Complexity Management Methodology for Fuzzy Systems with Feedforward Rule Bases

Alexander Gegov*, David Sanders^, Boriana Vatchova+

*University of Portsmouth, School of Computing

Buckingham Building, Portsmouth PO1 3HE, United Kingdom

^University of Portsmouth, School of Engineering

Anglesea Building, Portsmouth PO1 3DJ, United Kingdom

+Bulgarian Academy of Sciences, Institute of Information and Communication Technologies

Block 2, 1113 Sofia, Bulgaria

Abstract: This paper proposes a complexity management methodology for fuzzy systems with feedforward rule bases. The methodology is based on formal methods for presentation, manipulation and transformation of fuzzy rule bases. First, Boolean matrices are used for formal presentation of rule bases. Then, binary merging operations are used for formal manipulation of rule bases. Finally, repetitive merging operations are used for formal transformation of rule bases. The formal methods facilitate the understanding and modelling of fuzzy systems in terms of interacting subsystems. In particular, the methods reduce the qualitative complexity in fuzzy systems by improving the transparency of the rule bases.

Keywords: fuzzy systems, complexity management, formal methods, rule bases, fuzzy networks, feedforward connections.

1. Introduction

Fuzzy systems are usually good at capturing the qualitative complexity of a wide range of problems by means of their linguistic modeling and approximate reasoning capabilities. However, this comes at a price because the associated operations during fuzzification, inference and defuzzification increase the quantitative complexity of the solution to these

problems. This price gets even higher as the amount of fuzzy operations increases as a result of the increased number of rules in the fuzzy system.

The number of rules in a fuzzy system is often an exponential function of the number of inputs to the system and the number of linguistic values that these inputs can take [1-4]. This exponential function has been used as a main indicator for the quantitative complexity of the associated fuzzy system.

There has been a growing interest recently in complexity issues of fuzzy systems [5-8]. This is due to the fact that fuzzy systems are already more widely used in large-scale applications where their quantitative complexity becomes more obvious. In particular, many methods have been developed for reducing this quantitative complexity. These are known as complexity reduction methods as they reduce the number of rules by reducing the number of inputs or the number of linguistic values that these inputs can take. The main objective in this case is to suppress the associated exponential function. These methods are classified into six groups and discussed below.

The first group of methods are aimed at removing less significant or merging similar linguistic values [9-11]. From these two strands, the one based on removal of linguistic values is more straightforward but it involves a higher risk as a result of the removal of the associated fuzzy set. On the other hand, the strand based on merging of linguistic values is more difficult for application due to the necessity to define a new fuzzy set for each of the merged linguistic values.

The second group of methods are aimed at removing less significant or merging similar inputs [12-14]. From these two strands, the one based on removal of inputs is more straightforward but it involves a higher risk as a result of the removal of the associated physical variable. On the other hand, the strand based on merging of inputs is more difficult

for application due to the necessity to justify physically the merging of the associated variables.

The third group of methods are based on singular value decomposition of the matrix representing the crisp values of the output from a fuzzy system [15-17]. As a result of this decomposition, the number of linguistic values for the inputs to the system is reduced. Although this group of methods can be quite effective in reducing the number of rules in a fuzzy system, they are applicable mainly for systems with two inputs. In the case of more inputs, the singular value decomposition process becomes quite complex as the dimension of the space in which the associated matrix is defined increases significantly.

The fourth group of methods are based on conversion of the intersection rule configuration of a fuzzy system into a union rule configuration with a smaller number of rules [18-20]. This group of methods can be quite effective in reducing the number of rules in a fuzzy system but they can only be applied to a special class of problems called ‘additively separable’. For problems that do not belong to this class, the conversion of the intersection rule configuration into a union rule configuration is not possible.

The fifth group of methods convert a fuzzy system into spatially decomposed subsystems as a result of which the overall number of rules is reduced [21-27]. In this case, the interactions among the subsystems are partially compensated and the resulting decomposed system has a decoupled structure. Although this group of methods have been widely used recently, the success of their application depends on the strength of interactions among the subsystems and the level of their compensation.

The sixth group of methods rearrange the inputs in a fuzzy system in a way that leads to the reduction of the number of rules [28-34]. In this case, the fuzzy system is decomposed into a multilayer hierarchical structure such that each layer has only two inputs and one output. Although these methods have become quite popular recently, they don’t offer clear

interpretation of the intermediate variables between the first and the last layer. Besides this, only two inputs are taken into account in each layer while all other inputs are ignored.

The above complexity reduction methodology for fuzzy systems has some drawbacks such as empirical nature and limited scope. The empirical nature of the methods in groups 1-2 and 5-6 assumes the use of a ‘trial and error’ approach that can be unreliable. Besides this, the limited scope of the methods in groups 3-4 makes them inapplicable some fuzzy systems. And finally, all methods in groups 1-6 do not reduce the qualitative complexity in terms of the opaqueness of rules.

Table 1 below shows a detailed comparison of complexity reduction methods for fuzzy rule based systems. This comparison summarises the attributes of the six groups of methods considered above in terms of their nature, scope and capability for reducing the qualitative complexity in fuzzy systems.

Table 1: Comparison of complexity reduction methods

<i>Method / Attribute</i>	<i>Nature</i>	<i>Scope</i>	<i>Qualitative complexity</i>
<i>First group</i>	<i>empirical</i>	<i>universal</i>	<i>unaffected</i>
<i>Second group</i>	<i>empirical</i>	<i>universal</i>	<i>unaffected</i>
<i>Third group</i>	<i>systematic</i>	<i>limited</i>	<i>unaffected</i>
<i>Fourth group</i>	<i>systematic</i>	<i>limited</i>	<i>unaffected</i>
<i>Fifth group</i>	<i>empirical</i>	<i>universal</i>	<i>unaffected</i>
<i>Sixth group</i>	<i>empirical</i>	<i>universal</i>	<i>unaffected</i>

This paper addresses the above drawbacks of the current complexity reduction methodology for fuzzy systems by proposing a novel complexity management methodology. The main advantages of this novel methodology are its systematic nature, universal scope and capability for reducing the qualitative complexity in fuzzy systems.

The remaining part of this paper is structured as follows. Section 2 introduces some theoretical preliminaries for fuzzy systems. Sections 3-5 describe the complexity management methodology for fuzzy systems in terms of formal methods for presentation, manipulation and transformation of rule bases. Section 6 illustrates this methodology for a feedforward fuzzy rule based network. Section 7 summarises the main advantages of the methodology and highlights future research directions.

A fuzzy system can be represented by the following rule base

where m is the number of inputs, n is the number of outputs and r is the number of rules [35-36]. In this case, i_p , $p=1,...,m$ represents the p -th input, v_{ips} , $p=1,...,m$, $s=1,...,r$ is the linguistic value of the p -th input in the s -th rule, o_q , $q=1,...,n$ represents the q -th output and v_{oqs} , $q=1,...,n$, $s=1,...,r$ is the linguistic value of the q -th output in the s -th rule.

A fuzzy system operates in three main stages - fuzzification, inference and defuzzification [37]. The inference stage includes three substages - application, implication and aggregation. In multiple-output fuzzy systems, each output is considered separately and in relation to the same set of inputs. Therefore, the three main stages above are applied repetitively for each output.

The maximum number of rules r in a fuzzy system is an exponential function of the number of inputs m and the number of linguistic values w that each input can take. If this number is a constant, the maximum number of rules is given by

$$r = w^m \quad (2)$$

where v is the number of linguistic values per input.

However, if the number of linguistic values that each input can take is not a constant, the maximum number of rules in a fuzzy system is given by

$$r = w_1 \dots w_m \quad (3)$$

where $w_p, p=1, \dots, m$ is the number of linguistic values that the p -th input can take.

Fuzzy rule bases have some important properties [38]. These properties describe the extent to which the permutations of linguistic values of inputs and outputs are present in the rule base. The properties also describe the type of mapping in the rule base between permutations of linguistic values of inputs in the ‘if’ part and permutations of linguistic values of outputs in the ‘then’ part. Four basic properties of fuzzy rule bases are introduced below by definitions. These definitions make use of logical equivalence, i.e. a property is present when the corresponding condition holds and vice versa. This logical equivalence also implies that a property is absent when the corresponding condition does not hold and vice versa.

Definition 1: A fuzzy rule base is complete if and only if all possible permutations of linguistic values of inputs are present in the ‘if’ part of the rule base.

Definition 2: A fuzzy rule base is exhaustive if and only if all possible permutations of linguistic values of outputs are present in the ‘then’ part of the rule base.

Definition 3: A fuzzy rule base is consistent if and only if every present permutation of linguistic values of inputs is mapped to only one permutation of linguistic values of outputs.

Definition 4: A fuzzy rule base is monotonic if and only if every present permutation of linguistic values of outputs is mapped from only one permutation of linguistic values of inputs.

The aim of the proposed complexity management methodology for fuzzy systems is to provide formal methods for presentation, manipulation and transformation of rule bases. These methods facilitate the understanding and modelling of fuzzy systems in terms of interacting subsystems. In particular, the methods reduce the qualitative complexity in fuzzy systems by improving the transparency of the rule bases.

3. Formal Presentation of Fuzzy Systems

Fuzzy systems can be presented formally by Boolean matrices. The latter have been studied thoroughly by mathematicians and applied successfully by engineers in many areas. Some basic definitions for Boolean matrices are given below.

Definition 5: An $m \times n$ null Boolean matrix is a matrix with m rows and n columns all of whose elements are equal to 0.

Definition 6: An $m \times n$ universal Boolean matrix is a matrix with m rows and n columns all of whose elements are equal to 1.

Definition 7: A Boolean matrix is square if and only if the number of its rows is equal to the number of its columns.

Definition 8: A Boolean matrix is homogenous if and only if its row and column labels are of the same type.

The basic operations that can be applied to elements of Boolean matrices are ‘addition’ and ‘multiplication’. They are both binary operations as they can only be applied to two operands. In the case of more than two elements, each of the two operations can be applied in a sequential manner, i.e. only two elements are considered at each step and the result from the current step becomes an operand in the next step. The ‘addition’ operation has the effect of

taking the ‘maximum’ of the elements whereas the ‘multiplication’ operation has the effect of taking the ‘minimum’ of the elements. Both operations are commutative, i.e. the result is not affected if the positions of the two elements are swapped.

In terms of the values of the first and the second element, there are four different permutations for the ‘addition’ operation which are described by the following equations:

$$1 + 1 = \max(1, 1) = 1 \quad (4)$$

$$0 + 1 = \max(0, 1) = 1 \quad (5)$$

$$1 + 0 = \max(1, 0) = 1 \quad (6)$$

$$0 + 0 = \max(0, 0) = 0 \quad (7)$$

Similarly, there are four different permutations for the ‘multiplication’ operation which are described by the following equations:

$$1 \cdot 1 = \min(1, 1) = 1 \quad (8)$$

$$0 \cdot 1 = \min(0, 1) = 0 \quad (9)$$

$$1 \cdot 0 = \min(1, 0) = 0 \quad (10)$$

$$0 \cdot 0 = \min(0, 0) = 0 \quad (11)$$

Boolean matrices are multiplied in almost the same way as conventional matrices, i.e. matrices whose elements can take any values. Each element in a Boolean matrix product $A*B$ can be obtained by multiplying each row from the first matrix A with its counterpart column from the second matrix B . In this case, the row index of an element $A*B$ is the same as the index of the corresponding row from the matrix A whereas the column index of an element in $A*B$ is the same as the index of the corresponding column from the matrix B .

The multiplication compatibility rule for Boolean matrices is the same as the rule for conventional matrices, i.e. the number of columns in the first matrix must be equal to the number of rows in the second matrix. The only difference is that instead of applying the

arithmetic ‘addition’ and ‘multiplication’ operations on elements of the matrices, we apply the ‘maximum’ and ‘minimum’ operations, respectively.

The space complexity associated with the Boolean matrix for a rule base can be evaluated in the general case as follows. If a rule base has m inputs taking x linguistic values each and n outputs taking y linguistic values each, then the Boolean matrix for this rule base will have x^m rows and y^n columns, i.e. it will have $x^m \cdot y^n$ elements.

Table 2 below shows in a simplified context how the number of elements in the Boolean matrix for a rule base increases with the increase of the number of inputs to / outputs from this rule base and the number of linguistic values that these inputs / outputs can take.

Table 2: Space complexity of Boolean matrix for a rule base

$m, n / x, y$	$x=2, y=2$	$x=3, y=3$	$x=4, y=4$
$m=1, n=1$	4	6	16
$m=2, n=2$	16	81	256
$m=3, n=3$	64	729	4,096

The formal presentation of a fuzzy rule base by a Boolean matrix is described by Algorithm 1 and illustrated by Examples 1-3.

Algorithm 1:

1. Sort all possible permutations of linguistic values of inputs $v_{ips}, p=1,...,m, s=1,...,r$ from the rule base in an ascending order.
2. Sort all possible permutations of linguistic values of outputs $v_{oqs}, q=1,...,n, s=1,...,r$ from the rule base in an ascending order.
3. Label the rows of the Boolean matrix with the sorted permutations of linguistic values of inputs $v_{ips}, p=1,...,m, s=1,...,r$.

4. Label the columns of the Boolean matrix with the sorted permutations of linguistic values of outputs v_{oqs} , $q=1,...,n$, $s=1,...,r$.

5. Go through all the elements of the Boolean matrix and set each element equal to 1 or 0 using steps 6 and 7.

6. If an element of the Boolean matrix reflects an existing mapping from an input onto an output permutation, set it equal to 1.

7. If an element of the Boolean matrix reflects a non-existing mapping from an input onto an output permutation, set it equal to 0.

Example 1:

A single-input-single-output fuzzy system is considered whereby the input i and the output o can take the linguistic values S (small), M (medium) and B (big). This system is described by the following rule base:

$$\begin{aligned} \text{If } i \text{ is } S \text{ then } o \text{ is } B \\ \text{If } i \text{ is } M \text{ then } o \text{ is } M \\ \text{If } i \text{ is } B \text{ then } o \text{ is } S \end{aligned} \quad (12)$$

The linguistic values S , M , and B can be substituted by the integers 1, 2 and 3, respectively. In this case, the rule base can be presented by the following Boolean matrix:

$$\begin{array}{cccc} i/o & 1 & 2 & 3 \\ 1 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 \end{array} \quad (13)$$

Example 2:

A two-input-two-output fuzzy system is considered whereby the inputs i_1 , i_2 and the outputs o_1 , o_2 can take the linguistic values S (small), M (medium) and B (big). This system is described by the following rule base:

$$\begin{aligned}
& \text{If } i_1 \text{ is } S \text{ and } i_2 \text{ is } S \text{ then } o_1 \text{ is } B \text{ and } o_2 \text{ is } B \\
& \text{If } i_1 \text{ is } S \text{ and } i_2 \text{ is } M \text{ then } o_1 \text{ is } B \text{ and } o_2 \text{ is } M \\
& \text{If } i_1 \text{ is } S \text{ and } i_2 \text{ is } B \text{ then } o_1 \text{ is } B \text{ and } o_2 \text{ is } S \\
& \text{If } i_1 \text{ is } M \text{ and } i_2 \text{ is } S \text{ then } o_1 \text{ is } M \text{ and } o_2 \text{ is } B \\
& \text{If } i_1 \text{ is } M \text{ and } i_2 \text{ is } M \text{ then } o_1 \text{ is } M \text{ and } o_2 \text{ is } M \\
& \text{If } i_1 \text{ is } M \text{ and } i_2 \text{ is } B \text{ then } o_1 \text{ is } M \text{ and } o_2 \text{ is } S \\
& \text{If } i_1 \text{ is } B \text{ and } i_2 \text{ is } S \text{ then } o_1 \text{ is } S \text{ and } o_2 \text{ is } B \\
& \text{If } i_1 \text{ is } B \text{ and } i_2 \text{ is } M \text{ then } o_1 \text{ is } S \text{ and } o_2 \text{ is } M \\
& \text{If } i_1 \text{ is } B \text{ and } i_2 \text{ is } B \text{ then } o_1 \text{ is } S \text{ and } o_2 \text{ is } S
\end{aligned} \tag{14}$$

The linguistic values S , M , and B can be substituted by the integers 1, 2 and 3, respectively. In this case, the rule base can be presented by the following Boolean matrix:

$i_1, i_2 / o_1, o_2$	11	12	13	21	22	23	31	32	33	(15)
11	0	0	0	0	0	0	0	0	1	
12	0	0	0	0	0	0	0	1	0	
13	0	0	0	0	0	0	1	0	0	
21	0	0	0	0	0	1	0	0	0	
22	0	0	0	0	1	0	0	0	0	
23	0	0	0	1	0	0	0	0	0	
31	0	0	1	0	0	0	0	0	0	
32	0	1	0	0	0	0	0	0	0	
33	1	0	0	0	0	0	0	0	0	

Example 3:

A three-input-three-output fuzzy system is considered whereby the inputs i_1 , i_2 , i_3 and the outputs o_1 , o_2 , o_3 can take the linguistic values S (small) and B (big). This system is described by the following rule base:

$$\begin{aligned}
& \text{If } i_1 \text{ is } S \text{ and } i_2 \text{ is } S \text{ and } i_3 \text{ is } S \text{ then } o_1 \text{ is } B \text{ and } o_2 \text{ is } B \text{ and } o_3 \text{ is } B \\
& \text{If } i_1 \text{ is } S \text{ and } i_2 \text{ is } S \text{ and } i_3 \text{ is } B \text{ then } o_1 \text{ is } B \text{ and } o_2 \text{ is } B \text{ and } o_3 \text{ is } S \\
& \text{If } i_1 \text{ is } S \text{ and } i_2 \text{ is } B \text{ and } i_3 \text{ is } S \text{ then } o_1 \text{ is } B \text{ and } o_2 \text{ is } S \text{ and } o_3 \text{ is } B \\
& \text{If } i_1 \text{ is } S \text{ and } i_2 \text{ is } B \text{ and } i_3 \text{ is } B \text{ then } o_1 \text{ is } B \text{ and } o_2 \text{ is } S \text{ and } o_3 \text{ is } S \\
& \text{If } i_1 \text{ is } B \text{ and } i_2 \text{ is } S \text{ and } i_3 \text{ is } S \text{ then } o_1 \text{ is } S \text{ and } o_2 \text{ is } B \text{ and } o_3 \text{ is } B \\
& \text{If } i_1 \text{ is } B \text{ and } i_2 \text{ is } S \text{ and } i_3 \text{ is } B \text{ then } o_1 \text{ is } S \text{ and } o_2 \text{ is } B \text{ and } o_3 \text{ is } S \\
& \text{If } i_1 \text{ is } B \text{ and } i_2 \text{ is } B \text{ and } i_3 \text{ is } S \text{ then } o_1 \text{ is } S \text{ and } o_2 \text{ is } S \text{ and } o_3 \text{ is } B \\
& \text{If } i_1 \text{ is } B \text{ and } i_2 \text{ is } B \text{ and } i_3 \text{ is } B \text{ then } o_1 \text{ is } S \text{ and } o_2 \text{ is } S \text{ and } o_3 \text{ is } S
\end{aligned} \tag{16}$$

The linguistic values S and B can be substituted by the integers 1 and 2, respectively. In this case, the rule base can be presented by the following Boolean matrix:

$$\begin{array}{c|cccccccc}
 i_1, i_2, i_3 / o_1, o_2, o_3 & 111 & 112 & 121 & 122 & 211 & 212 & 221 & 222 \\
 \hline
 111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 112 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 121 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 122 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 211 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 212 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 221 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 222 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{array} \quad (17)$$

It is obvious from the examples above that the Boolean matrix presentation of a fuzzy rule base facilitates the definition of the properties of the rule base. These properties can be implied directly from some properties of the Boolean matrix such as the number of non-zero elements in its rows and columns.

4. Formal Manipulation of Fuzzy Systems

Fuzzy systems can be manipulated formally using Boolean matrices. In this case, pairs of individual rule bases in a multiple rule based fuzzy system can be merged either horizontally, vertically or with respect to common inputs. The specific type of manipulation is chosen on the basis of the location of the corresponding rule bases with respect to each other. The remaining part of this section describes in detail three different methods for formal manipulation of fuzzy rule bases.

The process of merging two fuzzy rule bases in sequence into a single fuzzy rule base is called ‘horizontal merging’ and it is shown in Figure 1.

Figure 1: Horizontal merging of rule bases RB_1 and RB_2 into rule base RB

The horizontal merging operation is based on horizontal composition of rule bases which is identical to a Boolean matrix multiplication. This operation is binary in that it can be applied to only two operands at a time. The operands in this case are the Boolean matrices representing the operand rule bases. The result from the application of this operation is a single Boolean matrix representing the product rule base.

The space complexity associated with the Boolean matrix for the product rule base in the horizontal merging operation can be evaluated in the general case as follows. If the first operand rule base has $m1$ inputs taking $x1$ linguistic values each and the second operand rule base has $n2$ outputs taking $y2$ linguistic values each, then the Boolean matrix for the product rule base will have $x1^{m1}$ rows and $y2^{n2}$ columns, i.e. it will have $x1^{m1} \cdot y2^{n2}$ elements.

Table 3 below shows in a simplified context how the number of elements in the Boolean matrix for the product rule base in horizontal merging increases with the increase of the number of inputs to the first operand rule base, the number of outputs from the second operand rule base and the number of linguistic values that these inputs and outputs can take.

Table 3: Space complexity of Boolean matrix for product rule base in horizontal merging

$m1, n2 / x1, y2$	$x1=2, y2=2$	$x1=3, y2=3$	$x1=4, y2=4$
$m1=1, n2=1$	4	6	16
$m1=2, n2=2$	16	81	256
$m1=3, n2=3$	64	729	4,096

The application of the horizontal composition operation to Boolean matrices is described by Algorithm 2 and illustrated by Example 4.

Algorithm 2:

1. Label the rows of the product matrix with the row labels $v_{ips}, p=1,...,m, s=1,...,r$ from the first operand matrix.

2. Label the columns of the product matrix with the column labels v_{oqs} , $q=1,...,n$, $s=1,...,r$ from the second operand matrix.

3. Set each element of the product matrix equal to 1 or 0 by mapping it from the corresponding row in the first operand matrix and the corresponding column in the second operand matrix, as described in step 4.

4. Find the product matrix by multiplying the operand matrices using the operations for ‘addition’ and ‘multiplication’ of elements, as defined by Equations (4)-(11).

Example 4:

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices:

$$RB_1: \begin{array}{ccccc} & i_1/z_1 & 1 & 2 & 3 \\ 1 & & 0 & 1 & 0 \\ 2 & & 1 & 0 & 0 \\ 3 & & 0 & 0 & 1 \end{array} \quad (18)$$

$$RB_2: \begin{array}{ccccc} & z_1/o_1 & 1 & 2 & 3 \\ 1 & & 1 & 0 & 0 \\ 2 & & 0 & 0 & 1 \\ 3 & & 0 & 1 & 0 \end{array} \quad (19)$$

The horizontal merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1 * RB_2 = RB$ where RB will be presented by the following Boolean matrix:

$$RB: \begin{array}{ccccc} & i_1/o_1 & 1 & 2 & 3 \\ 1 & & 0 & 0 & 1 \\ 2 & & 1 & 0 & 0 \\ 3 & & 0 & 1 & 0 \end{array} \quad (20)$$

The process of merging two fuzzy rule bases in parallel into a single fuzzy rule base is called ‘vertical merging’ and it is shown in Figure 2.

Figure 2: Vertical merging of rule bases RB_1 and RB_2 into rule base RB

The vertical merging operation is based on vertical composition of rule bases which is identical to a Kronecker matrix product. This operation is binary in that it can be applied to only two operands at a time. The operands in this case are the Boolean matrices representing the operand rule bases. The result from the application of this operation is a single Boolean matrix representing the product rule base.

The space complexity associated with the Boolean matrix for the product rule base in the vertical merging operation can be evaluated in the general case as follows. If the first operand rule base has m_1 inputs taking x_1 linguistic values each and n_1 outputs taking y_1 linguistic values each, the second operand rule base has m_2 inputs taking x_2 linguistic values each and n_2 outputs taking y_2 linguistic values each, then the Boolean matrix for the product rule base will have $x_1^{m_1} \cdot x_2^{m_2}$ rows and $y_1^{n_1} \cdot y_2^{n_2}$ columns, i.e. it will have $x_1^{m_1} \cdot x_2^{m_2} \cdot y_1^{n_1} \cdot y_2^{n_2}$ elements.

Table 4 below shows in a simplified context how the number of elements in the Boolean matrix for the product rule base in vertical merging increases with the increase of the number of inputs to / outputs from the first operand rule base, the number of inputs to / outputs from the second operand rule base and the number of linguistic values that these inputs / outputs can take.

Table 4: Space complexity of Boolean matrix for product rule base in vertical merging

$m1, n1, m2, n2 /$ $x1, y1, x2, y2$	$x1=2, y1=2$ $x2=2, y2=2$	$x1=3, y1=3$ $x2=3, y2=3$	$x1=4, y1=4$ $x2=4, y2=4$
$m1=1, n1=1$ $m2=1, n2=1$	16	81	256
$m1=2, n1=2$ $m2=2, n2=2$	246	729	65,536
$m1=3, n1=3$ $m2=3, n2=3$	4,096	531,441	1,6777,216

The application of the vertical composition operation to Boolean matrices is described by Algorithm 3 and illustrated by Example 5.

Algorithm 3:

1. Construct all possible permutations of row labels $v_{ips}, p=1,...,m, s=1,...,r$ from the operand matrices and sort them.
2. Construct all possible permutations of column labels $v_{oqs}, q=1,...,n, s=1,...,r$ from the operand matrices and sort them.
3. Label the rows of the product matrix with the sorted permutations of row labels $v_{ips}, p=1,...,m, s=1,...,r$ from the operand matrices.
4. Label the columns of the product matrix with the sorted permutations of column labels $v_{oqs}, q=1,...,n, s=1,...,r$ from the operand matrices.
5. Go through all the elements of the operand matrices and set each element of the product matrix equal to 1 or 0, as described in steps 6 and 7.
6. If an element of the product matrix is mapped from a pair of non-zero elements in the product matrices, set this element equal to 1 in accordance with Equations (8)-(11).

7. If an element of the product matrix is mapped from a pair of elements in the product matrices at least one of which is zero, set this element equal to 0 in accordance with Equations (8)-(11).

Example 5:

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices:

$$RB_1: \begin{array}{ccccc} & i_1/o_1 & 1 & 2 & 3 \\ 1 & & 0 & 1 & 0 \\ 2 & & 1 & 0 & 0 \\ 3 & & 0 & 0 & 1 \end{array} \quad (21)$$

$$RB_2: \begin{array}{ccccc} & i_2/o_2 & 1 & 2 & 3 \\ 1 & & 1 & 0 & 0 \\ 2 & & 0 & 0 & 1 \\ 3 & & 0 & 1 & 0 \end{array} \quad (22)$$

The vertical merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1+RB_2=RB$ where RB will be presented by the following Boolean matrix:

$$RB: \begin{array}{ccccccccc} & i_1, i_2/o_1, o_2 & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\ 11 & & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 12 & & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 13 & & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 21 & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 22 & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 23 & & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 31 & & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 32 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 33 & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \quad (23)$$

The process of merging two fuzzy rule bases with common inputs into a single fuzzy rule base is called ‘output merging’ and it is shown in Figure 3.

Figure 3: Output merging of rule bases RB_1 and RB_2 into rule base RB

The output merging operation is based on vertical composition of rule bases which is identical to a modified Kroneker matrix product. This operation is binary in that it can be applied to only two operands at a time. The operands in this case are the Boolean matrices representing the operand rule bases. The result from the application of this operation is a single Boolean matrix representing the product rule base.

The space complexity associated with the Boolean matrix for the product rule base in the vertical merging operation can be evaluated in the general case as follows. If both operand rule bases have m inputs taking x linguistic values each, the first operand rule base has $n1$ outputs taking $y1$ linguistic values each, the second operand rule base has $n2$ outputs taking $y2$ linguistic values each, then the Boolean matrix for the product rule base will have x^m rows and $y1^{n1} \cdot y2^{n2}$ columns, i.e. it will have $x^m \cdot y1^{n1} \cdot y2^{n2}$ elements.

Table 5 below shows in a simplified context how the number of elements in the Boolean matrix for the product rule base in vertical merging increases with the increase of the number of inputs to both operand rule bases, the number of outputs from the first operand rule base, the number of outputs from the second operand rule base and the number of linguistic values that these inputs and outputs can take.

Table 5: Space complexity of Boolean matrix for product rule base in output merging

$m, n1, n2 / x, y1, y2$	$x=2, y1=2, y2=2$	$x=3, y1=3, y2=3$	$x=4, y1=4, y2=4$
$m=1, n1=1, n2=1$	8	27	64
$m=2, n1=2, n2=2$	64	6,561	4,096
$m=3, n1=3, n2=3$	512	19.683	262,144

The application of the output composition operation to Boolean matrices is described by Algorithm 4 and illustrated by Example 6.

Algorithm 4:

1. Label the rows of the product matrix with the common row labels $v_{ips}, p=1, \dots, m, s=1, \dots, r$ of the two operand matrices.
2. Label the columns of the product matrix with the sorted permutations of the column labels $v_{oqs}, q=1, \dots, n, s=1, \dots, r$ of the two operand matrices.
3. Go through all the elements of the operand matrices and set each element of the product matrix equal to 1 or 0, as described in steps 4 and 5.
4. If an element of the product matrix is mapped from two non-zero elements in the operand matrices, set this element equal to 1 in accordance with Equations (8)-(11).
5. If an element of the product matrix is mapped from two elements in the operand matrices such that at least one of them is zero, set this element equal to 0 in accordance with Equations (8)-(11).

Example 6:

The operand rule bases RB_1 and RB_2 are presented by the following Boolean matrices:

$$\begin{array}{rcccl}
 RB_1: & i_1/o_1 & 1 & 2 & 3 \\
 & 1 & 0 & 1 & 0 \\
 & 2 & 1 & 0 & 0 \\
 & 3 & 0 & 0 & 1
 \end{array} \tag{24}$$

$$\begin{array}{rcccl}
 RB_2: & i_1/o_2 & 1 & 2 & 3 \\
 & 1 & 1 & 0 & 0 \\
 & 2 & 0 & 0 & 1 \\
 & 3 & 0 & 1 & 0
 \end{array} \tag{25}$$

The output merging of RB_1 and RB_2 into a product rule base RB will be denoted by $RB_1;RB_2=RB$ where RB will be presented by the following Boolean matrix:

$$\begin{array}{rcccccccccc}
 RB: & i_1/o_1,o_2 & 11 & 12 & 13 & 21 & 22 & 23 & 31 & 32 & 33 \\
 & 11 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 & 12 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
 \end{array} \tag{26}$$

5. Formal Transformation of Fuzzy Systems

Fuzzy systems can be transformed formally using Boolean matrices. In this case, three or more individual rule bases in a multiple rule based fuzzy system can be merged either horizontally, vertically or with respect to common inputs. The specific type of manipulation is chosen on the basis of the location of the corresponding rule bases with respect to each other. The remaining part of this section describes in detail three different methods for formal transformation of fuzzy rule bases.

As all three merging operations are associative, it is possible to change the order of operations on three or more operand rule bases in the case of repetitive merging. Therefore, the changing of the order of operations for any three or more operand rule bases will not affect the product rule base.

The associativity property of the horizontal composition operation is illustrated by Example 7.

Example 7:

The rule bases in sequence RB_1 , RB_2 and RB_3 are presented by the Boolean matrices in Equations (27)-(29).

$$\begin{array}{rcl}
 RB_1: & i_1/z_{12} & 1 \quad 2 \\
 & 1 & 1 \quad 0 \\
 & 2 & 1 \quad 0
 \end{array} \tag{27}$$

$$\begin{array}{rcl}
 RB_2: & z_{12}/z_{23} & 1 \quad 2 \\
 & 1 & 0 \quad 1 \\
 & 2 & 1 \quad 0
 \end{array} \tag{28}$$

$$\begin{array}{rcl}
 RB_3: & z_{23}/o_3 & 1 \quad 2 \\
 & 1 & 0 \quad 1 \\
 & 2 & 0 \quad 1
 \end{array} \tag{29}$$

The associativity of horizontal merging of rule bases RB_1 , RB_2 and RB_3 is described by Figure 4 and Equation (30).

Figure 4: Associativity of horizontal merging of rule bases RB_1 , RB_2 and RB_3

$$\text{If } RB_{(1*2)*3} = (RB_1 * RB_2) * RB_3 \text{ and } RB_{1*(2*3)} = RB_1 * (RB_2 * RB_3) \tag{30}$$

$$\text{then } RB_{(1*2)*3} = RB_{1*(2*3)}$$

In this case, the horizontal merging of rule bases RB_1 , RB_2 and RB_3 from left to right will give the same result as their horizontal merging from right to left and the product rule base will be presented by the following Boolean matrix:

$$\begin{array}{rcl}
 RB_{(1*2)*3}=RB_{1*(2*3)}: & i_1/o_3 & 1 \quad 2 \\
 & 1 & 0 \quad 1 \\
 & 2 & 0 \quad 1
 \end{array} \tag{31}$$

The associativity property of the vertical composition operation is illustrated by Example 8.

Example 8:

The rule bases in parallel RB_1 , RB_2 and RB_3 are presented by the Boolean matrices in Equations (32)-(34).

$$RB_1: \begin{array}{ccc} i_1/o_1 & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \quad (32)$$

$$RB_2: \begin{array}{ccc} i_2/o_2 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \quad (33)$$

$$RB_3: \begin{array}{ccc} i_3/o_3 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \quad (34)$$

The associativity of vertical merging of rule bases RB_1 , RB_2 and RB_3 is described by Figure 5 and Equation (35).

Figure 5: Associativity of vertical merging of rule bases RB_1 , RB_2 and RB_3

$$\text{If } RB_{(1+2)+3} = (RB_1 + RB_2) + RB_3 \text{ and } RB_{1+(2+3)} = RB_1 + (RB_2 + RB_3) \quad (35)$$

$$\text{then } RB_{(1+2)+3} = RB_{1+(2+3)}$$

In this case, the vertical merging of rule bases RB_1 , RB_2 and RB_3 from top to bottom will give the same result as their vertical merging from bottom to top and the product rule base will be presented by the following Boolean matrix:

$$RB_{(1+2)+3}=RB_{1+(2+3)}: i_1, i_2, i_3 / o_1, o_2, o_3 \quad \begin{array}{c|cccccccc} & 111 & 112 & 121 & 122 & 211 & 212 & 221 & 222 \\ \hline 111 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 112 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 121 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 122 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 211 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 212 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 221 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 222 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \quad (36)$$

The associativity property of the output composition operation is illustrated by Example 9.

Example 9:

The rule bases in parallel RB_1 , RB_2 and RB_3 are presented by the Boolean matrices in Equations (37)-(39).

$$RB_1: \quad \begin{array}{c|cc} i/o_1 & 1 & 2 \\ \hline 1 & 1 & 0 \\ 2 & 1 & 0 \end{array} \quad (37)$$

$$RB_2: \quad \begin{array}{c|cc} i/o_2 & 1 & 2 \\ \hline 1 & 0 & 1 \\ 2 & 1 & 0 \end{array} \quad (38)$$

$$RB_3: \quad \begin{array}{c|cc} i/o_3 & 1 & 2 \\ \hline 1 & 0 & 1 \\ 2 & 0 & 1 \end{array} \quad (39)$$

The associativity of vertical merging of rule bases RB_1 , RB_2 and RB_3 is described by Figure 6 and Equation (40).

Figure 6: Associativity of output merging of rule bases RB_1 , RB_2 and RB_3

$$\begin{aligned} \text{If } RB_{(1;2);3} &= (RB_1 ; RB_2) ; RB_3 \text{ and } RB_{1;(2;3)} = RB_1 ; (RB_2 ; RB_3) \\ \text{then } RB_{(1;2);3} &= RB_{1;(2;3)} \end{aligned} \quad (40)$$

In this case, the output merging of rule bases RB_1 , RB_2 and RB_3 from top to bottom will give the same result as their output merging from bottom to top and the product rule base will be presented by the following Boolean matrix:

$$RB_{(1;2);3}=RB_{1;(2;3)}: i/o_1,o_2,o_3 \quad \begin{matrix} 111 & 112 & 121 & 122 & 211 & 212 & 221 & 222 \end{matrix} \quad (41)$$

11	0	0	0	1	0	0	0	0
12	0	1	0	0	0	0	0	0

6. Application to Feedforward Fuzzy Systems

The proposed complexity management methodology is applied to the modelling of a feedforward fuzzy system with four rule bases and is evaluated comparatively in terms of model transparency. This type of system is like an initial fuzzy network that is shown in Figure 7.

Figure 7: Initial fuzzy network

The locations of the rule bases in this initial fuzzy network are presented by a grid structure with horizontal levels and vertical levels as follows:

<i>level/layer</i>	<i>layer 1</i>	<i>layer 2</i>	(42)
<i>level 1</i>	$RB_1, i_1, o_{11}, o_{12}$	$RB_2, i_{21}, i_{22}, o_2$	
<i>level 2</i>	$RB_3, i_3, o_{31}, o_{32}$	$RB_4, i_{41}, i_{42}, o_4$	

The connections between the rule bases in the initial fuzzy network are presented by a similar grid structure as follows:

<i>level/layer</i>	<i>layer 1</i>	<i>layer 2</i>	(43)
<i>level 1</i>	$o_{11} = i_{21}$ $o_{12} = i_{41}$	o_2	
<i>level 2</i>	$o_{31} = i_{22}$ $o_{32} = i_{42}$	o_4	

The feedforward connections $o_{12}=i_{41}$ and $o_{31}=i_{22}$ in the initial fuzzy network are crossing their paths in a complex way. In this case, it is not possible to merge horizontally the rule bases in the first layer RB_1 and RB_3 with the rule bases in the second layer RB_2 and RB_4 . Therefore, it is necessary to convert the initial fuzzy network into a final fuzzy network with no crossing of the connections. Algorithm 5 describes the process of this conversion.

Algorithm 5:

1. Find rule bases RB_{11} and RB_{12} such that $RB_1 = RB_{11}; RB_{12}$.
2. Find rule bases RB_{31} and RB_{32} such that $RB_3 = RB_{31}; RB_{32}$.
3. Find a rule base RB_{E1} such that $(RB_{11}+RB_{31}) * RB_2 = RB_{E1}$.
4. Find a rule base RB_{E2} such that $(RB_{12}+RB_{32}) * RB_4 = RB_{E2}$.
5. Find the rule base RB_E such that $RB_{E1}; RB_{E2} = RB_E$.

In the above algorithm, RB_E is the equivalent rule base for the initial fuzzy network from Equations (42)-(43) whereas RB_{E1} and RB_{E2} are the equivalent rule bases for the two fuzzy sub-networks of the final fuzzy network that is shown in Figure 8. In particular, steps 1-2 illustrate the output merging of the two pairs of rule bases (RB_{11}, RB_{12}) and (RB_{31}, RB_{32}) from the final network into the rule bases RB_1 and RB_3 from the initial network, respectively. Similarly, steps 3-4 show the derivation of the two rule bases RB_{E1} and RB_{E2} from the final

network by vertical and horizontal merging of the six rule bases above. Finally, step 5 describes the output merging of RB_{E1} and RB_{E2} into RB_E .

Figure 8: Final fuzzy network

The locations of the rule bases in this final fuzzy network are presented by a grid structure with horizontal levels and vertical levels as follows:

level/layer	layer 1	layer 2	(44)
level 1	RB_{11}, i_1, o_{11}	$RB_2, i_{21}, i_{22}, o_2$	
level 2	RB_{12}, i_1, o_{12}		
level 3	RB_{31}, i_3, o_{31}		
level 4	RB_{32}, i_3, o_{32}	$RB_4, i_{41}, i_{42}, o_4$	

The connections between the rule bases in the final fuzzy network are presented by a similar grid structure as follows:

<i>level/layer</i>	<i>layer 1</i>	<i>layer 2</i>	(45)
<i>level 1</i>	$o_{11} = i_{21}$	o_2	
<i>level 2</i>	$o_{12} = i_4$		
<i>level 3</i>	$o_{31} = i_{22}$		
<i>level 4</i>	$o_{32} = i_{42}$	o_4	

The feedforward connections $o_{12}=i_{41}$ and $o_{31}=i_{22}$ in the final fuzzy network are crossing their paths in a simple way. In this case, it is possible to merge vertically the rule bases in each of the two pairs of rule bases in the first layer (RB_{11}, RB_{31}), (RB_{12}, RB_{32}) and then to merge the two product rule bases ($RB_{11}+RB_{31}$), ($RB_{12}+RB_{32}$) horizontally with the rule bases in the second layer RB_2 and RB_4 , respectively, in accordance with steps 3-4 in Algorithm 5. It is also possible to merge the outputs of the equivalent rule bases RB_{E1} and RB_{E2} for the two fuzzy sub-networks of the final fuzzy network into an equivalent rule base RB_E for the initial fuzzy network in accordance with step 5 in Algorithm 5.

The proposed complexity management methodology is evaluated comparatively in terms of model transparency for the initial and the final fuzzy network. These two fuzzy network models are also compared to a general fuzzy system model that represents the current complexity reduction methodology.

The model transparency index used is given by the formula

$$(s+z)/(m+n) \quad (46)$$

where s is the number of subsystems, z is the number of connections, m is the number of inputs and n is the number of outputs. The formula implies that the model transparency increases with the increase in the number of subsystems and connections or with the decrease in the number of inputs and outputs.

The transparency figures obtained for the fuzzy system, the initial fuzzy network and the final fuzzy network are 0.25, 2.00 and 2.50, respectively. This shows that the two fuzzy

networks are between 8 and 10 times superior to the fuzzy system in terms of modelling transparency and capability for reducing qualitative complexity. For more complex cases, this superiority would be even bigger as the number of subsystems and connections usually increases more significantly in comparison to the increase of the number of inputs and outputs, i.e. the numerator in Equation (46) grows at a higher rate than the denominator.

Table 6: Transparency of fuzzy system, initial fuzzy network and final fuzzy network

<i>Fuzzy model</i>	<i>Fuzzy system</i>	<i>Initial fuzzy network</i>	<i>Final fuzzy network</i>
<i>Transparency index</i>	0.25	2	2.50

7. Conclusion

The proposed complexity management methodology for feedforward fuzzy rule based systems improves the transparency of the models used. This allows the structure of a fairly complex process in terms of interacting sub-processes to be reflected explicitly in the model. As a result, any complex process can be modelled by a fuzzy network in a more transparent way than by a fuzzy system due to the better visibility inside the process. This also leads to better understanding of the modelled process.

The proposed methodology is based on formal methods for presentation, manipulation and transformation of fuzzy rule bases. These methods make use of Boolean matrices for formal presentation of rule bases, binary merging operations for formal manipulation of rule bases and repetitive merging operations for formal transformation of rule bases.

The proposed methodology is illustrated for feedforward fuzzy networks with a fairly small number of sub-networks, connections, inputs and outputs. However, it can be easily extended to feedback fuzzy networks with an arbitrarily large number of sub-networks, connections, inputs and outputs. In this case, all binary merging operations presented can be

applied repetitively in a flexible way by using the associativity property. This would lead only to a linear increase of the associated quantitative complexity.

Acknowledgement

The first author would like to thank the Faculty of Technology at the University of Portsmouth for the granted research sabbatical that made possible the writing of this paper.

References

- [1] Guven M, Passino K (2001) Avoiding exponential parameter growth in fuzzy systems. IEEE Transactions on Fuzzy Systems 9/1 : 194-199
- [2] Kim Y, Ahn S, Kwon W (2000) Computational complexity of general fuzzy logic control and its simplification for a loop controller. Fuzzy Sets and Systems 111 : 215-224
- [3] Lazzerini B, Marcelloni F (2000) Reducing computation overhead in MISO fuzzy systems. Fuzzy Sets and Systems 113 : 485-496
- [4] Pal N, Eluri V, Mandal G (2002) Fuzzy logic approaches to structure preserving dimensionality reduction. IEEE Transactions on Fuzzy Systems 10/3 : 277-286
- [5] Gegov A (1996) Distributed fuzzy control of multivariable systems. Kluwer, Dordrecht
- [6] Gupta M, Kiszka J, Trojan G (1986) Multivariable structure of fuzzy control systems. IEEE Transactions on Systems, Man and Cybernetics 16/5 : 638-655
- [7] Wan F, Shang H, Wang L, Sun Y (2005) How to determine the minimum number of fuzzy rules to achieve given accuracy: a computational geometric approach to SISO case. Fuzzy Sets and Systems 150 : 199-209
- [8] Xiong N, Litz L (2002) Reduction of fuzzy control rules by means of premise learning – method and case study. Fuzzy Sets and Systems 132 : 217-231
- [9] Roubos H, Setnes M (2001) Compact and transparent fuzzy models and classifiers through iterative complexity reduction. IEEE Transactions on Fuzzy Systems 9/4 : 516-524

- [10] Setnes M, Babuska R, Verbruggen H (1998) Rule-based modelling: precision and transparency. *IEEE Transactions on Systems, Man and Cybernetics* 28/1 : 165-169
- [11] Setnes M (1995) Fuzzy rule base simplification using similarity measures. MSc thesis, Delft University of Technology
- [12] Lacrose V (1997) Complexity reduction of fuzzy controllers: application to multivariable control. PhD thesis, Toulouse Laboratory for Systems Analysis and Architecture
- [13] Jamshidi M (1997) Large scale systems: modelling, control and fuzzy logic. Prentice Hall, Upper Saddle River
- [14] Yan J, Ryan M, Power J (1994) Using fuzzy logic. Prentice Hall, New York
- [15] Yam Y, Baranyi P, Yang C (1999) Reduction of fuzzy rule base via singular value decomposition. *IEEE Transactions on Fuzzy Systems* 7/2 : 120-132
- [16] Tao C (2001) Comments on “reduction of fuzzy rule base via singular value decomposition”. *IEEE Transactions on Fuzzy Systems* 9/4 : 675-676
- [17] Simon D (2000) Design and rule base reduction of a filter for the estimation of motor currents, *International Journal of Approximate Reasoning* 25 : 145-167
- [18] Combs W, Andrews J (1998) Combinatorial rule explosion eliminated by a fuzzy rule configuration. *IEEE Transactions on Fuzzy Systems* 6/1 : 1-11
- [19] Mendel J, Liang Q (1999) Comments on “combinatorial rule explosion eliminated by a fuzzy rule configuration”. *IEEE Transactions on Fuzzy Systems* 7/3 : 369-371
- [20] Trillas E, Alsina C (2002) On the law $[(p \text{ and } q) \text{ then } r] = [(if \text{ } p \text{ then } r) \text{ or } (if \text{ } q \text{ then } r)]$ in fuzzy logic. *IEEE Transactions on Fuzzy Systems* 10/1 : 84-88
- [21] Chen S, Yu F, Chung H (2002) Decoupled fuzzy controller design with single-input fuzzy logic. *Fuzzy Sets and Systems* 129 : 335-342

[22] Gegov A, Frank M (1995) Hierarchical fuzzy control of multivariable systems. Fuzzy Sets and Systems 72 : 299-310

[23] Molloy, S (2002) Fuzzy Control of multiple-input multiple-output processes. PhD thesis, Delft University of Technology

[24] Gegov A, Frank M (1995) Decomposition of multivariable systems for distributed fuzzy control. Fuzzy Sets and Systems 73 : 329-340

[25] Xu C (1991) Linguistic decoupling control of fuzzy multivariable processes. Fuzzy Sets and Systems 44 : 209-217

[26] Xu C, Lu Y (1989) Decoupling in fuzzy systems: a cascade compensation approach. Fuzzy Sets and Systems 29 : 177-185

[27] Galichet S, Boukezzoula R, Foulloy L (2004) Explicit analytical formulation and exact inversion of decomposable fuzzy systems with singleton consequents. Fuzzy Sets and Systems 146 : 421-436

[28] Huwendiek O, Brockmann W (1999) Function approximation with decomposed fuzzy systems. Fuzzy Sets and Systems 101 : 273-286

[29] Joo M, Lee J (2002) Universal approximation by hierarchical fuzzy system with constraints on the fuzzy rule. Fuzzy Sets and Systems 130 : 175-188

[30] Joo M, Lee J (2005) A class of hierarchical fuzzy systems with constraints on the fuzzy rules. IEEE Transactions on Fuzzy Systems 13/2 : 194-203

[31] Lee M, Chung H, Yu F (2003) Modelling of hierarchical fuzzy systems. Fuzzy Sets and Systems 138 : 343-361

[32] Raju G, Zhou J, Kisner R (1991) Hierarchical fuzzy control. International Journal of Control 54/5 : 1201-1216

[33] Wang L (1999) Analysis and design of hierarchical fuzzy systems. IEEE Transactions on Fuzzy Systems 7/5 : 617-624

[34] Bolognani S, Zigliotto M (1998) Hardware and software effective configurations for multi-input fuzzy logic controllers. *IEEE Transactions on Fuzzy Systems* 6/1 : 173-179

[35] Jang J, Sun C, Mizutani E (1997) *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Prentice Hall, Upper Saddle River

[36] Negnevitsky M (2002) *Artificial intelligence: a guide to intelligent systems*. Pearson Education, Harlow

[37] Ross T (2004) *Fuzzy logic with engineering applications*. Wiley, Chichester

[38] Gegov A (2007) *Complexity management in fuzzy systems*. Springer, Berlin