FUZZY SYSTEMS

There are many ways of representing knowledge in general; we will consider here only the very basic aspects of knowledge representation in a fuzzy expert system. Most basic is the representation of data. Next is the idea of representing knowledge about reasoning processes, usually represented in fuzzy expert systems by fuzzy production rules which are discussed later in this article. Fuzzy expert systems add two major elements to knowledge representation in nonfuzzy expert systems: (1) the addition of basic data types not found in conventional systems and (2) an expanded rule syntax which facilitates reasoning in terms of words rather than numbers and permits approximate numerical comparisons. We will first consider the representation of data.

However, let us first explain the basic differences between fuzzy expert systems and fuzzy control. Both systems have a fuzzy rule base, where fuzzy expert systems are designed to model human experts in the areas of decision making and fuzzy control models human operators in control of a process. In fuzzy control the inputs to the fuzzy rule base are usually real numbers, representing measurements on the process, and the outputs are also real numbers representing how to change certain variables in the process to achieve better performance. In fuzzy expert systems the inputs are real numbers, character strings, or fuzzy sets, representing data on the decision problem, and the outputs are real numbers, character strings, or fuzzy numbers representing possible actions by the decision makers. Fuzzy control answers the question of "how much" to change process variables and fuzzy expert systems answer the question "what to do?" or "what is it?" Since both systems are based on a fuzzy rule base, some techniques useful in fuzzy control (discussed below) are presented because they are also useful in fuzzy expert systems.

Basic data types available for individual data items include numbers and character strings. Some nonfuzzy expert system shells include with the basic data types a measure of the confidence that the value for the data is valid. Individual data items are usually collected into named groups. Groups may be simple structures of several data items, lists, frames, or classes.

Fuzzy expert systems include additional data types not found in conventional expert systems: discrete fuzzy sets, and fuzzy numbers. A fuzzy set is similar to an ordinary set (a collection of objects drawn from some universe), with one major difference: to each member of the set is attached a grade of membership, a number between zero and one, which represents the degree to which the object is a member of the set. In probability theory, all the probabilities must add to one. However, in fuzzy set theory, all the grades of membership need not add to one.

In general, the members of discrete fuzzy sets are words describing some real-world entity. Table 1 contains two examples of discrete fuzzy sets. Fuzzy set speed describes a numeric quantity. In Table 1 we see that speed could certainly be described as Medium, but could also be described as Fast with almost equal certainty. This represents an ambiguity. Ambiguities such as these need not be resolved, because they add robustness to a fuzzy expert system. Also in Table 1, discrete fuzzy set "fault," whose members are words describing different possible faults, describes a nonnumeric categorical quantity. We are certain that the fault is not in the fuel system; it is probably in the ignition or electrical systems, although it just might be in the hydraulic system. Since these categories are mutually exclusive, we have not an ambiguity but a contradiction. Unlike ambiguities, contradictions must be resolved before our program is done.

Table 1. Discrete Fuzzy Sets

Speed		Fault	
Member	Membership Degree	Member	Membership Degree
Stop	0.000	Fuel	0.000
Slow	0.012	Ignition	0.923
Medium	1.000	Electrical	0.824
Fast	0.875	Hydraulic	0.232

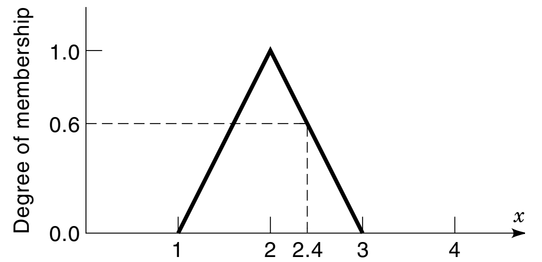

Figure 1. A fuzzy set describing uncertainty that a value x is equal to two.

Fuzzy numbers, like statistical distributions, represent uncertain numerical quantities. Fuzzy numbers may have any of several shapes; most common are piecewise linear (triangles and trapezodoids), piecewise quadratic (s-shaped), and normal (Gaussian). A typical fuzzy number is shown in Fig. 1. From Fig. 1 we see that the confidence that 2.4 belongs to "fuzzy 2" is 0.6.

Fuzzy production rules usually are of the IF . . . THEN . . . types. The IF part of the rule, called the *antecedent,* specifies conditions which the data must satisfy for the rule to be fireable; when the rule is fired, the actions specified in the THEN part, the *consequent,* are executed. Some simple fuzzy antecedents follow.

IF size is Small AND class is Box, THEN . . .

In this antecedent, size and class are discrete fuzzy sets; Small is a member of size and Box is a member of class.

IF weight is about 20, THEN . . .

Here weight is a scalar number. "About" is an adjective (called a *hedge*) which modifies the scalar 20, converting the scalar 20 to a fuzzy number. The resulting fuzzy number "about 20" could be triangular (as in Fig. 1) with base on the interval [16, 24] and vertex at 20. The comparison between weight and "about 20" is an approximate one, which can hold with varying degrees of confidence depending on the precise value for weight.

IF speed is Fast AND distance is Short, THEN . . .

This apparently simple antecedent is a little more complex than it appears, since speed and distance are scalar numbers, and Fast and Short are members of discrete fuzzy sets describing speed and distance. Fuzzy numbers are used to define Fast and Short. When the truth value of the antecedent is evaluated, the number speed must be converted into a grade of membership of Fast, and similarly for distance and Short. This conversion is called *fuzzification,* and it requires that fuzzy numbers be assigned to each fuzzy set member. Membership functions for the members of the discrete fuzzy set distance are shown in Fig. 2.

This type of rule antecedent is so common in fuzzy control that a shorthand notation has evolved; Short, Medium, and Long are considered simply fuzzy numbers, the clause "distance is Short" is considered a comparison for approximate equality between distance and Short, and the fuzzy set of which Short, Medium, and Long are members may not even be named.

In evaluating the truth value of an antecedent, the confidences that the individual clauses are true and that the rule itself is valid are combined. We are using "truth value" and "confidence" to mean the same thing. Confidence is discussed more in the section on uncertainty. The most common way of doing this is to use the Zadeh operators: truth value of A AND B = min(truth values of A and of B), truth value of A OR B = max(truth values of A and of B). There are many other combination rules. If the antecedent truth value exceeds an assigned threshold value and the rule is enabled for firing, the rule is said to be fireable.

The THEN part of a rule is called the *consequent,* and it consists of instructions to be executed when the rule is fired. The truth value with which the consequent is asserted is usually the truth value of the antecedent. Consequent actions may be for input or output of data, information to the user, and the like, but the most important consequent actions are those which modify data. Some consequent actions are described below.

action: write 'Hello, World!'

action: read x

The above two instructions need no explanation.

$$\text{action: } MA = \frac{MA + X}{T + DT}$$

Here a new value for *MA* is computed from values for variables assigned in the rule antecedent. The confidence in the new value for *MA* will be the antecedent confidence. The system may provide that the old confidence value not be overwritten unless the new value has confidence greater than (or possibly equal to) the old confidence value.

action: class is Desk

Here the confidence in member Desk of fuzzy set class will be set to the antecedent confidence, assuming that the new confidence value exceeds any old confidence value.

action: z is Small

In fuzzy control systems, z would be a scalar number. In fuzzy reasoning systems, z is likely to be a discrete fuzzy set, of which "small" is a member. One or the other syntaxes may be used depending on the particular fuzzy expert system shell being used. The effect of this action is to store the antecedent confidence as the grade of membership of "small". If z is a scalar number and Small is a fuzzy set member with its corresponding fuzzy number, as is the case in fuzzy control, this consequent operation is complex. To begin with, there are almost invariably other rules fired concurrently with other consequents such as z is Medium, z is Large, and so forth; the consequent z is Small cannot be applied except together with the other rule consequents. Now the inverse operation to fuzzification, called *defuzzification,* must be carried out. There are many ways of defuzzification used by fuzzy control engineers, which are beyond the scope of this article. Two ways of defuzzification are now described.

If the fuzzy numbers corresponding to Small, Medium, and Large are singletons (i.e., only a single value has a nonzero grade of membership), defuzzification can be very simple; a weighted average of these values can be taken, using the confidences in Small, Medium, and Large assigned by the rules as weights.

If the fuzzy numbers are not singletons, the conclusions of all rules which have fired are then aggregated (unioned) to get the fuzzy conclusion of the whole system; it is then defuzzified to a real number. In control applications, this number may then output to the controller or be used to compute the new control value; in fuzzy reasoning applications, the number may be output to the user or used as input to succeeding reasoning stages.

We have only discussed one rule; a fuzzy expert system usually has multiple blocks of rules (rules grouped together to perform a certain job) and a network of blocks of rules. With many blocks of rules the output from certain blocks become input to other blocks of rules.

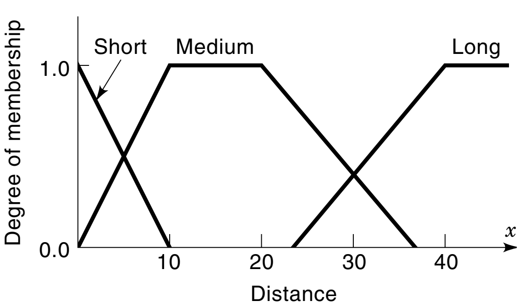

Figure 2. Three fuzzy sets describing uncertainty that a distance is Short, Medium or Long.

KNOWLEDGE ACQUISITION

Knowledge acquisition is the translation of information about a system into fuzzy production rules and expert data bases which are to model the system. Classically, this is done by a domain expert, thoroughly familiar with the application field, and a knowledge engineer, familiar with artificial intelligence techniques and languages. In one of the first fuzzy expert systems (1) the authors generated fuzzy rules like "If size is Large and vertical position is Low and region touches the border, then class is Lung." These directly represent how an expert classifies regions seen in a medical echocardiogram, with Lung a member of the fuzzy set of region classifications.

The fuzzy expert system is to automatically classify the regions. However, the linguistic variables "Large," "Low," and so on, are all defined by fuzzy numbers. Unlike control applications, a well-written fuzzy expert system for classification is usually insensitive to the precise values for the membership functions.

An alternative method of rule generation, sometimes useful when there are masses of historical numeric input data, is to automatically generate the fuzzy rules from these data. These procedures may be useful in fuzzy control and also in what has become known as "Data Mining." Suppose we have data on input and output of a process which are in the form of real numbers or fuzzy numbers. The procedures to automatically generate the rules involve various techniques such as gradient descent methods (2–5), least squares (6,7), genetic algorithms (8,9), fuzzy c-means (10), fuzzy-neural methods (11–14), heuristics (15), and other methods (16). We will briefly discuss two of the procedures to automatically generate the fuzzy rules.

Perhaps the simplest method is the heuristic procedure presented in Ref. 15 since it does not require iterative learning. Suppose we have data (x_p, y_p), $1 \leq p \leq m$, where $x_p = (x_{1p}, x_{2p})$, on the process to be modeled. The inputs are x_p and the outputs are y_p. Assume that the x_{1p}, x_{2p}, y_p are all in the interval [0, 1]. We wish to construct rules of the form

$$\text{If } x_1 \text{ is } A_{1i} \text{ and } x_2 \text{ is } A_{2j}, \text{ Then } y = b_{ij}$$

for $1 \leq i, j \leq K$. In the consequent b_{ij} is a real number. The A_{1i} (A_{2j}) are triangular fuzzy numbers which partition the interval [0, 1]. Given inputs (x_{1p}, x_{2p}) the output from this block of rules is computed as follows:

$$y = \frac{\sum_{i=1}^{K} \sum_{j=1}^{K} A_{ij}(x_{1p}, x_{2p}) b_{ij}}{\sum_{i=1}^{K} \sum_{j=1}^{K} A_{ij}(x_{1p}, x_{2p})}$$

where $A_{ij}(x_{1p}, x_{2p}) = A_{1i}(x_{1p})A_{2j}(x_{2p})$. The $A_{1i}(x_{1p})$ (and $A_{2i}(x_{2p})$) denote the membership value of the fuzzy set A_{1i} (A_{2i}) at x_{1p} (x_{2p}). The b_{ij} are defined as

$$b_{ij} = \frac{\sum_{p=1}^{m} W_{ij}(x_{1p}, x_{2p}) y_p}{\sum_{p=1}^{m} W_{ij}(x_{1p}, x_{2p})}$$

where $W_{ij}(x_{1p}, x_{2p}) = (A_{1i}(x_{1p})A_{2j}(x_{2p}))^{\alpha}$, for some $\alpha > 0$. This simple heuristic appears to work well in the examples presented.

Another approach, also with no time-consuming iterative learning procedures, was presented in Ref. 16. They proposed a five-step procedure for generating fuzzy rules by combining both numerical data on the process and linguistic information from domain experts. They argued that the two types of information (numerical/linguistic) alone are usually incomplete. They illustrated their method on the truck backer-upper control problem.

UNCERTAINTY REPRESENTATION

The term "uncertainty" itself is not well defined, and may have several related definitions. *Imprecision* in a measurement causes uncertainty as to its accuracy. *Vagueness* is likely to refer to the uncertainty attached to the precise meaning of descriptive terms such as *small* or *fast.* We have lumped all these meanings together under the term "confidence." In fact, the confidence we place in a value is usually itself subject to uncertainty, as are the precise values we define for a membership function. Most fuzzy expert system shells can handle only one level of uncertainty; confidence, grades of membership, and membership functions are usually taken as accurate. While this is not intellectually satisfactory, in practice it poses few if any problems; providing for more than one level of uncertainty would cause a giant increase in system complexity, with little gain achieved.

Let us now have a look at the different methods that can be used for uncertainty modeling. We will not discuss the differences between probability and fuzzy set theory. For more details on the ongoing debate on probabilities versus fuzzy sets in uncertainty modeling see Refs. 17 and 18.

Whatever method of modeling uncertainty is selected, you will have the problem of deciding how the uncertainties are to be propagated through the system. For systems such as FLOPS, which do not routinely combine fuzzy numbers, uncertainty propagation is not usually a problem. If, however, fuzzy numbers are combined (as in fuzzy control), it may be a problem to be dealt with.

Probability Approach

Probability theory has been used from the very beginning to handle uncertainty in expert systems. The goal of this approach is to find suitable probability distributions. There are two main and different interpretations of probability. First, probability can be seen as a relative frequency. In this context, probability describes an objective uncertainty. On the other hand, probability can be interpreted as a measure of belief. This way of interpreting probability leads to subjective imprecision. The last interpretation seems to be more suitable in knowledge-based systems. In order to define a probability distribution function, experts are asked about the numerical value of some parameters. By specifying these quantities a suitable subjective probability distribution function can be found. For combining individual uncertainties, Bayes' theorem has often been employed.

Dempster–Shafer Approach

Dempster–Shafer's theory of evidence is another way of handling uncertainty. It is motivated by Dempster's theorem (19,20), and it is based on the assumption that a partial belief in a proposition is quantified by a single number in the unit interval. Furthermore, different beliefs can be combined into a new one. However, this method leads to exponential complexity (21), and we refer the reader to Refs. 22, and 23 for more details.

Fuzzy Set Approach

The use of fuzzy sets for designing knowledge-based systems was suggested by Zadeh (24). His intention was to (a) overcome the problem of vague concepts in knowledge-based systems being insufficiently described by only using zeros and ones, and (b) use fuzzy sets for representing incomplete knowledge tainted with imprecision and uncertainty. Using fuzzy sets, uncertainty can easily be described in a natural way by membership functions which can take values in the whole unit interval [0, 1].

Let us take a closer look at the representation of uncertainty in a knowledge-based system using the fuzzy set approach.

Real Numbers. In this approach real numbers, instead of fuzzy sets, are used to describe uncertainty in the data. Let us assume that for the linguistic variable "human's height" we have five terms: very short, short, medium, tall, and very tall. Each of these terms is defined by a fuzzy number A_i, where A_1 defines very short, A_2 defines short, and so on. Let x be a number that can be the height of a person. If $A_1(x) = 0$, $A_2(x) = 0.3$, $A_3(x) = 0.8$, $A_4(x) = 0.2$, and $A_5(x) = 0$, then we can express x as (0, 0.3, 0.8, 0.2, 0) showing the uncertainty about the height of this person. So, maybe the person's height is medium.

Fuzzy Numbers. Another way of modeling uncertainty is to use fuzzy sets where the membership functions have to be chosen according to an expert's knowledge. Fuzzy numbers with a limited menu of shapes can easily be used for this task because they can be represented by only three or four values. In order to minimize the amount of parameters, Gaussian fuzzy sets are used. The membership functions represent a Gaussian function which can be coded by a mean value (membership degree one) and a variance value σ (25).

RULE-BASED REASONING

We first discuss a very popular method of rule-based reasoning called *approximate reasoning.* Then we present an alternative procedure used in the fuzzy expert system shell called FLOPS. Consider a block of rules

$$\text{If } x \text{ is } A_i, \text{ then } y \text{ is } B_i$$

for $1 \leq i \leq m$. All the fuzzy sets will be fuzzy subsets of the real numbers. One can easily handle more clauses in the rule's antecedent than simply "x is A_i," but for simplicity we will only consider the single clause. Given an input $x = A$, the rules are executed and we obtain a conclusion $y = B$. There are two methods of obtaining $y = B$ (26): (1) First infer, then aggregate (FITA); and (2) first aggregate, then infer (FATI). Let us first consider FITA. There are three steps in FITA: (1) First model the implication $(A_i \rightarrow (B_i)$ as a fuzzy relation R_i; (2) combine A with R_i, called the *compositional rule of inference,* as $A \circ R_i = B_i'$; and (3) then aggregate all the B_i' into B. However, there are a tremendous number of different ways to get B from A. Numerous papers have been published comparing the various methods (see, for example, Refs. 27–34) and guidelines for picking a certain procedure to accomplish specific goals. In fact, 72 methods of modeling the implication have been studied (35). The methods of computing $A \circ R_i$ and aggregating the B_i' usually employ t-norms and/or t-conorms (36), and we have plenty of more choices to make for these operators.

FATI also has three steps: (1) Model the implication $(A_i \rightarrow (B_i)$ as a fuzzy relation R_i; (2) aggregate the R_i into one fuzzy relation R; (3) compose A with R to get B as $A \circ R = B$. Again, there are many choices on how to compute the R_i, R, and B from A and R. One solution to this dilemma of choosing the right operators to perform approximate reasoning is to decide on what properties you want your inferencing system to possess and then choose the operators that will give those properties. For a single rule system "If x is A, then y is B" we say the inferencing is consistent if given input $x = A$, then the conclusion is $y = B$. That is, $A \circ (A \rightarrow B) = B$. If all the fuzzy sets are normalized (maximum membership is one), then there are a number of ways to perform approximate reasoning so that the rule is consistent (37). For FITA to be consistent we require $B = B_k$ if $A = A_k$ for some k, $1 \leq k \leq m$. FATI is consistent if $A_k \circ R = B_k$. In Ref. 38 the authors give a sufficient condition for FATI to be consistent, and in Ref. 39 the authors argue that, in general, FATI is not consistent, but FITA can be consistent if you use a consistent implication from the single rule case and you also use a special method of aggregating the B_i' into B. Demanding consistency will greatly narrow down the operators you can use in approximate reasoning.

An alternative method of rule-based inferencing is used in FLOPS (discussed below in the section on software). Here the rules are used to construct discrete fuzzy sets and approximate reasoning is not applicable.

Initially, numeric input variables are fuzzified to create discrete fuzzy sets. From there on, reasoning can be done with fuzzy set member words such as Large (a member of discrete fuzzy set size) and Left (a member of discrete fuzzy set horizontal position) rather than in terms of numerical values. Consider the following slightly simplified FLOPS rule:

IF size is Small AND x-position is Center
AND y-position is Very-High, THEN class is Artifact;

The confidence that each clause in the antecedent is valid is computed. For clauses such as "size is Small," the confidence is simply the grade of membership of Small in discrete fuzzy set size. Other clauses might involve comparisons, Boolean or fuzzy, between two data items; in this case, the confidence that the clause is valid is the fuzzy AND (by default, minimum) of the confidences in the data items and the confidence that the comparison holds. The minimum of the confidences in the individual clauses (and the rule confidence, if less than 1) is the confidence that the entire antecedent is valid. This confidence is stored as the grade of membership of Artifact in discrete fuzzy set class. Since there is no fuzzy set in the consequence of the above rule (Artifact is not fuzzy), the implication cannot be modeled as a fuzzy relation and approximate reasoning is not applicable. FLOPS then reasons with confidences in discrete entities in which there is placed a single scalar confidence; many of these are discrete fuzzy sets.

OPTIMIZATION

Once a fuzzy expert system has been designed, it depends on a large set of parameters such as the weights (confidence values in the rules, etc.), the number of rules, the method of inference, and the position and shape of the fuzzy sets which define the linguistic variables in the rules. The optimization, also called the tuning or calibration, of the fuzzy expert system is a process of determining a best value for these parameters. "Best" is defined as those values of the parameters which maximize, or minimize, some objective functions. At first, researchers suboptimized with tuning some of the parameters while holding the rest fixed. A number of papers (5,40–44) presented various techniques to minimize the number of rules needed in the rule base. A basic method was the use of genetic algorithms.

Another group of papers (8,9,45–50) was concerned with tuning the membership functions of the fuzzy sets with the use of a genetic algorithm, a popular technique. Gradient decent methods were also employed to tune the fuzzy sets. The next step, possibly using a genetic algorithm, will be to tune the whole fuzzy expert system. One would need to code a whole rule, its weights, and all the fuzzy sets in the rule, as part of one individual in a population of individuals evolving toward the optimal solution. Using binary coding a single rule will produce a fairly long vector of zeros and ones. Add to this vector all the other rules so that an individual in the population is the whole fuzzy expert system. Append to this vector the types of rule inferencing methods you wish to investigate. If we are to have 2000 individuals in the population and we wish to go through 10,000 generations in the genetic algorithm, we see that the computation becomes enormous. Hence researchers have been content to attack only parts of the whole optimization problem. Let us now briefly discuss two of these methods of tuning a fuzzy expert system.

In Ref. 42 the authors tune the rules in a fuzzy expert system designed for a classification problem. The problem has two objectives: (1) Maximize the number of correctly classified patterns, and (2) minimize the number of rules. A set of fuzzy if-then rules is coded as one individual in the genetic algorithm. The fitness function for the algorithm is a convex combination of the two objectives (maximize the number of correctly classified patterns and minimize the number of rules).

In Ref. 47 the authors optimize the fuzzy sets in a fuzzy expert system used for control. They assume that there is a data set available on the process, and the objective is to minimize the squared error function defined from the input–output data. The fuzzy if-then rules, the method of inference, and the defuzzifier are all held fixed. All the fuzzy sets are trapezoidal fuzzy numbers. A member of the population is a coded vector containing all the trapezoidal fuzzy numbers in the fuzzy expert system. Their tuning method worked well in the application (the inverted pendulum problem) presented where the population size was small, and there were only seven if-then rules in the system.

VALIDATION AND IMPLEMENTATION

An expert system is a model of how an expert thinks; like all models, it must be tested before routine use (validation). It is of the utmost importance to use different data sets for tuning and validation. If the entire data set is gathered at one time, it is common to split the data set into two: one for tuning, and the other for validation. You may also employ a domain expert to validate the fuzzy expert system. Suppose the system was designed to classify regions seen in a medical echocardiogram. To validate the system we compare how it classifies regions to how an expert classifies the same regions on a new series of echocardiograms.

Once the fuzzy expert system has been validated, it is ready for use. If it is to be used for control, then it will usually run on-line and have to be very fast and is now ready to be implemented in hardware. That is, to get the speed to be used on-line you may need to obtain hardware for the system. Fuzzy expert systems not used for control have not (in the past) usually run on-line and do not need to be very fast. However, with the dramatic increases in computer speed continually occurring, running online in real time is now possible. A tremendous increase in speed can also be realized by hard coding an expert system into (for example) the C or C++ languages. A further increase in speed can be achieved by the use of interval rather than fuzzy logic.

HARDWARE

For the development of knowledge-based systems, special hardware is seldom needed. For this task a suitable software tool is more important than fast hardware. However, as knowledge-based systems become larger, it is no longer suitable to use a single processor. Therefore, the data collection and the actions of the system should be logically and geographically distributed in order to speed up the computational expense (51). Special fuzzy hardware can overcome this problem.

In Ref. 52 the authors present a fuzzy component network which consists of fuzzy sensors, fuzzy actuators, and fuzzy inference components. All these components can be configured. However, a special language is needed for this task. In order to take advantage of special hardware, this fuzzy hardware configuration language has to be integrated into a fuzzy expert system developing tool. Another way is to transform the developed expert system into special hardware. However, this approach seems to be unsuitable because modifications of the system (which seem likely) can lead to needed changes in the hardware.

Slowly, as fuzzy expert systems (and fuzzy controllers) were developed and became more sophisticated, special hardware was suggested to implement the various components of these systems. Today there is much more interest in obtaining hardware for fuzzy systems as evidenced by the recent edited book devoted solely to fuzzy hardware (53).

SOFTWARE

While there are many excellent software packages available for constructing fuzzy control systems, there are only a few designed for more general fuzzy reasoning applications. Notable for being based in Artificial Intelligence technology are FRIL, a fuzzy superset of PROLOG; FLOPS, a fuzzy superset of OPS5; and Fuzzy CLIPS, a fuzzy superset of CLIPS. While not directly based in AI technology, METUS is highly developed from a computer science viewpoint and has achieved considerable success in noncontrol problems.

All these systems are powerful, and they embody facilities which are not possible even to enumerate let alone describe in detail. The descriptions furnished here are certainly incomplete. While all systems are capable of application in diverse fields, the precise facilities furnished depend somewhat on the fields in which they have had the most use. FRIL has probably had the most diverse applications, ranging from aircrew modeling to vision understanding. FLOPS has been applied primarily to medical and technical fields. Fuzzy CLIPS has found its greatest use in engineering, and METUS has been used primarily in the financial world.

FRIL

Created originally by James F. Baldwin, FRIL offers the advantages of Prolog plus those of fuzzy systems theory. Prolog has been one of the two dominant AI languages (the other being LISP) and has been especially popular in Europe. FRIL provides the critical data types of discrete fuzzy sets, whose members may be symbols or numbers, and continuous fuzzy sets such as fuzzy numbers. FRIL's fundamental data structure is a list, each term of which can be a data item or a list.

Rules in FRIL reverse the IF (antecedent) THEN (consequent) construction discussed above. The first element of a FRIL rule corresponds to the consequent; the second element corresponds to the antecedent. For example, the rule

((illness of X is flu)(temp of X is high)
(strength of X is weak)(throat of X is sore)): (0.9,1)

corresponds to the following IF-THEN rule:

rule rconf 0.9 IF temp of X is high
AND strength of X is weak
AND throat of X is sore,
THEN illness of X is flu

In the first case, the symbols : (0.9,1) mean that if the antecedent clauses are true, we are at least 0.9 sure that the consequent clause (illness of X is flu) is true. Similarly, in the second case the symbols rconf 0.9 mean that we are 0.9 confident that the rule is valid—that is, that if the antecedent holds, the consequent is true.

Being constructed as a superset of a powerful AI language, FRIL in turn can be very powerful, but is not an easy language to learn unless one has previous experience with Prolog. Fortunately there is excellent documentation in the form of a text which includes a demonstration diskette (54).

FLOPS

FLOPS was created by Douglas Tucker, William Siler, and James J. Buckley (55) to solve a pattern recognition problem involving very noisy images. FLOPS is a fuzzy superset of OPS5, a well-known AI production system for constructing expert systems. FLOPS added fuzzy sets, fuzzy numbers, and approximate numerical comparisons to OPS5's capabilities. Most FLOPS applications have been medical or technical, as distinct from business or control applications. Two rule-firing modes are offered: sequential and parallel. Suppose (as is often the case) that more than one rule is concurrently fireable. In sequential mode, one rule is selected for firing; the rest are stacked for backtracking—that is, for firing if the path chosen does not work out. In parallel mode, all concurrently fireable rules are fired effectively in parallel; any resulting conflicts for memory modification are then arbitrated by the inference engine.

Like OPS5, FLOPS is a forward-chaining system; however, backward chaining is easily emulated. As is usually the case with production systems, there is a lot of system overhead in checking which rules are newly fireable. FLOPS employs the popular RETE algorithm to reduce this overhead. The parallel mode of FLOPS also considerably reduces system overhead, since instead of checking for rule fireability after each rule is fired we check only after each block of rules is fired. This typically reduces system overhead by roughly a factor of six.

A simple basic blackboard system is employed to transfer data between external programs (for example, C++) and FLOPS. A standardized simple relational database format is used for this purpose. FLOPS can call external programs for special purposes when a rule-based system is inappropriate, or it can call other FLOPS programs. Internally, FLOPS programs are organized by rule blocks, with metarules to control rule block fireability and activation of external non-AI programs and other FLOPS programs. Recursion can be used for problems where its use is indicated, ranging from the toy problem Tower of Hanoi through solution of ordinary differential equations. A special command is furnished for real-time on-line applications.

To reduce the number of rules in an application, FLOPS programs may shift expert knowledge from rules to a database of expert knowledge, with rules written to interpret that database, or to generate rules automatically from the expert knowledge database. Program learning may involve writing rules to generate other rules.

A program development environment TFLOPS is furnished for creating FLOPS programs. Debugging facilities include (a) inspection of data and fireable rule stacks and (b) a simple explain facility for tracing data backwards through modification and creation and checking why rules are or are not fireable.

Fuzzy Clips

Inspired by Robert Lea of NASA and created by the National Research Council of Canada under the direction of Robert Orchard, this language is a fuzzy superset of CLIPS, a nonfuzzy expert system shell developed by the NASA Johnson Space Flight Center. Its availability on the Internet without charge is certainly an added plus (56).

A program development environment is furnished which permits editing, running, and viewing programs. While discrete fuzzy sets are not available as data types, the use of certainty factors attached to character strings creates *fuzzy facts,* which can be used individually in much the same manner as members of discrete fuzzy sets.

A simple control rule, written in Fuzzy Clips, is

(defrule rule_pos_pos (error positive)(rate positive)
⟹ (assert (output negative)))

where rule_pos_pos is the name of the rule. We require that the error and rate both be positive; if these are true, then the confidence that output is negative is set to the antecedent confidence. A debugging facility is provided by a flexible WATCH command, amounting to a sophisticated trace of a program run.

METUS

METUS, written by Earl Cox (57,58), is a powerful tool for fuzzy reasoning even though it is not based on an existing AI system. Its use has been primarily in business and financial applications, in a client–server environment. Its origin is in Reveal, a fuzzy expert system by Peter Llewellyn Jones of the United Kingdom. Metus provides both forward and backward chaining, a sophisticated blackboard system, and program development facilities. METUS employs a flexible if-then-else rule syntax and is especially notable for its advanced use of hedges, which are modifying adjectives applied to fuzzy sets. Simple METUS rules are:

IF costs are High, THEN margins are Weak;
else margins are Strong

and

if speed is very Fast, then stopping time is Increased.

Also provided are time lags for a time sequence of numerical data, as in

if sales $[t - 1]$ are Low but inventory $[t]$ is Moderate,
then buying risk is Elevated.

METUS supplies a number of multivalued logics in addition to the well-known Zadeh max–min operators, along with a number of defuzzification techniques. Metarules permit executing external non-METUS programs, executing other METUS programs (called policies), and enabling rules or lists of rules.

BIBLIOGRAPHY

1. J. J. Buckley, W. Siler, and D. Tucker, Fuzzy expert system, *Fuzzy Sets Syst.,* **20**: 1–16, 1986.
2. H. Ichihashi and T. Watanabe, Learning control system by a simplified fuzzy reasoning model, *Proc. IPUM'90,* 1990, pp.417–419.
3. H. Ishibuchi et al., Emperical study on learning in fuzzy systems by Rice test analysis, *Fuzzy Sets Syst.,* **64**: 129–144, 1994.
4. J. S. R. Jang, ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Trans. Syst. Man Cybern.,* **23**: 665–685, 1993.
5. H. Nomura, I. Hayashi, and N. Wakami, A learning method of fuzzy inference rules by decent method, *Proc. 1st IEEE Int. Conf. Fuzzy Syst.,* San Diego, 1992, pp. 203–210.
6. M. Sugeno and G. T. Kang, Structure identification of fuzzy model, *Fuzzy Sets Syst.,* **28**: 15–33, 1988.
7. T. Takagi and M. Sugeno, Fuzzy identification of systems and its application to modeling and control, *IEEE Trans. Syst. Man Cybern.,* **15**: 116–132, 1985.
8. C. L. Karr and E. J. Gentry, Fuzzy control of pH using genetic algorithms, *IEEE Trans. Fuzzy Syst.,* **1**: 46–53, 1993.
9. H. Nomura, I. Hayashi, and N. Wakami, A self-tuning method of fuzzy reasoning by genetic algorithm, *Proc. Int. Fuzzy Syst. Control Conf.,* Louisville, KY, 1992, pp. 236–245.
10. M. Sugeno and T. Yasukawa, A fuzzy-logic-based approach to qualitative modeling, *IEEE Trans. Fuzzy Syst.,* **1**: 7–31, 1993.
11. C. M. Higgins and R. M. Goodman, Fuzzy rule-based networks for control, *IEEE Trans. Fuzzy Syst.,* **2**: 82–88, 1994.
12. Y. Lin and G. A. Cunningham, A new approach to fuzzy-neural system modeling, *IEEE Trans. Fuzzy Syst.,* **3**: 190–198, 1995.
13. M. Russo, Comments on "A new approach to fuzzy-neural system modeling," *IEEE Trans. Fuzzy Syst.,* **4**: 209–210, 1996.
14. H. Takagi and I. Hayashi, NN-driven fuzzy reasoning, *Int. J. Approximate Reason.,* **5**: 191–212, 1991.
15. K. Nozaki, H. Ishibuchi, and H. Tanaka, A simple but powerful heuristic method for generating fuzzy rules from fuzzy data, *Fuzzy Sets Syst.,* **86**: 251–270, 1997.
16. L. X. Wang and J. M. Mendel, Generating fuzzy rules by learning from examples, *IEEE Trans. Syst. Man Cybern.,* **22**: 1414–1427, 1992.
17. D. Dubois and H. Prade, Fuzzy sets—A convenient fiction for modeling vagueness and possibility, *IEEE Trans. Fuzzy Syst.,* **2**: 16–21, 1994.
18. G. J. Klir, On the alleged superiority of probabilistic representation of uncertainty, *IEEE Trans. Fuzzy Syst.,* **2**: 27–31, 1994.
19. A. P. Dempster, Upper and lower probabilities induced by multivalued mapping, *Ann. Math. Stat.,* **38**: 325–339, 1967.
20. G. Shafer, Constructive probability, *Synthese,* **48**: 1–60, 1981.
21. P. Orponen, Dempster's rule of combination is #*P*-complete, *Artif. Intell.,* **44**: 245–253, 1990.
22. I. R. Goodman and H. T. Nguyen, *Uncertainty Models for Knowledge-Based Systems,* New York: Elsevier, 1985.
23. R. Kruse, E. Schwecke, and J. Heinsohn, *Uncertainty and Vagueness in Knowledge Based Systems,* Berlin: Springer-Verlag, 1991.
24. L. Zadeh, A theory of approximate reasoning, in D. Michie, J. Hayes, and L. Mickulich (eds.), *Machine Intelligence,* Amsterdam: Elsevier, 1979, Vol. 9, pp. 149–194.
25. D. Cayrac, D. Dubois, and H. Prade, Handling uncertainty with possibility theory and fuzzy sets in a satellite fault diagnosis application, *IEEE Trans. Fuzzy Syst.,* **4**: 251–269, 1996.
26. I. B. Turksen and Y. Tian, Constraints on membership functions of rules in fuzzy expert systems, *Proc. 2nd IEEE Int. Conf. Fuzzy Syst.,* San Francisco, 1993, pp. 845–850.
27. Z. Cao and A. Kandel, Applicability of some implication operators, *Fuzzy Sets Syst.,* **31**: 151–186, 1989.
28. S. Fukami, M. Mizumoto, and K. Tanaka, Some considerations on fuzzy conditional inference, *Fuzzy Sets Syst.,* **4**: 243–273, 1980.
29. E. E. Kerre, A comparative study of the behavior of some popular fuzzy implication operators on the generalized modus ponens, in L. A. Zadeh and J. Kacprzyk (eds.), *Fuzzy Logic for the Management of Uncertainty,* New York: Wiley, 1992, pp. 281–295.
30. R. Martin-Clouaire, Sematics and computation of the generalized modus ponens: The long paper, *Int. J. Approximate Reason.,* **3**: 195–217, 1989.
31. M. Mizumoto and H.-J. Zimmermann, Comparison of fuzzy reasoning methods, *Fuzzy Sets Syst.,* **8**: 253–283, 1982.
32. M. Mizumoto, Comparison of various fuzzy reasoning methods, *Proc. IFSA Congr., 2nd,* Tokyo, 1987, pp. 2–7.
33. D. Park, Z. Cao, and A. Kandel, Investigations on the applicability of fuzzy inference, *Fuzzy Sets Syst.,* **49**: 151–169, 1992.
34. C. Romer and A. Kandel, Applicability of fuzzy inference by means of generalized Dempster–Shafer theory, *IEEE Trans. Fuzzy Syst.,* **3**: 448–453, 1995.
35. J. B. Kiszka et al., The inference of some fuzzy implication operators on the accuracy of a fuzzy model I, II, *Fuzzy Sets Syst.,* **15**: 111–128, 223–240, 1985.
36. G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic,* Upper Saddle River, NJ: Prentice Hall, 1995.
37. P. Magrez and P. Smets, Fuzzy modus ponens: A new model suitable for applications in knowledge-based systems, *Int. J. Intell. Syst.,* **4**: 181–200, 1989.
38. A. DiNola, W. Pedrycz, and S. Sessa, An aspect of discrepancy in the implementation of modus ponens in the presence of fuzzy quantities, *Int. J. Approximate Reason.,* **3**: 259–265, 1989.
39. J. J. Buckley and Y. Hayashi, Can approximate reasoning be consistent?, *Fuzzy Sets Syst.,* **65**: 13–18, 1994.
40. F. Guely and P. Siarry, Gradient decent method for optimizing fuzzy rule bases, *Proc. IEEE Int. Conf. Fuzzy Syst., 2nd,* San Francisco, 1993, pp. 1241–1246.

41. C. C. Hung and B. R. Fernandez, Minimizing rules of fuzzy logic system by using a systematic approach, *Proc. 2nd IEEE Int. Conf. Fuzzy Syst.,* San Francisco, 1993, pp. 38–44.
42. H. Ishibuchi et al., Selecting fuzzy if-then rules for classification problems using genetic algorithms, *IEEE Trans. Fuzzy Syst.,* **3**: 260–270, 1995.
43. R. Rovatti and R. Guerrieri, Fuzzy sets of rules for system identification, *IEEE Trans. Fuzzy Syst.,* **4**: 89–102, 1996.
44. R. Rovatti, R. Guerrieri, and G. Baccarani, An enhanced two-level Boolean synthesis methodology for fuzzy rules minimization, *IEEE Trans. Fuzzy Syst.,* **3**: 288–299, 1995.
45. S. Abe, M.-S. Lan, and R. Thawonmas, Tuning of a fuzzy classifier derived from data, *Int. J. Approximate Reason.,* **14**: 1–24, 1996.
46. H. Bersini, J. Nordvik, and A. Bornari, A simple direct fuzzy controller derived from its neural equivalent, *Proc. 2nd IEEE Int. Conf. Fuzzy Syst.,* San Francisco, 1993, pp. 345–350.
47. F. Herrera, M. Lozano, and J. L. Verdegay, Tuning fuzzy logic controllers by genetic algorithm, *Int. J. Approximate Reason.,* **12**: 299–315, 1995.
48. C. L. Karr, Design of an adaptive fuzzy logic controller using a genetic algorithm, *Proc. 4th Int. Conf. Genet. Algorithms,* San Diego, 1991, pp. 450–457.
49. C. Perneel et al., Optimization of fuzzy expert systems using genetic algorithms and neural networks, *IEEE Trans. Fuzzy System,* **3**: 300–312, 1995.
50. P. Thrift, Fuzzy logic synthesis with genetic algorithms, *Proc. 4th Int. Conf. Genet. Algorithms,* San Diego, 1991, pp. 509–513.
51. S. S. Iyengar and R. J. Kashyap, Distributed sensor networks: Introduction to the special section, *IEEE Trans. Syst. Man Cybern.,* **21**: 1027–1031, 1991.
52. J.-F. Josserand and L. Foulloy, Fuzzy component network for intelligent measurement and control, *IEEE Trans. Fuzzy Syst.,* **4**: 476–487, 1996.
53. A. Kandel and G. Langholz (eds.), *Fuzzy Hardware Architectures and Applications,* Boston: Kluwer Academic Publishers, 1998.
54. J. F. Baldwin, T. P. Martin, and B. Pilsworth, *Fril: Fuzzy and Evidential Reasoning,* New York: Wiley, 1995.
55. W. Siler, [Online]. Available www: FLOPS: http://users.aol.com/wsiler
56. Fuzzy Clips: http://ai.iit.nrc.ca/fuzzy
57. E. Cox, *Fuzzy Systems Handbook,* San Diego: Academic Press, 1994.
58. E. Cox, [Online]. Available www: METUS: http://www.metus.com

JAMES J. BUCKLEY
University of Alabama at Birmingham
WILLIAM SILER
Southern Dynamic Systems
THOMAS FEURING
University of Münster