Reduction in Space Complexity And Error Detection/Correction of a Fuzzy controller

F. Vainstein[1], E. Marte[2], V. Osoria[3], R. Romero[4]

[1]Georgia Institute of Technology, 210 Technology Circle Savannah, GA 31407
feodor.vainstein@gtsav.gatech.edu
[2]Pontificia universidad Católica Madre y Maestra
Autopista Duarte Km. 1 ½, Santiago de los Caballeros, Dominican Republic
edwin.marte@ engineeringroup.com
[3]Universidad Tecnológica de Santiago, Av. Estrella Sadhalá, Santiago de los Caballeros
valentin.osoria@verizon.net.do
[4]Instituto Tecnologico de Santo Domingo, Av. Los Proceres, Galá, Santo Domingo
rafael.romero@engineeringroup.com

Keywords: Space Complexity, Reduction, Fuzzy, Error Detection, Controller

Abstract: Fuzzy controllers prove to be very useful for practical applications, especially in the cases when there is no appropriate mathematical model of behavior of the controlled object. Control signal is computed by fuzzy controller with the use of rule base table. In this paper we propose a mathematical method for reduction in space complexity of the system by decreasing the number of address lines in the memory used to store If-Then rules. The idea of the method is to use variable radix in representing integers. We also propose incorporation of error-correcting codes in the memory used to store If-then rules without substantial increasing of space complexity and application of signature analysis for error detection/location in a fuzzy controller.

1. Introduction

Fuzzy sets and some basic ideas pertaining to their theory were first introduced in 1965 by Lofti A. Zadeh, a Professor of Electrical Engineering at the University of California a Berkeley. The development of fuzzy set theory and fuzzy logic experimented changes since their introduction. Therefore the "Fuzzy Boom" (since 1989) has been characterized by a rapid increase in successful industrial applications that have netted impressive revenues.

Major research centers have been established devoted to this field. This all been accompanied by a tremendous increase in the number of contributors as well as in the number of relevant publications, including several dedicated journals.

Braunstingl [1] developed a wall-following robot that used a fuzzy logic controller and local navigation strategy to determine its movement. The fuzzy logic controller uses the input variables to control the firing of 33 rules.

A fuzzy system developed by Surmann [2] controls the navigation of an autonomous mobile robot. The entire system has about 180 fuzzy rules that associate 30 fuzzy inputs with 11 outputs. Potentially, the number of fuzzy rules can be very large.

The contribution of this paper is to tackle the space complexity of the system by decreasing the number of addresses lines used to store If-Then rules. Also the incorporation of error correcting codes in the memory used to store If-Then rules without substantial increasing space complexity as well as application of signatures analysis for error detection/location in a fuzzy controller.

2. Fuzzification, Rule Base and Defuzzification

Fuzzy controllers follow standard procedures for their design, which consists of fuzzification, control rule base establishment, and defuzzification as shown by fig. 1. We are using a fuzzy controller with a sensor 0 and sensor k where r_0 and r_k denote the number of membership possible values.

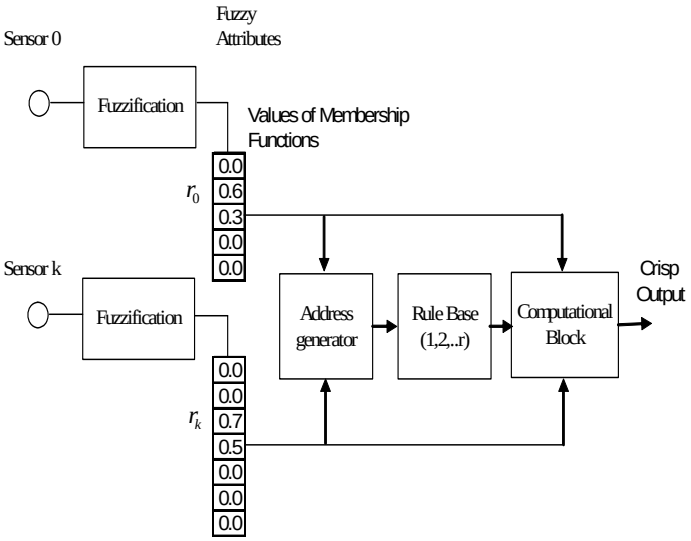

Fig. 1 Basic Fuzzy Controller Block diagram

2.1 Fuzzification

Fuzzification is mapping from the crisp domain into the fuzzy domain. Fuzzification also means the assigning of linguistic value, defined by relative small number of membership functions to variable.

2.2 Rule Base

The rule base is in fact a big database of rules that keep the knowledge of how is best to control the system.

2.3 Computational block

The Computational Block runs the inference engine which goes through all the rules, evaluating the firing strength of each rule which in turn is proportional to the truth-value of the preconditions.

After all the rules are computed, we have the firing strength of each rule. A problem then arises - we might have several rules with similar consequents, but different firing strength. Such a situation will result with different membership values for the same output. Here the defuzzifier comes in.

2.4 Defuzzification

The defuzzifier block task is to receive as input the membership values of the outputs, or in other words, the fuzzy outputs. Then it returns the actual numerical output, which may be a drug dose, a desired temperature, or any other variable. There are many methods of solving that problem and all of them aggregate the membership values of the outputs, in some form of an average, to find out the actual output.

3. Address Generation

Fig. 2 below shows a classical fuzzy controller implementation with two sensors, velocity and position.

Normally different values of position and velocity will trigger different and probably simultaneous rules which will lead the computational block to take a decision with these values. Base Rules in a fuzzy controller are preset values for decision taking, as an example for the controller in Fig.2 we created a 2-dimensional table where we are implementing five different decisions (1 to 5). Therefore the computational block for delivering 2 outputs in this system will need 3 bits for each output to represent these values in classical binary format (Fig. 3).

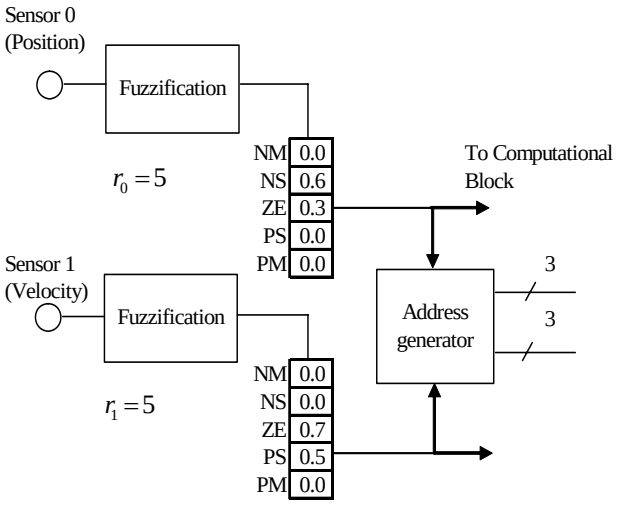

Fig. 2 Example of Address Generation

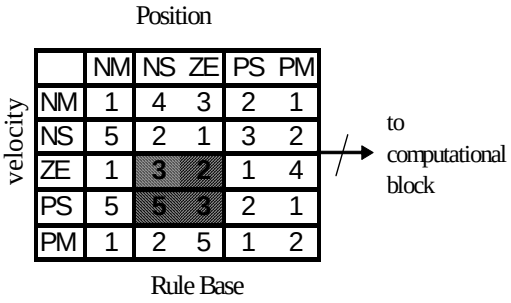

Fig. 3 Bidimensional Table for Decision
Take of the fuzzy Controller

3.1 Number of Lines and Space Complexity

Denote by N_0 the number of input address lines of the ROM storing Rule Base Table.

With the straightforward approach, the value of N_0 is obtained from the following formula:

$$N_0 = \sum_{i=0}^{k} [\log_2 r_i] \tag{1}$$

Example: Let k=9, $r_0 = r_1 = ... = r_9 = 5$ Using (1) we obtain $N_0 = 30$.

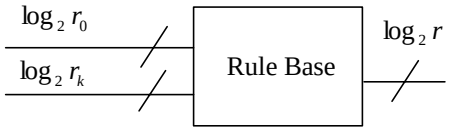

Fig. 4 Straight forward Address Generation

Let us assume that Rules Base is stored in a ROM. The space complexity of a Rom is proportional to 2^N where N is the number of address lines in the Rom (See Fig. 5).

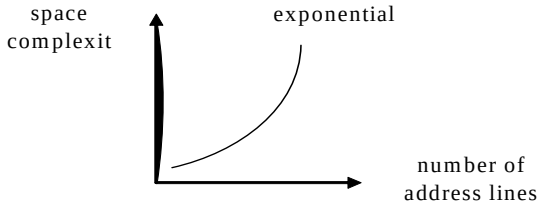

Fig. 5 Space Complexity Vs Address Lines

4. Variable Radix Numbers (Multi Radix)

In this paper we introduce a new representation of numbers. Unlike the usual decimal, and binary numbers, where the radix is fixed, in our representation the radix differ from position to position. We call this numbers Variable Radix (or Multy Radix) Numbers.

By Definition Multy Radix number can be written as follow:

$$a_r = (a_k, ..., a_0), \tag{2}$$

Where,

$$r = \{r_0, ..., r_k\}$$
$$a_0 \in \{0, ..., r_0 - 1\}$$
$$a_1 \in \{0, ..., r_1 - 1\}$$
$$\text{.....}$$
$$a_k \in \{0, ..., r_k - 1\}$$

The Multy Radix number a_r has the value

$$a_r = a_0 + a_1 r_0 + a_2 r_0 r_1 + ... + a_k r_0 r_1 ... r_{k-1} \tag{3}$$

Example:

Let's assume that $r_0 = 5, r_1 = 2, r_2 = 7$

Then the multi radix number 604 it then has the decimal value $a_r = 64_{10}$

Usage of multi radix numbers in a fuzzy controller will reduce the space complexity of the system.

4.1 Important Note

There exists natural one-to-one correspondence between the set of multi radix numbers and the set of fuzzy rules. It is demonstrated on Fig. 6 and Fig. 7.

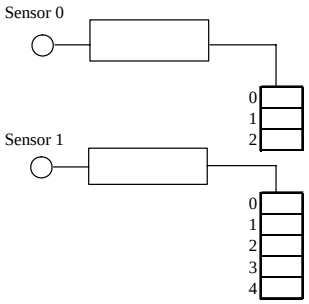

Fig. 6 Set of fuzzy Rules

	0	1	2	3	4
0	0 0 1	01 4	02 3	03 5	0 4 1
1	1 0 3	11 4	12 4	13 1	14 2
2	2 0 2	21 1	22 1	23 1	24 2

Fig. 7 One-to-One Mapping

4.2 Multi Radix to Binary Converter

Denote by $w_1 = r_0, w_2 = r_0 r_1, ..., w_k = r_0 r_1 ... r_k$.

Then $a_r = a_0 + a_1 w_1 + a_2 w_2 + ... + a_k w_k$ (4)

The block diagram of Multy Radix to Binary Converter is shown in Fig. 8 for the case of $r_0 = 5, r_1 = 5,\ r_2 = 3, r_3 = 6$

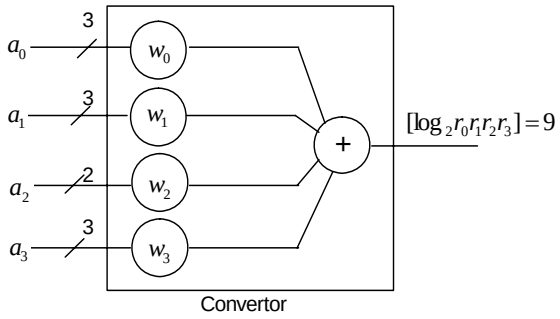

Fig. 8 Multi Radix to Binary Converter

3-input line ROMs can be used as multipliers by the constant w_i.

5. Reduction in space complexity

We can reduce the space complexity of the Rule Base by using Multi Radix numbers as shown in Fig. 9.

If we denote by N_0 the initial number of addresses lines and by N the number of addresses lines after the Multi Radix to Binary Converter then the following statement is true:

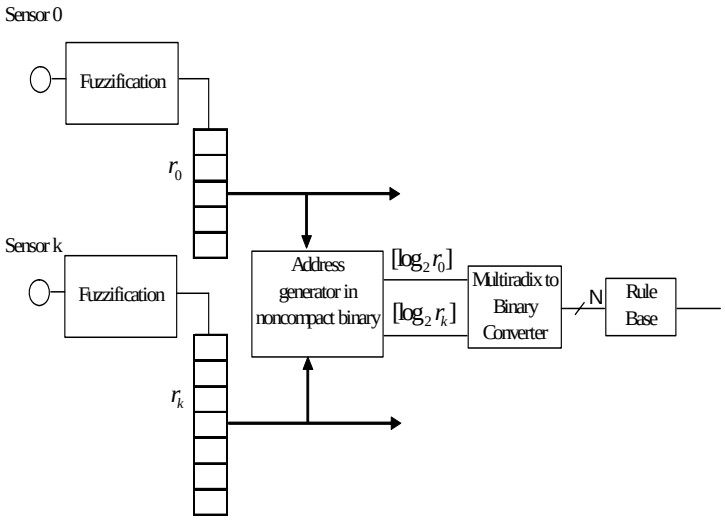

Fig. 9 Rule Base with Multi radix to Binary Converter

$$N = [\log_2 r_0 r_1 ... r_k] \le [\log_2 r_0] + ... + [\log_2 r_k] = N_0 \qquad (5)$$

Example:
Let $k = 9, r_0 = .. = r_9 = 5$. Then the number of initial addresses lines $N_0 = 30$.

The number of addresses lines after the Multi Radix to Binary Converter is equal to $N = [\log_2 r_k] = 24$

6. Data compression and error correcting codes in Rule Base

Data compression can be demonstrated by the following examples:

Example:
Suppose we have 2 Sensors $r_0 = r = 5, r = 5$. Normally, for this case it will take 15 bits for representing a single row as shown in Fig.10.

The string of numbers in the center row, as shown in Fig. 11, can be considered as a radix 5 number. Since this number has 5 digits (positions), the biggest possible number represented by this string is equal to $5^5 - 1$.

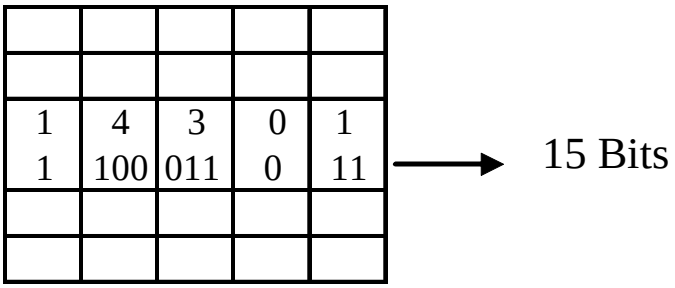

Fig. 10 Single Row 2 Sensors Representation

1	4	3	0	1

Fig. 11 Center Row

To convert it to binary we need $[\log_2 5^5 - 1] = 12$ bits. We saved 3 bits. These bits can be used for error correction

Example:

Suppose that we have 3 sensors, $r_0 = r_1 = r_2 = 5;\ r = 5$. In this case we have a number with 25 digits. The biggest possible number $5^{25} - 1$. To convert it to binary we need 59 bits. Therefore we saved 75-59=16 bits, see Fig. 12.

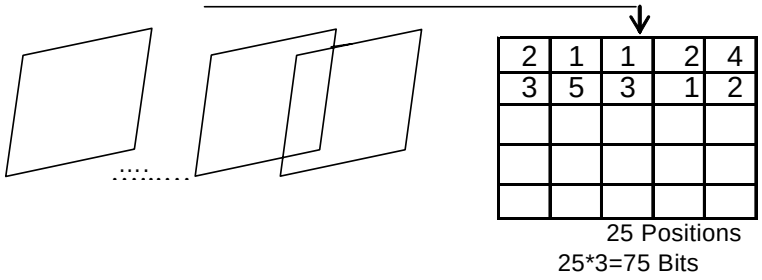

Fig. 12 3 Sensors representation

6.1 Rule Base with data compression and Error Detection/correction

The block diagram of the Rule Base with data compression and Error Detection/Correction is shown in Fig. 13

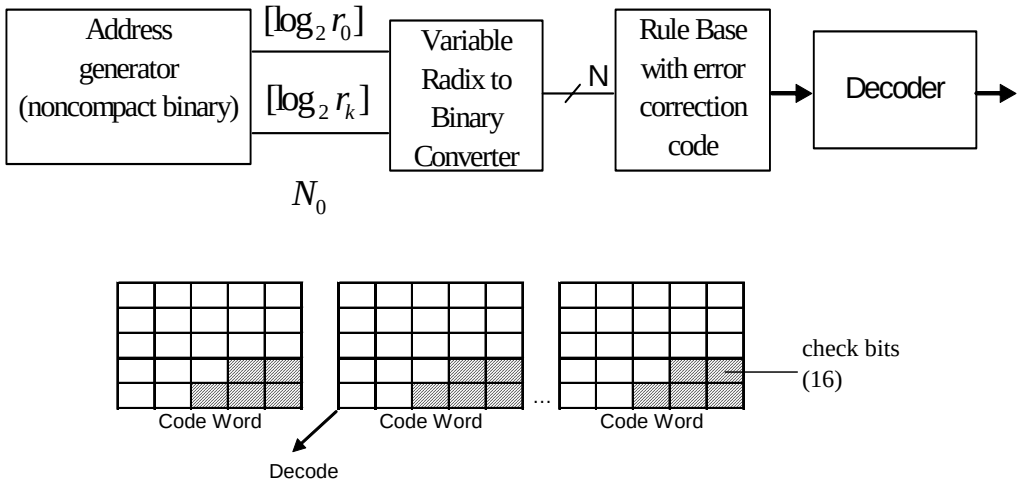

Fig. 13 Rule Base block Diagram with data compression and Error Detection/Correction

6.2 Testing of a Fuzzy Controller by signature Analysis of test Response

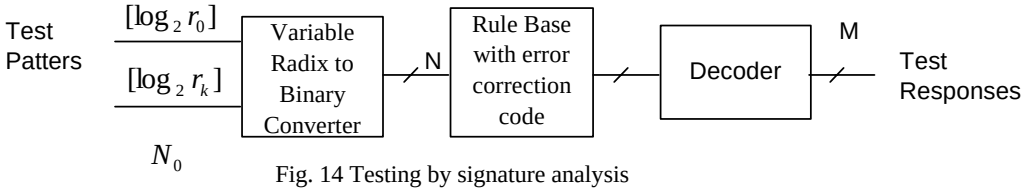

Fig. 14 Testing by signature analysis

Since initially we have N_0 address lines and after the decoder we have M address lines, we can consider a mapping

$$f : Z_2^{N_0} \longrightarrow Z_2^M \tag{6}$$

Note the Z_2^M can be considered as a field of 2^M elements $GF(2^M)$.

Error detection is performed the following way: first we precompute the signature

$$S_f = \sum \alpha_x f(x) \tag{7}$$

where $x \in Z_2^{N_0}, \alpha_x \in GF(2^M)$. S_f is to be considered fault-free.

The signature observed is computed with the same set of weights α_x

$$S_{\overline{f}} = \sum \alpha_x \overline{f}(x) \tag{8}$$

Here $\overline{f}$ is (possibly faulty) function $\overline{f} : Z_2^{N_0} \longrightarrow Z_2^M$ (9)

For testing if f and $\overline{f}$ are the same we compute the signatures. If the signatures S_f and $\overline{S_f}$ are the same, the test is passed.

7. Conclusion

In this paper we introduce a new representation of numbers – Variable Radix Number system. Using a Variable Radix Number system we decreased the number of addresses lines in a ROM that is used to store If-Then rules, thus reducing the space complexity of a fuzzy controller. Also we incorporated error correcting codes in the memory used to store If-Then rules without substantial increasing space complexity.

References

Braunsting R., J. Mujika, and J. P. Uribe, "A wall following robot with a fuzzy logic controller optimized by a genetic algorithm," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1995, pp. 77–82.

Gharieb W., G. Nagib "Fuzzy Intervention in PID Controller Design" in ISIE 2001, PUNSAN KOREA, pp 1639-1643.

Ibrahim A. S., "Nonlinear PID Controller Design Using Fuzzy Logic" in IEEE MELECOM 2002, pp 595-599.

Klir G. J., "Fuzzy Logic" in IEEE Potentials Oct/Nov 1995, pp 10-15.

Surmann H., J. Huser, and L. Peters, "A fuzzy system for indoor mobile robot navigation," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1995, pp. 83–88.

Wardana A. N. I., "PID-Fuzzy Controller for Grate Cooler in Cement Plant" in the *5th Asian Control Conf.* 2004, pp 1563-1567.