Towards Neural-Symbolic Integration: The Evolutionary Neural Logic Networks Paradigm

Athanasios Tsakonas

***Abstract*—This work presents the application of a new methodology for the production of neural logic networks into two real-world problems from the medical domain. Namely, we apply grammar guided genetic programming using cellular encoding for the representation of neural logic networks into population individuals. The application area is consisted of the diagnosis of Diabetes and the diagnosis of the course of Hepatitis patients. The system is proved able to generate arbitrarily connected and interpretable evolved solutions leading to potential knowledge extraction.**

***Index Terms*—Neural logic networks, Genetic programming, Diabetes, Hepatitis**

I. Introduction

Computational intelligence (CI), as a rapid growing technology has nowadays substituted traditional artificial intelligence (AI) and expert systems in a wide range of applications. The main advantage of these new techniques over AI's symbolic processing, lies in the ability of the CI systems to adapt to the problem environment, hence bypassing the stage of human knowledge acquiring - a mandatory step using expert systems. However, in a number of high-level decision tasks, common expert systems remain still applicable. The reason can be noticed into the need for symbolic representation of the knowledge into these systems, which is a feature that many CI systems have unremarkable success. In other words, it is considered that symbolic representation can be of significant value in these systems for humans, by making clear the inference process to users. Among CI methodologies, neural networks are powerful connectionist systems that still lack the element of complete and accurate interpretation into human-understandable form of knowledge and remain a black box for experts. To heal this situation, a number of alternative approaches have been proposed. Neural logic networks [1] belong to this category, and by their definition can be interpreted into a number of Prolog rules that consist an expert system. Virtually every logic rule can be represented into these networks and then transformed into Prolog commands. Although this model offers excellent results when used within the AI framework (i.e. building a system in a *top-down* process), the application of neural logic networks in CI's data mining tasks – considered a *bottom-up* procedure- has undergone limited success. The reason lies in that proposed systems suffered at least one of the following limitations:

- The extracted neural logic network cannot be interpreted into expert rules [1]-[2].
- The proposed methodology cannot express neural logic networks in their generic graph form [3].
- The user has to select topology and network connection model [1]-[2].

The application of neural logic networks into adaptive tasks seems promising: the extracted model will preserve its interpretability into a number of expert rules and there is not needed any knowledge-acquiring step. Moreover, a solution obtained this way, leads to potential knowledge extraction. Very recently, a new system, namely the evolutionary neural logic networks (ENLN), has been proposed [4] that fulfills those requirements. The new approach uses grammar-guided genetic programming to produce neural logic networks. The evolved solutions can be arbitrarily large and connected networks, since an indirect encoding is adopted. Also, neural logic networks produced by this methodology can always be interpreted into human-understandable expert rules, thus leading to potential knowledge extraction. In this work, we present in brief the methodology and then we apply and test its effectiveness into two real-world problems from the medical domain. In the first task, we examine the system for the diagnosis of Diabetes, and in the second application, we use the system for the diagnosis of the course of Hepatitis patients. The paper is organized as follows. Next section describes the theoretical background, presenting the neural logic networks concept and the grammar guided genetic programming. *Section III* deals with the design and the implementation of the ENLN system. The results and a following discussion are presented in *Section IV*. The paper ends with our conclusion and a description of future work in *Section V*.

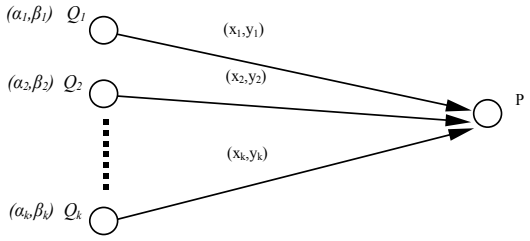
II. Background

A. Neural Logic Networks

The neural logic network is a finite directed graph. It is usually consisted by a set of input nodes and an output node. In its 3-valued form, the possible value for a node can be one of three ordered pair activation values (1,0) for "*true*", (0,1) for "*false*" and (0,0) for "*don't know*". Every synapse (edge) is assigned also an ordered pair weight *(x,y)* where *x* and *y* are real numbers. An example neural logic and its output value *(a,b)* of node `P` is shown in *Fig. 1*.

Athanasios Tsakonas is with the Aristotle University of Thessaloniki, Greece (telephone: +306-937-891-399, e-mail: tsakonas@stt.aegean.gr).

Different sets of weights enable the representation of different logical operations. It is actually possible to map any rule of conventional knowledge into a neural logic network. Neural logic networks can be expanded into fuzzy neural logic networks, enabling this way the handling of real valued attributes [1].



$$(a,b)=\begin{cases}(1,0) & if \sum_{j=1}^{k} a_j x_j - \sum_{j=1}^{k} b_j y_j \geq 1 \\ (0,1) & if \sum_{j=1}^{k} a_j x_j - \sum_{j=1}^{k} b_j y_j \leq -1 \\ (0,0) & otherwise \end{cases}$$

Fig. 1. Example neural logic network and its output value.

In *Fig. 2*, an example of a logical operator and its implementation in neural logic networks is shown.

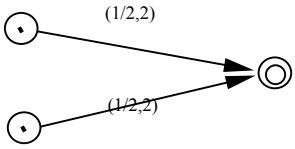


Fig. 2. An example logical operation for operating conjunction in neural logic networks.

In order to illustrate the interpretability of neural logic networks into expert rules, let us consider the following simple neural logic network, consisting of the priority rule:

Richer (X,Y) $\Leftarrow$ **priority** (House_Owner, Car_Owner, M/C_Owner)

Fig.3 depicts the neural logic network corresponding to the above rule. The interpretation of the network into *Prolog* rules is straightforward.

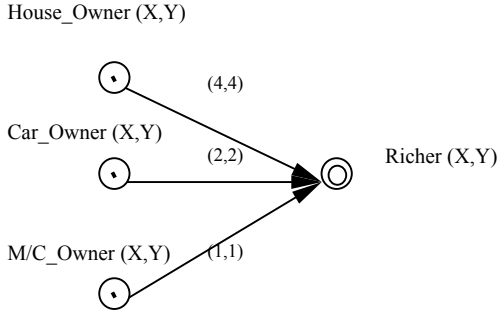


Fig. 3. Example neural logic network, corresponding to a priority rule.

The resulted Prolog commands are shown in *Fig.4*. Even though powerful in their definition, neural logic networks are not widely applied. The main reason can be located in the fact that for the known training methodologies [1]-[2], the refinement of the edge weights reduces significantly the interpretability of these networks to expert rules, thus depriving these networks from their valuable feature.

Some steps for the preservation of the interpretability have been performed by [3], without however the ability to express arbitrarily large and connected neural logic networks. For instance, a neural logic network, which performs the important logical operation of XOR, cannot be represented using the direct encoding of [3].

If *House_Owner*(X,Y)=(1,0) then *Richer*(X,Y)=(1,0)
If *House_Owner* (X,Y)=(0,1) then *Richer* (X,Y)=(0,1)
Supposing *House_Owner* (X,Y)=(0,0)
    If *Car_Owner*(X,Y)=(1,0) then *Richer* (X,Y)=(1,0)
    If *Car_Owner* (X,Y)=(0,1) then *Richer* (X,Y)=(0,1)
    Supposing *House_Owner* (X,Y)=(0,0) and *Car_Owner* (X,Y)=(0,0)
        If *M/C_Owner*(X,Y)=(1,0) then *Richer* (X,Y)=(1,0)
        If *M/C_Owner* (X,Y)=(0,1) then *Richer* (X,Y)=(0,1)
Supposing *House_Owner* (X,Y)=(0,0) and *Car_Owner* (X,Y)=(0,0)
and *M/C_Owner* (X,Y)=(0,0) then *Richer* (X,Y)=(0,0)

Fig. 4. Prolog rules equivalent to the neural logic network of *Fig.3*.

B. Grammar guided Genetic Programming

The prime advantage of genetic programming over genetic algorithms, is the ability to construct functional trees of variable length. This property enables the search for very complex solutions that are usually in the form of a mathematical formula - an approach that is commonly known as symbolic regression. Later paradigms extended this concept to calculate any *boolean* or programming expression. Thus, complex intelligent structures, such as fuzzy rule-based systems or decision trees have already been used as the desirable target solution in genetic programming approaches [5]-[8]. The main qualification of this solving procedure is that the feature selection, and the system configuration, derive in the searching process and do not require any human involvement. The potential gain of an automated feature selection and system configuration is obvious; no prior knowledge is required and, furthermore, not any human expertise is needed to construct an intelligent system. Nevertheless, the task of implementing complex intelligent structures into genetic programming functional sets in not rather straightforward. The function set that composes an intelligent system retains a specific hierarchy that must be traced in the GP tree permissible structures. This writing offers two advantages. First, the search process avoids candidate solutions that are meaningless or, at least, obscure. Second, the search space is reduced significantly among only valid solutions. Thus, a genotype - a point in the search space- corresponds always to a phenotype - a point in the solution space. This approach -known as legal search space handling method [9]- is applied in this work using context-free grammars.

C. Context-Free Grammars

Although powerful in its definition, the genetic programming procedure may be proved greedy in computational and time resources. Therefore, when the syntax form of the desired solution is already known, it is

useful to restrain the genetic programming from searching solutions with different syntax forms [10]-[11]. The most advantageous method to implement such restrictions among other approaches [12], is to apply syntax constraints to genetic programming trees, usually with the help of a context-free grammar declared in the Backus-Naur-Form (BNF) [13]. The BNF-grammar consists of terminal nodes and non-terminal nodes and is represented by the set {N,T,P,S} where N is the set of non-terminals, T is the set of terminals, P is the set of production rules and S is a member of N corresponding to the starting symbol. The construction of the production rules can be the most critical point in the creation of a BNF grammar, since these production rules express the permissible structures of an individual.

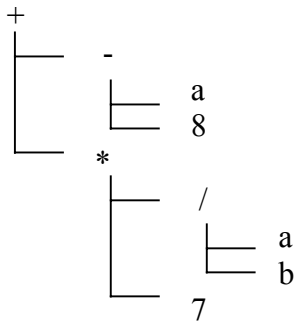


Fig. 5. Example genetic programming tree for the expression (a-8)+7*(a/b).

An example grammar expressing a class of individuals, which (among other grammars) may produce the program shown in *Fig. 5*, is composed by the following sets:

$$N = \{EXPR, OP\}$$
$$T = \{-,*,/,a,b,7,8\}$$
$$S = \langle EXPR \rangle$$

Then, P is expressed as shown in *Table I*.

TABLE I
GRAMMAR USED FOR A SIMPLE EXAMPLE TREE

Symbol	Rule
<EXPR>	::= <EXPR> <OP> <EXPR> | <VAR> | <NUMBER>
<OP>	::=- | * | /
<VAR>	::= a | b
<NUMBER>	::= 7 | 8

D. Cellular Encoding

Although mapping decision trees or fuzzy rule-based systems to specific grammars can be relatively easy to implement, the execution of massively parallel processing intelligent systems -such as the neural logic networks- is not forthright. In order to explore variable sized solutions, we applied *indirect encoding*. The most common one is the *cellular encoding* [15], in which a genotype can be realized as a descriptive phenotype for the desired solution. More specifically, within such a function set, there are elementary functions that modify the system architecture together with functions that calculate tuning variables. Current implementations include encoding for feed forward and Kohonen neural networks [16], [17] and fuzzy Petri-nets [18], [17]. In his original work, Gruau also used a context-free grammar - a BNF grammar- to encode indirectly the neural networks. On the other hand, in [18] a logic grammar - a context-sensitive one- is adapted to encode fuzzy Petri-nets. In our work, we show that as long as the depth-first execution of the program nodes of a GP tree is ensured -which is the default-, a context-free grammar such as a BNF grammar is adequate for expressing neural networks. Gruau's original work has been facing some skepticism [19] on the ability to express arbitrarily connected networks. Later developments [10] seem to offer less restrictive grammar, though the *cut* function in those implementations still maintained bounded effect. A similar technology, called *edge encoding*, developed by [20] is also today used with human competitive results in a wide area of applications.

III. Design and Implementation

A. Data Preprocessing and Genetic Programming Setup

We split each data set into a training set, a validation set and a test set. The training set consists of 50% of the data and the rest 50% is divided equally between the validation set and the test set. The separation of the examples into training, validation and test sets is performed in a loop manner. Specifically, for every four samples, the first two samples are assigned to the training set, the third sample is assigned to the validation set, and the fourth one is assigned to the test set. This process is repeated until all the examples are assigned a set. During the training phase, the validation set is typically used to avoid over-fitting. In all experiments, we used the same GP parameters.

Without claiming optimality, the GP parameters are presented in *Table II*. This optimization was decided after experimentation, since it is not possible to obtain a general principle regarding the most proper probability values for every case.

TABLE II
GENETIC PROGRAMMING PARAMETERS

Parameter	Value
Population:	2,000 individuals
GP implementation:	Steady-state G^3P
Selection:	Tournament with elitist strategy
Tournament size:	6
Crossover Rate:	0.35
Overall Mutation Rate:	0.65
Node Mutation Rate:	0.4
Shrink Mutation Rate:	0.6
Killing Anti-Tournament size:	2
Maximum allowed individualsize:	650 nodes
Maximum number of generations:	100

As it can be observed in *Table II*, the setup denotes our preference for significantly high mutation rates, especially shrink mutation [22] that slows down the code bloat caused by crossover operations.

B. System Grammar and Operating Functions

The system grammar is presented in *Fig.6*. Initial symbol (root) of a genetic programming tree can be a node of type <PROG>. The function set is as follows:

Function PROG: The function PROG creates the embryonic network that is used later by the functions S1, S2, P1 and P2 to be expanded. An alternative name for this function, which is used throughout this paper, is the term "ENLN".

Function S1 and S2: These functions enter a node in serial to the node that is applied.

Function P1 and P2: These functions enter a node in parallel to the node that is applied.

Function IN: The operation of function IN is to assign a variable to the input node that it is applied.

```
<PROG>    : =  PROG <PLACE1><SYNAPSE>
<PLACE1>: =  S1 <PLACE1><SYNAPSE><PLACE2>
             | P1 <PLACE1><PLACE1>
             | IN
IN        : =  Data attribute (system input)
<PLACE2>: =  S2 <PLACE2><SYNAPSE><PLACE2>
             | P2 <PLACE2><SYNAPSE><PLACE2>
             | E
E         : =  ∅
<SYNAPSE>    : =  LNK <NUM><CUT><SYNAPSE>
             | CNR <CNRSEL><K>
<NUM>     : =  NUM
<CUT>     : =  CUT
<CNRSEL>: =  CNRSEL
<K>       : =  K
NUM       : =  Integer in [1,256]
CUT       : =  Integer in [0,1]
CNRSEL    : =  Integer in [0,10]
K         : =  Integer in [0,9]
```

Fig. 6. Context free grammar for the production of neural logic networks within genetic programming framework.

Function E: The operation of function E is to mark the end of the expansion of the network.

Function LNK: This function provides the framework for the application of *cut* function. It actually enables the non-full connectivity of the network, a feature that offers larger solution search space.

TABLE III
OPERATIONS FOR FUNCTION CNR

Parameter	Calculation
0	Conjunction
1	Disjunction
2	Priority
3	At least k-true
4	At least k-false
5	Majority influence
6	Majority influence of k
7	2/3 Majority
8	Unanimity
9	If-Then operation, Kleene's model
10	Difference

Function CNR: This function performs the node inference. Based on the first parameter, the corresponding calculation is performed. The second parameter assists the calculation for the *at-least-k* and *majority-of-k* operators. Possible computations are shown in *Table III*. An alternative name for this function, which is used throughout this paper, is the term "Rule".

Function NUM, CUT, K, CNRSEL: These functions return an integer to be used by the corresponding calling functions.

Having discussed the system design, in the following session we shall apply the methodology in two medical domains.

IV. RESULTS AND DISCUSSION

A. Diagnosis of Diabetes

The desirable diagnosis in this problem is whether the patient has the diabetes symptoms, according to the criteria of World Health Organization. The population that was used for the collection of data lives near Phoenix in Arizona, US. A number of constraints have been applied for the selection of medical cases. Specifically, all the patients are female, at least 21 years old and originated from the race of Pima Indians [23]. The input features correspond either to physiological human features (f.ex. number or times being pregnant, body mass index etc.) or laboratory tests (f.ex. plasma glucose concentration a 2 hours in an oral glucose tolerance test). *Table IV* presents the domain characteristics, and *Table V* shows the input features in detail. This problem is interesting since there is unknown number of missing values, which is unknown data noise.

TABLE IV
DOMAIN DESCRIPTION FOR THE DIABETES PROBLEM

Parameter	Value
Domain	Diagnosis of Diabetes
Data Base	Diabetes
Input Features	8
Continuous Input Features	8
Discrete Input Features	0
Binary Input Features	0
Total Records	764
Training Set Records	382
Validation Set Records	191
Test Set Records	191
Missing data	Yes, unknown number
Normalized data	Yes

In literature [23], there is an application of the ADAP algorithm which works like a perceptron and is referred a 76% accuracy in unknown data. The solution that was extracted using our system, although it achieves a 72.3 % (138/191) accuracy in unknown data, it additionally enables the interpretation of the solution. The accuracy in the training data and in the validation set was 69.4% (265/382) and 69.7% (133/191) correspondingly.

We observe that from the 8 available features, the system selected to use only 2 of them. From the solution that is shown in *Fig. 7*, we may extract the following simple expert system:

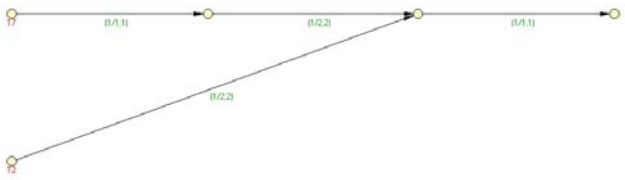
Q1 ⇐ **conjunction** (Body Mass Index)

Q ⇐ **conjunction** (Plasma Glucose Concentration a 2 Hours in an Oral Glucose Tolerance Test, **Q1**)

TABLE V
PARAMETER DESCRIPTION FOR THE DIABETES PROBLEM

Variable	Feature	Values/ Value range
T1	Number of times being pregnant	0-17
T2	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	0-199
T3	Diastolic blood pressure (mm Hg)	0-122
T4	Triceps skin fold thickness (mm)	0-99
T5	2-Hour serum insulin (mu U/ml)	0-846
T6	Body mass index (weight in kg/(height in m) ^{2})	0-67.1
T7	Diabetes pedigree function	0.078-2.42
T8	Age (years)	21-81

The interpretation of the above expert rules based on the fuzzy system inference, proposes the positive diagnosis of diabetes if two features are "high": the body mass index and the plasma glucose concentration a 2 hours in an oral glucose tolerance test. This expert system may be simplified, roughly speaking, into a proposition similar to "*the higher is the plasma glucose concentration a 2 hours in an oral glucose tolerance test and the larger is the body mass index, the more likely is the positive diabetes indication*". Moreover, is notable for the non-expert, the absence in this neural logic network of other distinguishing features -like the patient's age or the diabetes pedigree function - which might be expected to be included in the solution.



```
(ENLN (P1 (S1 (In T7) (Rule 0 0) E) (In T2)) (Rule
0 0))
```

Fig. 7. Solution description and the corresponding neural logic network for the Diabetes problem.

B. Diagnosis of the course of Hepatitis patients

This problem is concerned with the diagnosis of the course of Hepatitis to patients, based on the physiological and laboratory tests [24]. More specifically, with the aid of the given data, we need to predict whether the patient will survive or not. This domain is interesting since there is unequal distribution between the two classes, that is only 20,64 % (32/155) of the data correspond to patients that will live while the rest data is referred to patients that are expected to die. Moreover, there is about 5% of missing values, and a combination of binary and continuous input features. Since the data is composed by features having different value ranges, we preprocessed the data by normalizing them into the [1,2] interval in order to be able to apply the fuzzy neural logic networks model. Specifically, using this scheme, a value of 1 denotes the '*false'*, a value of 2 denotes the '*true'*, while a value of 0 signifies the '*unknown'*. In the non-binary features, values between 1 and 2 (which are the normalization borders) are also allowed. This coding is only a convention scheme since input values are actually decoded during the evaluation into the system to the usual `(x,y)` neural logic pairs. *Table VI* summarizes the characteristics of the Hepatitis domain, and *Table VII* shows in detail the input features of this database. In the literature, solutions that are proposed achieve an accuracy ranging from 80% [24] to 83% [25].

TABLE VI
DOMAIN DESCRIPTION FOR THE HEPATITIS PROBLEM

Parameter	Value
Domain	Diagnosis of the course for Hepatitis patients
Data Base	Hepatitis
Input Features	19
Continuous Input Features	6
Discrete Input Features	0
Binary Input Features	13
Total Records	152
Training Set Records	76
Validation Set Records	38
Test Set Records	38
Missing data	Yes, 146 values
Normalized data	Yes

The solution that was extracted using our system achieves accuracy 81.6 % (31/38) to unknown data (test set). The accuracy to the training data and validation set was 94.8% (72/76) and 68.4 % (26/38) correspondingly. The produced neural logic network is shown in *Fig. 8*.

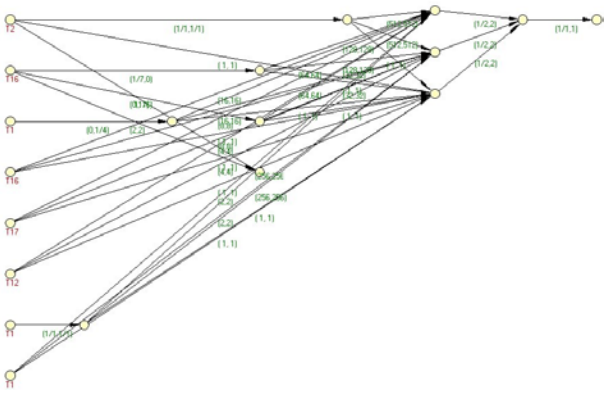
TABLE VII
PARAMETER DESCRIPTION FOR THE HEPATITIS PROBLEM

Variable	Feature	Values/Value range
T1	Age	10-80
T2	Sex (male/female)	0-1
T3	STEROID (true/false)	0-1
T4	ANTIVIRALS (true/false)	0-1
T5	FATIGUE (true/false)	0-1
T6	MALAISE (true/false)	0-1
T7	ANOREXIA (true/false)	0-1
T8	LIVER BIG (true/false)	0-1
T9	LIVER FIRM (true/false)	0-1
T10	SPLEEN PALPABLE (true/false)	0-1
T11	SPIDERS (true/false)	0-1
T12	ASCITES (true/false)	0-1
T13	VARICES (true/false)	0-1
T14	BILIRUBIN	0.39-4.00
T15	ALK PHOSPHATE	33-250
T16	SGOT	13-500
T17	ALBUMIN	2.1-6.0
T18	PROTIME	10-90
T19	HISTOLOGY (true/false)	0-1

Although the extracted solution is large and prevents obvious interpretation, the neural logic network maintains still its ability to be transformed into a (large, in this case) set of expert rules. Additionally, it is worth to note that, in order to make a decision, this proposed solution makes use of only 5 features from the available 19 ones.

The selected features are *AGE*, *SEX*, *ASCITES* (abnormal accumulation of clear yellow fluid in the peritoneal cavity), *SGOT* (enzyme tests of liver function / serum glutamic oxaloacetic transaminase), and *ALBUMIN* (protein found in blood and maintains the proper amount of water in it / serum albumin)

From the above decisive features, it is notable for the non-expert, the existence of both the age and the sex for the diagnosis of the course of Hepatitis.



```
(ENLN (S1 (P1 (P1 (S1 (In T2) (Rule 8 7) E) (P1
(S1 (In T16) (Rule 3 7) (P2 E (Link 3 4 (Link 15 3
(Rule 4 6))) (P2 E (Rule 2 5) E))) (P1 (S1 (In T1)
(Link 236 4 (Rule 4 4)) E) (P1 (In T16) (P1 (In
T17) (In T12)))))) (P1 (S1 (In T1) (Rule 8 7) E)
(In T1))) (Link 6 2 (Link 3 3 (Link 10 2 (Rule 2
8)))) (P2 E (Rule 2 9) (P2 E (Rule 5 11) E)))
(Rule 5 3))
```

Fig. 8. Solution description and the corresponding neural logic network for the Hepatitis problem.

V. Conclusion and Further Research

Neural networks are powerful connectionist systems that have been introduced in areas where symbolic processing systems of traditional artificial intelligence used to be applied. As a tool of computational intelligence, the adaptation of the neural network to the problem domain using an inductive method, offers advantage over expert systems where the knowledge must be acquired first, before the system development. Ever since their first application, interpretation of the obtained knowledge was a research target for neural networks. In the scope of this area, the neural logic networks have been proposed as a class of networks that by their definition preserve their interpretability into symbolic knowledge.

Until recently however, the application of an effective training / production method within the CI framework has not been successful. A novel system that uses genetic programming with indirect encoding that has been proposed recently [4], overcomes these problems, producing automatically designed and tuned neural logic networks, which always preserve their interpretability. In this work we applied the system into two real-world medical problems, the Diabetes diagnosis and the diagnosis of the course of Hepatitis patients. In both problems, the system has been proved capable of producing competitive to the literature results, which maintain their interpretability and lead to potential knowledge extraction.

Future work involves the application of the system in other areas, as well as the incorporation of recursive structures into the neural logic network architecture. Moreover, the *minimum description length principle* will be developed to be included as an anti-overfitting measure. Finally, parameter-tuning optimization of the underlying genetic programming algorithm is expected to offer better efficiency, hence it will be of primary importance among our future work.

References

[1] Teh H.H., *Neural Logic Networks: A New Class of Neural Networks*, World Scientific Pub Co, 1995.

[2] Tan A-H. and Teow L-N., "Inductive neural logic network and the SCM algorithm", *Neurocomputing*, vol. 14, 2 : 5, pp.157-176, 1997

[3] Chia H.W-K. and Tan C-L.,"Neural logic network learning using genetic programming", *Intl. Journal of Comp. Intelligence and Applications*, 1:4, 2001, pp 357-368.

[4] Tsakonas A., Aggelis V., Karkazis I. and Dounias G., "An Evolutionary System for Neural Logic Networks using Genetic Programming and Indirect Encoding", *Journal of Applied Logic*, accepted for publication, Elsevier, Spring 2004.

[5] Alba E., Cotta C. and Troya J.M., "Evolutionary Design of Fuzzy Logic Controllers Using Strongly-Typed GP", *Proc. 1996 IEEE Int'l Symposium on Intelligent Control*, pp. 127-132. New York, NY., 1996.

[6] Tsakonas A., Dounias G., "Hierarchical Classification Trees Using Type-Constrained Genetic Programming", *Proc. of 1st Intl. IEEE Symposium in Intelligent Systems*, Varna, Bulgaria, 2002.

[7] Tsakonas A., Dounias G., Axer H., and von Keyserlingk D.G., "Data Classification using Fuzzy Rule-Based Systems represented as Genetic Programming Type-Constrained Trees", *Proc. of the UKCI-01*, Edinbourgh, UK, pp 162-168, 2001.

[8] Tsakonas A. and Dounias G., "A Scheme for the Evolution of Feedforward Neural Networks using BNF-Grammar Driven Genetic Programming", *Proc. of Eunite-02*, Algarve, Portugal, 2002.

[9] Yu T. and Bentley P., "Methods to Evolve Legal Phenotypes", *Lecture Notes in Comp. Science 1498, Proc. of. Parallel Problem Solving from Nature V*, pp 280-291,1998.

[10] Gruau F., Whitley D. and Pyeatt L., "A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks", in Koza J.R., Goldberg D.E., Fogel D.B., Riolo R.L., Eds.*, Genetic Programming 1996: Proceedings of the First Annual Conf.*, pp 81-89, Cambridge, MA, MIT Press, 1996.

[11] Montana D.J., "Strongly Typed Genetic Programming", *Evolutionary Computation*, vol. .3, no. 2, 1995.

[12] N.Paterson and M.Livesey,"Evolving Caching Algorithms in C by GP", *Genetic Programming 1997*, pp 262-267, MIT Press, 1997.

[13] Naur P., "Revised report on the algorithmic language ALGOL 60", *Commun. ACM*, vol 6, No 1, pp 1-17, Jan 1963.

[14] Whigham P., "Search Bias, Language Bias and Genetic Programming", *Genetic Programming 1996*, pp 230-237, MIT Press, 1996

[15] Gruau F., "Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm", *Ph.D. Thesis*, Ecole Normale Superieure de Lyon, anonymous ftp:lip.ens-lyon.fr (140.77.1.11) pub/Rapports/ PhD PhD94-01-E.ps.Z.

[16] Gruau F., "On Using Syntactic Constraints with Genetic Programming", in P.J.Angeline, K.E.Jinnear,Jr., Eds., *Advances in Genetic Programming,* MIT,1996.

[17] Tsakonas A. and Dounias G., Decision Making in the Medical Domain: Comparing the Effectiveness of GP-Generated Fuzzy Intelligent Structures, *Proc. of Eunite-03*, Oulou, Finland, 2003.

[18] Wong M.L., "A flexible knowledge discovery system using genetic programming and logic grammars", *Decision Support Systems*, 31, 2001, pp 405-428.

[19] Hussain T., "Cellular Encoding: Review and Critique", *Technical Report, Queen's University*, 1997, http://www.qucis.queensu ca/home/hussain/web/1997_cellular_encoding_review.ps.gz

[20] J.Koza, F. Bennett, D. Andre and M. Keane, *Genetic Programming III: Automatic Programming and Automatic Circuit Synthesis*, Morgan Kaufmann, 2003.

[21] Ratle A. and Sebag M., "Genetic Programming and Domain Knowledge: Beyond the Limitations of Grammar-Guided Machine Discovery", Schienauer et al., Eds.*, Proc. of the 6th Conf. on Par. Probl. Solv. from Nature, LNCS*, Springer, Berlin, 2000, pp 211-220

[22] Singleton A., "Genetic Programming with C++", *BYTE Magazine*, Feb 1994.

[23] Smith,J.W., Everhart,J.E., Dickson,W.C., Knowler,W.C., and Johannes,R.S., "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus", *Proc. of the Symposium on Computer Applications and Medical Care*, pp. 261--265. IEEE Computer Society Press, 1988.

[24] Diaconis,P. and Efron,B. "Computer-Intensive Methods in Statistics", *Scientific American,* vol. 248, 1983.

[25] Cestnik,G., Konenenko,I and Bratko,I, "Assistant-86: A Knowledge-Elicitation Tool for Sophisticated Users". I.Bratko, N.Lavrac, Eds., *Progress in Machine Learning*, pp. 31-45, Sigma Press, 1987.