FUZZY-FORTH RULE BASED PRODUCTION SYSTEM FOR REAL TIME CONTROL SYSTEMS

L.E. Borges da Silva G.-E. April G. Olivier

Electrical Engineering Department
Ecole Polytechnique de Montreal
P.O. Box 6079 - station "A"
Montreal , Canada H3C 3A7
Tel.: (514) 340-4877

ABSTRACT

A fast and efficient FORTH based production system capable of working in a FUZZY environment is described. This production system provides a practical means for structuring knowledge applicable to complex real time control systems. The ordinary meaning of a conditional statement, i.e., a Boolean condition-action pair, was extended in order to cope with the concept of membership function, which allows managing fuzzy variables. A fuzzy variable assumes so called "linguistic values", such as " absolutely large", "small and negative" or even "near zero" instead of numerical values. The fuzzy set approach provides a "natural" means for structuring knowledge applicable to the intelligent solution of engineering problems.
The FUZZY-FORTH production system divides the knowledge in two parts: the "passive rule set" which contains all the available knowledge, and the "active rule set" which contains part of the "passive rule set" and will produce meaningful inferences. The "active rule set" can be up-dated at any time by the rules themselves. This versatility allows the implementation of control systems with several levels of intelligence, which is very useful in systems with real-time learning capability.
The system runs on an in-house version of FORTH, implemented on a Motorola 6809 microprocessor.

INTRODUCTION

FUZZY-FORTH (a Rule-based Production System), was designed to cope with the concept of "linguistic variables" introduced by L.A. ZADEH [ZAD73].
The Fuzzy Set theory is a relatively new concept which allows qualitative information to be represented mathematically and handled in a completely rigorous manner. The basic difference between the ordinary notion of a set and that of a fuzzy set can be seen in the following exemple.
Suppose a variable which represents the temperature of a certain room, on the closed interval [0°C, 40°C]. Further suppose that the measure is in the form of "about 20°C" or "greater than 25°C". An ordinary set which defines this can be expressed in terms of a membership function, μ, which can take values of either 0 or 1. If $\mu(T)=0$ then the temperature T is not a member of the set, otherwise if $\mu(T)=1$ then it is (fig.1.a).

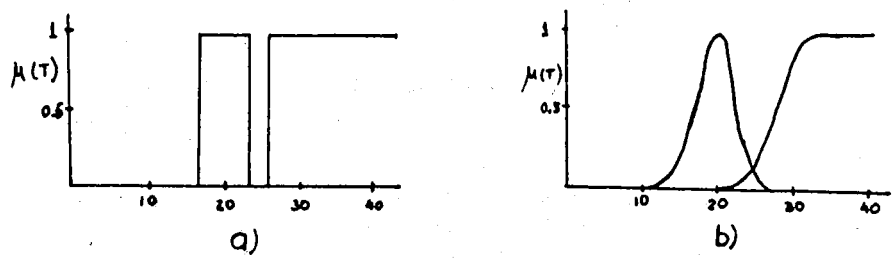

fig.1

A fuzzy set which represents the same idea has a membership function which take all values between 0 and 1 (fig.1.b). The ordinary set is deterministic and precise, having a definite transition from membership to non-membership. The fuzzy set, on the other hand, allows the qualitativeness of the measure to be reflected in a gradual membership transition. This means, that qualitative information can be represented mathematicaly and handled completely rigorously. Fuzzy sets may be combined in the same way as ordinary sets, using the same operators.
The union of two fuzzy sets is defined by the membership function

$$\mu_{A+B}(x) = \max (\mu_A(x) , \mu_B(x))$$

The intersection of two fuzzy sets is defined by

$$\mu_{A.B}(x) = \min (\mu_A(x) , \mu_B(x))$$

The negation is defined by

$$\mu_{\bar{A}}(x) = (1 - \mu_A(x))$$

We can notice that the intersection of these two measures produces a null set in the non-fuzzy case, but not in the fuzzy case. This is the basic property of the fuzzy concept. Fuzzy measures are not necessary mutually exclusive, providing information even in disjoint situations. This form of representation of the information is closer to day to day realities.
Using these concepts, we can define the "fuzzy conditional statements" [ZAD73] in the form

IF (fuel input is high) THEN (temperature is greater than 25°C)

The fuzzy conditional statement is a two dimensional fuzzy set in the product space of fuel and temperature, given by

$$\mu_R = \mu_{AxB} = \min (\mu_A(F) , \mu_B(T))$$

Naturally the behaviour of an entire process requires several fuzzy conditional statements (or rules) to be modelized. Each rule of a set of rules is evaluated separately through the MIN operator and the results are combined using the MAX operator. To simplify the explanation, let us suppose the actions are deterministic . Thus, given a set of rules, the membership of each rule is evaluated and action is taken according to the rule with the maximum value of membership. This approach doesn't eliminate the possibility that the action part of the rule also be described in terms of fuzzy sets,

nor the possibility that each rule participates in a manner proportionnal to its own pertinence or weight. That depends on which fuzzy structure is being implemented, and FUZZY-FORTH was designed to permit these implementations, with very few modifications.

FUZZY-FORTH RULE BASED PRODUCTION SYSTEM

FUZZY-FORTH was designed to edit and run fuzzy conditional statements as fast and efficiently as possible, using all the possibilities of FORTH.
FUZZY-FORTH can be divided into two main structures, i.e. the rule editor and the Inference Engine.
The FUZZY-FORTH rule editor is contained in a set of FORTH defining words to help construct the rules.
A rule is defined in the form :

```
RULE: (name) IF (variable) IS (fuzzy label) OR (fuzzy label) ... AND
             (variable) IS (fuzzy label) OR (fuzzy label) ... AND
             .......
             (variable) IS (fuzzy label) OR (fuzzy label) ...
        THEN .... (actions) .... END-RULE
```

Once created, a rule has a name like any other FORTH word, and is placed into the FORTH dictionary, being part of what we call the "passive rule set", i.e. the knowledge base which contains all the available knowledge.
Rules, however, differ from other FORTH words in the fact that they have three CFAs, one in the usual position, and the other two being called the conditionnal-CFA and the action-CFA. When a rule's name is executed the effect is to include its conditionnal-CFA in the "active rule set" which contains the set of all active knowledge. The structure of the active rule set is such that it looks like an ordinary FORTH word, with the conditional-CFA's of all active rules sequentially listed. This allows the highly efficient FORTH inner interpreter (NEXT) to be used as the actual Inference Engine. When the active rule set is executed, all the conditional-CFA's it contains are executed and then, usually, the action-CFA of the most relevant rule is executed. The active rule set can be modified at any time, i.e. rules can add or remove other rules or themselves, or even reinitialize the active rule set. This versatility is made possible by the addition to FUZZY-FORTH of a few words like INCLUDE, EXCLUDE, INIT-IE, RUN-IE, which do this job.
Another very important function implemented into FUZZY-FORTH is the "modifier", which allows to change the fuzzy sets in a certain way. These modifiers have names like "very", "not", "extremely", etc. When they appear before a fuzzy set, they change its characteristics function in a specific way. The modifiers are defined by

```
MOD: (name) ( operation ) ;MOD
```

Defined in this way, they may then be introduced in a rule like

```
....(variable) IS (modifier) (fuzzy label) OR .....
```

The MOD: ;MOD structure is designed in such way that several modifiers can be placed before the fuzzy label (for instance IF variable IS not very hot THEN open the valve).

DICTIONARY STRUCTURE OF THE RULE

When a rule is being written, the effect is to place all information about this rule into the dictionary, at places reserved for that by the immediate words defined to do this job. FUZZY-FORTH defines a rule in the same way that other word is defined in FORTH, and places it into the dictionary.
For a very simple rule like

```
RULE: Rule1 IF Temperature IS Not Very High THEN close the window END-RULE
```

the dictionary structure is shown in fig.2 .

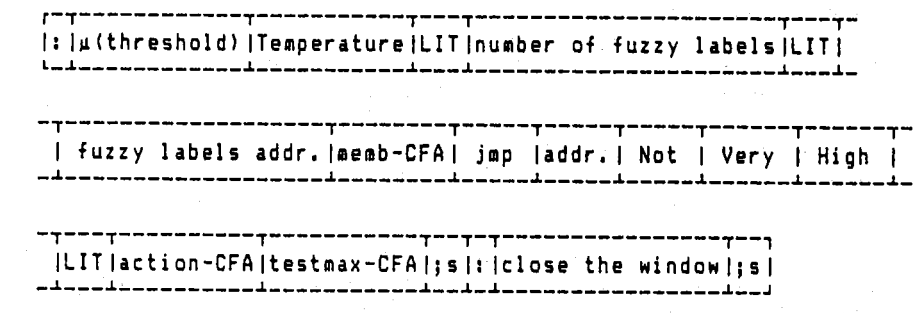

fig.2

The immediate defining word RULE:, creates a special kind of word (a rule) with three CFAs, which we shall call respectively Rule-CFA, Condition-CFA and Action-CFA. The "<builds"'s part of this immediate defining word compiles the Rule-CFA and Conditional-CFA in the dictionary, and passes into compile mode to begin compiling the rest of the rule. The "does>"'s part executes the word INCLUDE which will be commented later.
The immediate word IF places into the dictionary the CFA of the function that will initialize the threshold value of membership.
The immediate word IS prepares places in the dictionary to receive information about the number of modifiers and fuzzy labels and where they are placed. The connective OR does nothing, and is only there to facilitate the comprehention of a written rule . The immediate word AND does the necessary calculations and adjusts the values initialised by the word IS, and places into the dictionary the CFA of the function which will calculate the membership for one rule (memb). The immediate word THEN does the same thing as AND and then prepares the dictionary to receive the action part of the rule, also compiles the CFA of the function which will calculate the maximum membership for all rules (testmax) . The immediate word END-RULE just compiles a ";" to end the rule after checking the validity of the compilation.
At this point with a little reflection, we can see that a rule can be written without condionnals, being made up only of actions which will allways be executed. This feature is very important in real-time control systems where the variables or variations of the variables have to be calculated just before running a set of rules. (Note that our in-house version of FORTH is case sensitive,which allows us to distinguish the fuzzy conditionals "IF"

"THEN" and operators "OR" "AND" from the regular FORTH conditionals "if" "then" and operators "or" "and").

DICTIONARY STRUCTURE OF THE ACTIVE RULE SET

The active rule set is structured like a ordinary FORTH word, where the Rule-CFAs are sequentialy aligned, ending by a ";s" (fig.3).

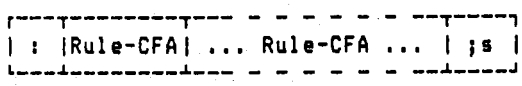

fig.3

The idea to structure the active rule set in such way, comes from the fact that in real-time controllers a looping engine is usually utilized to draw meaningful inferences. So the very efficient FORTH inner interpreter (NEXT) can be utilized as the Inference the Engine, reaching the limit speed of the language. When a rule's name is executed, it calls the FORTH word INCLUDE which looks for the end of the active rule set, places the new Rule-CFA at this place and moves the ";s" forward one position. This procedure determines a natural priority for the rules (although this is normally of little consequence in fuzzy controllers).
The word EXCLUDE, utilised as follows,

```
... EXCLUDE (rule's name) ...
```

looks for the positon of a Rule-CFA in the active rule set, takes it out, and reorders the entire active rule set. This word can be used as part of a certain action or by the designer directly from the keyboard, helping development. Another FUZZY-FORTH word that plays with the active rule set is RUN-IE, which just initialise the maximum membership of the rules, and using the standard FORTH word @EXECUTE, executes the active rule set.

CONTROL SYSTEM APPLICATION

FUZZY-FORTH was applied to control a asymmetrical four quadrant AC to DC power converter [OLIV84]. The control system is based on a three level hierarchical structure. The first level comprises the super-fast protection routines against all types of over-load and fault conditions. These routines are written directly in FORTH assembler and are always active. At this level we also find the short term controller which automatically handles the rapid transients due to source variations and the unpredictable nature of the load. This controller is described by fuzzy implications that handle the fuzzy model [TAK85] of the power converter . This level represents the "mechanical reflexes" of the system. The second level evaluates all the system parameters that the lower level needs to work properly. At this level fuzzy automatons are designed to adjust the system's output, or study the validity of certain operating points. These fuzzy automatons are also described in terms of fuzzy implications [WEE69]. This level represents the "skills" of the system. The third level, also a fuzzy automaton, looks at the overall behaviour of the system and instead of acting directly in the system's variables, activates the appropriate lower level routines to realise a precise task in order to

achieve the desired goal. This level represents the "intelligence" of the system. So the entire control system can be described as a composite fuzzy automaton and is completely implemented by FUZZY-FORTH.

CONCLUSION

The extensibility and speed of FORTH have been exploited to create a production system capable of real time control based on fuzzy set theory. The required speed was achieved by separating the knowledge base in two parts, namely the passive rule set, and the active rule set. The passive rule set encompasses all knowledge encoded in the system, while the active rule set is a subset of the latter, structured like a regular FORTH word, so that it can be executed repeatedly and rapidly by the FORTH inner interpreter. Obviously, for the system to evolve in a usefull manner, at least one of the rules in the active set must provide a means to change some or all of the said active rule set. To accomplish this, new constructs were needed in FORTH, namely "defer:", "define", ";define" and "undefine". These allow a rule to refer to as yet undefined words and yet produce no ill-effects when the rule is tested. The words declared using "defer:", can be defined later and all past references are automaticaly updated to the new definition. This allows the convenience of top-down programming, essential when one wishes to program a Fuzzy Automaton, which is a basic mathematical structure, at the lower levels of a multi-level control system.
The system was realized using an in house version of FORTH, which can run on a small developement system and allows compiled programs to be burned into ROM and run unmodified on a target board, allowing dedicated, turn-key control systems.

BIBLIOGRAPHY

[ZAD73] L.A.Zadeh - Outline of a New Approach to the Analysis of Systems and Decision Processes. IEEE Trans. SMC-3-1 , 28/44 (1973)

[WEE69] W.G.Wee, K.S.Fu - A Formulation of Fuzzy Automata and its Application as a Model of Learning Systems. IEEE Trans. SSC-5-3 , 215/223 (1969)

[TAK85] T.Takagi, M.Sugeno - Fuzzy Identification of System and Its Application to Modeling and Control. IEEE Trans. SMC-15-1 116/132 (1985)

[OLIV84] G.Olivier, G.E.April, L.E.B.da Silva - Analysis and Control of Four Quadrants Single Transformer Phase Controlled Converter. IEEE-IAS-1984 Annual Meeting - Conference Record