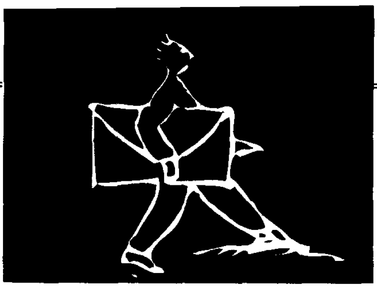


Soft Computing and Fuzzy Logic

LOTFI A. ZADEH, *University of California at Berkeley*

◆ *Soft computing is a collection of methodologies that aim to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost. Its principal constituents are fuzzy logic, neurocomputing, and probabilistic reasoning. Soft computing is likely to play an increasingly important role in many application areas, including software engineering. The role model for soft computing is the human mind.*

One of the deepest traditions in science is that of according respectability to what is quantitative, precise, rigorous, and categorically true. It is a fact, however, that we live in a world that is pervasively imprecise, uncertain, and hard to be categorical about. It is also a fact that precision and certainty carry a cost. Driven by our quest for respectability, we tend to close our eyes to these facts and thereby lose sight of the steep price we must pay for high precision and low uncertainty. Another visible concomitant of the quest for respectability is that in much of the scientific literature elegance takes precedence over relevance.

A case in point is the traveling salesman problem, which is frequently used as a testbed for assessing the effectiveness of various methods of solution. What is striking about this problem is the steep rise in computing time as a function of precision of solution. As the data in Table 1 show, lowering the accuracy to 3.50 percent reduces the computing time by an order of magnitude for a ten-fold increase in the number of cities.

A more familiar example that illustrates the point is the problem of parking a car. We find it relatively easy to park a car because the final position of the car is not specified precisely. If it were, the difficulty of parking would increase geometrically with the increase in precision, and eventually parking would become impossible.

Guiding principle. These and many similar examples lead to the basic premises and the guiding principle of *soft computing*.

The basic premises of soft computing are

◆ The real world is pervasively imprecise and uncertain.

♦ Precision and certainty carry a cost.

The guiding principle of soft computing is

♦ Exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, and low solution cost.

The label soft computing is growing in use. What does it mean? Where does it stand today and where is it headed? And what is the role of fuzzy logic in soft computing?

Sample application. Some of the most striking examples of the application of the guiding principle of soft computing are the data-compression techniques that play a key role in high-definition television and audio recording and reproduction.

For example, in NHK's Muse system, a motion-compensating technique determines the outline, direction, and speed of the moving body, then shifts the moving image without waiting to receive all the pixel data. The resulting moving image does not have the resolution of the still picture. Muse exploits the fact that the human eye cannot grasp the details of moving objects with the same precision as still objects. Even more impressive is what is achieved in the recently developed digital HDTV systems. For example, the General Instrument system, instead of transmitting data for every color dot in a blue sky, sends an instruction to paint the sky. The compression ratio this system achieves is on the order of 60 to 1. In audio recording and reproduction, similar ideas are embodied in Sony's MD-1 system and Philips' DCC.

In its current incarnation, the concept of soft computing has links to many earlier influences, among them my 1965 paper on fuzzy sets;[1] 1973 paper on the use of linguistic variables in the analysis and control of complex systems;[2] and 1979 report (1981 paper) on possibility theory and soft data analysis.[3]

Unlike traditional hard computing, soft computing is aimed at accommodating the pervasive imprecision of the real world. Although soft computing has not as yet had a visible impact on software engineering, it is likely to do so in the years ahead. Among the areas in which it is likely to be applied are programming languages, computer security, database management, user-friendly interfaces, automated programming, fault diagnosis, and networking.

In this article, I will focus on some of the basic ideas that underlie soft computing and relate them to its guiding principle.

SOFT COMPUTING AND FUZZY LOGIC

Basically, soft computing is not a homogeneous body of concepts and techniques. Rather, it is a partnership of distinct methods that in one way or another conform to its guiding principle. At this juncture, the dominant aim of soft computing is to exploit the tolerance for imprecision and uncertainty to achieve tractability, robustness, and low solution cost. The principal constituents of soft computing are fuzzy logic, neurocomputing, and probabilistic reasoning, with the latter subsuming genetic algorithms, belief networks, chaotic systems, and parts of learning theory. In the partnership of fuzzy logic, neurocomputing, and probabilistic reasoning, fuzzy logic is mainly concerned with imprecision and approximate reasoning; neurocomputing with learning and curve-fitting; and probabilistic reasoning with uncertainty and belief propagation.

In large measure, fuzzy logic, neurocomputing, and probabilistic reasoning are complementary, not competitive. It is becoming increasingly clear that in many cases it is advantageous to combine them. A case in point is the growing number of "neurofuzzy" consumer products and systems that use a combination of fuzzy logic and neural-network techniques.

**TABLE 1
SOLUTIONS FOR TRAVELING SALESMAN PROBLEM**

Number of cities	Accuracy	Computing time
100,000	1.00%	two days
100,000	0.75%	seven months
1,000,000	3.50%	3.5 hours

Source: New York Times, March 12, 1991.

In this article, I focus on fuzzy logic.

FUZZY LOGIC CONCEPTS

As one of the principal constituents of soft computing, fuzzy logic is playing a key role in what might be called high MIQ (machine intelligence quotient) systems.

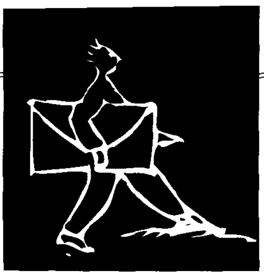
Two concepts within fuzzy logic play a central role in its applications.

♦ The first is a *linguistic variable;* that is, a variable whose values are words or sentences in a natural or synthetic language.[2]

♦ The other is a *fuzzy if-then rule*, in which the antecedent and consequents are propositions containing linguistic variables.[2]

The essential function of linguistic variables is that of granulation of variables and their dependencies. In effect, the use of linguistic variables and fuzzy if-then rules results — through granulation — in lossy data compression. In this respect, fuzzy logic mimics the remarkable ability of the human mind to summarize data and focus on decision-relevant information.

With regard to fuzzy logic, there is an issue of semantics that is in need of clarification. Specifically, it is frequently not recognized that the term fuzzy logic is actually used in two different senses. In a narrow sense, fuzzy logic (*FLn*) is a logical system — an extension of multivalued logic that is intended to serve as a logic of approximate reasoning. In a wider sense, fuzzy



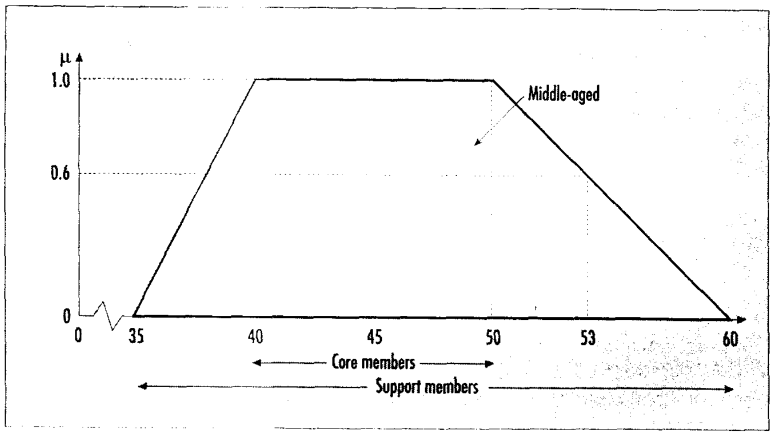


Figure 1. Interpretation of middle-aged as a linguistic value.

logic (*FLw*) is more or less synonymous with fuzzy set theory; that is, the theory of classes with unsharp boundaries. In this perspective, *FL* = *FLw*, and *FLn* is merely a branch of *FL*. What is important to recognize is that today the term fuzzy logic is used predominantly in its wider sense. It is in this sense that any field *X* can be "fuzzified" — and hence generalized — by replacing the concept of a crisp set in *X* by a fuzzy set. In application to basic fields such as set theory, arithmetic, topology, graph theory, probability theory, and logic, fuzzification leads to fuzzy set theory, fuzzy arithmetic, fuzzy topology, fuzzy graph theory, and fuzzy logic in its narrow sense.

Similarly, in application to applied fields like neurocomputing, stability theory, pattern recognition and mathematical programming, fuzzification leads to fuzzy neurocomputing, fuzzy stability theory, fuzzy pattern recognition, and fuzzy mathematical programming. What is gained through fuzzification is greater generality, higher expressive power, an enhanced ability to model real-world problems, and — most important — a methodology for exploiting the tolerance for imprecision, a methodology that fits the guiding principle of soft computing and thus serves to achieve tractability, robustness, and low solution cost.

Linguistic variables. A concept in fuzzy logic that plays a key role in exploiting the tolerance for imprecision is the linguistic variable. A linguistic variable, as its name suggests, is a variable whose values are words or sentences in a natural or synthetic language. For example, *age* is a linguistic variable if its *linguistic values* are *young*, *old*, *middle-aged*, *very old*, *not very young*, and so on. A linguistic variable is interpreted as a label of a fuzzy set that is characterized by a *membership function*, as illustrated in Figure 1. Thus, if *u* is a numerical age, say 53, then $\mu_{middle\text{-}aged}(53)$ is the *grade of membership* of 53 in *middle-aged*. Subjectively, you may interpret $\mu_{middle\text{-}aged}(u)$ as the degree to which *u* fits your perception of middle-aged in a specified context.

In a general setting, a linguistic variable, *V*, can be viewed as a microlanguage with context-free grammar and attributed-grammar semantics. The context-free grammar defines the legal values of *V*. For example, in the case of *age*, the legal values are *young*, *not young*, *not very young*, *quite old*, *middle-aged*, and so on. The attributed-grammar semantics provides a mechanism for computing the membership function of any value of *V* from the knowledge of the membership functions of the so-called *primary terms* — *young* and *old*, for example. A primary term plays the role of a generator whose meaning (its membership function) must be calibrated in context. For example, the meaning of *not very young* might be computed as

$$\mu_{not\ very\ young}(u) = 1 - (\mu_{young}(u))^2$$

where *very* plays the role of an intensifier and *young* is a primary term whose membership function is specified in context.

Most current applications of fuzzy logic employ a simpler framework, illustrated in Figure 2. Specifically, the membership functions are assumed to be triangular or trapezoidal, and the number of linguistic values is usually in the range of three to seven.

The concept of a linguistic variable plays a central role in the applications of fuzzy logic because it goes to the heart of the way in which humans perceive, reason, and communicate. Quintessentially, the use of words may be viewed as a form of data compression that exploits the tolerance for imprecision to achieve tractability, robustness, and economy of communication. This fits almost precisely the guiding principle of soft computing.

Granulation. In a related sense, the use of words may be viewed as a form of fuzzy quantization or more generally as *granulation*, as Figure 3 shows.

Basically, granulation involves a replacement of a constraint of the form

$$X = a$$

with a constraint of the form

$$X \text{ is } A$$

where *A* is a fuzzy subset of *U*, the universe of *X*. For example,

$$X = 2$$

might be replaced with

$$X \text{ is } small$$

In fuzzy logic, *X* is *a* is interpreted as a characterization of the possible values of X, with A representing a possibility distribution. Thus, the possibility that X can take a value u is given by

$$Poss\{X = u\} = \mu A(u)$$

It is in this sense that X is $\mu A(u)$, with possibility interpreted as ease of attainment or assignment, may be interpreted as an elastic constraint on X.

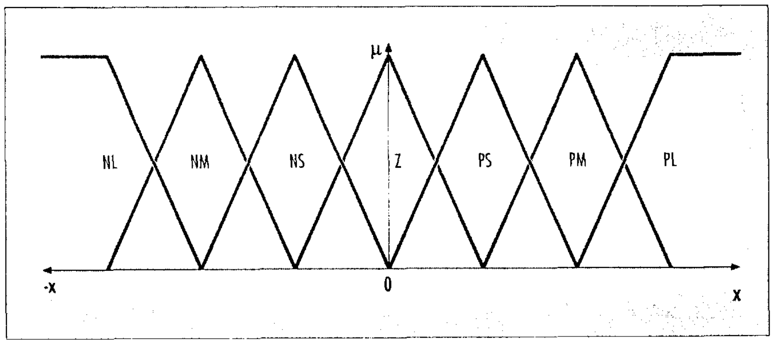


Figure 2. Triangular linguistic values, usually ranging from negative large, negative medium, and negative small (NL, NM, and NS) to zero (Z) to positive small, positive medium, and positive large (PS, PM, and PL).

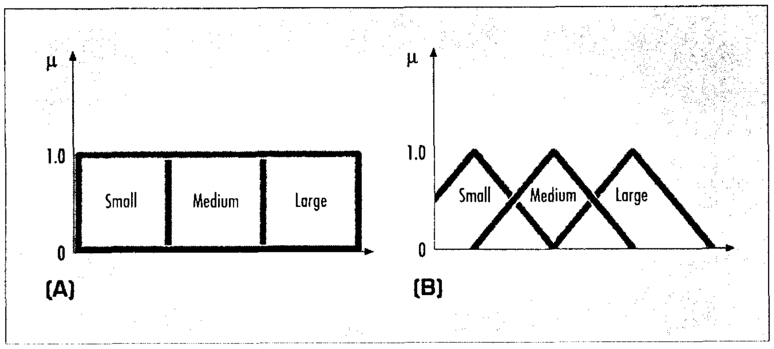


***Figure 3.** (A) Quantization versus (B) granulation (fuzzy quantization).*

Fuzzy rules and fuzzy graphs. The concept of a linguistic variable serves as a point of departure for what might be called the Fuzzy Dependency and Command Language.[4] As its name suggests, FDCL provides a system for representing and manipulating fuzzy dependencies and commands. Central to FDCL are the concepts of a *fuzzy rule* and *fuzzy graph* and the concomitant calculi of fuzzy rules and fuzzy graphs.[4,5] Employed in an informal way, these calculi have played and continue to play a key role in most of the applications of fuzzy logic.

To view the calculi of fuzzy rules and fuzzy graphs in a proper perspective, it is necessary to note that the representation and manipulation of dependencies is an essential part of the scientific method. Mathematics provides us with several calculi to do this, among them the calculi of differential equations, difference equations, matrices, linear and nonlinear mappings, and probabilistic dependencies. These calculi are effective when the dependencies are well-defined and amenable to numerical analysis. When they are not, the calculi of fuzzy rules and fuzzy graphs provide an alternative methodology aimed at results that are in the spirit of the guiding principle of soft computing.

More specifically, the use of the calculi of fuzzy rules and fuzzy graphs is indicated when the dependencies are

♦ ill-defined or too complex for conventional methods, or

♦ well-defined, but it is advantageous to use linguistic variables to exploit the tolerance for imprecision, thereby lowering the solution cost and achieving a higher MIQ.

Interestingly, the development of fuzzy-set theory was motivated by the first situation, but today most applications of fuzzy logic in the realm of consumer products are motivated by the second.

FDCL has many facets. Here, I shall sketch some of the basic ideas that underlie FDCL and the calculi of fuzzy rules and fuzzy graphs.

Like any language, FDCL is characterized by its syntax and semantics. The syntax of FDCL is concerned with the form of admissible fuzzy rules; the semantics is concerned with their meaning. It is important to note that FDCL is not a "fuzzified" version of a standard programming language, as is true of Fuzzy Prolog.[6]

Fuzzy rules. FDCL allows the use of a wide variety of fuzzy if-then rules, or simply fuzzy rules. A typical fuzzy rule relates m antecedent variables $X_1,\ldots,X_m$ to n consequent variables, $Y_1,\ldots,Y_n$ and has the form:

if X_1 is A_1 and ... X_m is A_m
then Y_1 is B_1 and ... Y_n is B_n

where $X = (X_1,\ldots,X_m)$ and $Y = (Y_1,\ldots,Y_n)$ are linguistic variables and $(A_1,\ldots,A_n)$ and $(B_1,\ldots,B_n)$ their respective linguistic values. For example:

if *Pressure* is *high* and *Temperature* is *high* then *Volume* is *small*

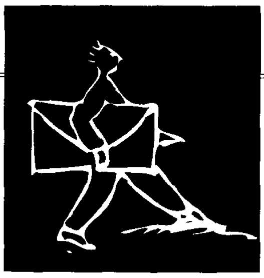
For simplicity, I will discuss only rules in which $m = n = 1$.

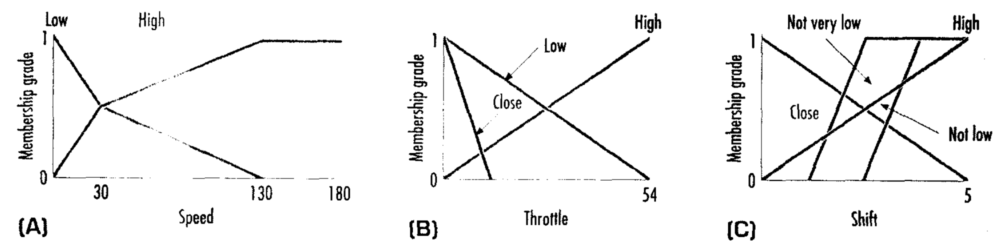
A rule can have a *surface structure* or a *deep structure*. The *surface structure* is the rule in its symbolic form:

if X is A then Y is B

Such a rule is said to be *uncalibrated*, which means that the membership functions of A and B are not specified.

The *deep structure* is the surface structure together with a characterization of the membership functions of linguistic values of variables. In this case, the rule is said to be *calibrated*.





Control rules
1. If (speed is low) and (shift is high) then (−3)
2. If (speed is high) and (shift is low) then (+3)
3. If (throt is low) and (speed is high) then (+3)
4. If (throt is low) and (speed is low) then (+1)
5. If (throt is high) and (speed is high) then (−1)
6. If (throt is high) and (speed is low) then (−3)

Figure 4. *Fuzzy rules used in Honda's fuzzy-logic transmission. Here, the meaning of the numeric values associated with the rules is not important; they only illustrate how rules are calibrated*

Figure 4 shows an example of the calibrated fuzzy rules used in Honda's fuzzy-logic transmission. As I explain later, one of the central problems in the applications of fuzzy logic is that of deriving the deep structure of a set of fuzzy rules from I/O data.

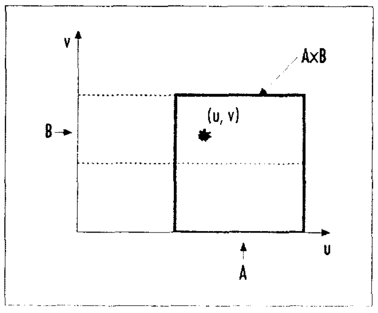


Figure 5. *$A \times B$ interpreted as a fuzzy point or a granule.*

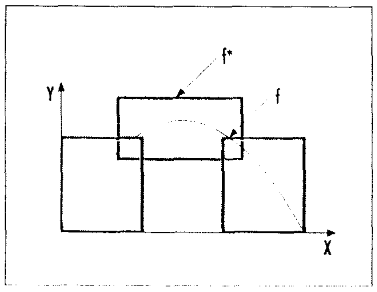


Figure 6. *Interpretation of a collection of fuzzy rules as a fuzzy graph.*

DERIVING RULES AND GRAPHS

In the semantics of FDCL, the basic questions are, what is the meaning of a single rule and what is the meaning of a collection of rules?

Deriving rules. Consider the simplest type of rule:

if X is A then Y is B

where A and B are linguistic values of X and Y, respectively. The question is, what is the meaning of this rule given the membership functions of A and B?

In fuzzy logic, the meaning of a proposition p is expressed as a *canonical form*

$p \rightarrow Z$ is C

where $\rightarrow$ means "translates into," Z is the constrained variable, and C is an fuzzy relation that plays the role of a fuzzy constraint on Z. What this implies is that the meaning of p is expressed as a fuzzy — or, equivalently, elastic — constraint on a designated variable. To illustrate, the meaning of the proposition *Mary is young* might be expressed as

Mary is young $\rightarrow$ *Age(Mary)* is *young*

where *Age(Mary)* is the focal variable and *young* is a fuzzy constraint on *Age(Mary)*.

Applying this concept of meaning representation to the fuzzy rule

if X is A then Y is B

we can express the meaning of the rule in question as a fuzzy constraint on the joint variable (X,Y). More specifically,

if X is A then Y is $B \rightarrow (X,Y)$ is $A \times B$,

where $A \times B$ is the Cartesian product of A and B. The membership function of $A \times B$ is given by

$$\mu_{A\times B}(u,v) = \mu_A(u) \wedge \mu_B(u)$$

where $\wedge$ is the conjunction operator, usually defined as min. $A \times B$ may be interpreted as a *fuzzy point* or a *granule*, as shown in Figure 5.

Deriving graphs. In the case of a collection of rules expressed as

if X is A_i then Y is B_i, $i = 1, ..., n$

and the meaning of the collection is defined as

if X *is* A_i then Y is B_i,
$(i = 1,...,n) \rightarrow (X,Y)$ is

$(A_1 \times B_1 + \ldots + A_n \times B_n)$

where + is used in place of $\vee$ to denote the disjunction operator, which is usually defined as max. For simplicity, the right-hand member of the collection may be written as

(X,Y) is $(\Sigma_i A_i \times B_i)$

The expression $\Sigma_i A_i \times B_i$ may be viewed as a superposition of fuzzy points or granules, as illustrated in Figure 7. In effect, it represents a coarse — or, equivalently, compressed — characterization of the dependency and for this reason it is called a *fuzzy graph*.[5] Thus, a collection of fuzzy rules is represented as a fuzzy graph. For example:

if X is *small* then Y is *small*
if X is *medium* then Y is *large*
if X is *large* then Y is *small*

is a coarse characterization of the dependency illustrated in Figure 6.

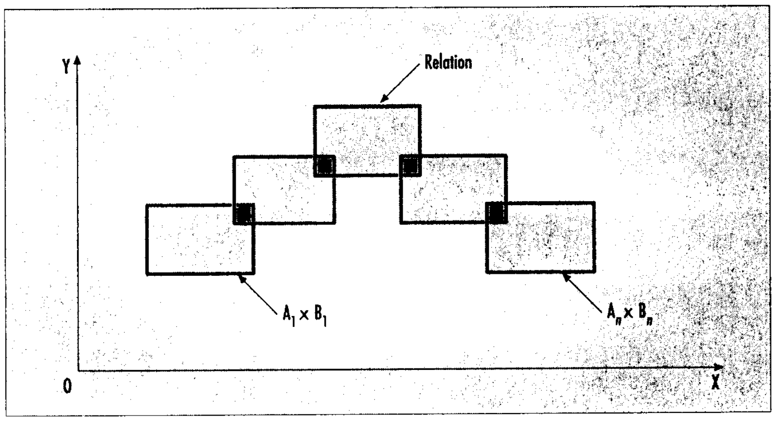


***Figure** 7. Representing a collection of fuzzy rules as a fuzzy graph, f^*, which approximates to f.*

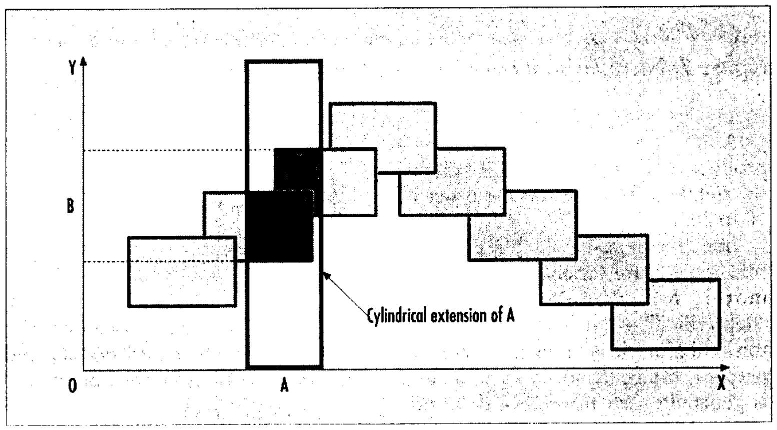


***Figure** 8. Interpolation of a fuzzy graph. The value of Y may be interpreted as the projection of the intersection of the fuzzy graph with the cylindrical extension of A.*

INTERPOLATION

If we interpret a collection of rules as a coarse representation of the functional dependence of Y on X, the *problem of interpolation* may be defined as that of computing the value of Y given a value of X that may not be a perfect match with any of the antecedent variables in the collection. More specifically, this problem can be expressed as the inference schema

(X,Y) is $(\Sigma_i A_i \times B_i)$

$$\frac{X \text{ is } A}{Y \text{ is} ?B}$$

in which $?B$ signifies that B is the object of computation. In graphical terms, as shown in Figure 8, the problem may be viewed as that of assigning a linguistic value to X and computing the corresponding linguistic value of Y.

In fuzzy logic, computation of B is carried out through the basic rule of inference, called the *compositional rule* of inference.[2] The rule in question reads

(X,Y) is R

$$\frac{X \text{ is } A}{Y \text{ is } R \bullet A}$$

in which the composition operation is defined by

$$\mu_{R \bullet A}(v) = sup_u(\mu_R(u,v) \wedge \mu_A(u))$$

in which $\mu_R(u,v)$ and $\mu_A(u)$ are, respectively, the membership functions of R and A.

In the example considered earlier, R is given by

$$R = (\Sigma_i A_i \times B_i)$$

in which

$$\mu_R(u,v) = \Sigma_i \mu_A(u) \wedge \mu_B(v)$$

Then

$$\mu_B(v) = \Sigma_i \mu_i \wedge \mu_B(v)$$

or, equivalently,

$$B = \Sigma_i \mu_i \wedge B_i$$

in which

$$\mu_i = (\mu_{A_i}(u) \wedge \mu_A(u))$$

The sequence of computations that leads to B is standard in most fuzzy logic applications and is usually implemented in software or hardware. In some implementations, called max-product implementations, the conjunction $\wedge$ is interpreted as the arithmetic product.

Interpolation lies at the heart of the utility of fuzzy rule-based systems because it makes it possible to employ a relatively small number of fuzzy rules to characterize a complex relationship between two or more variables. In a typical application in a consumer

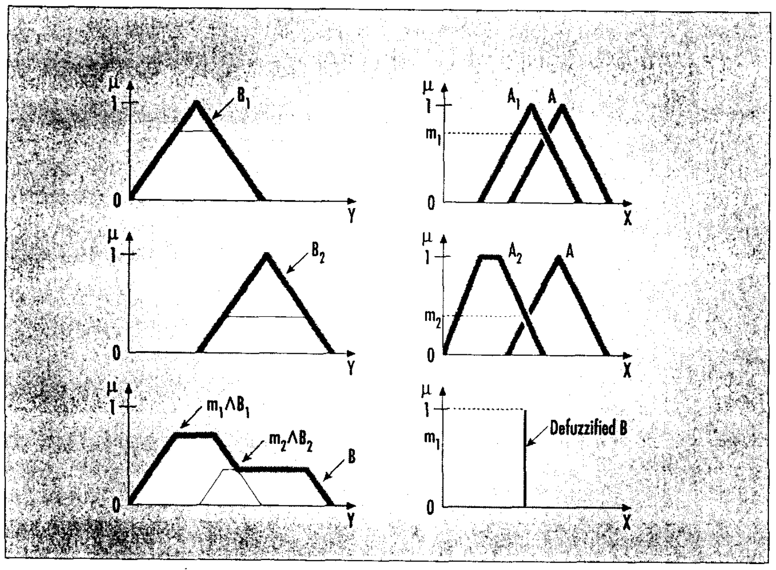


Figure 9. *Interpolation of two rules and defuzzification.*

product or industrial-control system, the number of rules is on the order of 10 to 20.[7,8]

In applications in which B plays the role of a control variable (an input to a motor, for example), B must be "defuzzified" — converted to a singleton — before it is applied. In current practice, the center-of-gravity method is generally used to achieve defuzzification. Figure 9 shows a simple example of interpolation and defuzzification.

In this figure, the rules are

if X is A_1 then Y is B_1
if X is A_2 then Y is B_2

and the input is

X is A

m_1 and m_2 are, respectively, the degrees to which A matches A_1 and A_2. The expression for the output is

$$\mu_B(v) = \Sigma_l \mu_l \wedge \mu_{Bl}(v)$$

which upon center-of-gravity defuzzification leads to a numerical value of B.

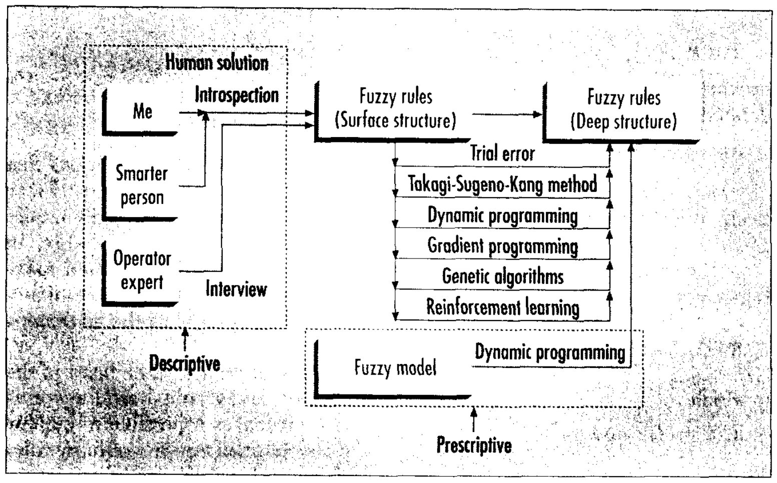


Figure 10. *Summary of alternative methods to deduce the deep structures of a set of rules.*

INDUCING RULES FROM OBSERVATION

One of the central problems in the applications of fuzzy logic relates to the question, how can rules be inferred from observations; that is, from the knowledge of a collection of I/O pairs? In the context of self-organizing systems, this problem was first formulated and analyzed by T.J. Procyk and E.H. Mamdani. Later, a seminal paper by T. Takagi and M. Sugeno made a major contribution.[9]

During the past several years, researchers have made important advances toward at least a partial solution to the problem by applying neural-network techniques or, more generally, dynamic and gradient programming,[7,10-11] Other promising approaches involve the use of genetic algorithms[12,13] and reinforcement learning.[14] Figure 10 summarizes the ways to derive the deep structure of a set of rules from the surface structure.

A basic idea underlying these approaches involves representing a fuzzy rule-based system as a multilayered structure, such as that shown in Figure 11.[11] In a simple version of this architecture[11] that is rooted in the Takagi-Sugeno-Kang approach,[9] the rules are assumed to be of the form

(if X_1 is A_{1i} and ... and X_m is A_{mi}
then $Y = b_i$), $i = 1,\ldots,n$

where b_i are constants (singleton consequents). If the numerical values of $X_1, \ldots X_m$ are $u_{1i}, \ldots u_{mi}$, respectively, and the grades of membership of $u_{1i}, \ldots u_{mi}$, in $A_{1i}, \ldots A_{mi}$ are $\mu_{1i}(u_{1i}),\ldots \mu_{mi}(u_{mi})$, then the combined degree to which the input n-tuple $X(u_{1i}, \ldots u_{mi})$ matches the antecedents is taken to be the product

$$m_i = \mu_{1i}(u_{1i}),\ldots \mu_{mi}(u_{mi}),$$

Then, defining the normalized weight w_i as

$$w_i = m_i / m_1 + \ldots + m_m$$

the output is expressed as

$$Y = \Sigma_i w_i b_i$$

Note that in this architecture there is no defuzzifier because the inputs X_1, ... X_m are assumed to be singletons.

In the application of gradient programming to this architecture, the membership functions of A_{1i}, ... A_{mi} are assumed to be triangular, trapezoidal, or Gaussian in form. Then, using backward iteration, the values of membership-function parameters are computed from right to left.[10-11] In this way, from the knowledge of I/O pairs we can compute the values of parameters and thereby induce the rules from observations.

The approach sketched here is one way the methodologies of fuzzy rule-based systems and neural networks can be combined, leading to "neurofuzzy" systems. Such systems are growing in number and visibility and are illustrative of the advantages derived from combining soft computing's constituent methodologies. In this context, it is important to note that interpolation and induction of rules from observations are key issues in both fuzzy logic and neurocomputing.

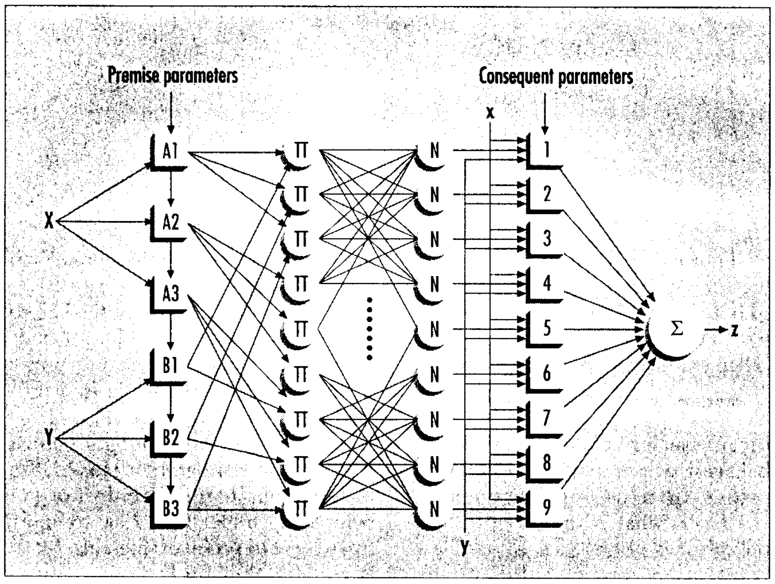


Figure 11. Representing a fuzzy system as a multilayered structure. Π and N denote multipliers and normalizers, respectively.

FUZZY BALL AND BEAM PROBLEM

An interesting problem that serves as a testbed for fuzzy logic is a variation on the ball and beam problem. The ball and beam problem, shown in Figure 12, is to transfer the ball from some initial position to a point in a specified positional set-interval and keep it in that interval.

Initially, the ball is placed in the notch at the center of the beam. The purpose of the notch is to constrain the initial value of Θ to be above a specified threshold. The same purpose would be served by a notch at each end of the beam.

In the fuzzy version of the problem, we want to transfer the ball from the notch to a point in a specified positional set-interval $[a_1, a_2]$ and keep it in that interval. Clearly, no approach that is model-dependent — in the sense of requiring a formulation of equations governing system behavior — can be employed because we do not know how to model a ball rolling or sliding on a rug-like surface. This rules out the use of classical control theory as well as any approach that requires simulation.

The set-interval may be viewed as a disjunctive goal. This feature makes it difficult to employ neural-network techniques.

By contrast, the problem is easy to solve with fuzzy logic because it is relatively easy for a human. In fact, the presence of a fuzzy layer makes the ball-and-beam problem easy for humans and difficult or impossible for alternative methodologies. As in most fuzzy-logic applications, the solution is in effect a translation of a human solution into FDCL. A human solution would normally involve seven steps:

1. Compile uncalibrated fuzzy or crisp rules from knowledge of natural laws, to govern the behavior of the ball and beam. For example

   if Θ is negative
   then $\ddot{Y}$ is positive
   if Θ is positive
   then $\ddot{Y}$ is negative
   the more negative Θ,
   the more positive $\ddot{Y}$
   the more positive Θ,
   the more negative $\ddot{Y}$

2. Construct a plan of action (an algorithm), expressed in terms of uncalibrated fuzzy rules of the form

   if *State* is *A* then *Action* is *B*

3. Test the system without trying to solve the problem
4. Calibrate the fuzzy rules in step 2 using metarules, rules that modify other rules
5. Test the algorithm constructed in step 2.
6. Refine the calibrated fuzzy rules derived in step 5.
7. Iterate steps 5 and 6 until the ball stays in the set-interval.

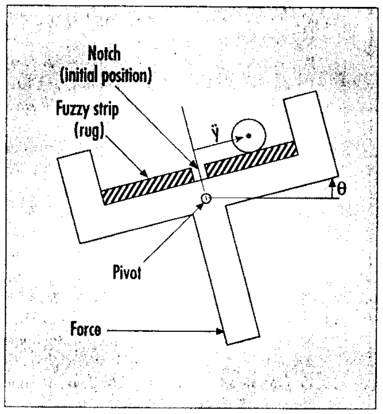


Figure 12. Fuzzy ball and beam problem.

Translating these steps into a collection of fuzzy rules expressed in FDCL is by no means a trivial problem. This is particularly true of the so-called gradual rules[15] of the form

the more X is A the more Y is B

because such rules describe the global behavior of a functional dependency rather than its local properties. However, what is important is that, though it is not easy, it is feasible to translate a human solution into FDCL, whereas it is not feasible to translate it into analytical techniques.

Now suppose you wanted the ball to reach the set interval at some time t and stay there. This is significantly more difficult for a human because it involves a conjunction of two goals:

♦ confine the motion of the ball to the prescribed set-interval $[a_1, a_2]$, and

♦ enter the set-interval at a time t which is constrained to lie in a prescribed temporal set-interval $[t_1, t_2]$.

In this case, formulating a human solution and translating it into FDCL is a real challenge. We do not yet completely understand how to apply fuzzy logic to problems like this. But it is evident that fuzzy logic — used alone or in combination with neurocomputing and probabilistic reasoning — is the methodology of choice when analytic models are impossible or hard to formulate.

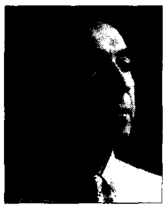
Although soft computing is still in its initial stages of evolution, it is rapidly growing in importance and visibility. In the years ahead, soft computing and its principal constituents — fuzzy logic, neurocomputing, and probabilistic reasoning — are likely to emerge as essential tools for the conception, analysis, and design of high MIQ systems. In the final analysis, the role model for soft computing is the human mind. ●

ACKNOWLEDGMENTS

This research was supported in part by the BISC Program, NASA Grant NCC 2-275, EPRI Agreement RP 8010-34 and MICRO State Program No. 92-180.

REFERENCES

1. L.A. Zadeh, "Fuzzy Sets," *Information and Control,* June 1965, pp. 338-353.
2. L.A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Trans. Systems, Man and Cybernetics*, 1973, pp. 28-44.
3. L.A. Zadeh, "Possibility Theory and Soft Data Analysis," *Mathematical Frontiers of the Social and Policy Sciences*, L. Cobb and R. M. Thrall, eds., Westview Press, Boulder, Colo., 1981, pp.69-129.
4. L.A. Zadeh, "Fuzzy Logic, Neural Networks and Soft Computing," *Comm. ACM*, Mar. 1994, pp. 77-84.
5. L.A. Zadeh, "On the Analysis of Large-Scale Systems," in *Systems Approaches and Environment Problems,* H. Gottinger, ed., Vandenhoeck and Ruprecht, Gottingen, 1974, pp. 23-37.
6. M. Mukaidono, Z.L. Shen, and L. Ding, "Fundamentals of Fuzzy Prolog," *Int'l J. Approximate Reasoning,* No. 3, 1989, pp. 179-193.
7. B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice-Hall, Englewood Cliffs, N.J., 1991.
8. T. Terano, K. Asai, and M. Sugeno, "Fuzzy Systems Theory and Its Applications," Academic Press, San Diego, Calif., 1992.
9. T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and its Applications to Modeling and Control," *IEEE Trans. Systems, Man, and Cybernetics 15*, 1985, pp. 116-132.
10. C.-T. Lin and C.S. George Lee, "Neural-Network-Based Fuzzy Logic Control and Decision System," *IEEE Trans. Computers*, Dec. 1991, pp. 1320-1336.
11. J.-S.R. Jang, "Self-Learning Fuzzy Controller Based on Temporal Back-Propagation," *IEEE Trans. Neural Networks*, May 1992, pp. 714-723.
12. C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I and Part II," *IEEE Trans. Systems, Man, and Cybernetics 20*, 1990, pp. 404-418.
13. C. Karr, "Genetic Algorithms for Fuzzy Controllers," *AI Expert*, Nov. 1991, pp. 26-33.
14. M.A. Lee and H. Takagi, "Integrating Design Stages of Fuzzy Systems Using Genetic Algorithms," *Proc. Int'l Conf. on Fuzzy Systems*, IEEE Press, New York, 1993, pp. 612-617.
15. D. Dubois and H. Prade, "Gradual Inference Rules in Approximate Reasoning," *Information Sciences*, 1992, pp. 103-122.



Lotfi A. Zadeh is professor emeritus of electrical engineering and computer sciences at the University of California, Berkeley, where he is director of the Berkeley Initiative in Soft Computing. His research interests are the theory of fuzzy sets and its applications to artificial intelligence, linguistics, logic, decision analysis, expert systems, and neural networks.

Zadeh is a graduate of the University of Teheran, MIT, and Columbia University. He has received honorary doctorates from the Paul-Sabatier University, France, the State University of New York at Binghamton, Dortmund University, Germany; and the University of Granada and Oviedo, Spain, in recognition of his development of the theory of fuzzy sets. He is a fellow of the IEEE, AAAS, ACM, and AAAI and a member of the National Academy of Engineering and the Russian Academy of Natu al Sciences.

Address questions about this article to Zadeh at the CS Div., Dept. of EECS, University of California, Berkeley, Berkeley, Calif. 94720; zadeh@cs.berkeley.edu