# A Fuzzy Expert System as a Stock Trading Advisor

Paulo E. Merloti

**Abstract**— this paper demonstrates a Fuzzy Expert System that works as a very simple trading system that receives buying or selling orders from a Fuzzy Expert System. Additionally, this paper also answers two other questions posed by assignment number two of class CS657.

— — — — — — — — ◆ — — — — — — — — —

## 1 INTRODUCTION

This paper is a report on the Assignment 2 for CS657 – Intelligent Machines and Systems for the Spring of 2005 semester.

This paper is organized in the following way: Section 2 is a discussion of consequences of setting some parameters in a fuzzification process such as set range, number of fuzzy sets, and overlap between sets. Section 3 is about the second problem posed in this assignment, where we exercise how to transform a crisp value into a fuzzy value, given a shaping function, a few fuzzy sets and their related range parameters.

Finally, section 4 is a presentation of FES, a Fuzzy Expert System that simulates a very simple stock trading system that decides how many stocks to buy or sell based on two input values (price and an indicator "MAD").

## 2 QUESTION 1: DISCUSSION ON FUZZY LOGIC

This question poses the following problem:

*"Suppose that the crisp variable x varies from Xmin Xmax, and is to be represented by a fuzzy variable.*

*a) What are the effects of increasing the number of fuzzy sets in the given range Xmax – Xmin?*

*b) If we increase the number of fuzzy set to a very large number what will result?*

*c) What are the effects of increasing the overlap between adjacent membership functions?"*

In addressing **item a)**, by increasing the number of fuzzy sets in the range Xmax – Xmin, we will cause the sets to be more specific (narrower) to a given point in the range, given that we keep the same overlap between sets. It will effectively reduce the probability of a value to be in one given set, as this set will cover less ground in the range (it will be squeezed by other sets). It will also cause the creation of more rules, and the increased resolution of the fuzzy associative memory. Rules will be more sensitive to a particular spot in the given range.

Regarding **item b)**, as we increase the number of fuzzy sets to a large number, sets will be less fuzzy (if we keep overlap constant) as each set will cover a very narrow range of values. Furthermore, one of the biggest advantages of fuzzy sets is the possibility of giving linguistic descriptors to fuzzy sets. If the number of sets is radically increased, we will soon run out of linguistic descriptors in order to describe all sets. For example, what would be in between

the sets "very very big" and "ultra very big"? And more importantly, would it make sense?

For **item c)**, the effects of increasing the overlap between sets would be decreased difference in degree of membership between sets. It would increase the possibility of one crisp value to be in many sets at the same time. While this could make sense in some applications, in general it is not a good idea as the number of rules triggered could increase significantly, which could also increase the computational cost of the system.

## 3 QUESTION 2: FUZZIFICATION OF AGES

In problem number 2 of the assignment, we are given the following function[1]:

$$S(x,a,b,c) = \begin{cases} 0 & x \le a \\ 2\left(\dfrac{x-a}{c-a}\right)^2 & a \le x \le b \\ 1 - 2\left(\dfrac{x-c}{c-a}\right)^2 & b \le x \le c \\ 1 & x > c \end{cases}$$

Where x is the crisp variable and a, b, and c are constant parameters with a<b<c. The "Z-Shaped" function is the complement of function S(x,a,b,c) and is defined as

Z(x,a,b,c) = 1-S(x,a,b,c)

Then, based on these two functions, 4 fuzzy sets are defined:

*Kid:*      $\mu_{Kid}(x)$ = Z(x,0,10,20)

*Young:*      $\mu_{Young}(x)$ = Z(x,0,20,40)

*Middle Age:*   $\mu_{MA}(x)$ = S(x,30,45,60)

*Old:*      $\mu_{Old}(x)$ = S(x,40,60,80)

The question posed in this problem is applying fuzzy logical operators and finding the fuzzy value of the following 3 statements for a 10 years old person and a 45 years old person:

a) Young but not Kid
b) Middle age or old
c) Middle age but not young

---

[1] The problem originally states "x≥c", but x>c is more appropriate in this case as it eliminates ambiguity at "c".
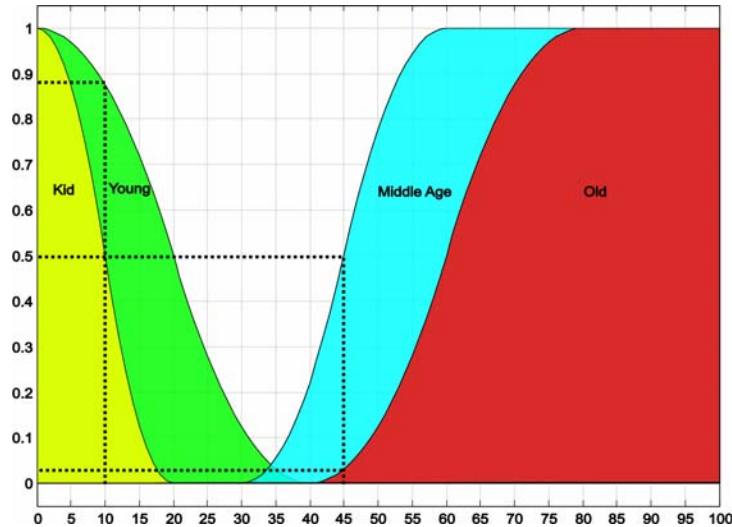
Fig. 1. Age fuzzy sets

Figure 1 show the given fuzzy sets on a graphic, assuming a range of 0 to 100 years.

For **item a)**, the expression "*Young but not Kid*" can be formally described as:

$\mu_{Young}(x) \wedge \neg \mu_{Kid}(x)$

Solving for x=10:

$\mu_{Young}(10) \wedge \neg \mu_{Kid}(10)$

$= MIN(\mu_{Young}(10), \neg \mu_{Kid}(10))$

$= MIN(Z(10,0,20,40),1-Z(10,0,10,20))$

$= MIN(1-S(10,0,20,40),1-(1-S(10,0,10,20)))$

$= MIN(1-S(10,0,20,40), S(10,0,10,20))$

$= MIN(0.8750,0.5000) = $ ***0.5000***

Solving for x=45:

$\mu_{Young}(45) \wedge \neg \mu_{Kid}(45)$

$= MIN(\mu_{Young}(45), \neg \mu_{Kid}(45))$

$= MIN(Z(45,0,20,40),1-Z(45,0,10,20))$

$= MIN(1-S(45,0,20,40),1-(1-S(45,0,10,20)))$

$= MIN(1-S(45,0,20,40), S(45,0,10,20))$

$= MIN(0.0000,1.0000) = $ ***0.0000***

Similarly, for **item b)** we will have the following (this time with less intermediate steps)

$\mu_{MA}(x) \vee \mu_{Old}(x)$

Solving for x=10:

$\mu_{MA}(10) \vee \mu_{Old}(10)$

$= MAX(0.0000,0.0000) = $ ***0.0000***

Solving for x=45:

$\mu_{MA}(45) \vee \mu_{Old}(45)$

$= MAX(0.5000,0.0313) = $ ***0.5000***

Finally, for **item c)** we have:

$\mu_{MA}(x) \wedge \neg \mu_{Young}(x)$

Solving for x=10:

$\mu_{MA}(10) \wedge \neg \mu_{Young}(10)$

$= MIN(0.0000,1-0.8750) = MIN(0.0000,0.1250) = $ ***0.0000***

Solving for x=45:

$\mu_{MA}(45) \wedge \neg \mu_{Young}(45)$

$= MIN(0.5000,1-0.0000) = MIN(0.5000,1.0000) = $ ***0.5000***

## 4   THE FINANCIAL FUZZY EXPERT SYSTEM

Here is the problem statement for financial fuzzy expert system:

"*Suppose that you are to design a fuzzy expert system for stock trading. The input to the fuzzy system is the current price of a company stock, and an indicator called MAD (moving average divergence). The fuzzy sets for the stock price are LO (low), MD (medium) and HI (high). Similarly the fuzzy sets for the MAD are N (negative), Z (zero) and P (positive). The outputs are BM (buy many), BF (buy few), DT (do not trade), SM (sell many) and SF (sell few). Suppose that there is a correlation between the price and the MAD indicator such that when MAD is positive today the stock price is expected to rise the next day, and when MAD is negative, the stock price is expected to fall the next day. Finally when the indicator does not show a noticeable trend, the next day price has no correlation with the MAD. However, there are sometimes exceptions to these rules due to political events and psychological mood of the market (e.g. casualties of occupation forces in Iraq , corporate scandals, homeland security department raising/lowering the alarm level, North Korea resuming nuclear activities, Israeli military occupying Palestinian cities, etc). The news about these events often produce random fluctuations in stock prices, which we will model as random numbers. Thus the stock share price p(i) and MAD denoted by m(i), at the open of the i-th day trading are modeled as*

$p(i) = 10 + 3sin(2\pi i/17)+0.6cos(2 \pi i/3)+\zeta(i)$

$m(i) = 0.3307cos(0.33i)-0.0542sin(0.33i)$

*where i=1,2,…,100 is the day number, $\zeta(i)=\rho(i)*(i\%4)$ and $\rho(i)$ is a random number between -1 and +1.*

*Design a fuzzy trading program to maximize your profit (or minimize your loss). Note that the fuzzy system can only find today's price and today's MAD and has no knowledge of future prices or future MAD given by the above equations. Simulate the system assuming that you can trade no more than once a day at the market open. You have $10,000 to invest, and the maximum number of trades each time is 1000 shares. You can also borrow on margin at 0.3% a day up to 70% of the current value of your stock holding (Example: you have spent all your 10,000 and bought stocks that are currently worth $8000. You believe that the price will go up so you decide to borrow the maximum amount which is $8000*0.70 = $5600 to buy more stocks. If you pay back this borrowed money after 8 days by selling stocks,*
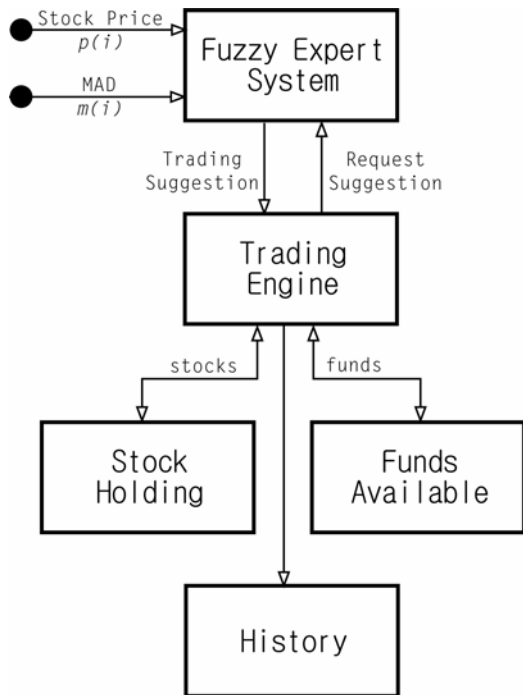
Fig. 2. FES Trading System block diagram

*then you will be charged interest $5600*8*0.003 = $134.4).*

It is important to distinguish the two main components of the system. The first one is the Trading System, responsible for buying, selling and managing everything that is related to the financial part of the system (stock holding, total assets, profits, losses, etc). The second one is the Fuzzy Expert System that like a Human expert, it is

asked what the best trade is for a given day, based on current stock price and a moving average divergence (MAD) index. This data flow of the system can be summarized in the block diagram of figure 2.

## 4.1 Architecture

Figure 3 shows the object model for the FES portraying the main classes, properties and methods of the system. For a detailed specification of methods, please refer to comments on the source code.

In this design, the main two classes are TTrade and TFuzzyExpert. They correspond to the "Fuzzy Expert System" and "Trading Engine" of figure 2. Following is a quick description of the core classes of this model:

**ClockObj**: It is a singleton available throughout the system and responsible for keeping the current day available to the GUI, Trade class and FuzzyExpert class. A class that wants to be informed of day changes must include an event method in the ClockObj broadcast list (TBroadcastList). When a day event occurs (day is advanced or maximum day is reached), all the objects in the broadcast list will be triggered. In this system, the Main Form (GUI) is the one responsible for advancing the days of the ClockObj.

**Main Form:** Allows the user to interact with the system.

**TTrade:** It is the class that performs stock trades (buy, sell) and keeps all the related values updated. The properties of the TTrade are valid only for the current day (ClockObject.CurrentDay). When TTrade is created, it includes itself into the ClockObj broadcast list, and every time a day is advanced, TTrade automatically performs a trade (PerformTrading). In this method, before buying or selling stocks, TTrade
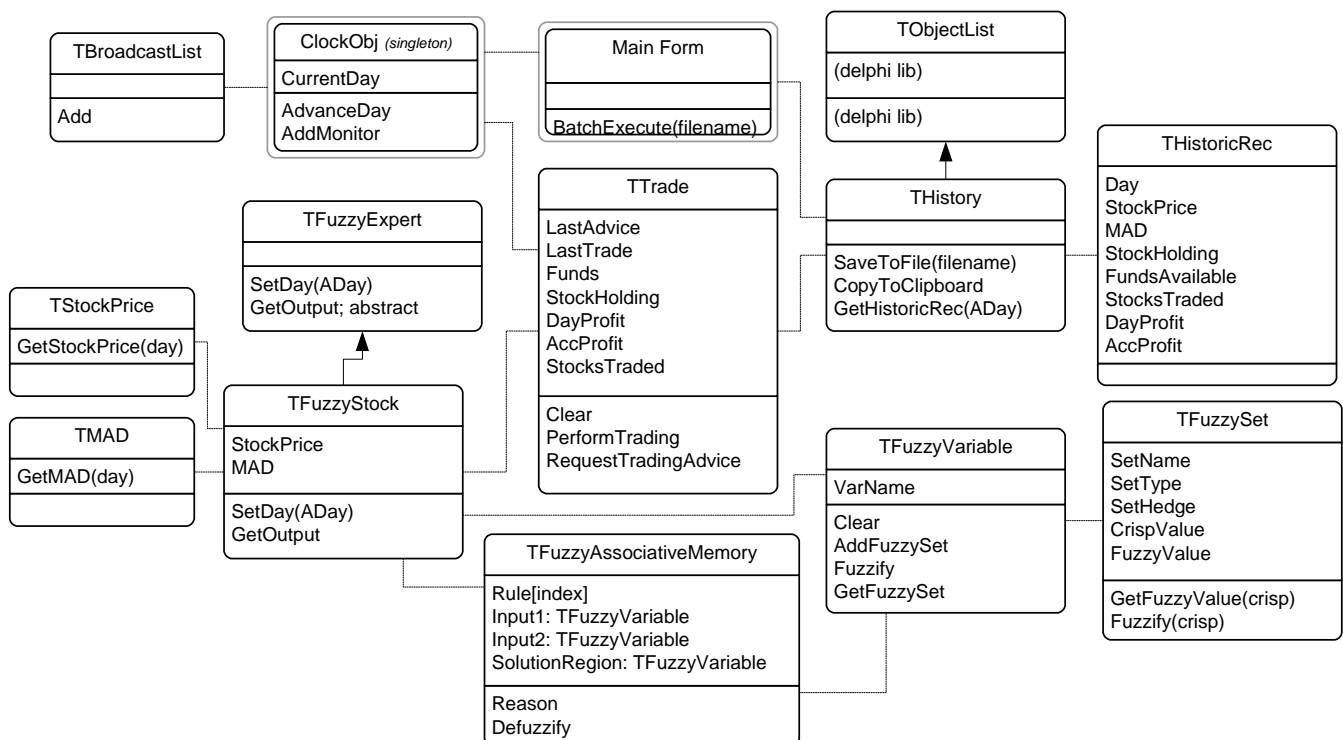


Fig. 3. FES Object Model

requests from TFuzzyStock a suggestion of how many stocks to buy or sell.

**THistory:** Stores a sequence of snapshots (THistoricRec) over time containing all the financial information for a given day (stock price, mad, funds available, stocks traded, day profit, accumulated profit and so on). It also implements methods for saving the history to a text file or copying it to the Windows clipboard.

**TFuzzyExpert:** Abstract class for a Fuzzy Expert System.

**TFuzzyStock:** Specialized fuzzy expert class for the stock trading system. When TFuzzyStock is requested to give advice, it retrieves the current stock price from TStockPrice and TMad (in a real situation, these two classes would lie outside the boundaries of the system and would be accessed through some defined interface) and returns a crisp suggestion of how many stocks to buy or sell based on the rules implemented on TFuzzyAssociativeMemory class and fuzzified stock price and mad value implemented by TFuzzyVariable.

**TFuzzyVariable:** Implements a fuzzy variable (input or output). In this system, there are three such sets: stock price (input), mad (input) and trading suggestion (output). This class is responsible for taking a crisp value and retrieving the degree of membership in each one of the fuzzy sets that compose that particular fuzzy variable.

**TFuzzySet:** Describes a fuzzy set with type (triangular, trapeze), and hedge values. It is also responsible for retrieving the fuzzy value related to a crisp value.

A basic sequence of events can be illustrated in the sequence diagram of figure 4. In that scenario, TTrade initiates the TFuzzyStock with a new day, which will cause the TFuzzyStock to request a stock price and mad value for that day. Following, TTrade requests a stock trade suggestion by invoking the method GetOutput from TFuzzyStock. In order to obtain the trade suggestion, TFuzzyStock first scales the stock price and mad values to the [0:1] range and then submit the crisp normalized values to be fuzzified by the respective fuzzy variable objects. After the inputs are fuzzified in the TFuzzyVar classes, the fuzzy associative memory is requested to reason (TFAM.Reason) over those fuzzy values. By looking at the internal rules table, TFAM will obtain the fuzzy output values related to the solution region. TFuzzyStock then
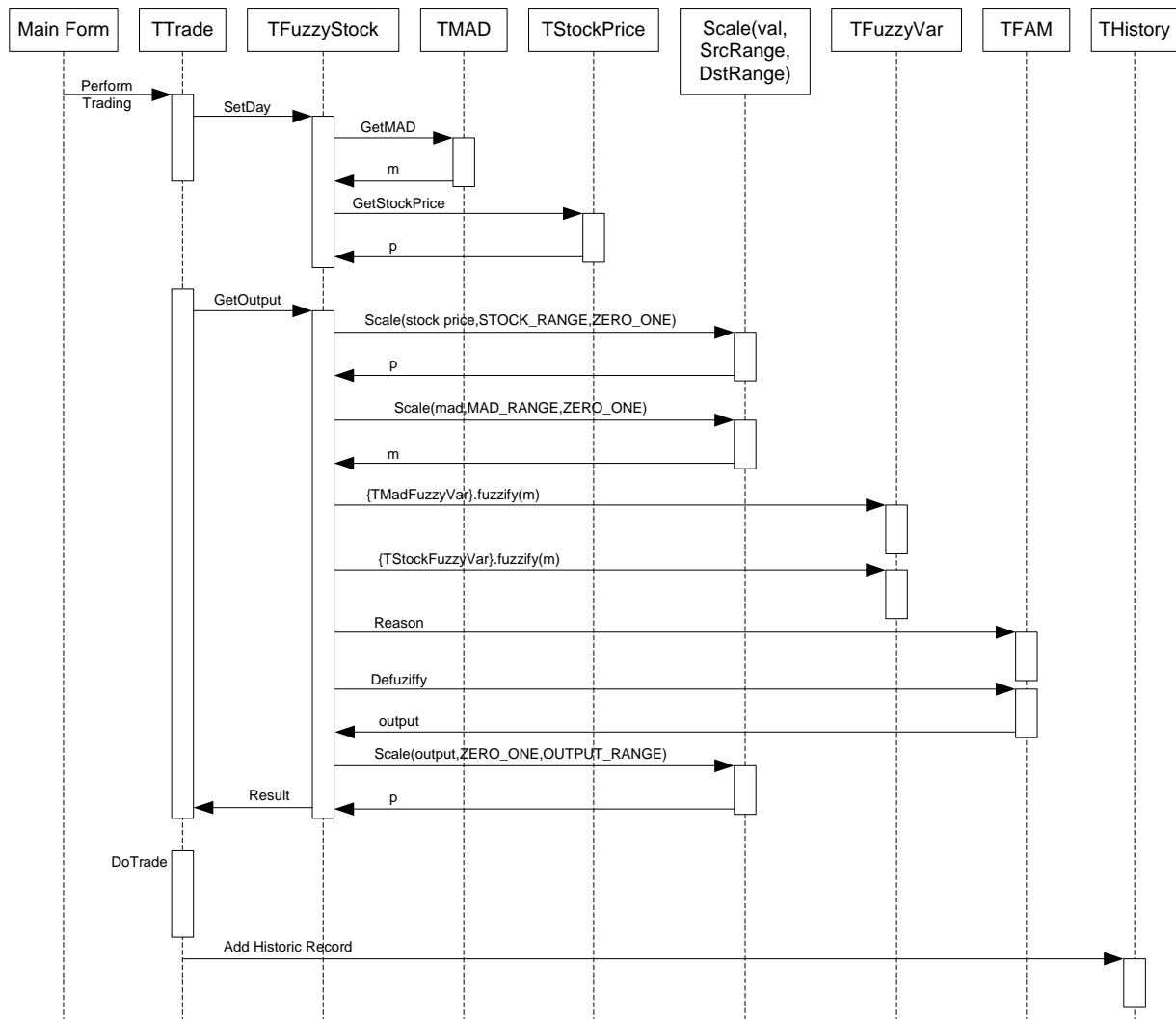


Fig. 4. Sequence diagram for performing a trade operation

commands TFAM to defuzzify the solution region (TFAM implements the centroid method) and return that output. The last step performed by TFuzzyStock is to re-scale the crisp output value that is in the [0:1] range back to the desired output range, that is returned to the TTrade class as the trading suggestion.

With that value in hands, the TTrade class performs the trade following the business rules. Not always it is possible to perform the suggested operation, either because there are no funds available or because it exceeds the operation limit for the day.

After the trade is performed, a record is added to the THistory list that will be used later for information purposes.

### 4.2 Algorithms

In this section we will show the basic algorithms applied in the Fuzzy Expert System Engine. These include the scaling algorithm, the fuzzification process for a fuzzy set, the reasoning and defuzzification algorithms for the fuzzy associative memory.

In listing 1, we can see the scaling algorithm. Usually inputs have different ranges, for instance the *stock price* may vary from $0.00 to $20.00 and the *mad* indicator can vary from -3 to +3. These differences in scale may present a problem for the inference engine when it's time to perform AND/OR operations. To avoid these problems, we have to normalize the inputs to a unique range. Also, after the reasoning is done and a normalized output value is found, we also translate it to a range that makes sense to the output unit. The scaling function needs 3 parameters, the value to be scaled, the source range and destination range. The calculation is performed based on the line equation with source range placed on the abscissa axis and the destination range placed on the ordinate axis. The output of the function is value in function of the line equation.

Listing 2 shows how fuzzification is performed. For this application, there are three possible types of fuzzy sets: left trapeze, triangle and right trapeze. It is assumed that slopes of the triangle and trapeze are linear. The hedge points are defined in a 3-element vector from left to right. For instance, if the set shape is of a triangle, there will be 4 possible fuzzy values: what it is to the left or to the right of the triangle is zero; if a crisp point falls into the triangle, then we calculate the fuzzy value according to the ascendant edge or descendant edge of the triangle, and if a crisp value is to the right of the triangle base then the fuzzy value is also zero. In a similar manner, to calculate the fuzzy value of a trapeze, we assume that for a left trapeze for example, there is nothing at this left. If a crisp point falls in the plateau area, fuzzy value is one. If it falls in the falling edge, the fuzzy value is calculated according to the equation of line of that slope, and if it falls to the right of a left trapeze, then the fuzzy value returned is zero. The same concept applies to the right trapeze. Note that calculating the equation of line in this case is done by utilizing the same scaling algorithm described in listing 1.

After fuzzy sets are obtained, the fuzzy associative memory (FAM) object performs the reasoning about those values, which is nothing more than triggering a set of pre-

```
Listing 1 – Scale Algorithm
function Scale(Value, Src, Dst): double;
begin
  m := (Dst.end-Dst.start)/(Src.end-Src.start);
  b := Dst.start - m*Srcstart;
  Return(m*Value +b);
end;


Listing 2 – Fuzzification Algorithm
function TFuzzyVariable::Fuzzify(crisp)
  for i := 0 to FuzzySets.Count-1 do
    FuzzySets[i].Fuzzify(crisp);
end;

procedure TFuzzySet::Fuzzify(crisp)

  case SetType of

  LeftTrapeze:
    begin
      if  (crisp >= Hedge[0])
      and (crisp <  Hedge[1]) then
        fuzzy := 1
      else if (crisp >= Hedge[1])
          and (crisp <= Hedge[2]) then
        fuzzy := Scale(
          crisp,
          Range(Hedge[1],Hedge[2]),
          ONE_ZERO)
      else
        fuzzy := 0;
    end;

  RightTrapeze:
    begin
      if crisp < Hedge[0] then
        fuzzy := 0
      else if (crisp >= Hedge[0])
          and (crisp <= Hedge[1]) then
        fuzzy := Scale(
          crisp,
          Range(Hedge[0],Hedge[1]),
          ZERO_ONE)
      else if (crisp > Hedge[1])
          and (crisp <= Hedge[2]) then
        fuzzy := 1;
    end;

  Triangle:
    begin
      if crisp < Hedge[0] then
        fuzzy := 0
      else if (crisp >= Hedge[0])
          and (crisp < Hedge[1]) then
        fuzzy := Scale(
          crisp,
          Range(Hedge[0],Hedge[1]),
          ZERO_ONE)
      else if (crisp >= Hedge[1])
          and (crisp <= Hedge[2]) then
        fuzzy := Scale(
          crisp,
          Range(Hedge[1],Hedge[2]),
          ONE_ZERO)
      else
        fuzzy := 0;
    end;
  end;

end;
```

defined rules according to the two fuzzy inputs (stock price and mad value). Listing 3 shows the reasoning algorithm. The FAM table is implemented as an array of 9 structures (record 0 to 8) with the following format:

*Antecedent1: string*
*Antecedent2: string*
*Consequent: string*

This record effectively represents a fuzzy rule of the type "IF Antecedent1 AND Antecedent2 THEN Consequent". Because we have 3 fuzzy sets for the stock input (LO, MD, HI) and 3 fuzzy sets for the mad input (N, Z, P), there are nine of such rules. The algorithm starts by going over each one of these rules and evaluating the antecedent part of the rule. The value obtained is associated with that rule for later processing. In the second stage of the algorithm, we go over each one of the rules again in order to aggregate one single value for each one of the output fuzzy sets. For example, the output fuzzy set "Buy Many" (BM) may have appeared in more than one rule and may have a different fuzzy value for each one of the rules. The final aggregated value for the BM fuzzy set will be the maximum value of all fuzzy values for that set defined in the RuleVal array (OR operation). The result of this algorithm is a solution region, composed by a series of output fuzzy sets (i.e.: BM, BF, DT, SF, SM) with a fuzzy value attached to each one of these sets.

Finally, the defuzzification algorithm is shown in Listing 4, and while it seems complex at first glance, it is really simple. The method utilized to retrieve a crisp value from the solution region is the "centroid". If you imagine the output fuzzy sets represented on a graphic, the centroid (COG) is the clipped area under the fuzzy set functions divided by the area under the fuzzy set functions, as seen below.

$$COG = \frac{\int_a^b \mu(x)x\,dx}{\int_a^b \mu(x)\,dx}$$

In our application, *a, b* are the starting and ending point or our output range respectively. $\mu(x)$ is the membership value at point x. We have chosen dx=0.01 for the discretization of this expression in algorithm 4. Note that in order to retrieve $\mu(x)$, we need to consider all the sets that overlap over x. We do this by OR'ing the membership values of the fuzzy sets over x.

### 4.3 Software Features

The Fuzzy Expert System application (FES) was developed in Delphi 6 with no third-party components. The choice of using Borland's Delphi comes from its extremely rapid application development cycle, much optimized native code that compares to machine level language and powerful graphic user interface development. The software can be started in two ways, the regular Windows double-clicking or by a command prompt call. When launched using the command prompt, the application expects a input parameter containing the output file name. No user interface will be shown, and a text file with the history of

```
Listing 3 – TFAM.Reason
procedure Reason;
begin
  SolutionRegion.Clear;

  //first calculate fuzzy value for each rule
  //in the FAM table
  for each Rule r in the FAM table
  begin
    a1 := StockVariable.
      FuzzySet(Rule[r].Antecedent1).
      FuzzyValue;
    a2 := MadVariable.
      FuzzySet(Rule[r].Antecedent2).
      FuzzyValue;

    //AND of two antecedents
    RuleVal[r] := Min(a1,a2);
  end;

  //then aggregate output sets into solution
  //region
  for each Rule r in the FAM table
  begin
    uRule := RuleVal[r];
    uSet  := Get output fuzzy value for
             consequent fuzzy set of Rule[r];
    //effectively OR'ing the several rules that
    //mention a given output fuzzy set
    if uRule > uSet then
      SolutionRegion.
        FuzzySet(Rule[r].Consequent).
        FuzzyValue := uRule;
  end;
end;
```

```
Listing 4 – TFAM.Defuzzify
function TFAM::Defuzzify: double;
begin
  sum := 0; wSum := 0;
  nOfSets := SolutionRegion.FuzzySets.Count;

  //we divide the solution region in 101 bins
  for i := 0 to 100 do
  begin
    maxFuzzy := 0;

    //for each bin, we get the maximum (OR)
    //membership value of Fuzzy sets at point i
    for j := 0 to nOfSets-1 do
    begin
      curve  := SolutionRegion.
                FuzzySets[j].
                GetFuzzyValue(i/100);
      capped := SolutionRegion.
                FuzzySets[j].
                FuzzyValue;
      //retrieve the membership value for fuzzy
      //set "j"
      fuzzyVal := Min(curve,capped);
      //OR'ing the membership values of
      //different fuzzy sets at bin "i"
      if fuzzyVal > maxFuzzy then
        maxFuzzy := fuzzyVal;
    end;

    //integrating
    sum  := sum + maxFuzzy;
    wSum := wSum + (maxFuzzy * (i/100));
  end;

  Return(wSum/sum);
end;
```
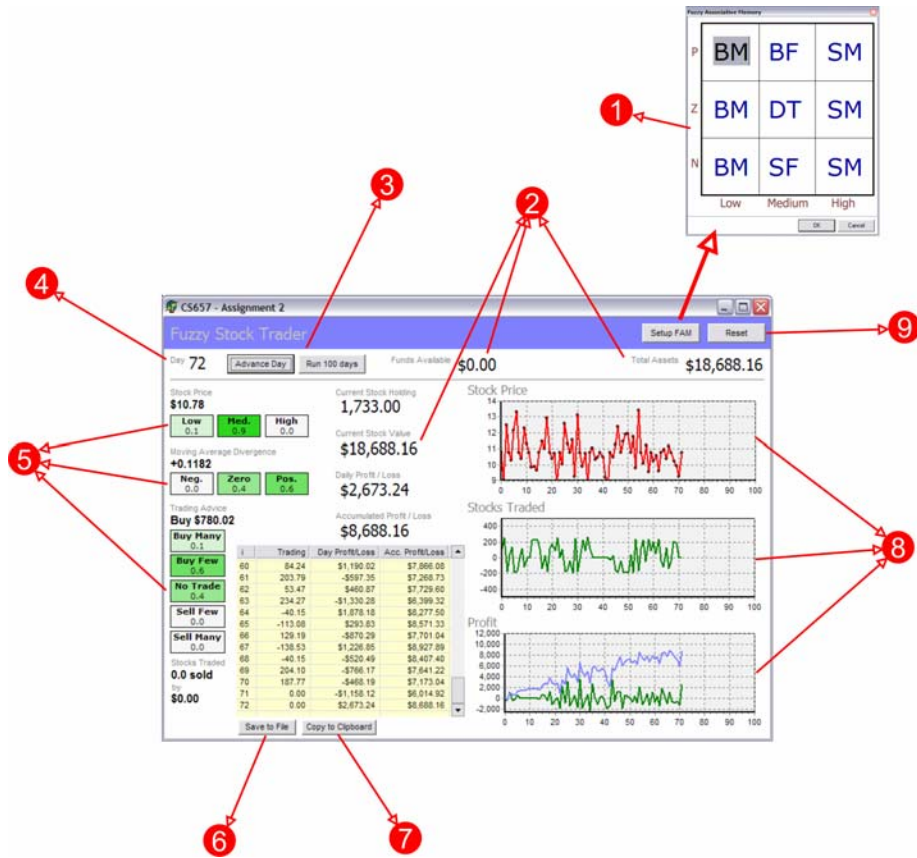
Fig. 5. FES features

transactions will be created in the directory and file name specified by the command line parameter, as shown in the following example:

```
FES "c:\temp\xp1.txt"
```

The above line will cause FES to generate an output file "xp1.txt" in the "c:\temp" local folder using the default FAM.

When the user launches the application through Windows, the user interface depicted in fig. 5 will show up. FES presents the following features:

1. Customizable FAM table.
2. Indicators of the current situation.
3. Control buttons to either advance the clock one day or run 100 days in a batch.
4. Day indicator
5. Gradient fuzzy membership indicators.
6. Save history to file (format given in the course webpage)
7. Copy history to clipboard. (Microsoft Excel compatible)
8. Graphics for daily stock price, amount of stocks traded and profit/loss. In the profit/loss graphic, the green series shows daily profit/loss, while the blue series shows accumulated profit/loss.
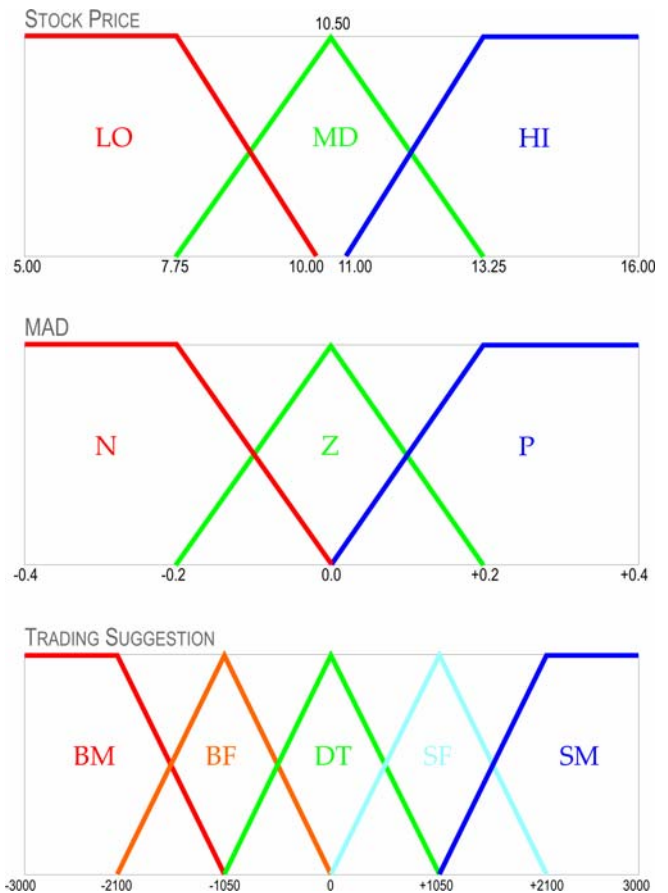9. Reset Button



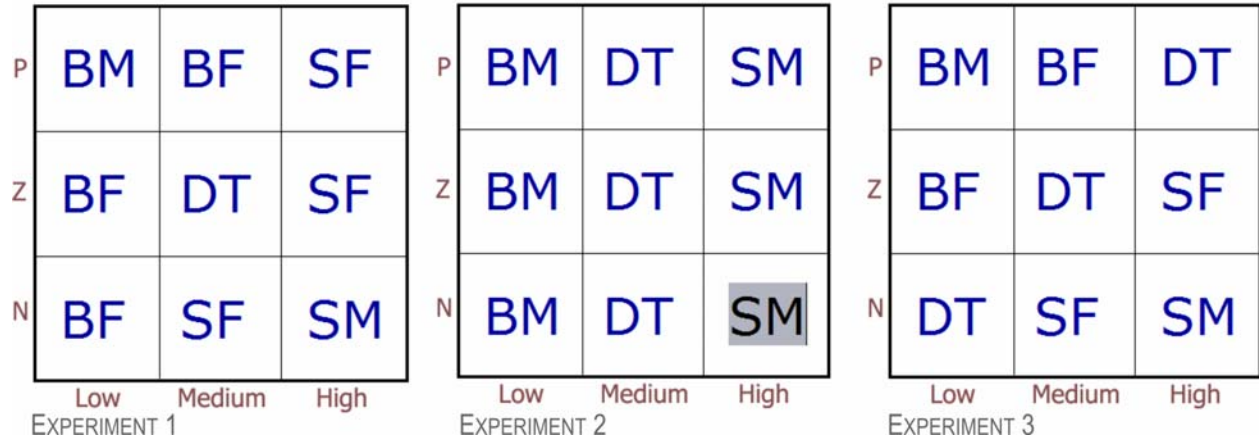Fig. 6. Fuzzy variables and fuzzy sets

Fig. 7. Fuzzy associative memory (FAM) tables for each one of the experiments

### 4.4 Experiments

Before performing the experiments, it is necessary to reveal the setup of the system so results may be reproduced. In the experiments, FAM table were varied. Although equally important, the scaling and distribution of fuzzy sets over fuzzy variables was kept constant. This system has three fuzzy variables, being two inputs and one output. Their respective fuzzy variables are shown in figure 6. Informal experiments demonstrated that results are very likely to be affected by the adjustment of these parameters.

Each experiment consists of 10 runs of 100 days with a given FAM table. At the end of the run, we recorded the accumulated profit for that run. In Table 1 we see the average accumulated profit of 10 runs for each experiment and its standard deviation. While recording the final profit at the end of the 100th day is ok for this application, more thought should be put in the case of other applications. We noted that in some cases, two experiments may result in the same accumulated profit, but the profit curve along the 100 days was pretty different for two given experiments.

Figure 7 shows the FAM tables used in each one of the experiments, and figure 8 shows a chart of each run by experiment.

The logic behind experiment 1 (xp1) is that if stock is low today (LO) and *mad* indicates that it will be high tomorrow (P), then it's time to buy many (and perhaps sell tomorrow if *mad* confirms). At the opposite side of the table, if stock price is high today (HI) and *mad* indicates that it is going to be low tomorrow then it's time to sell. If stock price is low,

but *mad* indicates that tomorrow the price will either decrease or stay the same, then se buy only a few. The symmetry also applies for the HI side of the table. If stock price is medium today, then selling or buying will depend on what we think will happen tomorrow. If *mad* indicates price will increase then we buy a few. If price will decrease then we sell a few stocks, and if *mad* does not indicate anything and stock price is medium, then the suggestion should be for not trading stocks. Actually, this is the only entry for not trading (DT) in experiment 1.

Experiment 2 is designed to test the hypothesis that in simple systems like these, with high degree of randomness, the rule of thumb of stock market still applies: sell high, buy low. In this experiment we eliminate the *mad* dimension altogether by setting BM when stock price is low, DT when it's medium and SM when it's high.

Experiment 3 is very similar to Experiment 1, except that when stock is low today and we think it is going to be even lower tomorrow, we do not trade and wait for tomorrow. The same is true when the stock price is high today and *mad* indicates that it is going to be even higher tomorrow, in this case we just idle.

### 4.5 Conclusion

In this paper we demonstrated a very simple fuzzy expert system for trading stocks. The environment proposed is a very simplified model of the real world stock market, and stock prices are manufactured to simulate the stock market environment.

Three experiments were performed using different sets of fuzzy rules, and we saw that the best performing experiment (the one that regularly resulted with the higher accumulated profit) was experiment number 2. Curiously xp2 is the one that does not uses the *mad* indicator. Experiment 1 and 3 had results very similar, but xp1 performed better than xp3. One can infer that such difference comes from the number of trades performed during the virtual time allotted for the experiment (100 days). The FAM table for experiment 3 has three entries with DT, while xp1 only has the central cell set as DT.

TABLE 1
EXPERIMENT RESULTS

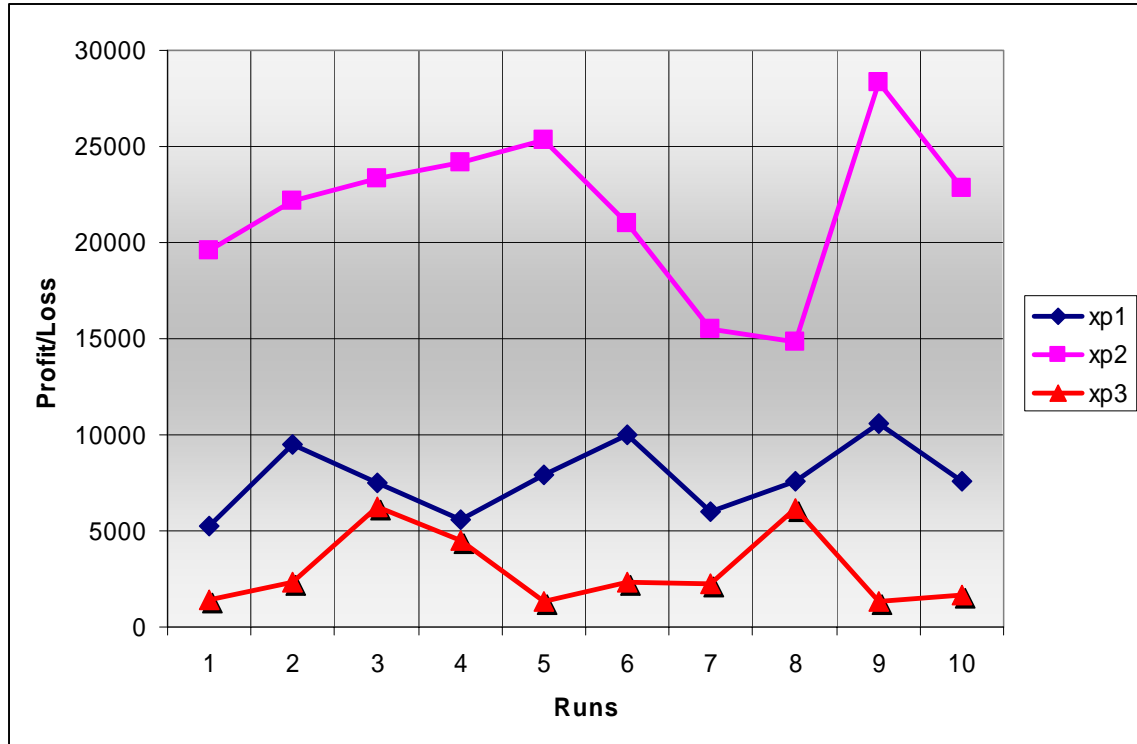| Experiment | Avg. Accumulated Profit | σ |
|---|---|---|
| xp1 | 7756.22 | 1834.96 |
| xp2 | 21,713.21 | 4179.71 |
| xp3 | 2951.39 | 1939.60 |

Fig. 8. Performance chart of the three experiments

The excellent performance of xp2 didn't come as a surprise, as we know that the golden rule of the stock market is buy when it is low, and sell when it is high. The only problem with this rule in the real world is that there are no upper boundaries on the stock price range as we have in this toy application, which makes things more complicated. By relying on an indicator such as *mad* to guide our decisions today may be dangerous if there is no real basis that the index really forecasts the future, it may bias our decisions towards undesired trades.

Finally, we have tested three configurations of FAM tables, with different performance obtained from each experiment. Although xp2 was the best performer among the three experiments, we don't know if it is definitely the best one. The only way to discover the best configuration of the FAM table would be trying all the possibilities, but even in a simple application like this, the task becomes very time consuming (for a 3x3 grid of 5 possible values for each grid, there are almost 2 million combinations). As a suggestion for future study, one approach for the discovery of a quasi-optimal solution would be genetic algorithms. If we imagine the 3x3 table as a vector of 9x1 elements, then this vector could be interpreted as the chromosome, and each gene would have a possibility of 5 values instead of the traditional binary values. For each generation, a population individual would run "his version" of the experiment a number of times, and the fitness function could be the average accumulated profit of a series of runs. This approach was used with success in other areas of AI such evolving Cellular Automata with Genetic Algorithms. [1]

## REFERENCES

[1] M. Mitchell, J. P. Crutchfield, P. T. Hraber. "*Evolving Cellular Automata to Perform Computations: Mechanisms and Impediments*" SFI Working Paper Abstract, Santa Fe Institute, 1993.