Data Classification using Fuzzy Rule-Based Systems represented as Genetic Programming Type-Constrained Trees

Athanasios Tsakonas[1], George Dounias[1], Hubertus Axer[2], Deidrich Graf v.Keyserlingk[2]
[1]Dept.of Business Administration,University of the Aegean,
Michalon 7,82100 Chios, Greece
Phone:+30-271-35000, Fax: +30-271-35099
Email: tsakonas@stt.aegean.gr, g.dounias@aegean.gr
[2]Dept.of Anatomy I,RWTH Aachen,
Pauwelsstr. 30, D-52057 Aachen,Germany
Phone +49-241-8089100, Fax +49-241-8888431
Email:{hubertus,keyser}@cajal.medizin.rwth-aachen.de

Abstract
This paper presents a genetic programming implementation of fuzzy rule based systems for data classification. The genetic programming is used not only as a variable-length search mechanism but the fuzzy rule-based functionality is incorporated inside the genetic programs. The model is tested on real world data from the medical domain and is compared to other computational intelligent approaches. Results denote the operability of our model and seem promising in domains where a large parameter space or incomplete domain knowledge restrains the user from applying well-defined fuzzy classifiers.

Keywords: *Fuzzy Rule-Based Systems, Genetic Programming, Data classification, Aphasia*

1.Introduction

Since their first emergence, the fuzzy rule-based systems were used extensively in many areas of human activity. Their success can be valued to the existing real-world data fuzziness, which binds the non-fuzzy approaches from being effective. The simplest fuzzy rule-based classifier can be considered the *Mamdani* model using the *min-max* criterion [Mamdani and Assilian 1975]. This model can be considered as a set of competitive rules. Each rule consists of an *antecedent set*, a *fuzzy inference system* and a *consequent set*. Antecedent sets are used to fuzzify the inputs -using *membership functions*-, namely to translate a number to a fuzzy linguistic term. The fuzzy inference system assigns the proper value to the rule -called *firing strength*- and the consequent set is used to characterize the output with a fuzzy linguistic term. When a fuzzy classifier is used, the consequent set assigns a specific class to the output. The rule with the maximum *strength* is supposed to *fire*, namely to give the decision output. While simple to implement, the fuzzy rule base has to be provided by a training procedure when domain knowledge does not exist. Various computational intelligence-related methods have been developed for this reason, including neural [Jang 1998], genetic algorithms [Herrera and Verdegay 1996] and hybrid or heuristic methods [Nauck and Kruse 1996]. Specifically, genetic algorithms (GA) [Goldberg 1989] have been used either for the determination of the rule bases or the membership functions or both [Li and Ng 1996]. Genetic algorithms have been proved capable of solving large scale or complex problems and they are commonly used as search mechanism when direct search is impossible. The main concept is the maintenance of a population of candidate solutions evolving through genetic operations. An extension to the inspiration of genetic algorithms is the Genetic Programming (GP) advance [Koza 1992], where the main problem of GAs, concerning the fixed problem definition is avoided by using variable-length trees instead of fixed-sized individuals. Moreover, the GP theory enabled the use of functional tree-nodes, which opened a new world of computational intelligent approaches like the symbolic regression problem solving [Koza et al 1999]. Consequently, genetic programming was used for the training of fuzzy rule-based systems [Alba et al. 1996a]. In their excellent work, [Alba et al. 1996b] describe the use of the GP as a search mechanism - in the *cart-centering problem*-, in order to produce valid fuzzy rule-based systems, which are directed then to a fuzzy controller. The ability however of the GP to maintain functional nodes inside the program structure, enables us to incorporate the functionality of a fuzzy rule-based system directly into the GP-tree architecture. Thus, a GP-program can behave like a fuzzy rule-based classifier. In this paper, this latter approach is presented together with a real world example application. We implemented the BNF-Grammar in a way that satisfies the grammar used in [Alba et al. 1996b]. The GP tree-nodes were given functionality, aiming both at simulating a Mamdani FRB model using the min-max criterion, and at producing a useful, for the determination of the fitness value, output, during the training phase. The real world application is exercised in the medical domain, demonstrating the capability of the GP-training approach to apply feature extraction and generate fuzzy rule bases, as well as the functionality of the FRB incorporation in genetic programming trees. The paper is organized as follows: in the next section we describe the main characteristics of the Genetic Programming. Section 3 presents the fuzzy rule-based systems emphasizing in Mamdani classifiers. The implementation of these classifiers in a Genetic Programming tree structure is demonstrated in Section 4. Section 5 shows a real-world application of the model. Discussion follows in section 6, and, finally, our conclusions are presented in section 7.

2. Genetic Programming (GP)

Genetic algorithms in general, are inspired by the Darwinian mechanism, which is responsible for the nature evolvement. They are different than the direct search methods, where one candidate solution is optimized through succeeding repetitions of the algorithm. In genetic algorithms, a set of solutions, called *population*, is maintained, and the optimum solution is found by subsequent genetic operations among the members of the population, which are called *chromosomes* or *individuals*. These individuals are evaluated according to the problem and a *fitness measure* is given to them. In the classic (generational) genetic algorithms, two populations are used, one for buffering the current population and the second for buffering the next population. When all individuals have been evaluated, a *generation* is supposed to be complete, the next population becomes the current and the procedure continues. In *steady-state* genetic algorithms only one population is maintained and each new individual with better fitness is inserted to the current population taking the place of a less fitted individual. The genetic operations may differ slightly between various implementations, but they can be classified in three major types: *selection (or reproduction)*, *crossover* and *mutation*. The selection mechanism ensures that good individuals (namely having large fitness) will be preferred among less fitted members. There is a variety of methods for selection, such as stochastic sampling with/without replacement, tournament with/without elitist strategy, etc. Crossover is the operation of exchanging genetic material between two selected individuals, called *parents*, in order to produce two new individuals, called *offsprings*. Mutation is the random substitution of a part of the genetic material of a selected individual. When dealing with genetic algorithms, one has to determine a fixed-sized array representing the chromosome. Nevertheless, often the solution size is unknown. This drawback of genetic algorithms has led the research in extensions such as the *messy genetic algorithms* and *the genetic programming*. In the latter concept, which is dealt in this work, individuals are represented as tree structures with variable length. While we may represent a function as an individual, a new methodology for the development of symbolic regression systems has evolved. In genetic programming, a tree node can symbolize a more or less primitive function such as addition, multiplication, boolean algebra operators etc. In *Figure 1,* it is shown such an individual representing a simple numerical function.

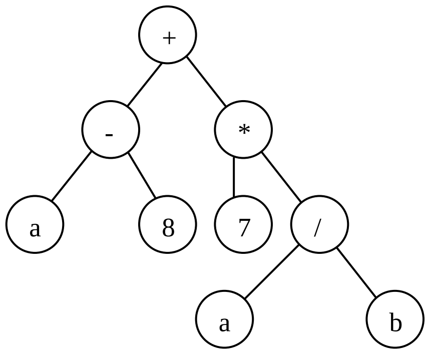

Figure 1. Tree representation of the program (expression): (a-8)+7*(a/b)

Crossover in genetic programming is the exchange of sub-trees between two selected parents. Mutation may happen with two ways: the first resembles to the mutation of genetic algorithms, where a node is substituted randomly. The other, more advanced, way, called *shrink mutation* [Singleton 1994], selects a sub-tree of an individual and promotes it randomly to a parental node of the same individual. As it is seen in *Figure 1*, two types of nodes exist, which determine correspondingly two sets. *Functional* nodes are nodes having arguments (e.g. the subtraction symbol in *Figure 1*), and *terminal* nodes are nodes, which do not take arguments (e.g. the *a* variable in *Figure 1*). Although this simple characterization is sufficient to produce numerical expressions, there are cases where a stricter tree hierarchy has to be defined. Hence, a grammar is usually adapted, which guides the genetic operations in order to ensure the validity of new individuals. In this paper, such a grammar is used for the representation of a fuzzy rule-based classifier into an individual.

3. Fuzzy Rule-Based Systems (FRBS) for data classification

Fuzzy sets are an extension to the classic sets, which have a crisp boundary. According to [Zadeh 1965], fuzzy sets play an important role in human thinking. In fuzzy sets, the transition for a value from belonging to a set and not belonging to the set is gradual and characterized be the membership function [Jang et al. 1997]. A fuzzy set is defined as:

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

where the $\mu_A(x)$ is a *membership function* for the fuzzy set *A*. The membership functions are described as a mathematical formula. The *X* is called the *universe of discourse*, and it may be comprised by discrete or continuous values. When the universe of discourse X is a continuous space, several fuzzy sets are used, most times covering the X uniformly. These fuzzy sets often are given linguistic terms such as "Small" or "Medium", thus they are called *linguistic variables*. These linguistic variables are used in fuzzy rules, which are interpreted as fuzzy relations using *fuzzy reasoning*. Fuzzy reasoning contains inference rules, which derive conclusions from a set of fuzzy rules and input data. A fuzzy if-then rule can be in the form:

$$if\ x\ is\ A\ then\ y\ is\ B$$

where the "x is A" is the antecedent (or premise) set, and "y is B" is the consequent (or conclusion) set. In fuzzy reasoning, the traditional two-valued logic, the *modus ponens*, is used in a generalized form. Namely, a fact may be more or less true based on the truth of another fact. In the Mamdani classifier model, using the max-min composition, several steps are followed to perform fuzzy reasoning. Firstly, we compare the input data with the antecedent sets of the fuzzy rules and we get the degrees of compatibility (called *weights*) with respect to these antecedent sets. Then, we combine these degrees using fuzzy “AND” or “OR” to obtain a *firing strength*, which shows the degree that a rule is satisfied. The max-min criterion, when only “AND” operators are used, will assign as firing strength the smaller of the antecedent degrees of compatibility. Finally, we obtain the overall output between the consequent sets of the rules. When the max-min composition is used, the rule with the larger firing strength will be the system's output. This fuzzy rule-based model is adapted throughout this paper.

4. FRBS for data classification in GP architecture

For the incorporation of fuzzy rule-based systems into a genetic program, several steps must be accomplished:

a) A BNF-grammar is constructed to guide the programs’ architecture.
b) The nodes of a program are active, by means of implementing a fuzzy inference rule-based system.

The attribute (a) is demonstrated in previous work [Montana 1993], [Alba et al. 1996a], including a genetic programming implementation for the construction of fuzzy rule bases [Alba et al. 1996b]. However, in the latter approach, the constructed trees are driven in a fuzzy controller. Thus, due to the (b) attribute of our system no separate fuzzy system is needed for the evaluation of the trees. Such an implementation has to follow various specifications in order to ensure that:

a) The resulting tree can be directly implemented later in a fuzzy controller or fuzzy software, namely it builds competitive fuzzy if-then classifying rules.
b) The determination of the fitness value would enable fast and accurate solution.

Therefore, the following parameters were built. The definition of the grammar is shown in *Table I*. This grammar describes a fuzzy system model with four inputs and one output. In this example, one of the inputs is considered to have different value range than the others. Thus, two groups of antecedent sets (group A and group B) are constructed, each of them covering the different value's range of the according inputs, with a different also number of antecedent sets.

Table I. BNF grammar of program trees. The trees are described in a prefix notation. Words in bold denote valid program nodes.

Grammar used for the GP Tree	
<TREE>	<RL> |<RULE>
<RL>	**RL** <TREE> <TREE>
<RULE>	**RULE** <COND> <CLASS>
<COND>	<IF_A> | <IF_B> | <AND>
<IF_A>	**IF_A** <INP_A> <FS_A>
<IF_B>	**IF_B** <INP_B> <FS_B>
<AND>	**AND** <COND> <COND>
<CLASS>	**THEN** <OUT> <CLASS_VALUE>
<FS_A>	**S_A | M_A | L_A**
<FS_B>	**VS_B | S_B | M_B | L_B | VL_B**
<INP_A>	**X1**
<INP_B>	**X2 | X3 | X4**
<CLASS_VALUE>	**CLASS1 | CLASS2 | CLASS3** …
<OUT>	**Y**

Based on the above grammar, a sample program is shown in *Figure 2*. The contour section corresponds to the following rule:

IF *X1* is *LARGE* **AND** *X2* is *SMALL* **THEN** *Y* is *CLASS1*

In order to implement a *Mamdani* fuzzy model with the *min-max* criterion, we select the minimum weight of antecedent sets for each rule and the maximum weighted rule to fire. The functions used to describe the fuzzy mechanism correspond to the words with bold in Table I. In Table II we present their working.

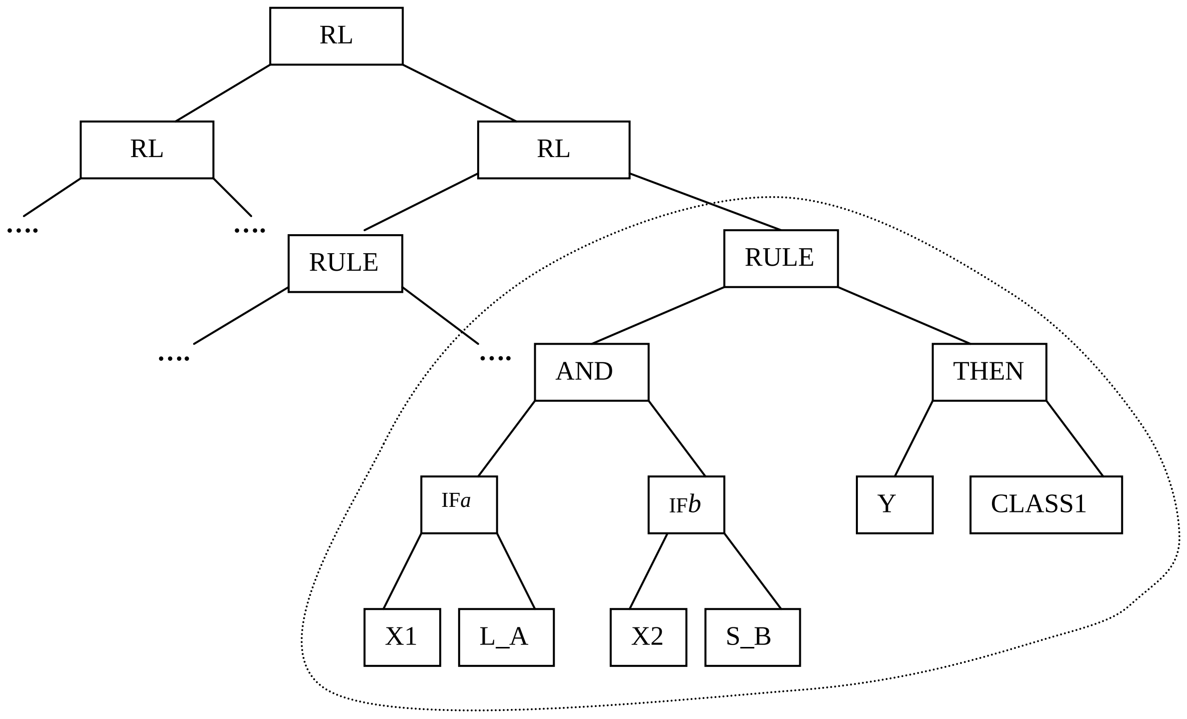

Figure 2. Genetic Programming tree architecture of fuzzy rule-based classifying systems.

Table II. Functions adapted in genetic programming implementation for the simulation of a Mamdani-model classifier behavior.

Function	Operation
RL (arg1,arg2)	If absolute(arg1)>absolute(arg2) then return arg1; else return arg2
RULE(arg1,arg2)	Return arg1*arg2
IF_A(arg1,arg2)	Fuzzify (arg1), based on the (arg2) value, return weight
IF_B(arg1,arg2)	Fuzzify (arg1), based on the (arg2) value, return weight
AND(arg1,arg2)	Return minimum(arg1,arg2)
THEN(arg1,arg2)	If arg1=arg2 then return 1; else return -1
L_A, M_A, L_B, etc.	Return a constant value (e.g 0 for L_A, 10 for M_A, 500 for L_B etc.).
CLASS1, CLASS2, etc.	Return a constant value (e.g 1 for CLASS1, 2 for CLASS2, etc.).
X1,X2, etc.	System inputs (assuming a numerical value)
Y	System output (assuming a numerical value)

It is worth to note that the fuzzification happens in *IF_A* and *IF_B* nodes. For example, if the implementation uses Gaussian membership functions -just like the example in *Section 5*-, then for a given Gaussian range *a* (standard for each of the *IF_A* and *IF_B* nodes) , a center *c=arg2* and a value *x=arg1*, the node output will be the following:

$$e^{-\frac{1}{2}\left(\frac{x-c}{a}\right)^2}$$

The *THEN* node returns 1 if for the examining example the output (*arg1*) belongs to the class described by *arg2* and -1 else. The reason to use this mechanism, together with the *RL* node working, is to be able to know (when the tree evaluation is complete) whether the fired rule was true or false. If the fired rule describes a false consequent set, the program value will be negative. While a program describes a full rule base, during the training will produce either positive or negative values based on the training set records. The fitness value is comprised by the following equation:

$$F = \sum_{t=0}^{n-1} (1 : f_t > 0 \,|\, 0 : f_t \leq 0) \qquad (1)$$

where, F is the program fitness, *t* is a record in the training set, *n* is the number of training records, and f_t is the program output for the record *t.*

The implementation in this paper was accomplished in a steady-state, grammar-driven genetic programming algorithm, using tournament selection. The steady-state approach for this grammar-driven model has shown within previous work [Alba et al. 1996b], faster search of the solution space, in contrast to the classic (generational) genetic programming. It is also less greedy in computer sources, making possible the adaptation of larger populations. The tournament selection is an efficient way of selection and is commonly used in steady-state genetic algorithms. The operators used were: reproduction, crossover, node mutation and shrink mutation. Due to the considerably large parameter space, mutation plays an important role in our model, while in [Alba et al. 1996b], only the crossover operator is necessary. Shrink mutation was proved valuable in reducing the code bloat caused by crossover operations, which often drove the solution in self-repeating rule bases. A Kill tournament is also used which replaces the worst of two randomly selected individuals. It was included a maximum age (set at 10 generations) for a program before substitution even if this is the best solution, in order to increase population divergence by avoiding the dominance of one good individual. The initialization of the population was random with respect to the grammar rules. The applied restrictions were:

a) During initialization, the root node must be a <TREE> type node
b) Crossover is allowed only between equal type nodes
c) Node mutation is allowed only in terminal nodes and the new value must be of the same type
d) Shrink mutation is allowed only between <TREE> type nodes, or between <COND> type nodes

These rules were applied to the example application, which follows in the next section.

5. Real-world example application

The problem used to test the efficiency of the model was the diagnosis of aphasia's subtypes. Aphasia [Axer et.al 2000b] is a human disease, often due to brain damage. Its effects usually are the patient's

disability in the usage or comprehension of words, as well as difficulties in reading or writing or articulation. A physician in a free interview performs the clinical diagnosis of Aphasia. Major types of aphasia are *Broca's Aphasia* (or *Motor* or *Expressive Aphasia), Wernicke's Aphasia* (or *Sensory* or *Receptive Aphasia), Global Aphasia* (or *Total Aphasia), Anomic Aphasia* and *Conduction Aphasia.*
Standardized examinations have been developed for the diagnosis of Aphasia, such as the Aachen Aphasia Test (AAT), which is commonly used in German speaking countries [Huber et al. 1984]. Our system inputs are these AAT scores, in total 30 parameters, each of them corresponding to a different Aphasia's symptom. These parameters have different value ranges. Specifically, P0-P5 range from 0 to 5, T0, N0, C0 and V0 range from 0 to 100, T1-T5 range from 0 to 10, N1-N5, C1-C3 and B1-B4 range from 0 to 30, and V1-V2 range from 0 to 60. Since different physicians perform the score ratings, fuzziness in these ratings is to be expected [Jantzen et al. 2000]. This diagnosis of Aphasia's subtypes has gained recently the focus of various CI implementations [Axer et.al 2000a]. Thus, we intent here to use these data to construct a Mamdani fuzzy classifier, keeping finally only the determinant's features. Firstly, we selected to use Gaussian membership functions, due to resemblance to the normal distribution, which may portray the scoring's deviations from characteristic values. For inputs P0-P5 three Gaussian membership functions were used: Small, Medium and Large, having centers in 0, 2.5 and 5 accordingly, and a width of 2.5. For the rest of the parameters, 11 Gaussian membership functions were available by the grammar, each of them having width of 10 and center ranging from 0 to 100 (by every 10 values). The latter selection was done mainly for the demonstration of the model's efficiency, rather than describing domain knowledge on possible scoring concentrations. The system parameters are presented in *Table III*. In order to compare the model's efficiency with previous works in this domain, we selected to classify four of the major aphasia's subtypes: Broca's aphasia, Wernicke's aphasia, Global aphasia and Anomic aphasia. The training set size and the test set size were kept similar to these previous works.

Table III. The GP parameters

Genetic Programming parameters	
Available data records (concerning four major aphasia subtypes) :	145
Data used for training:	78 (54% of the available data, containing the four classes proportional)
Data used for testing:	67 (46% of the available data, containing the four classes proportional)
Population:	2,000 individuals
GP implementation:	Steady-state Grammar-driven GP
Selection:	Tournament with elitist strategy
Tournament size:	6
Crossover Rate:	0.6
Overall Mutation Rate:	0.4
Node Mutation Rate (proportional to overall mutation rate):	0.4
Shrink Mutation Rate (proportional to overall mutation rate):	0.6
Killing Anti-Tournament size:	2
Maximum allowed formula size:	200 tree-nodes per individual

Figure 3 shows the best individual's fitness during training. The results are presented in *Table IV*. They are presented together with previous research using neural networks [Axer et.al 2000a], and genetic programming solutions using crisp rules [Dounias et al.2002].

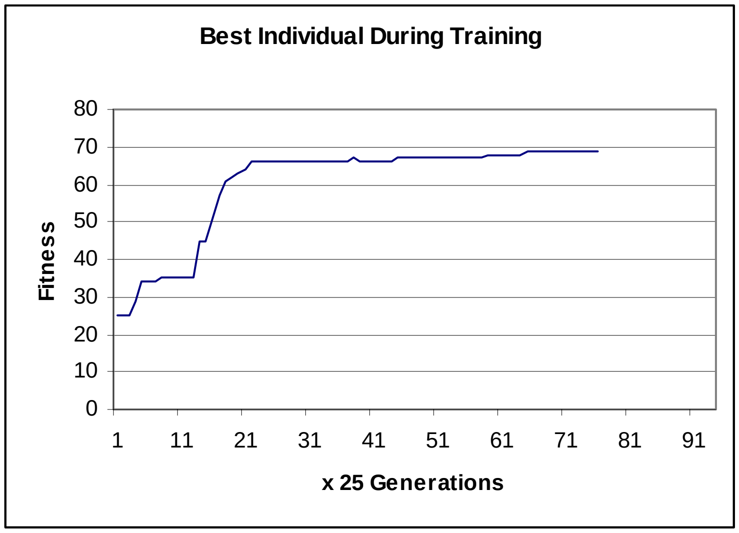

Figure 3. The best individual's fitness during training.

The solution presented here was accomplished after 1786 generations. It is a set of 11 fuzzy competitive rules in the *If-Then* form, which classified correctly the 88.5% (69 out of 78 cases) of the training set. The *Rx* fuzzy set's name, can be translated as *"is about X"*. For example, the *R40* fuzzy set has a meaning of *"is about 40"*, the *R0* fuzzy set has a meaning of *"is about 0"* etc. It is noted for the less experienced reader, that these rules are not meant to be used without a fuzzy classifier, since they correspond to a competitive fuzzy rule base, where fuzzification precedes and weights are assigned to each rule when examining a patient case. They can give though a rough explanation of the system's decision methodology.

IF B2 is R40 **AND** B1 is R40 **AND** V0 is R60 **AND** C3 is R0 **THEN** Broca Aphasia
IF B4 is R0 **AND** B1 is R40 **AND** C1 is R0 **THEN** Broca Aphasia
IF B4 is R0 **AND** C1 is R0 **AND** C1 is R0 **AND** B2 is R50 **THEN** Broca Aphasia
IF B1 is R40 **AND** C1 is R0 **THEN** Wernicke Aphasia
IF B4 is R0 **AND** B2 is R40 **AND** N0 is R100 **THEN** Wernicke Aphasia
IF V0 is R60 **AND** V0 is R60 **AND** B2 is R50 **THEN** Wernicke Aphasia
IF N2 is R40 **AND** B2 is R50 **AND** V0 is R60 **AND** B2 is R40 **AND** P1 is Medium **AND** C1 is R0 **AND** V0 is R40 **AND** C1 is R0 **THEN** Wernicke Aphasia
IF B2 is R40 **AND** V0 is R40 **THEN** Wernicke Aphasia
IF C0 is R0 **AND** P1 is Medium **AND** N2 is R40 **THEN** Global Aphasia
IF N0 is R100 **THEN** Anomic Aphasia
IF B2 is R40 **AND** B1 is R40 **AND** C1 is R0 **THEN** Anomic Aphasia

This fuzzy rule base classified correctly the 79.1% (53 out of 67 cases) of unknown data (test data). As it can be seen, these rules can be further simplified, while identical antecedent sets exist in different rules which classify the same class.

Table IV. Comparison between NN and GP models.

System	Error in test set
NN (spontaneous speech model)	14 %
NN (comprehensive model)	7.6 %
GP (simple crisp rules model)	20.2 %
GP (most accurate crisp rules model)	15.4 %
GP for FRBS (this paper's model)	20.9%

6. Discussion

As it is shown in *Table IV*, the results are competitive to those of previous GP-implementations. These implementations constructed a 3-crisp rule base. However, it was required to have separate training runs for

the extraction of each rule. The accuracy of these rules in the training set reached almost the 100% and fell in 79.8% and 84.6% in test set (~20% loss of accuracy). Since this paper's approach had 88.5% accuracy in training set and 79.1% in test set (~10% loss of accuracy), it seems that, in this domain, fuzzy rules provide more robust generalization. We suggest that with a more proper configuration of the membership functions and GP tuning, a higher classification score -in both sets- could be achieved easily.
The results remain however less accurate than the neural net models. Although promising for better accuracy, this paper's approach has an extra advantage over those of NNs, where prior domain knowledge is needed in general, and it was used in that case for the selection of inputs.
From equation (1) it is shown that we assigned the fitness value to the sum of correct classifications, rather than using the sum of weights. As the training proceeds, the best fitness is increased, having its upper limit to the total number of training records. Experiments have shown us that when the sum of weights is used as a fitness measure, the training procedure promotes a small number of strong rules, which prevent less strong but more descriptive rules to evolve. Thus, one could say that during the training phase, for a given classification score of an individual, its sum of weights must be small enough to permit the evolvement and the incorporation (by crossover) of rules describing other classes. The calculation of a more proper fitness value embedding the sum of weights remains in research though. Also, by setting negative fitness values to zero, we enable the survival of strong antecedent sets throughout training. These sets, may describe a different class, which may be found probably by later genetic operations.
Consequently, the optimal classification score is dependent on the proper selection of membership functions. To solve this problem, we may follow a variety of approaches. A histogram analysis of the values for each input may reveal areas where these values are concentrated. Alternatively, domain knowledge may help in the determination of membership functions. Another approach is to leave the genetic programming process to compute the membership functions. However, with the latter implementation, the search space for the GP increases dramatically, and we consider that it should be avoided if prior domain knowledge exists. An example of this approach could be a three-parameter function *IF_Gauss* (arg1, arg2, arg3) -instead of the *IF_A* and *IF_B* functions - where *arg1* is the input value, *arg2* is the center of Gaussian and *arg3* the width of the Gaussian. The arguments *arg2* and *arg3* may be derived by number nodes or as results of numerical operations implemented in sub-trees.

7.Conclusions and further research

In this paper, a genetic programming implementation for the search of fuzzy rule-based systems was presented. The aim of the FRBS was the data classification. Firstly, we implemented the structure of a Mamdani model of FRBS into GP functions using a BNF-grammar. Then we added functionality to the GP-tree nodes to simulate the Mamdani classifier output. The model was tested in real-world medical data. These results are encouraging for the effectiveness of this system. The main attributes of our model are both the feature extraction and the FRBS determination. Thus, an advantage over standard FRBS classifiers is that no prior knowledge is needed for the selection of proper system inputs. Hence, problems consisted of a large number of possibly correlated inputs can be resolved more effectively by our model, whereas the definition of inputs in a standard FRBS is essential. Also, the system's output, being a fuzzy rule-based system, can easily implemented further in a fuzzy classifier using software or hardware. Further research may include the selection of a more proper fitness measure or the incorporation of hierarchical rule-based systems. Also, the expansion of the grammar to include the determination of membership functions' parameters would make the algorithm independent from the initial determination of fuzzy sets. Further experimentation in other domains (e.g. financial, manufacturing systems, etc.) would offer an integrated view on the efficiency and the operability of the model.

References

[Alba et al. 1996a] E.Alba, C.Cotta, J.M.Troya, "Type-Constrained Genetic Programming for Rule-Base Definition in Fuzzy Logic Controllers", in John R. Koza, David E. Goldberg, David B. Fogel, Rick L. Riolo (eds.), Proc. of the 1st Ann. Conf. on Genetic Programming, Stanford Univ., Cambridge, MA. The MIT Press, pp 255-260, 1996.
[Alba et al. 1996b] E.Alba, C.Cotta, J.M.Troya, "Evolutionary Design of Fuzzy Logic Controllers Using Strongly-Typed GP", In *Proc. 1996 IEEE* Int'l Symposium on Intelligent Control., pp. 127-132. New York, NY., 1996

[Axer et.al 2000a] Hubertus Axer, Jan Jantzen, Georg Berks, Diedrich Graf v.Keyserlingk, "Aphasia Classification Using Neural Networks", in Proc. ESIT 2000, pp 111-115, Aachen, 2000

[Axer et.al 2000b] Hubertus Axer, Jan Jantzen, Georg Berks, Dagmar Südfeld, Diedrich Graf v.Keyserlingk, "The Aphasia Database on the Web: Description of a Model for Problems of Classification in Medicine", in Proc. ESIT 2000, pp 104-110,Aachen, 2000

[Banzhaf et al. 1998a] Wolfgang Banzhaf, Peter Nordin, Robert E.Keller, Frank D.Francone, "Genetic Programming - An Introduction", Morgan Kaufmann Publishers, Inc., 1998

[Dounias et al. 2002] G.Dounias, A.Tsakonas, J.Jantzen, H.Axer, B. Bjerregaard, D.G.v.Keyserlingk, " Genetic Programming for the Generation of Crisp and Fuzzy Rule Bases in Classification and Diagnosis of Medical Data" submitted to NF-2002, Habana, Cuba, 2002

[Goldberg 1989] D.E.Goldberg, "Genetic Algorithms in Search, Optimisation. and Machine Learning", Addison-Wesley, 1989

[Herrera and Verdegay 1996] F.Herrera, K.L.Verdegay , (Eds.),"Genetic Algorithms and Soft Computing", Physica-Verlag Heidelberg, 1996

[Huber et al. 1984] W.Huber, K.Poeck, D.Weniger, "The Aachen Aphasia Test", in Rose F.C.,"Advances in Neurology. Vol. 42:Progress in Aphasiology.", Raven, New York,1984

[Jang et al. 1997] Jang J.-S.R., Sun C.-T., Mizutani E., “Neuro-Fuzzy and Soft Computing”, Matlab Curicculum Series, 1997

[Jantzen et al. 2000] J.Jantzen, H.Axer,D.G.v.Keyserlingk, "Diagnosis of Aphasia Using Neural and Fuzzy Techniques", Lecture Notes, Aachen Summer School: Soft Computing in Medicine, 2000

[Jang 1998] J-S.R. Jang, “Neuro-fuzzy modeling for nonlinear dynamic system identification”, in E.H. Ruspini, P.P. Bonissone, W. Pedrycz (eds.), “Handbook of Fuzzy Computation”, Institute of Physics Publishing, Dirak House, Temple Back, Bristol BS1 6BE UK, 1998

[Koza 1992] John R.Koza, "Genetic Programming - On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992

[Koza et al 1999] John R.Koza, Forrest H.Bennett III, David Andre, Martin A. Keane, "Genetic Programming III", Morgan Kaufmann Publishers, Inc., 1999

[Li and Ng 1996] Y.Li,K.C.Ng,"Uniform Approach to Model-Based Fuzzy Control System Design and Structural Oprimisation", in F.Herrera, K.L.Verdegay , (Eds.),"Genetic Algorithms and Soft Computing", Physica-Verlag Heidelberg, 1996

[Mamdani and Assilian 1975] E.H.Mamdani and S.Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller", in International Journal of Man-Machine Studies, Vol. 7(1), pp 1-13, 1975

[Montana 1993] D.J.Montana, "Strongly Typed Genetic Programming", Bolt Beranek & Newman, Inc., Technical Report #7866, 1993

[Nauck and Kruse 1996] Nauck Detlef and Kruse Rudolf, “Designing Neuro-Fuzzy Systems Through Backpropagations”, in Witold Pedrydz (Ed.), “Fuzzy Modeling – Paradigms and Practice”, pp 203-231, Kluwer Academic Publishers, 1996

[Singleton 1994] A.Singleton, "Genetic Programming with C++", BYTE Magazine, February 1994

[Tsakonas et al. 2001] A.Tsakonas, G.Dounias, C.Papadopoulos, "The Throughput Rate of Short Exponential Production Lines with Finite Intermediate Buffers Using Genetic Programming Approximation Techniques", accepted for publication in Proc. of MCCS-2001,Vinnitsa, Ukraine, 2001

[Zadeh 1965], L.A.Zadeh, "Fuzzy Sets", in Information and Control, Vol. 8, pp 338-353, 1965