

# First-order Cascade ARTMAP

Rodrigo Basilio, Gerson Zaverucha, and Valmir C. Barbosa

Programa de Engenharia de Sistemas e Computação – COPPE  
Universidade Federal do Rio de Janeiro  
Caixa Postal 68511, 21945-970 Rio de Janeiro, RJ, Brasil.  
{rbasilio, gerson, valmir}@cos.ufrj.br

**Abstract.** First-order theory refinement using neural networks is still an open problem. Towards a solution to this problem, we define a First-Ord extension of the Cascade ARTMAP (FOCA) system, using Inductive Logic Programming techniques. To present such a first-order extension of Cascade ARTMAP, we: a) modify the network structure to handle first-order objects; b) define first-order versions of the main functions that guide all Cascade ARTMAP dynamics, the choice and match functions; c) define a first-order version of the propositional learning algorithm, that approximates Plotkin’s least general generalization (lgg). Results show that our initial goal, learning logic programs using neural networks, has been achieved.

## 1 Introduction

The Cascade ARTMAP [21] system is a knowledge based neural network (KBNN) [20], like KBANN [22], RAPTURE [11], and C-IL<sup>2</sup>P [6], that has been shown to outperform other purely analytical or inductive systems in the task of propositional theory refinement: a prior incomplete and/or partially correct propositional symbolic knowledge about a problem domain is given to a theory refinement system and it is revised by training the system with examples.

Three main advantages of Cascade ARTMAP over other KBNN systems are: a) the initial rule structure of the network created by the insertion algorithm is preserved during training (this a major problem when using backpropagation training for the task of theory refinement) - this facilitates rule extraction allowing a direct comparison of the extracted rules to the originally inserted rules; b) it is an incremental learning system; c) it combines instance-based learning with rule induction<sup>1</sup>.

All these characteristics are inherited from the fuzzy ARTMAP system [4], of which Cascade ARTMAP is an extension that allows the representation of intermediate attributes of rule-based knowledge and multi-step inference (rule chaining).

Inductive logic programming (ILP)[10] augments the expressive power of inductive learning and theory refinement tasks to (first-order) logic programming [23]. This has allowed ILP systems to handle problems like mutagenicity, carcinogenicity, drug design, language learning [15], and benchmark problems like East-West train problem [12].

Most ILP systems, such as FOIL [17], GOLEM [13], TIM [9], and PROGOL [14], use a covering algorithm to generate hypothesis. However, as pointed out by Bratko (whose HYPER is a recent exception [3]), systems that use a covering algorithm have difficulties in learning multiple predicates and recursive definitions, and will have unnecessarily long clauses in the hypothesis. Cascade ARTMAP, by contrast, uses a non-covering algorithm to generate each hypothesis a whole, like other KBNN systems.

Our main goal is to combine ILP techniques with Cascade ARTMAP, defining a unified hybrid first-order KBNN system, named FOCA, which takes advantage of its parent methodologies while attempting to overcome their limitations. To present such a first-order extension of Cascade ARTMAP, we: a) modify the network structure to handle first-order objects; b) define first-order versions of the choice (similarity) function and of the vigilance criterion (functions that guide all Cascade ARTMAP dynamics); c) define a first-order version of the learning algorithm (the weight update), which is related to [8] and approximates Plotkin’s least general generalization (lgg) [16]. The FOCA system can be viewed like an ILP system: for each example, the system generates a bottom-clause (or most specific clause, similarly to PROGOL) with respect to the initial domain theory (background knowledge) and search for the most similar rule already encoded in the network that satisfies the vigilance criterion, which verifies if this rule is close enough to the example. If this rule meets the criterion, the bottom clause is generalized with it, otherwise the next most similar rule is considered. If there is no rule satisfying the criterion, a new one is created from the bottom-clause.

---

<sup>1</sup> This characteristic is shared by the Rule Induction from a Set of Exemplars (RISE) system [5].

Related work includes the following. The ILP systems GOLEM and TIM are also bottom-up systems that use and approximate lgg, but not combined with incremental clustering. [1] uses the ILP system LINUS [10] that transforms a first-order learning task to attribute-value form, and applies a neural network as its attribute-value learning algorithm. This can only be done to a restricted class of problems. [2] presents a first-order KBNN based on radial basis networks. The learning algorithm only deals with the numeric part of the theory, which has no rule chaining and is recursion-free. [7] shares the same objectives with this work, but the learning part is still being developed and uses a backpropagation-based KBNN.

The remaining parts of this paper are organized as follows. Section 2 presents an introduction to Cascade ARTMAP, Section 3 describes the FOCA system, Section 4 presents some experimental results, and Section 5 contains conclusions and directions for future work.

## 2 Cascade ARTMAP

We only review Fuzzy ARTMAP [4], since Cascade ARTMAP is an extension of it that allows rule insertion and rule cascading (chaining), and these are done similarly in FOCA.

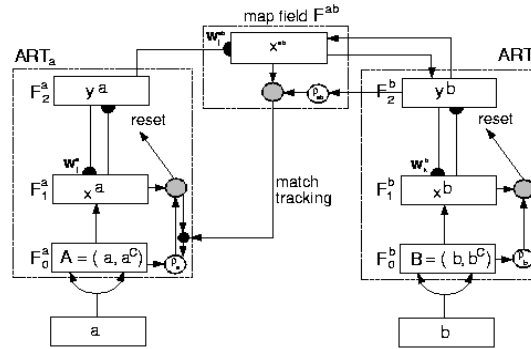


Fig. 1. Fuzzy ARTMAP system [4]

Fuzzy ARTMAP (see Fig. 1) incorporates two Fuzzy ART modules,  $ART_a$  and  $ART_b$ , which are linked together via an inter-ART map field  $F^{ab}$ . Each ART module is a clustering (unsupervised) system. We can view each example presented to Fuzzy ARTMAP as a rule, the input part being the body (antecedent) and the output part (class) being the head (consequent) of the rule.  $ART_a$  receives the examples' bodies and constructs a clustering scheme, aggregating similar bodies in the same cluster (a node in the  $F_2^a$  layer). Each cluster has a prototype (a weight vector  $w_j$  connecting the cluster  $j$  to all nodes in  $F_1^a$ ) that is the generalization of the bodies that belong to that cluster.  $ART_b$  receives examples' heads and similarly constructs a clustering scheme in  $F_2^b$ . The map field links each category formed in each ART module consistently.

Each Fuzzy ART module has three layers:

- 1)  $F_0$ , with the same number of nodes as the module's input vector (unless we use a normalization scheme like complement coding as in Fig. 1). Vector  $\mathbf{I}$  denotes activation of  $F_0$  ( $\mathbf{A}$  in  $ART_a$  and  $\mathbf{B}$  in  $ART_b$ ).
- 2)  $F_1$ , with the same number of nodes as  $F_0$ . The nodes in these two layers are linked together by one-to-one weights. Vector  $\mathbf{x}$  denotes the activation of  $F_1$ .
- 3)  $F_2$ , which is fully connected with  $F_1$ . The most important weights in the ART network are the weights between one node  $j$  in  $F_2$  and all nodes in  $F_1$ , denoted by the vector of adaptive weights  $w_j$ , because they keep the information about the prototypes of the network. Vector  $\mathbf{y}$  denotes the activation of  $F_2$ .

The dynamics of Fuzzy ART depends on a choice parameter  $\alpha > 0$ , a learning rate parameter  $\beta \in [0,1]$ , and a vigilance parameter  $\rho \in [0,1]$ . For Fuzzy ARTMAP, there are two additional parameters: the minimum value of the vigilance parameter of  $ART_a$ , called the baseline parameter  $\rho_a$ , and the vigilance parameter of Map Field  $\rho_{ab}$ .

We can view the Fuzzy ARTMAP dynamics as a propositional inductive learning system: for each example presentation (see step 1 of the algorithm below), the most similar rule already encoded in the Fuzzy ARTMAP (steps 2 and 3) and satisfying the vigilance criterion is chosen to generalize with this example (step 4a); if there is no rule satisfying the criterion, a new one is created from the example.

The following is the Fuzzy ARTMAP learning algorithm.

### Algorithm

For each example  $e=(a,b)$  in the set of examples  $E$ , do:

**step 1** (Input presentation):  $\rho_a = \overline{\rho}_a$ ,  $\mathbf{x}^a = \mathbf{A}$ ,  $\mathbf{x}^b = \mathbf{B}$ .

**step 2** (Category selection): **2.1**) For each ART module, calculate the choice function  $\mathbf{T}_j$  (the degree to which  $\mathbf{w}_j$  is a subset of the input  $\mathbf{I}$ ) for each node in  $F_2$  and select the node  $J$  that has the greatest value. The *choice function*  $\mathbf{T}_j$  is defined by:

$$\mathbf{T}_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \quad (1)$$

where input  $\mathbf{I}$  is  $\mathbf{A}$  in  $\text{ART}_a$  and  $\mathbf{B}$  in  $\text{ART}_b$ ,  $\wedge$  is the fuzzy intersection defined by  $(p \wedge q)_i = \min(p_i, q_i)$ , and the norm  $|\cdot|$  is defined by

$$|\mathbf{p}| = \sum_i |p_i| \quad (2)$$

**2.2**) For each node pre-selected in 2.1 ( $J$  in  $F_2^a$  and  $K$  in  $F_2^b$ ), we calculate the *match function*  $\mathbf{m}_j$  (the degree in which the input is a subset of the prototype  $\mathbf{w}_j$ ), defined as

$$\mathbf{m}_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{I}|} \quad (3)$$

If the match function for node  $J$  and/or  $K$  is greater than or equal to  $\rho$  (vigilance criterion) then we say that *resonance* happens and the respective  $F_2$  layer is activated; so for  $\text{ART}_a$ :  $\mathbf{y}_j=1$  and  $\mathbf{y}_j = 0$  for  $j \neq J$  (winner-take-all activation, similarly for  $\text{ART}_b$ ). Otherwise, go to 2.1 to select a new node in the respective module. If no category can be chosen, let  $J$  and/or  $K$  be the new nodes created dynamically, where  $\mathbf{w}_J^a=1$ ,  $\mathbf{w}_J^{ab}=1$ ,  $\mathbf{y}_J^a=1$  and  $\mathbf{y}_J^b=0$  for  $j \neq J$  and/or  $\mathbf{w}_K^b=1$ ,  $\mathbf{y}_K^b=1$ ,  $\mathbf{y}_K^a=0$ , for  $k \neq K$ .

**step 3** (Verification in Map Field): In the Map Field,  $\mathbf{x}^{ab} = \mathbf{w}_J^{ab} \wedge \mathbf{y}^b$ . If  $|\mathbf{x}^{ab}| / |\mathbf{y}^b| \geq \rho_{ab}$  then go to 4a, otherwise go to 4b.

**step 4a** (Learning):  $\mathbf{w}_J^a, \mathbf{w}_K^b$  will be updated:

$$\mathbf{w}_j^{(\text{new})} = \beta(\mathbf{I} \wedge \mathbf{w}_j^{(\text{old})}) + (1-\beta)\mathbf{w}_j^{(\text{old})} \quad (4)$$

and  $\mathbf{w}_J^{ab}$  will be updated :

$$\mathbf{w}_j^{ab} = \beta_{ab}(\mathbf{y}^b \wedge \mathbf{w}_j^{ab}) + (1-\beta_{ab})\mathbf{w}_j^{ab} \quad (5)$$

(fast learning corresponds to setting  $\beta=1$ ).

**step 4b** (Match Tracking): match tracking is activated,  $\rho_a = \mathbf{m}_J^a(A)+\varepsilon$ ,  $\mathbf{T}_J = 0$  and we go back to step 2.

## 3 The FOCA System

A logic program is a set of clauses:  $h \leftarrow b_1, \dots, b_n$  where  $h$  is an atom and  $b_1, \dots, b_n$  are literals (positive or negative atoms). The general learning framework [Muggleton, 1999] from ILP is: given an initial domain theory (background knowledge)  $B$ , a set of positive examples  $E^+$  and a set of negative examples  $E^-$ , the goal is to build a hypothesis  $H$ , where  $B, E^+, E^-$  and  $H$  are logic programs, satisfying the following conditions:

**Necessity:**  $B \not\models E^+$  (B does not cover  $E^+$ )

**Sufficiency:**  $B \wedge H \models E^+$  (B and H covers  $E^+$ )

**Weak Consistency:**  $B \wedge H \not\models \text{false}$

**Strong Consistency:**  $B \wedge H \wedge E^- \not\models \text{false}$

In systems that handle noise, the sufficiency and strong consistency are not required.

Generally in ILP, an example is a ground instance  $p_i(c_1, \dots, c_n)$  of a target concept  $p_i(V_1, V_2, \dots, V_n)$  and its description is logically encoded in  $B$ . The hypothesis space is generally ordered by  $\theta$ -subsumption. Some ILP bottom-up (specific to general) systems search the hypothesis space using the *relative least general generalization operator* for two clauses ( $\text{rlgg}(e_1, e_2)$ ) [16], which can be calculated in two steps [9]: we calculate the saturation (bottom-clause) of  $e_1$  and  $e_2$ ,  $\perp_1$  and  $\perp_2$ , respectively, and then we calculate the  $\text{lgg}(\perp_1, \perp_2)$  [16].

Each neural network of the ART family can be viewed as a propositional bottom-up system, where the intersection (fuzzy-and) is the generalization operator (only fast learning mode). The First-Order Cascade ARTMAP (FOCA) is also a bottom-up system: an example is saturated with respect to background knowledge  $B$  generating a bottom-clause  $\perp_i$  and the most similar rule already encoded in the network satisfying the vigilance criterion is chosen to generalize with  $\perp_i$  using a  $\text{lgg}$  based operator (so it is in fact a  $\text{rlgg}$ -based operator); if there is no rule satisfying the criterion a new one is created from  $\perp_i$ . Similarly to

Progol, FOCA also use language bias like mode declarations and the depth mode language  $L_i(M)$  to restrict the size of a bottom-clause [14].

Before presenting the algorithm in Section 3.4 and the performance evaluation in 3.5, we describe the architecture in 3.1, the new choice and match functions, in 3.2 and the lgg-based operator that we use in 3.3.

### 3.1 Architecture

The architecture of the FOCA system is shown in Fig. 2.

In Cascade ARTMAP, the information that passes through links is a number. Now in FOCA, the information is a set of tuples of terms, where each tuple has an associated number. The bottom clause's body is the input of  $ART_a$  and its head is the input of  $ART_b$ .

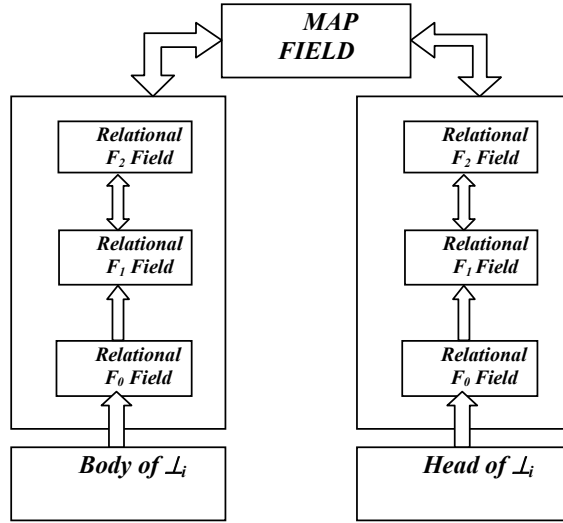


Fig. 2. Overview of the FOCA system

Each neuron in  $F_0$  and  $F_1$  represents a predicate symbol of the problem. We define  $\text{predicate}(k)$  as the predicate symbol that neuron  $k$  represents. The activation of each neuron  $k$  in  $F_0$  is the set  $A_k$ , consisting of tuples of terms of  $\text{predicate}(k)$  that appears in the input of an ART module. For instance, for the input  $I = \{\text{father}(Z,S), \text{father}(Z,D), \text{mother}(Q,S), \text{mother}(Q,D)\}$  we have  $A_k = \{(Z,S),(Z,D)\}$  and  $A_j = \{(Q,S), (Q,D)\}$ , where  $\text{predicate}(k) = \text{father}$  and  $\text{predicate}(j) = \text{mother}$ .

Each neuron in  $F_1$  has a link with the neuron in  $F_0$  that represents the same predicate. Initially, let the union of all  $A_k$  with the respective predicate symbol applied to each tuple be  $X = \{p(V_1, \dots, V_n) \mid (V_1, \dots, V_n) \in A_k, \text{ where } \text{predicate}(k) = p, \text{ for each } k \in F_0\}$  ( $X$  is  $X^a$  in  $ART_a$  and  $X^b$  in  $ART_b$ ). In Cascade ARTMAP, the vector  $x^a$  represents a working memory: the input and all attributes activated by the rule chaining. Here, similarly, the set  $X^a$  will also hold all literals inferred during the rule chaining. The steps 4b and 5a of the algorithm show how  $X^a$  is updated with the literals activated in rule chaining.

Each cluster  $j$  in the  $F_2$  layer has a set  $W_{jk}$ , associated to each link with a neuron  $k$  in  $F_1$ .  $W_{jk}$  holds all tuples of terms from  $\text{predicate}(k)$  that belongs to the relational prototype, defined as  $W_j = \{p(t_1, \dots, t_n) \mid (t_1, \dots, t_n) \in W_{jk}, \text{ where } \text{predicate}(k) = p, \text{ for each } k \in F_1\}$ . If a cluster  $j$  codes more than one example,  $W_j$  can be viewed as an approximate rlgg of all saturations of examples coded by  $j$ . Associated with each literal  $p$  there is a number, named  $\text{Fuzzy}(p) \in [0,1]$ , that represents its fuzzy information. Initially, if a literal does not have such number the default value is one.

### 3.2 The First-Order Choice and Match Functions

The choice and match functions were developed based on [19] but having the goal that in the propositional case, they would be reduced to their propositional versions.

Given a category  $J$  in  $F_2^a$ , and the set  $X^a$ , the first-order equations of choice and match functions are

$$T_j(X) = \frac{\sum_{(w,x) \in INT_{WX}} Fuzzy(w) \wedge Fuzzy(x)}{\sum_{(w,x) \in INT_{WX}} Fuzzy(w) + \alpha} \quad (6)$$

$$m_j(X) = \frac{\sum_{(w,x) \in INT_{WX}} Fuzzy(w) \wedge Fuzzy(x)}{\sum_{(w,x) \in INT_{WX}} Fuzzy(x)} \quad (7)$$

where  $INT_{WX} \subseteq W_j \times X^a$  is the relational intersection defined as

$$INT_{WX} = \{(w,x) | m \in m_{OP}, w = m\theta_w, x = m\theta_x\} \quad (8)$$

where  $m_{OP}$  is defined as

$$m_{OP} = \max_{\theta_w, \theta_x} |O \cap P|$$

where  $O\theta_w = W_j$ ,  $P\theta_x = X^a$ ,  $\theta_w$  and  $\theta_x$  are renaming substitutions. If we have more than one candidate for  $INT_{WX}$ , we choose arbitrarily. The choice and match functions in  $F_2^b$  are calculated similarly.

### 3.3 Learning

For a cluster  $J$  in  $ART_a$  and the working memory  $X^a$ , the learning operator is defined as

$$W_j = algg(W_j, X^a) \quad (9)$$

where the operator  $algg$  is defined as

$$algg(W_j, X^a) = \{lgg(w,x) | (w,x) \in INT_{WX}\} \quad (10)$$

The  $algg$  operator, unlike  $lgg$ , is dependent on the order: two different sequences of calculations can produce two different clauses. This learning definition only supports the fast learning setting, because we eliminate all literals in  $W_j$  that is not in  $INT_{WX}$ .

The fuzzy information is updated by

$$Fuzzy(w) = \beta * \min(Fuzzy(w), Fuzzy(x)) + (1 - \beta) * Fuzzy(w) \quad (11)$$

where  $(w, x) \in INT_{WX}$ . Similarly, in  $F_2^b$ , the prototype set and fuzzy information is also updated accordingly.

### 3.4 Algorithm

The algorithm of FOCA, shown below, is similar to the Cascade ARTMAP algorithm. The notion of  $\psi(J)$  set is defined similarly to [21].

For each example  $e$  in  $E$ , do

**Step 1 (Bottom-Clause generation):** the example  $e$  is pre-processed and the bottom clause  $\perp_i$  is calculated. The body of  $\perp_i$  is presented to  $ART_a$  and the head of  $\perp_i$  to  $ART_b$ .

**Step 2 (Show  $\perp_i$ ):**  $\rho_a = \overline{\rho_a}$ . In  $F_1^a$ ,  $X^a = \text{body of } \perp_i$ .

**Step 3 (Cluster selection): 3.1)** the choice function  $T_j^a(X^a)$  (eq.(6)) is calculated for each node  $j$  in  $F_2^a$ .

**3.2)** Let  $J$  be the chosen node in 3.1. If  $J$  pass in the vigilance criterion  $m_j^a(X^a) \geq \rho_a$ , go to step 4<sub>b</sub> else go to 3.1. If no category can be chosen in  $F_2^a$ , go to step 4<sub>a</sub>.

**Step 4a (New node):** let  $J$  be a new node created dynamically. In  $ART_b$ ,  $X^b = \text{head of } \perp_i$ . **4.1)** we calculate the choice function for each node  $k$  in  $F_2^b$ . **4.2)** Let  $K$  be the chosen node in 4.1. If  $K$  pass in the vigilance criterion  $m_k^b(X^b) \geq \rho_b$  then go to 4.3 else go to 4.1. If no category can be chosen in  $F_2^b$ , let  $K$  be a new node in  $F_2^b$ . **4.3)** each node in  $\psi(J)$  and node  $K$  in  $F_2^b$  will be updated using eq. (9) and (11). The weight  $w_j^{ab}$  is updated using eq (5).

**Step 4b (Inference):**  $x^{ab} = w_j^{ab}$  and the Map Field is activated. The weights one-to-one between  $F^{ab}$  and  $F_2^b$  activate  $F_2^b$ . For each node  $k$  in  $F_2^b$ , the choice function is defined as  $T_k^b = x_k^{ab}$ . The winning node  $K$  is:

$$T_K^b = \max \{T_k^b : \text{for each node } k \text{ in } F_2^b\}$$

When  $K$  is chosen  $y_K^b = 1$  e  $y_k^b = 0$  for each  $k \neq K$ .

$X^b = W_k^b \theta$ , where  $\theta = \theta_w^{-1} \theta_x$ , and  $\theta_w, \theta_x$  are the renaming substitutions used to calculate  $INT_{wX}$  for the cluster J in  $F_2^a$  in step 3. If  $X^b$  holds a target atom then go to step 5b, else go to step 5a.

**Step 5a (Update):**  $X^a$  is updated as follows:

$$X^a = X^a \cup X^b$$

and then we go to step 3.

**Step 5b (Matching):**  $X^b = \text{head of } \perp_i$  and we calculate  $m_k^b(X^b)$ , imposing for the variables in the body that appear in the head the same substitutions in  $\theta_w$  and  $\theta_x$  used to calculate  $INT_{wX}$  for the cluster J in  $F_2^a$  in step 3. If  $m_k^b(X^b) \geq \rho_b$  then go to step 6a, else go to step 6b.

**Step 6a (Resonance):** Here we have resonance and each activated node in  $\psi(J)$  and K in  $F_2^b$  will be updated using eq. (9) and (11).

**Step 6b (Mini-Match Tracking):** A prediction error occurs, and so we have mini-match tracking:  $\rho_a = m_z^a(X^a) + \epsilon$ , where  $m_z^a(X^a) = \{\min(m_k^a(X^a) \mid k \in \psi(J)\}$ ,  $T_z^a = 0$ , and we go back to step 3.

**Example.** We present a simple trace of the algorithm above for the East-West train problem [12]. Consider that we have already presented two examples. Therefore, there are already in each module two clusters: clusters 1 and 2 in  $ART_a$  associated with clusters 1 and 2 in  $ART_b$ , respectively. The  $F_2$  layer of each ART module is shown below.

$F_2^a$ Layer ( $ART_a$ )	
Cluster 1	$W_1^a = \{\text{has\_car}(X,V), \text{open}(V), \text{long}(V), \text{shape}(V, \text{rectangle}), \text{has\_car}(X,W), \text{not long}(W), \text{not open}(W), \text{shape}(W, \text{rectangle}), \text{has\_car}(X,Z), \text{not long}(Z), \text{open}(Z), \text{shape}(Z, \text{rectangle})\}$
Cluster 2	$W_2^a = \{\text{has\_car}(A,B), \text{not open}(B), \text{long}(B), \text{shape}(B, \text{rectangle}), \text{has\_car}(A,C), \text{not long}(C), \text{open}(C), \text{shape}(C, \text{rectangle})\}$

$F_2^b$ Layer ( $ART_b$ )	
Cluster 1	$W_1^b = \{\text{eastbound}(X)\}$
Cluster 2	$W_2^b = \{\text{not eastbound}(A)\}$

Let  $\bar{\rho}_a = 0$ ,  $\alpha_a = \alpha_b = 0.001$ ,  $\rho_{ab} = \rho_b = 1$ . Now, we present the third example: (step 1)  $e = \text{eastbound}(\text{east2})$ . The bottom-clause is generated:

$$\perp_3 = \{\text{eastbound}(E) \leftarrow \text{has\_car}(E,F), \text{not long}(F), \text{open}(F), \text{shape}(F, \text{u\_shaped}), \text{has\_car}(E,G), \text{open}(G), \text{not long}(G), \text{shape}(G, \text{bucket}), \text{has\_car}(E,H), \text{not open}(H), \text{not long}(H), \text{shape}(H, \text{rectangle})\}$$

In step 2  $X^a = \text{body of } \perp_3$ . In step 3, we calculate the choice and match functions for each node:

For cluster 1, the relational intersection is

$$INT_{wX} = \{(\text{has\_car}(X,V), \text{has\_car}(E,F)), (\text{open}(V), \text{open}(F)), (\text{has\_car}(X,Z), \text{has\_car}(E,G)), (\text{open}(Z), \text{open}(G)), (\text{not long}(Z), \text{not long}(G)), (\text{has\_car}(X,W), \text{has\_car}(E,H)), (\text{not open}(W), \text{not open}(H)), (\text{not long}(W), \text{not long}(H)), (\text{shape}(W, \text{rectangle}), \text{shape}(H, \text{rectangle})\}$$

using substitutions  $\theta_w = \{E/X, H/W, G/Z, F/V\}$  and  $\theta_x = \emptyset$ .

Since  $\text{Fuzzy}(w) = \text{Fuzzy}(x) = 1$ , for all  $(w,x) \in INT_{wX}$ , then  $T_1^a(X^a) = 9/(12+\alpha) \cong 0.75$  and  $m_1(X^a) = 9/12 = 0.75$ .

For cluster 2, the relational intersection is

$$INT_{wX} = \{(\text{has\_car}(A,C), \text{has\_car}(E,G)), (\text{open}(C), \text{open}(G)), (\text{not long}(C), \text{not long}(G)), (\text{has\_car}(A,B), \text{has\_car}(E,H)), (\text{not open}(B), \text{not open}(H)), (\text{shape}(B, \text{rectangle}), \text{shape}(H, \text{rectangle})\}$$

using substitutions  $\theta_w = \{E/A, H/B, G/C\}$  and  $\theta_x = \emptyset$ .

Then,  $T_2^a(X^a) = 6/(8+\alpha) \cong 0.75$  and  $m_2(X^a) = 6/12 = 0.5$ .

Therefore, node 1 is chosen and it passes in the vigilance criterion (step 3.2). In step 4<sub>b</sub>, we calculate the choice function in  $F_2^b$ :  $x^{ab} = w^{ab}_1 = (10)$ ,  $T_1^b = 1$  and  $T_2^b = 0$ . Then, node 1 in  $F_2^b$  is chosen and  $X^b = W^b_1 \theta_w^{-1} \theta_x = \{\text{eastbound}(E)\}$ .  $X^b$  holds a target atom then we go to step 5b, and we calculate

$\text{INT}_{WX} = \{\{\text{eastbound}(X), \text{eastbound}(E)\}\}$

using substitutions  $\theta_w = \{E/X, H/W, G/Z, F/V\}$  and  $\theta_x = \emptyset$ .

Then,  $m_1^b(X^b) = 1/1 = 1$ .

In step 6a, the nodes 1 in each module are updated using (9) and (11). The final state of the  $F_2^a$  layer is shown below.

F <sub>2</sub> <sup>a</sup> Layer (ART <sub>a</sub> )	
Cluster 1	$W^a_1 = \{\text{has\_car}(M,N), \text{open}(N), \text{has\_car}(M,O), \text{not long}(O), \text{not open}(O), \text{shape}(O, \text{rectangle}), \text{has\_car}(M,P), \text{not long}(P), \text{open}(P)\}$
Cluster 2	$W^a_2 = \{\text{has\_car}(A,B), \text{not open}(B), \text{long}(B), \text{shape}(B, \text{rectangle}), \text{has\_car}(A,C), \text{not long}(C), \text{open}(C), \text{shape}(C, \text{rectangle})\}$

### 3.5 Performance Evaluation

To classify a test example, we only execute steps 1, 2, 3 (in case it needs to go to 4a, we stop and return no), 4b, 5a, 5b (here, if  $m_k^b(X^b) \geq \rho_b$  then we return yes else no) of the algorithm 3.4.

## 4 Experiments

The system was tested in some Machine Learning domain problems. The first three problems are defined in [10]: learning the concept of an arch, family relationships, and Eleusis problem (layouts 1,2,3). The fourth problem is the east-west train problem with 10 examples [12]. They do not have a test set and each theory was correctly learned.

The Illegal KRK endgames problem [10] has a test set. We use the same experimental methodology for comparison purpose. We obtained from the rules extracted from FOCA 97.92% accuracy, with 0.005% of standard deviation. Other ILP systems' performances are presented for comparison in table 1 below.

**Table 1.** Results in the KRK problem

SYSTEMS	ACC. (5 SETS OF 100 EX.)
FOIL	90.8% sd 1.7%
LINUS-ASSINTANT	98.1% sd 1.1%
LINUS-NEWGEM	88.4% sd 4.0%
FOCA	97.92% sd 0.005%

## 5 Conclusions

This work contributes to bridge the gap between symbolic and connectionist learning systems. A first-order extension of the neural network Cascade ARTMAP, the FOCA system, was presented, by extending to first-order the structure of the network, the choice and match functions, and the learning algorithm. Since (Fuzzy) ARTMAP is composed of two (Fuzzy) ART modules, this work could be also be seen as presenting a first-order (Fuzzy) ART, a first-order clustering system.

The experimental results show that FOCA can learn first-order theories as an ILP system. Nevertheless, they have not explored FOCA's main advantage: a first-order theory refinement system.

Therefore, we would like to apply FOCA on such problems. We will also apply FOCA in other ILP applications: mutagenicity, drug design, language learning [15] and character recognition [24].

Like the ART family, FOCA is also sensitive to the order of the examples' presentation. We can overcome this problem by using an ensemble, as in [4]. We also plan to investigate the effect of the algg operator, since it approximates the lgg and some relevant literals may be lost.

## Acknowledgments

The first author is financially supported by CAPES and the other two are partially financially supported by the Brazilian Research Council CNPq. We would like to thank Jude Shavlik, David Page and Ah-Hwee Tan for useful discussions.

## References

1. R. Basilio, G. Zaverucha and A. Garcez. Inducing Relational Concepts with Neural Networks Via the LINUS System, In Proc. Fifth International Conference on Neural Information Processing (ICONIP'98), Vol. 3 pp. 1507-1510, Japan, 1998.
2. M. Botta, A. Giordana and R. Piola. FONN: Combining First Order Logic with Connectionist Learning. Proc. of the 14th International Conference on Machine Learning ICML-97, pp. 46-56 Morgan Kaufmann, 1997.
3. I. Bratko. Refining Complete Hypotheses in ILP. ILP-99, LNAI 1634, pp. 44-55, Springer-Verlag, 1999.
4. G. A. Carpenter, S. Grossberg, J.H. Reynolds, N. Markuzon and D.B. Rosen Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Trans. Neural Networks 3, pp.698-713, 1992.
5. P. Domingos. Rule Induction and Instance-Based Learning: a Unified Approach, IJCAI, vol. 2, pp.1226-1232, 1995.
6. A. S. Garcez, G. Zaverucha. The Connectionist Inductive Learning and Logic Programming System. Applied Intelligence Journal, F. Kurfess (editor), Vol. 11, Number 1, pp. 59-77. 1999.
7. N. A. Hallack, G. Zaverucha, and V. C. Barbosa. Towards a hybrid model of first-order theory refinement. In Hybrid Neural Systems. LNAI 1778, Springer-Verlag, 2000.
8. D. Haussler. Learning Conjunctive concepts in structural domains. Machine Learning 4, pp. 7-40, 1989.
9. P. Idestam-Almqvist. Efficient Induction of Recursive Definitions by Structural Analysis of Saturations. Advances in Inductive Logic Programming. Ed. Luc De Raedt. pp 192-205, IOS Press, 1996.
10. N. Lavrac and S. Dzeroski. Inductive Logic Programming: Techniques and Applications. Ellis Horwood Series in Artificial Intelligence, 1994.
11. J.J. Mahoney and R. J. Mooney. Combining Connectionist and Symbolic Learning Methods to Refine Certainty-factor Rule-bases. Connection Science 5 pp. 339-364, 1993.
12. D. Michie, S. Muggleton, D. Page, A. Srinivasan. To the international computing community: a new East-West Challenge. Technical Report, Oxford University Computing Laboratory, Oxford, UK, 1994.
13. S. Muggleton and C. Feng. Efficient Induction of Logic Programs. In Muggleton, S. (Eds) Inductive Logic Programming, pp. 453-472. Academic Press, London, 1992.
14. S. Muggleton. Inverse Entailment and Progol. In New Generation Computing 13, pp 245-286, 1995.
15. S. Muggleton. Inductive logic programming: issues, results and the LLL challenge. Artificial Intelligence, 114 pp. 283-296, December 1999.
16. G. Plotkin. A further note on inductive generalization. Machine Intelligence, v.6. University Press, 1971.
17. J. R. Quinlan. Learning logical definitions from relations. Machine Learning 5, pp.239-266, 1990.
18. J. R. Quinlan. Learning first-order definitions of functions. Journal of Artificial Intelligence Research 5, 139-161, 1996.
19. J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. ILP-98, LNAI 1446. Springer-Verlag, pp. 271-280, 1998.
20. J. Shavlik. Framework for Combining Symbolic and Neural Learning. Machine Learning, 14, pp.321-331, 1994.
21. Ah-Hwee Tan. Cascade ARTMAP: Integrating Neural Computation and Symbolic Knowledge Processing. IEEE Trans.on Neural Networks. pp.237-250, vol. 8, n.2, 1997.
22. G. Towell and J. Shavlik. Knowledge-Based Artificial Neural Networks. Artificial Intelligence, pp.119-165, vol. 70, 1994.
23. S. Wrobel. First Order Theory Refinement. Advances in Inductive Logic Programming. Ed. Luc De Raedt. pp 14-33, IOS Press, 1996.
24. B. Kijirikul and S. Sinthupinyo. Approximate ILP Rules by Backpropagation Neural Network: A Result on Thai Character Recognition. 9th International Workshop on Inductive Logic Programming, LNAI 1634, pp. 162-173, Springer-Verlag, 1999.