A Fuzzy-based Approach against Contradictions and Incompleteness within Requirement Specification

GERHARD H. SCHILDT, DANIELA KAHN
Institute of Computer-Aided Automation
Vienna University of Technology
Treitlstr. 1., 1040 Vienna
AUSTRIA

Abstract: - One of the most severe problem due to software-development is to guarantee that there are no contradictions and no incompletion within a certain requirement specification. We did a new approach how to handle this problem, because we are especially engaged in fuzzy controllers in real-time Systems. The presented method is limited to rule bases as a part of fuzzy systems. The fundamental idea is to calculate all possible rules within a rule base automatically, and afterwards, reduce the number of rules with backtracking strategies and contradiction checks. Additionally, an analysis of rules by a human expert can be done.
First of all our contribution introduces the fundamental problem, and then describes the method of calculating all possible rules. Furthermore it is shown how to reduce the number of rules by backtracking and contradiction checks. An application for a fuzzy controller is shown. Normally this fuzzy controller has two input variables: *error* and the *variation of error.* The number of possible rules depends on the number of variables as well as the number of fuzzy sets for each input and output variable. It is shown how the reduction of number of rules can be done. It can be seen that this approach cannot be used to detect conflicts due to timeliness in real-time systems, but it can be a useful approach to detect operational system states that otherwise would not be considered.

Key-Words: - fuzzy logic control, rulebase limitation, backtracking, contradiction check.

1 Introduction

One of the most severe problems due to software development is to guarantee that there are no bugs within a certain specification. Because just those bugs within a specification will cause a lot of manpower effort to correct it, or in worst case, to redesign a software system. This problem was recognized and as reaction the so-called water-fall model was introduced in the field of software engineering.

2 Estimation of the maximal rule number

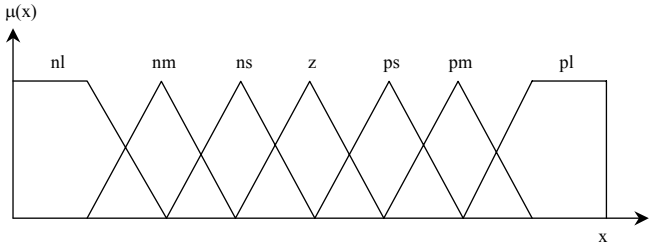
Let us assume that a fuzzy system has ***n*** input variables (e.g.: *error, variation of error*, and o*ther input variables*), every variable may be described by $m_i (i = 1...n)$ fuzzy sets for input variables and y output variables. Every output variable may be described by $z_j (j = 1...y)$ fuzzy sets. Furthermore fuzzy operators like ***AND, OR, NAND*** and ***NOR*** can be used to describe a single rule like as an example

IF a=… AND b=… OR c=… NOR d=… NAND e =… **THEN** conclusion .

Then the upper boundary of the number of fuzzy rules can be determined as

$$\boldsymbol{R} = \prod_{i=1}^{n} m_i \cdot 4^{(n-1)} \cdot \prod_{j=1}^{y} z_j \quad . \qquad (1)$$

The complete amount of rules can be generated, automatically. This estimation can be done if in the phase of system analysis one can be sure that no input or output variable was forgotten. Figure 1 shows as an example a description of a variable by a certain number of fuzzy sets.



Legend:
n{l,m,s} = negative large/middle/small
z = zero
p{l,m,s} = positive large/middle/small

Fig. 1: Membership functions

3 Analysis of complete rule base

The next step is to reduce the number of rules by different methods. We have to look for efficient procedures to reduce the maximum number of rules.

3.1 Computer-Aided analysis using backtracking

Let us assume the following type of rule as an example

IF a=… NAND b=… OR c=… AND d=…
THEN conclusion

and analysis found out that the condition IF a=… NAND b=… never will be fulfilled, one can neglect all other possible rules containing this expression as a part of premise. We assume that a considerable part of potential rules may be neglected due to this fact.

3.2 Contradiction check

A future model of technical process should enable a so-called contradiction check, i.e. it does not make sense for a heating controller to put heating power to maximum if sensors measured that actual temperature is too high.

3.3 Inspection of rule base by human expert

Applying *backtracking* and *contradiction check* should reduce the potential number of rules, but if the expected reduction rate is not high enough, one may try to reduce number of rules by inspection of rule base done by a human expert. This procedure may be based on *clustering analysis* within a rule base. That means, for sufficient operation of a fuzzy controller the rule base presented as a matrix should show certain *clustering effects*, otherwise the rule base is not appropriate to the technical process. Furthermore, if one selects one rule as a certain element within the rule base matrix, neighboring elements should contain neighboring rules only (see Fig. 2); otherwise those rules have be checked, especially.

We have to grant that this analysis is not a compelling evidence for the correctness of a rule base but at least it produces certain indications for the correctness of the contents of the rule base. Thus, it should not applied for safety-related process control. Nevertheless, this method assists a system specialist in designing a hardware-/software system.

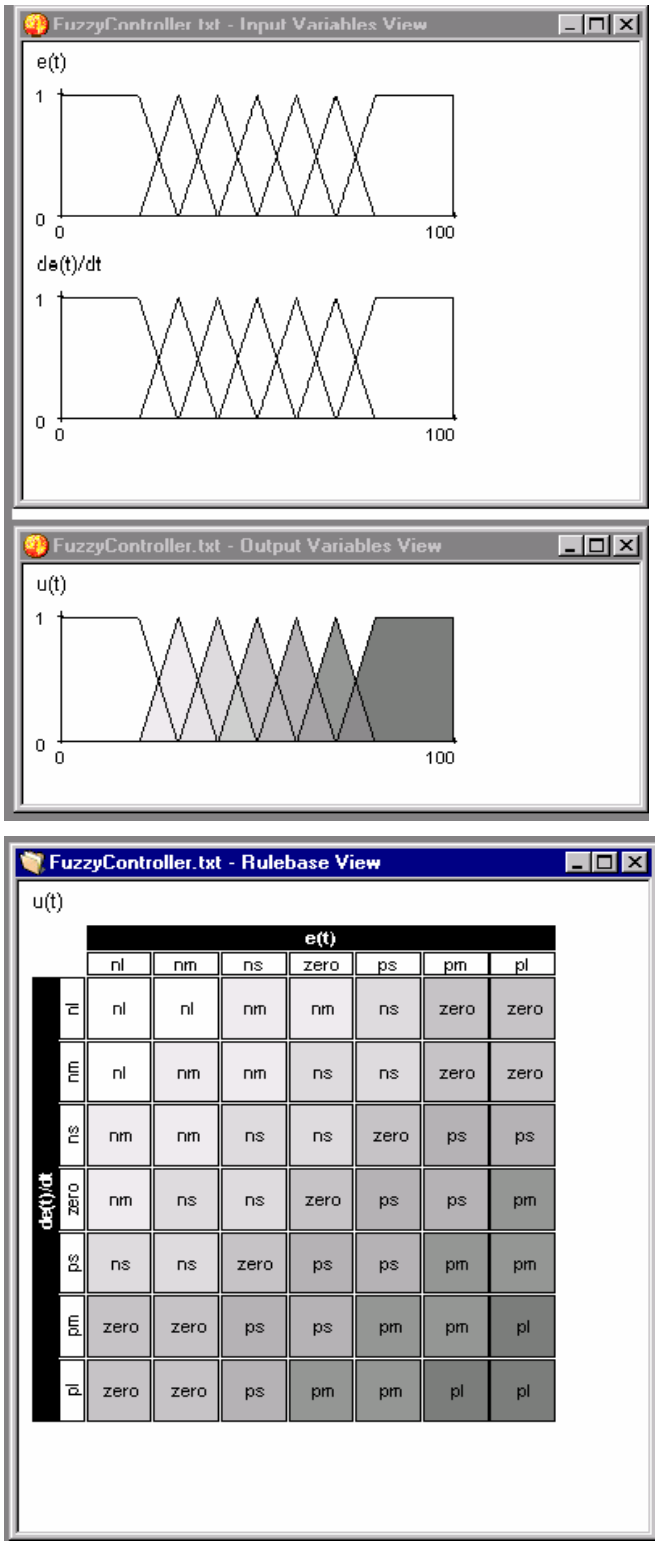
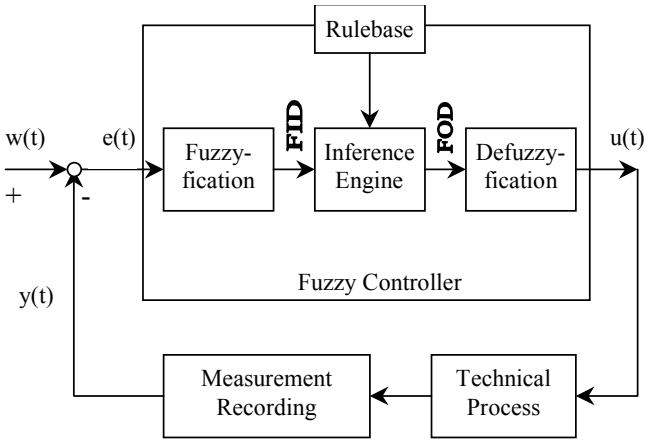


Fig. 2: Input variables e(t), de(t)/dt, output variable u(t), and rule base matrix

4 Single loop feedback system

As an example we will present a single loop feedback system as a fuzzy controller to demonstrate the method.



Legend:

variable	type of variable
w(t):desired value	analogue value ("crisp" value)
e(t)=w(t)-y(t)=error	analogue value
u(t):control signal	analogue value
y(t):measured value	analogue value
FID: Fuzzyfied input data	fuzzy value
FOD: Fuzzyfied output data	fuzzy value

Fig. 3: Fuzzy controller

At first, let us discuss a simple fuzzy controller comprising two input variables like *error* and *variation of error* with only one output variable (conclusion), which is a usual controller application, and let us assume that linguistic description comprises e.g. 7 fuzzy sets for each input and output variable, then we can calculate the maximum number of rules as R = 1.372 rules considering 4 possible fuzzy operators (AND, OR, NAND, and NOR). Rules have the following structure:

IF a = … AND b= … THEN conclusion,
IF a=… OR b=… THEN conclusion,
IF a=… NAND b=… THEN conclusion,
or
IF a=… NOR b=… conclusion

Then the maximum number of rules can be calculated as max number of rules:

$$R = (7*7) * 4 * (7) = 1.372 \text{ rules.} \quad (2)$$

This calculation bases on the following assumptions:

Input variables:
- Error (7 fuzzy sets)
- Variation of error (7 fuzzy sets)

Output variable:
- Variable Heater (7 fuzzy sets)

One of fuzzy operators (AND, OR, NAND, NOR) shall be applied.

Although we found out that number of potential rules increases more than exponentially, for usual fuzzy-based PID-controllers with a limited number of variables and fuzzy sets for every variable the maximum number of rules ***R*** is not so huge that all rules can be checked in order to reduce the number of necessary rules.

An essential module of a fuzzy controller is the *rule base* comprising a certain number of rules. It was found that the number of rules necessary for efficient control of a technical process is limited to a certain upper boundary. For example, about 80 rules are sufficient to control temperature and steam pressure of a conventional reactor. Contents of the rule base is a certain modelling of technical process and looks like the following source code section:

FIU Source Code

```
$ FILENAME: temp/temp3.fil
$ DATE:                09/18/2000
$ UPDATE:     09/23/2000

$ Temperature controller: Three inputs, two outputs
$ INPUT(S):    Error, Var(iationOf)_Error
$ OUTPUT(S): Var(iationOf)_Heater

$ FIU HEADER
fiu tvfi (min max)*8;

$ DEFINITION OF INPUT VARIABLE(S)
invar Error " " : -1.0 () 1.0 [
P_Large                          (@0.6, 0, @1.0, 1)
P_Medium                (@0.3, 0, @0.6, 1, @1.0, 0)
P_Small                 (@0.0, 0, @0.3, 1, @0.6, 0)
Zero                    (@-0.3,0, @0.0, 1, @0.3, 0)
N_Small                 (@-0.6,0, @-0.3,1, @0.0, 0)
N_Medium               (@-1.0,0, @-0.6,1, @-0.3,0)
N_Large                          (@-1.0,1, @-0.6,0)

];
invar Var_Error " " : -1.0 () 1.0 [
P_Large                          (@0.6, 0, @1.0, 1)
P_Medium                (@0.3, 0, @0.6, 1, @1.0, 0)
P_Small                 (@0.0, 0, @0.3, 1, @0.6, 0)
Zero                    (@-0.3,0, @0.0, 1, @0.3, 0)
N_Small                 (@-0.6,0, @-0.3,1, @0.0, 0)
N_Medium               (@-1.0,0, @-0.6,1, @-0.3,0)
N_Large                          (@-1.0,1, @-0.6,0)

$ DEFINITION OF OUTPUT VARIABLE(S)
outvar Var_Heater " " : -1.0 () 1.0 * (
P_Large        = 0.8
P_Medium    =  0.4
P_Small        = 0.2
Zero           = 0.0
```

```
N_Small      = -0.2
N_Medium     = -0.4
N_Large      = -0.8
);
$ RULES

if Error  is P_Small  and Var_Error is N_Medium
                      then Var_Heater is N_Medium;
if Error  is P_Small  and Var_Error is N_Small
                      then Var_Heater is N_Small;
if Error  is P_Small  and Var_Error is N_Large
                      then Var_Heater is N_Small
if Error  is P_Small  and Var_Error is N_Medium
                      then Var_Heater is N_Small;
if Error  is P_Small  and Var_Error is N_Small
                      then Var_Heater is Zero;
if Error  is P_Small  and Var_Error is N_Large
                      then Var_Heater is Zero;
if Error  is P_Small  and Var_Error is N_Medium
                      then Var_Heater is Zero;

if Error  is P_Large  and Var_Error is P_Small
                      then Var_Heater is P_Medium;
if Error  is P_Large  and Var_Error is P_Large
                      then Var_Heater is P_Large;
if Error  is P_Large  and Var_Error is P_Medium
                      then Var_Heater is P_Large;
if Error  is P_Large  and Var_Error is P_Small
                      then Var_Heater is P_Large;
if Error  is P_Large  and Var_Error is P_Large
                      then Var_Heater is P_Large;
if Error  is P_Large  and Var_Error is P_Medium
                      then Var_Heater is P_Large;
if Error  is P_Large  and Var_Error is P_Small
                      then Var_Heater is P_Large;
```

Fig. 4: Rule base of a conventional thermal reactor (example)

There are fundamental advantages in modelling the technical process by such a rule base:

- The contents of the rule base can be validated by an expert such as an experienced operator, who has a lot of technical knowledge of the process, he is responsible for, but has no deeper knowledge on computer science.
- The real-time behavior can be judged better than for conventional PID controllers. This is because the real-time behavior may be tuned by the modification of
- contents of the rule base by editing certain rules;
- certain membership functions;
- certain scaling factors during the fuzzification process;
- the inference algorithms.

For a limited number of input and output variables as well as a limited number of fuzzy sets for each variable the maximum number of possible rules can be generated, automatically.

5 Conclusions

In software-driven systems we always have the severe problem that software will never be error-free. Either there are *contradictions* or *incompleteness* within the specification, so that we never can be sure of error-free software. Our contribution presented that there exists an upper boundary for the number of possible rules for a fuzzy controller with a limited number of input and output variables. Furthermore, the number of fuzzy sets to describe a certain variable should be limited, too. The total set of possible rules can be generated, automatically. Afterwards one can reduce the total number of rules by *backtracking*, *contradiction check* and *special inspection* of rule base in order to avoid contradictions and incompleteness within requirement specification. At that time we are on the way to develop appropriate tools to assist these checks.

Acknowledgement

The authors wish to thank Prof. Dr. Kastner, who is very engaged in the field of fuzzy process control, for helpful discussions.

References:

[1] C. H. Jung, et al., Fuzzy Controller for the Steam Generator Water Level Based on Real-time Tuning Algorithm, *ANS-Meeting*, Pennsylvania, 1996.

[2] L. van den Durpel, D. Ruan, Fuzzy Model Based Control of a Nuclear Reactor, *FLINS94*, 1994

[3] A. A. Averkin: Fuzzy Logic Acquisition and Simulation Modules for Expert Systems to Assist Operator´s Decision for Nuclear Power Stations, *FLINS94*, 1994.

[4] C. H. Chou, H. C. Lu, A heuristic self-tuning fuzzy controller", *Journal Fuzzy Sets and Systems*, 1996.

[5] G. H. Schildt, A Fuzzy Controller for NPPs, *FLINS96*, 1996.

[6] Y. J. J.Buckle, Stability and the fuzzy controller, *Journal Fuzzy Sets and Systems*, Vol. 77, 1996.

[7] Y. G. Oh, et al., Application of Fuzzy Logic for Monitoring and Diagnosis of Loose Parts in Nuclear Power Plants, *IASTED Pittsburgh*, 1998 USA.

[8] Aptronix Inc., FIDE application, Note 001-920727, 1992 USA.

[9] W. Kastner: Yet Another Fuzzy Logic Control Shell, *TEMPUS Modify'97*, pp. 125-131, 1997.