Musical Applications of Fuzzy Logic

Peter Elsea

Principles of Fuzzy Logic

Fuzzy sets are distinguished from traditional sets by the concept of partial membership. Partial memberships are descriptive, a number indicating how closely each member represents the ideal of the set. In fuzzy logic, the set "tall" in the domain of height above 5 ft. would be something like figure 1.

{ 0 0 0 0 0 0 0.1 0.2 0.4 0.6 0.8 0.9 1.0 1.0 … }

Figure 1. A fuzzy set for "tall".

The numbers μ are the grade of membership in the set. μ is normalized to be a number from 0 to 1. The zeros at the left end of example 1 suggest that heights below 5'6" are not considered tall, and ones at the right indicate that any height above 6'0" is tall. Between the two values is a transition zone where the membership is fractional. With this mechanism, a person of height 5'9" can be described as "somewhat tall" with a tallness membership of 0.6. This makes it possible to do meaningful comparisons within boundary regions that are poorly managed by traditional methods.

The form $\mu_A(x)$ means the membership of x in set A.

The rules for set operations under fuzzy logic differ only slightly from those of standard (or crisp) logic.
A union $A \cup B$ of two fuzzy sets has the maximum of the values for each member.
The intersection $A \cap B$, contains the minimum values found for each member.
The fuzzy complement A' is 1-μ for each member.
The product AB is the product of each member pair.
The power function A^n is μ^n for each member.
A^2 and $A^{0.5}$ are specially defined as concentration (CON) and dilation (DIL) respectively.
The bounded sum $A \oplus B$ is the sums of the member pairs limited to 1.
The bounded difference $A \Theta B$ is $\mu_A(x)$- $\mu_B(x)$ limited to 0.
For complete explanations of fuzzy operations see [Zadeh 1993].

Functions

In fuzzy logic, a characteristic such as "Tall" may be represented by a membership function in an appropriate continuous domain such as height. A domain will support several sets of related characteristics such as short, very short and so on. The functions may be a simple increase or decrease across a portion of the domain, or may be trapezoidal, with 0 membership at the ends of the domain, and membership of 1 at one or more intermediate loci (or vice versa). Trapezoidal sets need not be symmetrical. The transitions may be linear or some type of curve, such as a sigmoid. These functions are often represented graphically, and are usually programmed as lookup tables. Such tables are used in either direction: values may be returned for locations, and locations returned for values, with interpolation if appropriate. It is assumed that the end values are extended as needed beyond the listed domain.

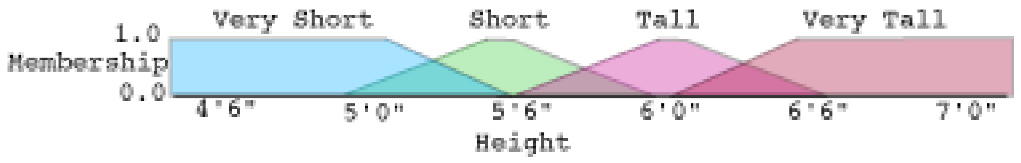

Figure 2. Fuzzy sets illustrated by functions.

Fuzzy Numbers

A particularly useful type of fuzzy set is the fuzzy number [Dubois 1993]. This is a representation of the concept "around n." In a typical implementation, the value n would have a membership in the set N of 1, n + 1 and n-1 would have a membership between 0 and 1, n+2 and n-1 would have lower memberships and so on to values that are not near n and have memberships of 0. It is possible to do math with fuzzy numbers but this seldom arises in musical applications. The fuzzy number is often used for approximate equalities, where for fuzzy number N, x==N is interpreted as μN(x). This principle has been successfully expanded to give reliable pattern matching for noisy data.[Dubois 1983]

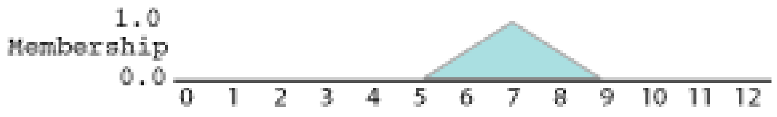

Figure 3. A fuzzy number

Making Decisions in Fuzzy Logic

Reasoning in fuzzy logic is usually based on inference, statements of the type "for value x, if A or B then C" and "for value x, if D and E then C". Under fuzzy logic, the condition A_x may be the membership of x in set A, and likewise for condition B. The result r of the expression A or B would be $\max(A_x,B_x)$ a value between 0 and 1. If C is a single action, r can be a scalar for the action. If C is a set, a result set is generated by taking $\min(r, C_i)$ for all of C. This clipped set can then be used in further steps of reasoning.

In complex reasoning, several inferences are evaluated and the results combined. As an example, consider the factors that might determine optimum performance tempo. These can be stated as a set of rules:

Perform near the marked tempo.
If players are virtuosos, play faster.
If music is difficult, play slower.

There are five quantities that can be represented by fuzzy sets, which are shown graphically in figure 4. Skill of the performers may be a complex issue, or a separate value for each player. In this example let the conductor rate them from one to ten. Then the minimum value would be used. Difficulty of music is also complex, and probably changes throughout the piece. It might be based on average number of notes per measure.

The universe of discourse for the results is the range of practical tempi. "Near the marked tempo" might be a fuzzy set corresponding to allegro, adagio, and so on, or it might be a fuzzy number generated by shifting a triangular function to a specific metronome setting. "Faster" is a function that increases across this domain, and slower is a decreasing function. Note that these functions may overlap or there may be a gap. Slow is not the compliment of fast.

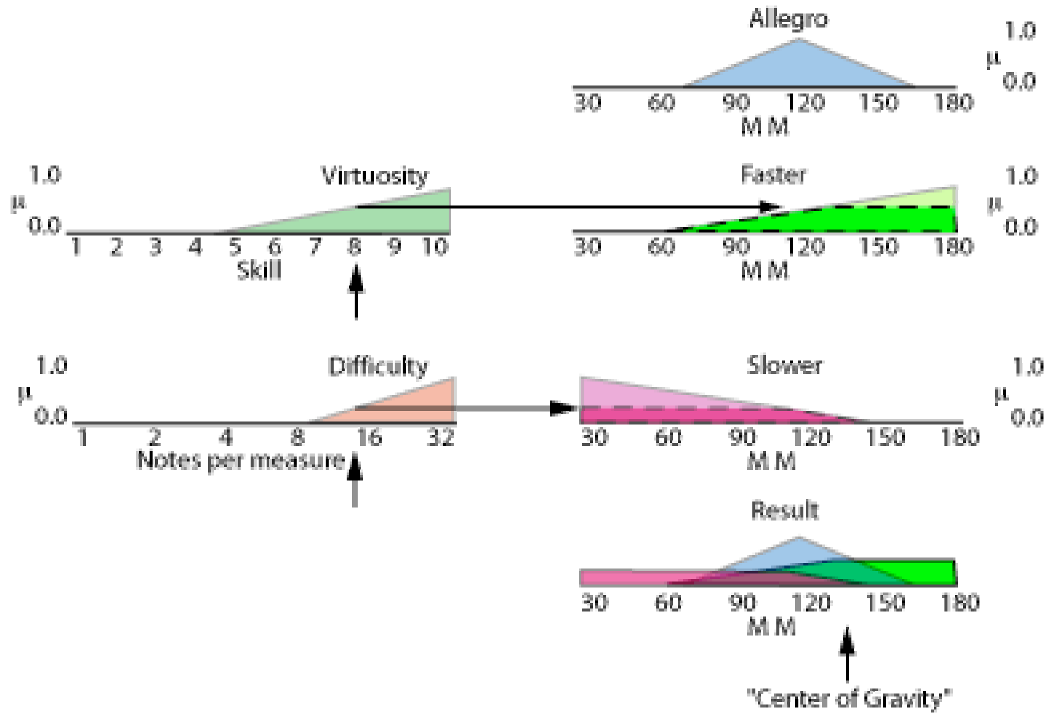

Figure 4. Basic decision mechanism.

To find optimum tempo, evaluate all three rules and take the union[1] of the result sets. A solution value can be extracted in many ways, but the most common is finding the centroid or "center of gravity" of the result.[Cox 1993, p 249]

$$\frac{\sum_{i=1}^{i=n} \mu_R(i) * i}{\sum_{i=1}^{i=n} \mu_R(i)}$$

This method will produce a reasonable tempo for any set of conditions. It is very easy to add rules such as "if the hall is reverberant, play slowly". The result set of the new rule is simply included in the final union. It is easy to turn rules on and off to test their effect, or to weight a rule to adjust its importance.

Building Expert Systems

An expert system can be built from a set of rules and a library of appropriate functions. The rules can be developed from an accurate description of procedures. In fact, there is a body of techniques for mapping linguistic statements to rules and to math operations on the core sets. For instance, the set "very fast" might be produced by squaring the membership values in "fast". The values and shape of the functions used may be subjective or empirically derived. The systems will begin to produce rough results as soon as a few rules are in place, and will become more accurate as rules are added and adjustments are made to the sets.

The Application of Fuzzy Logic to Music

There are many features of music that can be described as "fuzzy". As an example, consider the mapping of Midi velocity to traditional dynamics. Any performer understands the concept of mezzo-forte as meaning the middle of the dynamic range, but an actual performance will vary quite a bit around the median point. Likewise, a forte passage will probably cover half of the possible velocities, with a good deal of overlap with the mezzo range. Figure 5 shows how velocity might map to memberships in fuzzy sets for the traditional dynamics. These sets could be used either to assign a dynamic marking to a recorded passage or to determine appropriate velocities for a particular dynamic.

[1] In some situations, the final set can be constructed by summing the memberships of the result sets. Do this when several rules affect the same part of the resultant domain and you don't want one them to hide each other. It is not necessary to normalize the solution set before finding the centroid.

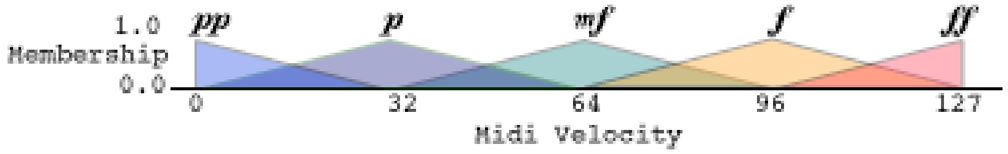

Figure 5. The fuzzy relationship of dynamic markings to velocity.

It is important to remember that these are not distributions of any kind. The low membership of forte at velocity 67 simply indicates that a velocity of 67 is not particularly loud. A velocity of 32 also has a value in the set forte, a value of 0.

A Fuzzy Approach to Producing Music Structures

Fuzzy sets can be used to represent musical structures[2]. Let M represent the C major scale, with memberships in the universe of pitch class [p_0,…, p_{11}]:

M= {1 0 1 0 1 1 0 1 0 1 0 1}.

Since the universe of pitch classes is defined modulo 12, transposition of any musical structure is possible by right rotation, equivalent to:

$\mu_T(i) = \mu_M((i+t)\mathrm{mod}12)$

The symbol >> will be used to denote right rotation.
Any major scale is then produced by right rotation of the C major scale. D major will be represented by M>>2.

M>>2 = {0 1 1 0 1 0 1 1 0 1 0 1 }

The interval of a third can be represented by the structure:

TT= {0 0 0 1 1 0 0 0 0 0 0 0}

This includes both the major and minor third. Again, rotation can be used to move the root. Then the question, what is a third above D in the scale of C major can be answered by (TT>>2) ∩ M.

TT>>2 = {0 0 0 0 0 1 1 0 0 0 0 0}

M = {1 0 1 0 1 1 0 1 0 1 0 1}

(TT>>2) ∩ M= {0 0 0 0 0 1 0 0 0 0 0 0}

However, this approach does not generalize to other scales. A harmonic minor scale H would be

H ={ 1 0 1 1 0 1 0 1 1 0 0 1}.

But finding the third above A flat implies:

[2] This is not the set of musical set theory.

(TT>>8) ∩ H ={1 0 0 0 0 0 0 0 0 0 0 0 1}

This is not a useful answer. Fuzzy logic allows us to define the third as :

F3= {0 0 0 1 0.9 0 0 0 0 0 0 0 }

The partial membership 0.9 indicates that the interval of 4 half steps is a bit less representative of the third in a minor scale. Thus

(F3>>8) ∩ H = {0.9 0 0 0 0 0 0 0 0 0 0 0 1}

This can be easily reduced to a single answer by finding the locus of the highest membership. The locus of highest membership is common enough in music applications to justify a definition: TOP^n A = the set of loci of the top n membership values, which can be determined by a simple iterative procedure.

This is not to imply that the question cannot be answered by traditional sets. It can, simply by considering the two types of third separately:

```
If (MT>>8) ∩ H
then (MT>>8) ∩ H
else (mT>>8) ∩ H
```

The fuzzy approach is a bit more efficient, and such efficiencies add up in a complex program. However, a more important benefit is that the definition of the fuzzy third contains a preference for the minor. The major could be encouraged by a simple change in the membership values, which would likely only be defined once in a program. A similar modification in the traditional code requires changing the order of operation everywhere it occurs, which could mean substantial revision. This trivial example survives generalization- that is, in most applications fuzzy logic yields simpler code that is easier to modify than traditional approaches.

Table 1 defines fuzzy sets for common intervals above and below a given root.

F2	{ 0 0.9 1 0 0 0 0 0 0 0 0 0 }	F2b	{0 0 0 0 0 0 0 0 0 0 1 0.9}
F3	{ 0 0 0 1 0.9 0 0 0 0 0 0 0 }	F3b	{ 0 0 0 0 0 0 0 0 0.9 1 0 0 }
F4	{ 0 0 0 0 0 1 0.5 0 0 0 0 0 }	F4b	{ 0 0 0 0 0 0 0.5 1 0 0 0 0}
F5	{ 0 0 0 0 0 0 0.9 1 0.5 0 0 0 }	F5b	{ 0 0 0 0 0.5 1 0.9 0 0 0 0 0 }
F6	{ 0 0 0 0 0 0 0 0 1 0.9 0 0 }	F6b	{ 0 0 0 0.9 1 0 0 0 0 0 0 0 }
F7	{ 0 0 0 0 0 0 0 0 0 0 0.9 1 }	F7b	{ 0 0.9 1 0 0 0 0 0 0 0 0 0 }

Table 1.

Fuzzy Algorithms

A more subtle benefit of fuzzy logic is that construction algorithms tend to follow human paradigms very closely. Here is a common procedure for building a chord on a given root r in a scale W:

1. Add a third above the root
2. Add a fifth above the root

Step one was described above. Step 2 is identical, except with the fuzzy fifth interval:

F5={0 0 0 0 0 0 0.9 1 0.5 0 0 0}.

This includes the diminished interval and the remote possibility of an augmented fifth. The complete chord is the union of root, third and fifth.

$R = ((S>>r) \cap W) \cup ((F3>>r) \cap W) \cup ((F5>>r) \cap W)$

Where S = {1 0 0 0 0 0 0 0 0 0 0 0 }

The set S is the singleton set which is rotated to designate a particular pitch. This rotation can also be indicated by S_r, and the intersection with the scale set can be omitted if we are assured the root is in the scale, or we are harmonizing chromatic motion. The result set R can be converted into a list of pitch classes by the TOP[3] procedure.

This algorithm works well in major keys, and in natural and harmonic minor when the goal is to favor major and minor sonorities over diminished. When the diminished sound is preferred, the algorithm is:

1. Add a third above the root.
2. Add a third above the third.

The chord is again a union
$R = ((S \geq\geq r) \cap W) \cup ((Fz3 \geq\geq r) \cap W) \cup ((Fz3 \geq\geq TOP(Fz3 \geq\geq r)) \cap W)$

This algorithm works nicely in natural and harmonic minor, favoring diminished chords a bit. In a practical application, the algorithm would be selected according to context.

Chords of the seventh which are natural to the scale can be derived by adding an interval of a fifth above the third of a triad constructed using either of the above algorithms.

$R = ((S \geq\geq r) \cap W) \cup ((Fz3 \geq\geq r) \cap W) \cup ((Fz5 \geq\geq r) \cap W) \cup ((Fz5 \geq\geq TOP(Fz3 \geq\geq r)) \cap W)$

When non-scale seventh chords are desired, they can be created directly by rotation of a template set

Dom7 = {1 0 0 0 1 0 0 1 0 0 1 0}

Similar sets may be constructed for other flavors of seventh chord.

Simple Harmonizer

To build a complete triadic harmonizer, we find the three candidate chords for a given pitch, then choose the best according to context. To find the chord with p as the third, take

$$r6 = TOP^1((F3b>>p) \cap W)$$

where F3b is the third below set from table 1. Then

$$R6 = S_{r6} \cup S_p \cup ((F5>>r) \cap W)$$

Likewise, for the chord with p as a fifth find

$$r6\text{-}4 = TOP^1((F5b>>p) \cap W),$$

then:

$$R6\text{-}4 = S_{r6\text{-}4} \cup ((F3>>r6\text{-}4) \cap W) \cup S_p$$

In other words find the fifth below the given pitch, and the third above that. (To build 7th chords, we find the fifth above the third.)
We then test the three candidates against the previous chord. To favor progressions with common tones, we construct a result set:

$$Rc = \left\{ \frac{\sum(Cp \cap R)}{3}, \frac{\sum(Cp \cap R6)}{3}, \frac{\sum(Cp \cap R6-4)}{3} \right\}$$

The set reflects the desirability of each choice.

To favor specific progressions we compare the roots. Here is a chart of common progressions with sets that suggest some favoritism:[Kosta&Payne 116]

Cp	Followed by	or	or	Sets D_{CP}
I	Any but vii			{1 0 1 0.5 0.5 1 0 1 1 1 0 0}
ii	V	vii		{0 0 0 0 0 0 1 0 0 0 1}
iii	vi			{0 0 0 0 0 0 0 0 0.5 1 0 0}
IV	V	ii	vii	{0 0 1 0 0 0 0 1 1 0.5 0 0}
V	I			{1 0 0 0 0 0 0 0 0 0 0 0 }
vi	IV	ii		{ 0 1 0 0 0 1 0 0 0 0 0 0}
Vii	V	I		{1 0 0 0 0 0 0 1 0 0 0 0}

The sets D_{CP} represent the desirability of possible following chords. This is reminiscent of the Markov process, but the values do not have to add up to

anything. Two equally desirable chords may both have a possibility[3] of 1, and less interesting choices may have arbitrarily lower memberships.
From these sets we produce

$$Rd = \{S_r \cap D_{CP}, S_{r6} \cap D_{CP}, S_{r6\text{-}4} \cap D_{CP},\}$$

A third rule may be simply to avoid too many repetitions of the same chord. The concept "too many repetitions" would be represented by a set like

$$T = \{\ 0\ 0\ 0.2\ 0.4\ 0.6\ 0.8\ 1\ 1\ 1\ \ 1\}$$

We then keep a running count of occurrences n of each chord in the last 10 and build the set

$$Rt = \{\mu Tm(n_{Sr}), \mu Tm(n_{Sr6}), \mu Tm(n_{Sr64})\}$$

To select the winning chord we combine Rcom, Rdes and the complement of Rt in some way. Simply adding the sets gives:

$$Rf = \{\mu Rc\ (r) + \mu Rd\ (r) + \mu Rt'(r),$$
$$\mu Rc\ (r) + \mu Rd\ (r) + \mu Rt'(r),$$
$$\mu Rc\ (r) + \mu Rd\ (r) + \mu Rt'(r)\}$$

The winning chord is then TOP[1] Rf, with no further processing.

More applications

The fuzzy process can be applied to any musical decision. Here is an outline for a simple phrase generator.

Phrase Generator

Interval direction

- If early in phrase go up
- If late in phrase go down
- If near top of range go down
- If near bottom of range go up
- If high in range go down
- If low in range go up
- If late in phrase and above attractor go down
- If late in phrase and below attractor go up
- Dither

Interval size

1. If too many small intervals pick medium

[3] The concept of fuzzy memberships representing possibility was introduced in [Zadeh 1978]

2. If too many medium intervals pick small or large
3. If too many large intervals pick medium
4. Dither

In this application the phrase length would be an input variable that would scale the indexing for the "where in the phrase" clauses, as well as determine the number of pitches generated. An "attractor" is some desirable target pitch, perhaps tonic. The calculation of start points and attractors would be fuzzy and related to the structure of the piece. Dither is a random fuzzy number that can be added to any decision to make the process less deterministic.

To produce a value from direction rule 1, you would define a downward sloping function for "early in the phrase" such as (to pseudocode):

```
Early[] = {1.0 1.0  0.8 0.6 0.4 0.2 0 0 0 0};
Direction = 0;
Direction -=  Early[where];
```

Then:

```
Late[] = {0 0 0 0 0.2 0.4 0.6 0.8 1.0 1.0};
High[] = {0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0.2 0.4 0.6 0.8 1.0 1.0 1.0 1.0};
Low[] = {1.0 1.0 1.0 1.0  0.8 0.6 0.4 0.2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0};
AboveAttractor[] = {0  0.2 0.4 0.6 0.8 1.0 1.0 1.0 1.0}; // shift to attractor
BelowAttractor [] = { 1.0  0.8 0.6 0.4 0.2 0}; //shift to attractor - 6


Direction += Late[where];
Direction -= High[lastnote];
Direction += Low[lastnote];
Direction -= min(Late[where] ,  AboveAttractor [Lastnote]);
Direction +=min (Late[where], BelowAttractor [Lastnote]);
Direction += random(0.5) – 0.25;    // dither
```

The direction will be used later in branching to interval choice. To choose an interval size, define:

```
TooMany = {  0 0.2 0.4 0.6 0.8 1.0 1.0 1.0};
SmInt = { 0 1.0 1.0 0.5 0.2 0 0 0 0 0 0 0};
MedInt = { 0 0 0 0.5 0.1 0.5 0 0 0 0 0 0};
LrgInt = { 0 0 0 0 0 0.5 0 1.0 0.5 0.6 0 0};
```

History of small, medium and large intervals are kept by

Histogram += {SmInt [lastInt], MedInt [lastInt],LrgInt[LastInt]};

Top(Histogram) and Direction will point to the direction and class of interval to use in a case statement. The choice within the class could be random or conditioned by more rules.

Rhythm Recognizer

The problem of parsing rhythm is central to the problem of transcribing music from real performances. There have been many crisp logic attempts that foundered on problems such as distinguishing dotted from triplet rhythms. The fuzzy approach is to classify pattern sets by the number of events during a beat. Fitting the times of the onsets against a set of fuzzy descriptions of all patterns (to a desired resolution) will quickly produce a best fit. Figure 6 show an example of this.

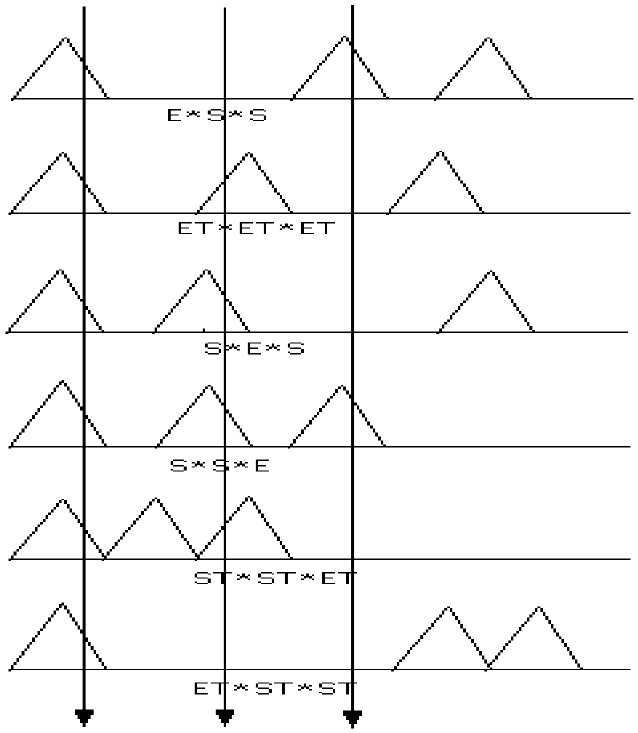

Figure 6.

The downward arrows in figure 6 represent the observed times. The triangles represent fuzzy sets for some[4] of the combinations of three events in a quarter note. The membership of a time in each set is the intersection of a line and triangle. A simple confidence value is generated by summing the memberships of the three times in each set. The best fit here is S*S*E.

Of course, this is after a process of identifying the beats by examining all notes according to a rule set that might go like this:

A note is on the beat if:

- It is close to the metronome tick.
- It is close to time predicted by previous beats.
- It is long in duration.
- It is accented.
- It is new chord.

And so on. After actual beats are found, "missing" beats are identified by interpolation.

These examples are only a sampling of opportunities for use of fuzzy logic in solving musical problems. In the author’s experience, no program will make exclusive use of a single approach: standard logic is the most efficient when the choices are clear cut, and some processes are best described by math functions. However, in the majority of cases where outcomes are traditionally determined by rules of thumb, fuzzy logic is the clearest route to the goal.

pqe

Bibliography

All works cited as in readings* can be found in

Dubois, D,H. Prade and R Yeager ed., Readings in Fuzzy Sets For intelligent Systems Morgan Kaufman, San Mateo CA, 1993.

Cox, E.D. Fuzzy Systems Handbook, AP Professional 1994

Dubois D. , H Prade, and C Testamale Weighted Fuzzy Pattern Matching. org pub 1983, readings* p. 112 ff

Dubois D. and H Prade, Fuzzy Numbers, An OverView. Org pub 1993, readings* p. 112 ff

[4] The complete group of sets is determined by the desired resolution.

Zadah, L.A., Fuzzy Sets as a Basis for a Theory of Possibility. org pub1978, reprinted in Selected Papers by L.A. Zadeh (ed Yeager, Ovichinnikov, Tong, Nguyen) New York: Wiley
Zadah, L.A., Fuzzy Sets. 1993, readings* p.27 ff