replaced by another type of controller (e.g. PI, PID, etc)
- The outputs, inputs and states of the controlled system are crisp values, while inside the FLC we use fuzzy sets
- $\Rightarrow$ the need to transform crisp values into fuzzy sets (operation called *fuzzification*), or to transform fuzzy sets into crisp values (*defuzzification*)

The general scheme of the fuzzy control

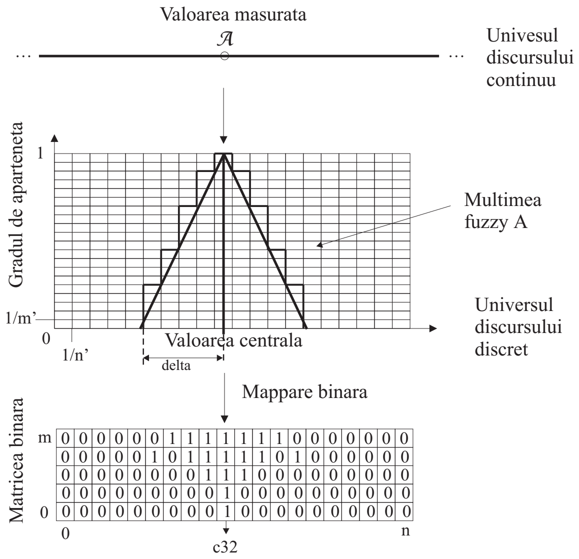
- Hence, the interface of the FLC with external word is realized by the *Fuzzifier* (for inputs), and by the *Defuzzifier* (for outputs)
- *Knowlege Base* is the rule base containing the fuzzy rules used to control a certain controlled process (hence, the knowledge base depends on the application, i.e., on the controlled process)
- *Decision Making Logic* implements the fuzzy inference and is application independent
- The FLC can have other parameters like: maximum number of inputs and outputs, maximum number of rules that can be stored in the knowledge base, input-output response time, etc (will be detailed later)
- In the following slides we will discuss the Fuzzifier, the Defuzzifier, and the fuzzy inference (decision making logic)

Outline

Digital fuzzification: purpose

- The transformation of a crisp value of the input fact into a fuzzy set (i.e., the fuzzification process) compensates for the loss of accuracy of the measurement process (i.e, using sensors) and of the digitization processes (A-D conversion). If the internal domain of representation of fuzzy sets inside the FLC is different from the domain of representation of the digitized value, then a mapping process has to take place.
- If fuzzification is applied, an input fact will trigger more fuzzy rules than in the case when the input fact remains crisp (singleton), which means that the process of going from one fuzzy rule to the other to be smoother (because the two rules will be active in the same time, in different degree). Hence, the controll process will be smooth.
- The drawback is that, if fuzzification is applied, the inference process will need more computations than in the case of singleton inputs.

The fuzzification process



m	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	1	1	1	1	1	0	1	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Figure 2: The fuzzification process

The fuzzification process

- The measured value of an output (in this case $\mathcal{A}$, see fig 2) or of a state of the controlled process is analogue, having a continuous universe of discourse
- This analogue value is discretized (by A-D conversion followed by normalization), then it is fuzzified into a discrete universe of discourse
- After fuzzification we obtain a fuzzy set A, which can be either a single value (a singleton – no fuzzification performed in this case), or a vector of arbitrary shape (triangle, Gaussian, etc)
- Usually we represent a fuzzified value (i.e., the fuzzy set A) as an isosceles triangle with the base having the length of $2 \cdot \Delta$ (if $\Delta = 0$, then the triangle becomes a singleton)
- The universe of discourse is discretized into n' equal intervals, while the membership function is discretized into m' equal intervals.

The fuzzification process

- The internal representation, inside the FLC, of the fuzzy set A is made on a universe of discourse with n elements and a membership function represented on m bits, i.e., with integer values in the interval $[0, 2^m - 1]$
- Which means that the internal representation of the fuzzy set A is a Boolean matrix that has the size $m * n$
- We prefer to represent the membership functions as integer numbers on m bits, instead of real numbers in the interval $[0, 1]$
- If $n \neq n'$, a nonlinear mapping is needed between the two universes of discourse
- The universe of discourse of the real discretized input values can have negative values, but for the internal representation of fuzzy sets inside the FLC, we prefer to use positive numbers between 0 and $n - 1$ (a linear mapping has to be performed)
- In general we prefer that $m' = 2^m$

Outline

Digital fuzzy inference: fuzzy rules

- We describe the set of N fuzzy rules from the rule base of the FLC
- Each rule can have in general K inputs and L outputs, but in most cases we prefer FLCs with a single output
- $A1, A2, \ldots AK$ denote the input K linguistic variables (e.g. speed, distance, or control error e, the derivative of the error de, the second derivative of the error $d2e$, etc), while $B1, \ldots, BL$ denote the L output linguistic variables (e.g. acceleration, the control signal u, etc)
- $A_i^1, \ldots, A_i^K$ represents the linguistic terms of the input linguistic variables $1, \ldots K$ of the rule i (e.g. for the variable speed, the terms can be very small, small, etc; for the linguistic variable error, the terms can be negative large, approx zero, positive large, etc)
- Similar, $B_i^1, \ldots, B_i^L$ represent the terms of the output linguistic variables $1, \ldots, L$ of rule i
- O graphic representation of fuzzy inference for the case when there are 2 inputs and one output, and the input fact is crisp

Digital fuzzy inference: fuzzy rules

R_1: IF $A1$ IS A_1^1 AND $A2$ IS A_1^2 AND ... AND AK IS A_1^K THEN
$B1$ IS B_1^1 IS $B2$ IS B_1^2 AND ... AND BL IS B_1^L,

ALSO
R_2: IF $A2$ IS A_2^1 AND $A2$ IS A_2^2 AND ... AND AK IS A_2^K THEN
$B1$ IS B_2^1 AND $B2$ IS B_2^2 AND ... AND BL IS B_2^L,

ALSO
$\vdots$
R_i: IF $A1$ IS A_i^1 AND $A2$ IS A_i^2 AND ... AND AK IS A_i^K THEN
$B1$ IS B_i^1 AND $B2$ IS B_i^2 AND ... AND BL IS B_i^L,

ALSO
$\vdots$
R_N: IF $A1$ IS A_N^1 AND $A2$ IS A_N^2 AND ... AND AK IS A_N^K THEN
$B1$ IS B_N^1 AND $B2$ IS B_N^2 AND ... AND BL IS B_N^L.

Example of fuzzy inference when the input fact is not fuzzified

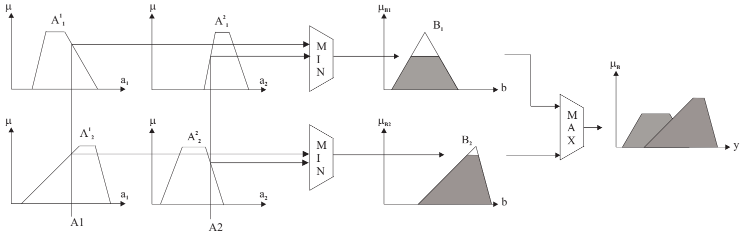


Figure 3: Example of fuzzy inference with 2 inputs and one output when the input fact is not fuzzified (i.e, it is crisp)

Example of fuzzy inference when the input fact is fuzzified

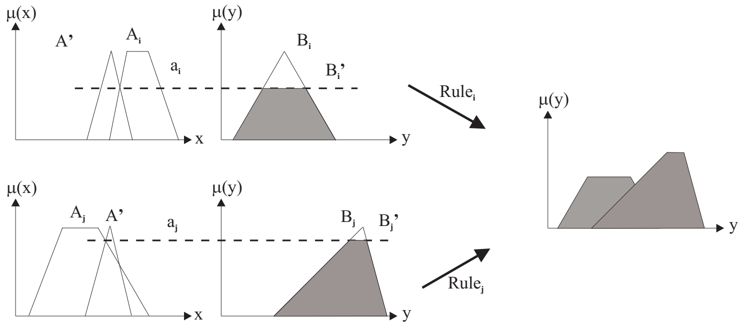


Figure 4: Example of fuzzy inference with one input and one output when the input fact is fuzzified

Outline

Digital defuzzification

- The output of the fuzzy inference process is a fuzzy set, usually non-normal, obtained by the union of the outputs of the active fuzzy rules (e.g. fig 3, 4)
- In some cases it is possible to use directly this fuzzy set: e.g. in expert systems we can try to “identify” this set, i.e., to give it an interpretation as a fuzzy set
- In control engineering problems (and in most cases, in general) we want to apply a crisp control signal to the controlled process, not a fuzzy set
- In order to obtain this crisp value from the output fuzzy set, the defuzzification operation is needed.
- By defuzzification we understand the transformation of a fuzzy set into a crisp value

Digital defuzzification

- The most used defuzzification method is Centre OF Area (COA), named also Centre of Gravity (COG), illustrated in figure 5
- In fig 5, on the abscissa it is represented the universe of discourse, and on the ordinate, the degree of membership
- The output crisp value will be the abscissa of the center of gravity of the figure (i.e. fuzzy set) resulted after the fuzzy inference
- Formula for COG is: $\frac{\int \mu(y) \cdot y dy}{\int \mu(y) dy}$ or, for the discrete case, $\frac{\sum_{j=1}^{n} \mu(y_j) \cdot y_j}{\sum_{j=1}^{n} \mu(y_j)}$
- The defuzzification usually involves a division operation, hence it is very slow and resource consuming in case of hardware implementation of the FLC
- There exist other defuzzification methods: Smallest of Maxima (SOM), Mean of Maxima (MOM), etc

Digital defuzzification

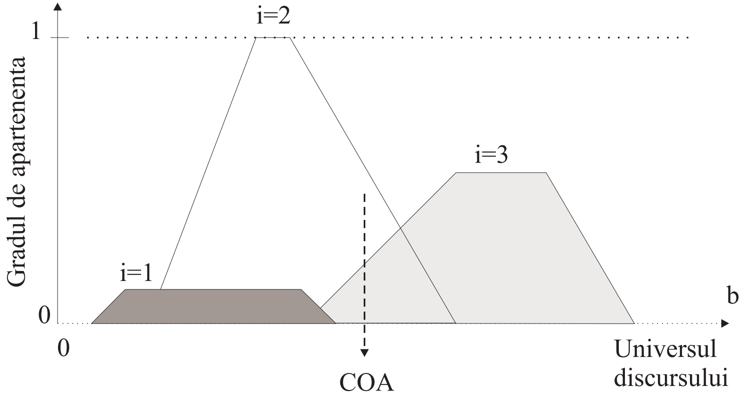


Figure 5: Digital defuzzification

Outline

Outline

Digital FLC characteristics

- Performance measurements provided by the FLC manufacturers:
    1. number of fuzzy inferences per second, where by fuzzy inference is understand sometimes an operation defined by a single rule, or, an operation defined by a part of a rule
    2. number of elementary fuzzy operations (MIN or MAX) per second
- These parameters cannot provide a reliable measurement of the real speed of the FLC.
- Therefore, Patyra (in [Pat96]) suggested to characterize the FLC by the input-to-output delay, denoted θ_{IN-OUT}:
  This is defined as the total delay time from the moment when data (the fact) are provided at the inputs of the FLC, till the moment when the crisp action is generated at the output of the FLC.

Digital FLC characteristics

The most important parameters of a FLC are:

- number of input variables (i.e., of inputs)(K)
- number of output variables (L) (outputs)
- number of linguistic rules in the knowledge base (N)
- number of membership functions in the input universe of discourse (i.e., the number of terms of an input linguistic variable) (MB_{IN})
- number of membership functions in the output universe of discourse (i.e., the number of terms of an output linguistic variable) (MB_{OUT})
- number of binary vectors that characterize the membership functions (n)
- number of bits in a single binary vector (m)(the number of bits used to represent the membership functions)
- input-to-output delay time (θ_{IN-OUT})

Outline

Fuzzy Logic Controllers

Fuzzy inference means the obtaining of a new fact, B', from a fact A' and a rule $A \to B$, according to the formula:

$$B' = A' \bullet R_{A \to B} \tag{1}$$

When the circuit has two inputs and one output, the relation (1) becomes:

$$\begin{array}{l} R_1\text{: IF } A_1^1 \text{ AND IF } A_1^2 \text{ THEN } B_1\text{, ALSO} \\ R_2\text{: IF } A_2^1 \text{ AND IF } A_2^2 \text{ THEN } B_2\text{, ALSO} \\ \vdots \\ R_i\text{: IF } A_i^1 \text{ AND IF } A_i^2 \text{ THEN } B_i\text{, ALSO} \\ \vdots \\ R_N\text{: IF } A_N^1 \text{ AND IF } A_N^2 \text{ THEN } B_N. \end{array} \tag{2}$$

where R_1 .. R_N are fuzzy rules, A^1 si A^2 are input variables, and B is the output.

Fuzzy Logic Controllers

In these rules, by premise (or antecedent) we refer to the input variables, while the conclusion concerns the output fuzzy sets.
In the case of DISO (Double Input, Single Output) FLC, the premise is composed, the two parts of the premise (the sub-premises) being connected by the logic AND operator.
Hence, according to the definition of fuzzy implication, we obtain:

$$B' = (A^1, A^2) \bullet R \tag{3}$$

where $R = (A^1, A^2) \rightarrow B$.
Rules R_i are connected with the ALSO operator, which is interpreted as logic OR. Hence, the final result is obtained by the union of the rules R_i:

$$R = \bigcup_{i=1}^{N} R_i \tag{4}$$

Fuzzy Logic Controllers

The fact A' is $A' = (A1, A2)$, so the conclusion $B1$ is obtained applying the compositional rule of inference:

$$B1 = (A1, A2) \bullet R = (A1, A2) \bullet \bigcup_{i=1}^{N} R_i = \bigcup_{i=1}^{N} (A1, A2) \bullet R_i \tag{5}$$

The membership function of the set $B1$ can be computed with the MAX-MIN operator. For the rule R_i we obtain:

$$B1_i = (A1, A2) \bullet R_i \tag{6}$$

Fuzzy Logic Controllers

The corresponding membership function is defined as follows:

$$\begin{aligned}\mu_{B1_i}(b) = \; & MAXMIN(\mu_{A1}(a_1) \times \mu_{A2}(a_2), \mu_R(a_1, a_2, b)) = \\ & a_1 \in A1, a_2 \in A2\end{aligned}$$

$$MINMAX\{MIN[MIN(\mu_{A1}(a_1), \mu_{A_1^1}(a_1)), MIN(\mu_{A2}(a_2), \mu_{A_i^2}(a_2))], \tag{7}$$

$\mu_{B_i}(b)\}$, $a_1 \in A1, a_2 \in A2$

$$= MIN(\Omega_i, \mu_{B_i}(b)) b \in B$$

where, in this case, Ω_i is defined: $\Omega_i =$

$$MIN\{MAXMIN(\mu_{A1}(a1), \mu_{A_1^1}(a_1)), MAXMIN(\mu_{A2}(a_2), \mu_{A_1^2}(a_2))\} \tag{8}$$

$a_1 \in A1, a_2 \in A2$

Fuzzy Logic Controllers

So, the maximum between $B1_1$, $B1_2$, ..., $B1_N$ determines the final action $B1$, computed as the union:

$$B1 = \bigcup_{i=1}^{N} B1_i = MAX(B1_1, B1_2, \ldots, B1_N) \tag{9}$$

Fuzzy Logic Controllers: classic implementation

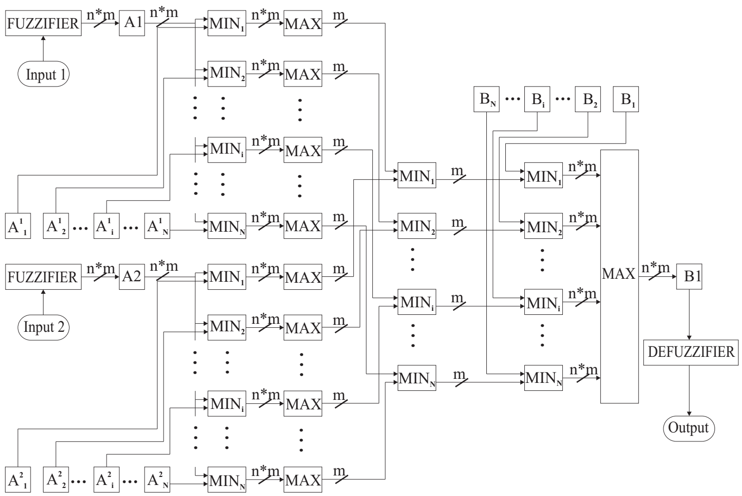


Figure 6: FLC DISO, classic implementation

Classic DISO FLC: the premises

- DISO - Double Input, Single Output
- The diagram from fig 6 is a block diagram, not a hardware diagram, because some blocks need a more complex hardware implementation: e.g., the first level of MIN blocks perform the minimum between n pairs of operands (m bits membership functions)
- Each of the two input facts (Input 1 and Input 2) are transformed into fuzzy sets by the two fuzzifiers
- These fuzzy sets are stored in the memories A1, respectively A2, both being $n * m$ bits vectors
- The linguistic terms from premises are stored into the N (where N is the number of fuzzy rules) memories (of $n * m$ bits) denoted $A_1^1, \ldots, A_N^1$, for the first input, and respectively $A_1^2, \ldots A_N^2$ for the second input
- Actually we need only MB_{IN} memories for each input, but in this case we have to establish how to associate the input terms with the rules

Classic DISO FLC: the premises

- First level of MIN circuits performs the minimum between the membership function of a term and the membership function of the fuzzified input fact; the inputs and the output are $n * m$ bits vectors
- According to the formula, these circuits compute $MIN(\mu_{A1}(a_1), \mu_{A_1^1}(a_1))$
- First level of MAX circuits compute Ω_i for each sub-premise of each rule
- Actually they determine the maximum of an array with n elements;
- The input is an $n * m$ bits vector, the output is a scalar (on m bits)

Classic DISO FLC: the premises

- For a hardware implementation we need a register at the output of each MAX circuit, where to store the partial result, and we compute the maximum between the current input of the MAX circuit and the partial result from the register
- This operation needs to be performed n times (i.e., for each element of the array)
- Second level of MIN circuits implements the AND operation between the two subpremises: the inputs and the output are scalars (on m bits)

Classic DISO FLC: the conclusions

- The third level of MIN circuits from the FLC belongs to the part of the FLC used for processing the conclusions
- These circuits perform the minimum between the membership function of the term from the conclusion of the rule (B_i for rule i), which is an $n * m$ bits vector, and the degree of activation Ω_i of rule i (which is an m bits scalar)
- The outputs of the circuits are $n * m$ bits vectors, namely B_i', the "truncated" fuzzy sets
- The same like for premises, instead of N memories of $n * m$ bits, one memory for each rule, we can use only MB_{OUT} memories, one for each linguistic term from conclusion
- In this case we have to perform the binding between the index i of rule i and the corresponding linguistic term

Classic DISO FLC: the conclusions

- Last level of MAX circuits implements the union operation $B1 = \bigcup_{i=1}^{N} B1_i = MAX(B1_1, B1_2, \ldots, B1_N)$ corresponding to the operator ALSO
- The inputs and the output are $n * m$ bits vectors
- There are several possible hardware implementations of this circuit, depending on the degree of parallelism involved
- The output of this MAX circuit is stored into the memory $B1$, of $n * m$ bits
- From there, it will be defuzzified and sent to the output of the FLC

Outline

Simplified FLC

- The inputs of this FLC are not fuzzified, they remain singletons.
- This will simplify the computations from premises,
- in the sense that the value Ω_i is obtained by the intersection between the fuzzy set A_i^1 and the fact $A1$ represented by a singleton.
- If the premises is composed, an AND is performed between these values.
- Conclusion of the rule is obtained as a minimum between Ω_i and the membership function of the set B_i (fig 3)

Conditions for simplifying the architecture of the FLC

- In order to obtain the intersection between the input fact and the fuzzy set representing the term of an input linguistic variable, we will use the value of the fact, normalized to the universe of discourse, as an address in the memory that contains the linguistic variable corresponding to that input.
- The terms of a linguistic variable, represented as fuzzy sets, intersect each other, hence an input fact can intersect more than one linguistic term (and consequently, it can activate more than one rule)
- In order to increase the speed of the circuit, we will process the rules in parallel, and we have to determine the parameter *MNSAR*: maximum number of simultaneous active rules [Chi92]

Conditions for simplifying the architecture of the FLC

- We can see that $MNSAR = MOF^K$, where MOF (maximum overlap factor) is the maximum number of linguistic terms that can overlap
- $MOF \geq 2$ and it is convenient to chose $MOF = 2$
- This choice imposes a limitation to the user regarding the choice of the terms of the input linguistic variables, but this limitation is not annoying (it is a natural that no more than two terms of a linguistic variable overlap)
- The terms will be separated into two disjunctive sets, odd terms and even terms, such that each of the two sets can be stored into a memory of n m bits locations (see fig 7)

Separation of the terms of a linguistic variable in even and odd terms

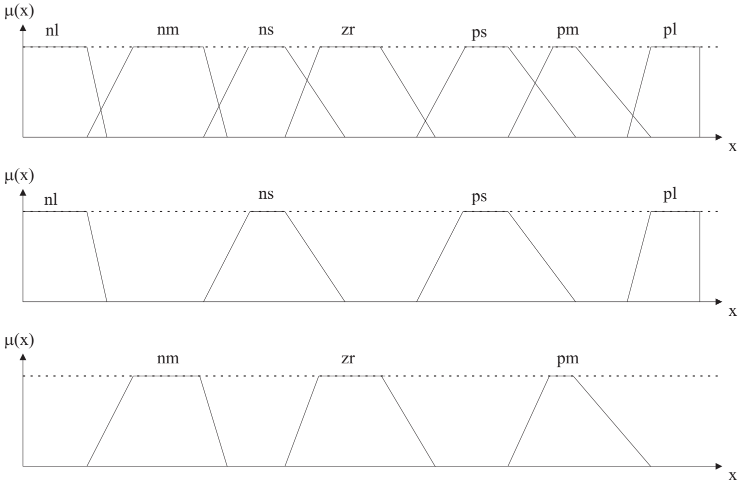


Figure 7: Separation of the terms of a linguistic variable in even and odd terms

Hardware diagram of the simplified FLC with 3 inputs

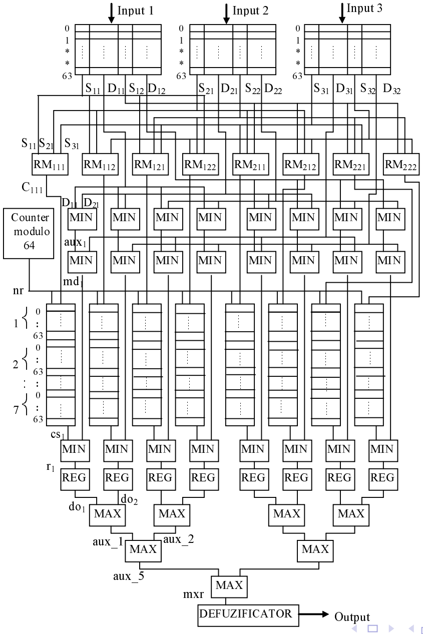
- In this case (i.e., $MOF = 2$) we will represent the MB_{IN} terms of each input linguistic variable using two arrays of n locations, each location containing the value of the membership function (the degree, on m bits) and the code of that linguistic term (the symbol).
- One array contain the odd terms, the other array contains the even terms
- Alongside the MB_{IN} terms, we need to add a symbol for 'no term', for the case when the input fact does not intersect any term in one of the two memories.

Hardware diagram of the simplified FLC with 3 inputs

- The circuit from figure 8 is a schema hardware diagram, not only a block diagram, like in figure 6 (classic DISO FLC)
- This circuit is MISO, with $K = 3$, hence we obtain $MNSAR = 8$,
- A triplet of inputs (representing the input fact) will determine the memories containing the input variable to show at their outputs three pairs (symbol, degree) $= (S_{ij}, D_{ij})$, $i, j \in \{1, 2, 3\}$.
- The symbols will be used as addresses for the rule memory (or memories) $RM_{111}, RM_{112}, \ldots, RM_{222}$; the content retrieved from the memory will be the term of the linguistic variable from the conclusion of the rule that has in premises these symbols.
- The symbol obtained from the memory rule will be used as an address in the memory that stores the output linguistic variable. The symbol is coded on $n * m$ bits.

Hardware diagram of the simplified FLC with 3 inputs

- The degress obtained from the input memories are processed in parallel with the processing of symbols:
- A MIN operation is performed between the three degrees
- This MIN corresponds to the logic AND between the sub-premises of a rule.
- The result of the MIN is the degree in which the premise of a certain rule is true (Ω in formulas 8).
- Performing MIN between this Ω and the membership function of the term of the linguistic output variable, we obtain the set B' corresponding to the activated rule.
- A union is performed between the consequences of the activated rules by the circuit MAX with 8 inputs, whose output enters into the defuzzifier.
- Registers are optional and are used in order to make a pipeline between the inference and defuzzification operations.



Input 1
Input 2
Input 3
0
1
*
*
63
S11 D11 S12 D12
S21 D21 S22 D22
S31 D31 S32 D32
S11 S21 S31
RM111
RM112
RM121
RM122
RM211
RM212
RM221
RM222
C111
D11 D21
Counter modulo 64
MIN
aux1
md1
nr
1
2
7
cs1
r1
REG
do1
do2
MAX
aux_1
aux_2
aux_5
mxr
DEFUZIFICATOR
Output

Outline

“Improved” FLC

It is based on the commutativity between MAX-MIN and UNION.
First, we build the individual fuzzy relations.
Fuzzy relation R_i (fuzzy implication for the first linguistic variable):

$$R_i^1 = A_i^1 \times B_i \tag{10}$$

For the second variable:

$$R_i^2 = A_i^2 \times B_i \tag{11}$$

R^1 and R^2 represent fuzzy partial relations, or sub-relations
The overall relation $R = R^1 \cup R^2$.

“Improved” FLC

For the first input variable $A1$ and the output B we obtain:

$$R^1 = R_1^1 \cup R_2^1 \cup, \dots, \cup R_N^1 \tag{12}$$

For the combination between the second input variable $A2$ and the output B:

$$R^2 = R_1^2 \cup R_2^2 \cup, \dots, \cup R_N^2 \tag{13}$$

Replacing the UNION operator with MAX, we obtain:

$$R^1(a_1, b) = MAX\{R_1^1(a_1, b), R_2^1(a1, b), \dots, R_N^1(a_1, b)\} \tag{14}$$

$$R^2(a_1, b) = MAX\{R_1^2(a_1, b), R_2^2(a1, b), \dots, R_N^2(a_1, b)\} \tag{15}$$

“Improved” FLC

Partial fuzzy relations are used in order to obtain the result of the fuzzy inference.
Output $B1$ related to the inputs $A1$ and $A2$, can be obtained using the MIN-superposition operator of the two partial relation:

$$B1 = MIN\{[A1 \bullet R^1(a_1, b)], [A2 \bullet R^2(a_2, b)]\} \qquad (16)$$
$$b \in B$$

$$B1 = MIN\{MAXMIN[A1, R1(a_1, b)], MAXMIN[A2, R^2(a_2, b)]\} \qquad (17)$$
$$b \in B$$

where $R^1(a_1, b)$ and $R^2(a_2, b)$ are given by the equations (14), respectively (15).

"Improved" FLC

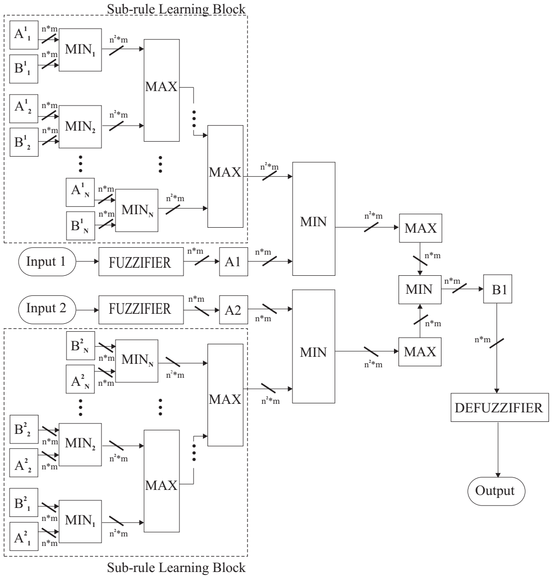


Figure 9: Improved FLC

“Improved” FLC: advantages and drawbacks

- The part represented in fig 9 inside the rectangles can be processed off-line (before facts appear at the inputs)
- This will decrease the input-to-output delay time (θ_{IN-OUT}) of the improved FLC, compared to classic FLC
- But, the main problem is that the fuzzy rules with mre than one sub-premise are not given usually in the form “IF premise1 THEN conclusion” and “IF premise2 THEN conclusion”, but “IF premise1 AND premise2 THEN conclusion”
- It means that it is difficult to formulate the rules with the input variables completely separated
- The attempts to start from the rules with several premises and to separate the premises can give bad results !!
- In the simulations that I performed with this type of FLC, it succeeded to control the controlled process only for rules with only one input !!
- There are however situations, but not very often, when it is possible to formulate the rules with the input variables separated

Ana-Maria Badulescu.
Studiul performantelor circuitelor fuzzy utilizand limbajul VHDL, (Performance study of fuzzy circuits using VHDL), Diploma project, supervisor Doru Todinca, University Politehnica Timisoara, Dep. of Computers, Facultaty of Automation and Computers, 1999.

Tzi-cker Chiueh.
Optimization of fuzzy logic inference architecture.
Computer, 25(5):67–71, May 1992.

Marek J. Patyra.
Design considerations of fuzzy logic controllers.
In Marek J. Patyra and D.M. Mlynek, editors, *Fuzzy Logic: Implementations and Applications*, pages 143–175. John Wiley & Sons Ltd. and B.G. Teubner, 1996.