Shared Multi-Space Representation for Neural-Symbolic Reasoning

Edjard de S. Mota and Yan B. Diniz

Federal University of Amazonas
Institute of Computing
Av. Rodrigo Octávio, 6200 CEP 69077-000 Manaus, Brasil
{edjard,ybd}@icomp.ufam.edu.br

Abstract. This paper presents a new neural-symbolic reasoning approach based on a sharing of neural multi-space representation for coded fractions of first-order logic. A multi-space is the union of spaces with different dimensions, each one for a different set of distinct features. In our case, we model the distinct aspects of logical formulae as separated spaces attached with vectors of importance weights of distinct sizes. This representation is our approach to tackle the neural network's propositional fixation that has defied the community to obtain robust and sound neural-symbolic learning and reasoning, but presenting practical useful performance. Expecting to achieve better results, we innovated the neuron structure by allowing one neuron to have more than one output, making it possible to share influences while propagating them across many neural spaces. Similarity measure between symbol code indexes defines the neighborhood of a neuron, and learning happens through unification which propagates the weights. Such propagation represents the substitution of variables across the clauses involved, reflecting the resolution principle. In this way, the network will learn about patterns of refutation, reducing the search space by identifying a region containing ground clauses with the same logical importance.

1 Introduction

This paper presents a new neural-symbolic reasoning approach based on neural sharing of multi-space representation for a coded portion of first-order formulae suitable for machine learning and neural network methods. The Smarandache multi-space [9] is a union of spaces with different dimensions, each one representing a different set of distinct features. We distribute across such a structure the different aspects of logical expressions along with vectors of weights of distinct sizes. With such a representation one can compute the *degree of importance*, that is *induced* by the resolution principle and unification across distinct dimensions of the logical structure during a deduction [12], taking such spaces into account.

There have been some efforts to deal with the neural network's propositional fixation [10], since it was argued in [4] that for some fragments of first-order logics such a limitation can be overcome, for instance [1, 7]. However, their attempt to provide robust and sound neural-symbolic learning and reasoning were unsuccessful, as they all lack practical useful performance [3], defying us to tackle this issue from a different perspective. Looking at Amao[1] structure sharing-based

[1] A cognitive agent we are developing at the Intelligent and Autonomous Computing group at IComp in UFAM

implementation [13, 2], as in most Prolog engines, we felt like transforming them into a structure sharing of code indexes and use it for neural learning computation.

Automated deduction based on Resolution Principle [12], reduces the search space by transforming the task of proving the validity of a formula to prove that its negation is inconsistent. The main struggle with doing first-order logic reasoning in connectionist approaches is that the variable binding of terms may lead to a huge, if not infinite, number of neurons for all elements of the Herbrand Base. We realized that, instead of doing this, neural reasoning could actually *points* to "neural regions" where the negation of a given formula were most likely to be inconsistent. The difference would be the use of a structured neural network trained to learn about regions of potential refutations before one is even requested. This is only possible if the network learns from the initial set of formulae and self-organize in regions of refutation.

In this paper, we introduce the *Shared Neural Multi-Space* (Shared NeMuS) of coded first-order expressions (CFOE), a weighted multi-space of CFOEs. The idea is to give a relative *degree of importance* for each element within it according to the element attributes and similarity with others structurally equivalent. Similarity defines the neighborhood of an element and neural learning is performed by the propagation of weights through unification. Such propagation represents the substitution of variables across the clauses involved, reflecting resolution principle for first-order logic[12]. In this way the network will learn about patterns of refutation to reduce the search space when queries are proposed.

Before describing the formalities of our approach, section 2 shows the fundamental aspects of the neural shared multi-spaces of CFOEs. In a Shared NeMuS one neuron represents logical expression and it may have many inputs of importance as well as outputs that influence others. We formally present the shared NeMuS for CFOEs in section 3 to capture the fundamentals described. In section 4 we detail the mechanisms to train such a structured neural net based on an adapted best-match similarity measure for learning patterns of resolution-based deduction. This innovative way of creating a structured neural network, shared NeMuS, may not fit in the standards of the machine learning field as discussed in section 5. Nonetheless, such a perspective can bring new light to the way neural-symbolic learning and reasoning is performed for first-order logic as we discuss in section 6.

2 Fundamentals of Neural Sharing of Multi-Space

We use Smarandache multi-space [9], which is a union of n spaces $A_1, \ldots, A_n$ in which each A_i is the space of a distinct observed characteristic of the overall space. For each A_i there is a different metric to describe a different side (or objetive), of the "major" side (or objective). In this perspective, first-order language has *atomic constants* (of the Herbrand universe), *function*, *predicate* with its literal instances, and *clause* spaces. Variables are used to refer sets of atomic terms via quantification, and they belong to the same space of atoms. Figure 1.(a) depicts a multi-space representation of first-order expressions with n clauses, at space 3, each one defined by a (possibly different) number of literals at space 2. Each literal is composed of terms either from function space 1 or constant space 0, or both. Lines from one element covers its compound terms at the space below.

The neural network embedded within such a multi-space is based on a chain of importance weights, having constant space as the basic level of importance. In their turn, weights of the constant space induce the importance weights of functions space, and both (constant and function) spaces induces weights of the predicate space according to literal instances within it. Finally, weights

of predicate space induces clauses importance weights. Figure 1.(b) depicts the neural multi-space of FOEs, in which weights are the (blue) arrows representing the influence of attributes from one space on objects at one or two space above them.

Different from traditional Artificial Neural Networks (ANN), one single neuron may have, along with its inputs (weights of influence), more than one output representing its influence upon more than one element at a space level above. From Figure 1.(b) constant a_3 affects literals l_1 and l_2 of clause C_1, and it affects l_1 of clause C_n. Note that there are two l_1 logical objects, but if both are positive/negative instances of the same predicate, then there should be just one neuron representation in this case rather then replicating information.

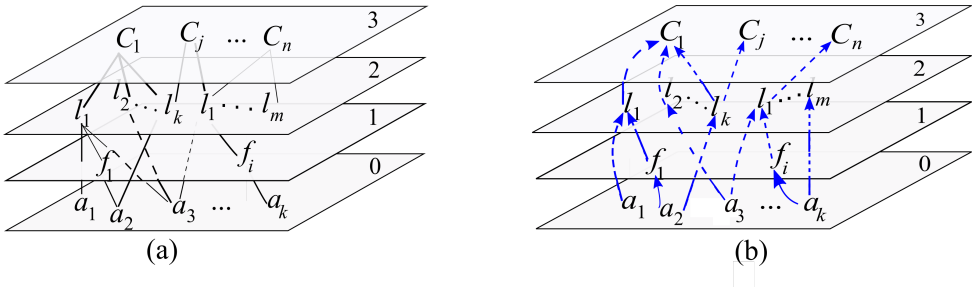

Fig. 1. (a) A general sharing multi-space of FOEs. (b) A neural sharing multi-space of FOEs

To avoid such repetition, we adopted the sharing of structure idea [13]: every logical neural element is a pair. The first component is the neuron symbol–attribute pair, formed by symbol code and a vector of indexes with the space each one belongs to. The second component is a vector of structured weights pointing to the elements the neuron exerts influence. A structured weight neuron is, in the case of constant neural space, a triple: the space index (0 up to 3), the code index of the symbol upon which it influences, and the value of the influence. A triple is used because atoms at level 0 can be attributes of functions (at level 1) or of a literal (at level 2), e.g. constant a_2, and most import is to tell the influence of a term on a function from a literal, like a_2 does on function f_1 as well as on literal l_k. For all other spaces, a structured weight is pair because, from space 1, every neuron will exert influence only on neurons at one level above.

When a shared NeMuS of FOE is generated all weight vectors of its components, at all levels, are set to zero to represent no previous learning. Then training is divided into two phases. In the first, every ground clause (clause with no variables), has its weights updated according to the code of its symbol components. This will create the importance of clauses, expressed in weights. Then, clauses that had their weights updated will propagate them via similarity of the predicate space, yielding regions associated to such similarity measure.

In the second phase, every clause with more than one literal and at least one variable, called *deduction rule*, is divided into two parts: conclusion and assumptions. For each assumption p of a deduction rule, with an index code i_p, a sort of neural unification is applied between p and its complementary literal with same index, if there is any, at the negative region of predicate space. The premisses with successful unification will update their weights from their components, and the weights of the conclusion will be updated by the weights of the shared variables. In the case

of functions, not only variable weights are updated, but the composition of predicate and variable weights will update the weights of functions.

3 NeMuS Framework for Coded First-order Expressions

3.1 Amao Logical Language

Amao[2] symbolic representation and reasoning component is a clausal-based formal system [14], in which clauses are divided into two categories. 1) *Initial Clauses*, say $\mathtt{B}$, are those belonging to the set of axioms plus the negation of the query; 2) *structured Clauses* are the ones derived by a sort of Linear Resolution [12]. Roughly, if $\mathtt{S}$ is a sentence or query, in clausal form, and $\mathtt{B}$ is the set of initial clauses, then a deduction of $\mathtt{S}$ from $\mathtt{B}$ corresponds to derive an empty clause, $\sqcup$, from $\{\sim \mathtt{S}\} \cup \mathtt{B}$, or according to Herbrand theorem, to prove that $\{\sim \mathtt{S}\} \cup \mathtt{B}$ is *unsatisfiable* and it yields the most general unifier for $\mathtt{S}$.

A set of logical formulae is represented by clauses of literals according to the following terminology. Predicates and constant or atomic symbols start with lowercase letters like $p, q, r, \ldots$ and $a, b, c, \ldots$, respectively. Variables start with capital letters, like $X, Y, \ldots$. A term is either a variable, a constant symbol or a function $f(t_1, \ldots, t_k)$ in which f represents a mapping from terms $t_1, \ldots, t_k$ to an "unknown" individual. If p is a symbol representing a predicate relation over the terms $t_1, \ldots t_n$, then $p(t_1, \ldots, t_n)$ is a valid atomic formula. Predicates and functions are compound symbols with similar structure, but with different logical meaning. A *literal* is either an atomic formula, L, or its negation $\sim L$, and both are said to be complementary to each other. A *Deduction Rule* is a disjunction of literals $\mathtt{L}_1$, ..., $\mathtt{L}_n$, written as $\mathtt{L}_1; \ldots; \mathtt{L}_1$. There may exist more than one positive literal, an so any Horn clause is represented by $Head; \sim body$, in which literals of the body are called assumptions.

Example 1. The following is a valid sequence of clauses, each with its unique index code.

1. $p(a)$. 3. $r(a)$. 5. $q(X, f(X))$; $\sim p(X)$
2. $p(b)$. 4. $r(c)$. 6. $s(X, f(Y))$; $\sim r(X); \sim p(Y)$

3.2 First-Order Expressions as Multi-Spaces

Amao symbolic reasoning component parses and translates a sequence of clauses into an internal structure of shared of data connected via memory address pointers. This representation is very efficient for dealing with symbols, and the idea of sharing data could be used to create computational efficient neural representations of clauses. Formal logic languages are structurally well defined, and such a structure can be thought as a structure of indexes. Instead of training a neural network with bare data like other approaches, e.g. [7], we decided to use an efficient encoding of shared structures, and turn them into spaces of index to build up a first-order neuronal multi-space.

For this purpose, Amao makes use of a symbolic hash mapping [8] (SHM), that maps symbolic objects of the language to a hash key within a finite range. Such a key is not the one used for

[2] Amao is the name of a deity that taught people of Camanaos tribe, who lived on the margins of the Negro River in the Brazilian part of the Amazon rainforest, the process of making *mandioca powder* and *beiju* biscuit for their diet.

learning because there may occur collisions. For this reason a separate chaining is used to place keys that collide in a list associated with index, in which every node contains the kind of occurred symbol. Counters were added so that to every new symbol parsed and "hashed", a code hash mapping (CHM) function generates the next natural number, starting from 1. In this way, every single symbol has a unique index, and such an index shall be the one used for neural learning mechanism. All codes compose what we call coded corpus defined as follows.

Definition 1 (First-order Coded Corpus (FOCC)) *Let $\mathcal{C}$, $\mathcal{F}$ and $\mathcal{P}$ be a finite sets of constants, functions and predicates, respectively. The* First-order Coded Corpus *is a triple of associative hash mappings $\langle f_C, f_F, f_P \rangle$, such that $f_C : C \to \mathbb{N}$, $f_F : F \to \mathbb{N}$ and $f_P : P \to \mathbb{N}$. The mappings f_F and f_P take into account the arity of each function and predicate, to generate their indexes $n \in \mathbb{N}$. Each element of a FOCC triple $\mathcal{C}$ shall be identified as $\mathcal{C}_C$, $\mathcal{C}_F$ and $\mathcal{C}_P$.*

Note that the uniqueness of a mapping is only within a corpus space, i.e. the code "1" will be the index of the first predicate found, as well as the first atom four in the case of formula $p(a)$ be the first clause parsed. Figure 2 depicts a possible FOCC generated from clauses of example 1.

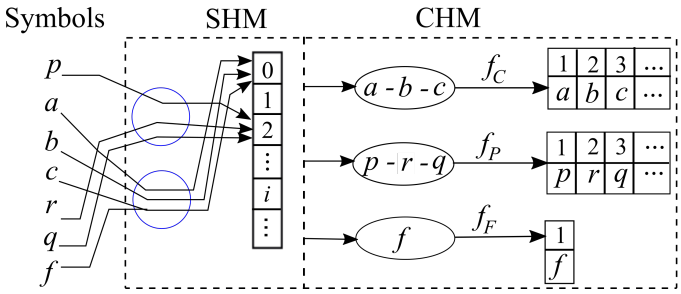

Fig. 2. First-order coded corpus of logical symbols from Example 1.

The result of parsing of any logical formula is passed to the corpus generation, which is also fed with variable indexes according to their clause scope. From a reasoning perspective, variables can be interpreted as an abstract way to talk about sets of atomic constants. For efficiency sake it is assumed that both belong to two different regions of the same space: positive region for constant symbols and negative for variable appearing in all clauses. The scope of each variable is bound via weights. The following two definitions capture these idea, in which $\mathbb{Z}_0$ means $\mathbb{Z} \setminus \{0\}$.

Definition 2 (Subject Binding) *Let $k \in \mathbb{N}$ be an index, $h \in \{1, 2\}$, $i \in \mathbb{Z}_0$ and $w \in \mathbb{R}$. The* Subject Binding *of k is the triple (h, i, w), and it represents that subject with index k* influences *object with index i at space h with measure w.*

The spaces a subject may influence are the function space (1) and the predicate space (2) (see Figure 1). As said above variables can be seen as a way to refer to sets of constants, either atoms or (mostly ground) functions. To be identified outside the subject space a variable shall always be a negative number, but its influence or subject binding will be accessed by its absolute value from the variable region of the subject space.

Definition 3 (Neural Subject Multi-Space (NeSuMS)) *Let* $C_\beta = [x_1, \ldots, x_m]$ *and* $V_\beta = [y_1, \ldots, y_n]$*, where each* $x_i(y_i)$ *is a vector of subject binding, be two subject binding spaces for constants and variables, respectively. A* Neural Subject Multi-Space *is the pair* (C_β, V_β)*.*

Functions and predicates have a different sort of binding, or importance, along with the information about their attributes. As they are both structurally alike, they are treated in the same way regarding their composition. Their binding is simpler than subject binding because they just need the logical element index at the space above and the value of such influence. In both cases, their attributes are uniquely identified by space, either zero (for subject space) or one (for function space), and the attribute index. In the case of space 0, if attribute index is less than zero this means that it refers to the variable region.

Definition 4 (Neural Compound Multi-Space (NeComMS)) *Let* $h_1, \ldots, h_m \in \{0, 1\}$ *be space indexes (for variable and function, i.e. 0 or 1),* $a_1, \ldots, a_m \in \mathbb{Z}_0$*,* $\overrightarrow{x_a^i} = [(h_1, a_1), \ldots, (h_m, a_m)]$ *a vector of pairs space-index of compound i,* $w_1, \ldots, w_n$ *are vectors of* Compound Binds $w \in \mathbb{Z}_0 \times \mathbb{R}$*. Then a* Neural Compound Multi-Space*, with k compounds, will be* $[(\overrightarrow{x_a^1}, \overrightarrow{\omega_1}), \ldots, (\overrightarrow{x_a^k}, \overrightarrow{\omega_k})]$*, in which every* $\overrightarrow{x_a^i}$ *may have a different size m as well as every* $\overrightarrow{\omega_i}$*, and* $i = 1 \ldots k$ *.*

Predicates, a part from the symbols uniquely indexed in the corpus, have their positive and negative occurrences, and so there will be two regions for predicate space too. This is one of the difference between predicates and functions, the other is their logical value. So, the spaces for them are defined as follows

Definition 5 (Function and Predicate Neural Multi-Spaces) *Let* C_f *,* C_p^+ *and* C_p^- *be NeComMS, such that* C_f *has index space one (1), and* C_p^+ *and* C_p^- *have both index space two(2). Then* C_f *is called a* Function Neural Multi-Space*, and every* $\overrightarrow{\omega}$ *appearing in* C_f *represents a vector of influences upon elements of space two (2). The pair* (C_p^+, C_p^-) *is called a* Predicate Neural Multi-Space *(LMS), in which every* $\overrightarrow{\omega}$ *appearing on both represent a vector of influences upon elements of space three (3) of clause.*

Clause spaces are simpler than compound spaces (functions and predicates) because clauses have "attributes" (their literals), but exert no influence upon spaces above, at least for the scope out our current research. One may think in terms of non-classical logics as adding other spaces composed of clauses that influence them. A clause is just an special case of a compound MS in which every weight vector has just one dimension pair $(\_, 0)$, where the _ symbol represents an anonymous logical object, and 0 represents no known influence to above spaces. The attributes will represent the literals that compose the clause.

Definition 6 (Neural Multi-Space of Clauses) *Let* $k_1, \ldots, k_m \in \mathbb{Z}_0$ *be predicate index codes, a* Neural Clause *at clause multi-space is* $C = ([(2, k_1), \ldots, (2, k_m)], [(\_, 0)])$*. A* Neural Multi-Space of Clauses *is simply* $[C_1, \ldots, C_n]$ *in which every* $C_i, i = 1..n$*, is a neural clause.*

Definition 7 (Shared NeMuS of CFOE) *Let* $\mathcal{S}$*,* $\mathcal{F}$*,* $\mathcal{P}$ *and* $\mathcal{C}$ *be a subject, function, predicate and clause neuronal multi-spaces. Then, we call a* Shared Neural Multi-Space of CFOE *to the ordered space* $\langle \mathcal{S}, \mathcal{F}, \mathcal{P}, \mathcal{C} \rangle$

4 Amao Learning Mechanism

In this section we present shared NeMuS learning process that is based on Kohonen [6] Self-Organizing Maps (SOM), although any learning mechanism could be used. Because shared NeMuS is not a standard matrix as in vector-spaces, distance measures are performed in different ways as it shall be clear in the sequel. The SOM training phase calculates the euclidean distance from the input vector to every neuron on the map. After that, it searches for the best match unity and updates the weights of every neuron in the neighborhood. The neighborhood of a single clause is defined by the index of a predicate. The following equation is used to update the weight vector:

$$\overrightarrow{\omega}_{(t+1)} = \overrightarrow{\omega}_{(t)} + \eta(\overrightarrow{\omega}_I - \overrightarrow{\omega}_{(t)}) \tag{1}$$

in which η is the learning rate, $\overrightarrow{\omega}_{(t)}$ and $\overrightarrow{\omega}_I$ are a multi-space vectors of weights, and t represents the epoch of interaction. We adapted the best match unity $\overrightarrow{\omega}_{bm}$ for our purposes, making it possible to apply resolution on clauses with complementary literals. In NeMuS this is easily obtained because the representation of any literal is its predicate index code in the positive region of the predicate space, and its complementary literal should have same index in the negative region, so the access is of complexity $O(1)$.

Training steps

This phase starts after the shared NeMuS structure had just been created from the compilation of the symbolic KB.The input for training is the KB itself and the steps are divided into two parts: one to deal with ground atomic formulae and the other deals with formulae with variables, henceforth called *deduction rules* (defined in section 3.1). Let $\overrightarrow{\omega}_I$ be an arbitrary input where its weights are represented by a CFOE, and $N_{KB} = \langle \mathcal{S}, \mathcal{F}, \mathcal{P}, \mathcal{C} \rangle$ a shared NeMuS.

Algorithm 1 Chain training

```
1: for every clause C ∈ 𝒞 do
2:     if C has just one literal then                          ▷ (Process of ground atoms.)
3:         for k ∈ C a index for predicate codes do
4:             ω⃗_(t+1) = [ω_C, ω_k, ω_1..., ω_m] + η(w⃗_I − [ω_C, ω_k, ω_1..., ω_m])
5:     if C has more than one literal then                     ▷ (Process of deduction rules.)
6:         for k a index for predicate codes ∈ C do
7:             if k > 0 then
8:                 for f a function attribute of predicate with code k do
9:                     for v a variable ∈ f do
10:                        {ω_C, ω_k} = {ω_k, ω_f} = {ω_bm, ω_bm}, for every function, or literal in clause C.
11:            else
12:                for a variable attribute v from predicate with code k do
13:                    {ω_C, ω_k} = {ω_bm, ω_bm}, for every literal ∈ C.
```

Refutation Pattern Learning Mechanism

The refutation pattern learning mechanism of Amao, called *NeMuS NeuraLogic Reasoner*, will try to find one refutation pattern for the input vector, has two important tasks that defines what was learned.

1. to recognize the refutation for a query (deduction rule inference with no premisses), it just needs to identify the region within its trained shared NeMuS for which all variable can be assigned a value.
2. to recognize the refutation for a ground formulae with more than one literal, it just need to compare the weight values of the input with the region indicated in the training phase, and if it is different, the answer is $false$, otherwise $true$.

Running Experiment on Refutation Pattern

The first test shows classical *Modus Ponens* reasoning with deduction rules having no restriction on the number of variables, and also when we have one level function. Using the NeMuS training on knowledge base presented in Example 1, we obtained these weights:

Symbolic Representation	Neural Representation
1. p(a).	(1.44, 0.84, 1.44)
2. p(b).	(1.44 0.84, 1.44)
3. r(a).	(3.12, 1.68, 2.04)
4. r(c).	(3.12, 1.68, 2.04)
5. q(X, f(X)); ∼p(X).	(1.04, 0.84, 1.44, 0.84, 1.44)
6. s(X, fY));∼r(X); ∼p(X).	(1.52, 1.68, 2.04, 0.84, 1.44, 1.68, 2.04, 0.84, 1.44)

Table 1. A NeMuS net trained

There are two important things to consider regarding the test results. The first is the translation of the symbolic input (query) into a NeMuS format with its vector of input weights $\overrightarrow{\omega}_I$. Second, identify the region this NeMuS object is most likely to belong. Furthermore, there must be a "kind of relation" between the input and the region which best matches it. For this Amao NeuraLogic reasoner creates a relation between region and the input. On the following table we present a best match selection from a single proof:

Proof $p(X)$:

1. Converting to $\overrightarrow{w_I}$: $\{1.44, 0.84, 0\}$
   The conversion of X is 0, because it is not in p.
2. Search for best match:

Distance $p(X) \leftrightarrow p(a)$	1.44
Distance $p(X) \leftrightarrow p(b)$	1.44
Distance $p(X) \leftrightarrow r(a)$	2.20617
Distance $p(X) \leftrightarrow r(c)$	2.20617
Distance $p(X) \leftrightarrow q(X, f(X))$	2.76261
Distance $p(X) \leftrightarrow q(X, f(Y))$	4.17248

With the information about the distance, NeuraLogic reasoner can define a relation between input and the best match solution. With the shortest distance 1.44, X can assume two values, $\{X/a\}$ and $\{X/b\}$. Now we are going to force a true and false for proposition, asking for:

Proof $s(a, f(c))$:

1. Converting to $\overrightarrow{\omega}_I : \{1.68,\ 0,\ 0.84,\ 0, 1.68,\ 2.04,\ 0.84,\ 1.44\}$
   From the translation we know that exist $r(a)$ and $p(c)$, and so their values are not 0. However, as it is not known whether there is a $s(a, f(c))$, their values for that positions are 0.
2. Search for best match give us: 2.49704.
   So now the shared NeMuS learn that 2.49704 is true.

Proof $s(c, f(b))$:

1. Converting to $\overrightarrow{\omega}_I : \{1.68,\ 0,\ 0.84,\ 0, 1.68,\ 0,\ 0.84,\ 0\}$

2. Search for best match give us: 3.53135
   With this shared NeMuS knows the maximum distance for true is 2.49704 and the answer is so far, that it's false.

Example 2. This example shows how first-order inductive learning can be easily dealt when recursive deduction rules are defined. For instance, to find a path on a graph can be simply defined with this knowledge base.

1. $link(a, b)$. 3. $link(c, d)$. 5. $path(XY)$; $\sim link(X, Y)$
2. $link(b, c)$. 4. $link(d, e)$. 6. $path(X, Y)$; $\sim link(X, Z); \sim path(Z, Y)$.

After knowledge base be represented it's possible to do training process:

Symbolic Representation	Neural Representation
1. link(a,b).	(3.35, 0.974, 1.44, 1.44)
2. link(b,c).	(3.35, 0.974, 2.28, 2.28)
3. link(c,d).	(3.35, 0.974, 3.12, 3.12)
4. link(d,e).	(3.35, 0.974, 3.96, 3.96)
5. path(X, Y);∼link(X, Y).	(4.89, 0.974, 3.39, 3.39, 0.974, 3.39, 3.39)
6. path(X,Y);∼link(X, Z);∼path(Z, Y).	(4.89, 0.974, 3.39, 3.39, 0.974, 3.39, 3.39, 0.974, 3.39, 3.39)

Table 2. Trained base of path between links problem.

Notice that there is a recursive rule, when $X \neq Y$ on clause 5, a value for Z is necessary on 6. So this search goes on until $link(X, Y)$ is true, or no path from X *to* Y is found. Our proposition is to give such a responsibility to NeuraLogic reasoner to perform an iterative process to verify if there is a path from X *to* Y by checking region weights. This is described in the following process to deal with **path(X, Y)**.

For i a weight $\in \mathcal{P}$
- If there's a index CFOE with $\overrightarrow{\omega}_i$ and $\overrightarrow{\omega}_Y$ with the same weight, answer true.
- Else

- If there's a index CFOE with weight $\overrightarrow{\omega}_i$ and $\overrightarrow{\omega}_X$ with the same weight
  $\overrightarrow{\omega}_X \leftarrow \overrightarrow{\omega}_i$
- Else the answer is false.

For now we can not avoid this iterative process to express a recursive execution, so NeuraLogic reasoner have only to give the right answer when it is asked for a link.

5 Related Work

Developing robust and sound, yet efficient, neural-symbolic learning and reasoning is the ultimate goal of the marriage between neural networks and symbolic (logical) reasoning[3]. The approach presented in this paper falls in the category of the ones pursuing for a feasible representation to overcome John McCarthy's claim that connectionist systems have propositional fixation[10], but which provides a feasible implementation to achieve useful performance.

Some recent approaches that sought to overcome this issue have proposed frameworks to allow expressive representation of complex nesting of symbols in first-order formulae. Komendantskaya proposed unification neural network [7], to allow first-order connectionist deduction. Practical results were not proven to be easily achieved for arbitrary first-order formulae having a (potential) infinite number of symbols. The proposed CFOE representation (section 3.2) has no such limit, and the sharing of neural CFOE makes the access of any neuron of $O(1)$ complexity in any case, while saving storage space. This is also an advantage when compared to Pinkas, Lima and Cohen, [11], who designed pools (tables) for symbols to allow the nesting of bindings and to keep track of unification. Despite the claimed efficiency when compared to the former, the pools are actually matrices representing directed acyclic graph. Sets of formulae with different numbers of terms and literal would generate sparse matrices compromising the complexity of the algorithms for learning and reasoning.

Guillame-Bert, Broda and Garcez, [5], encoded first-order formulae as vectors of real number from Cantor set aiming to provide neural-symbolic inductive learning about first-order rules. The type restriction on terms, but not on sub-terms, weakened the claimed expressive power. The generation of codes for large sets of first-order sentences may have an impact on the efficiency of the training process. Besides, our approach does not suffer the type restriction since it is already based on a multi-space concept where every logical symbol e well placed in its appropriate space.

6 Concluding Remarks

In this paper we presented a novel approach for neural-symbolic learning and reasoning of first-order logic. Our main purpose was to create a neural model that we could characterize *patterns of proof by refutation*, based on the resolution principle with unification for first order inference. There were two well known challenges to be tackled in or der to achieve this general and ambitious goal: to overcome the propositional fixation and a neural network architecture that could allow efficient computations. This means, Amao should perform reasoning faster than symbolic approaches as it should take advantage of having learned something about the domain.

These challenge were dealt with a little ingenuity of the shared NeMuS (Neural Multi-Space approach), which combines Smarandache multi-space modeling technique with sharing of structure concept from Boyer-Moore efficient implementation of Prolog engines. By separating in spaces

constants and variables, functions, predicates (literals) and clauses, we treated each of this logical objects as a type since each has specific computations for the overall neural computation of learning and reasoning.

Our main contribution was to show, like in Example 2 (in the end of section 4), that first-order neural-symbolic reasoning does not need to compute the entire Herbrand base (i.e. the set of ground atomic formulae). Amao used its trained shared NeMuS to iterate over the regions of similar ground atomic formulae and efficiently find a refutation or say the query does not follow from what it has learned. However, some interesting challenges remain to be tackled and we point some here.

- recursive deduction rules generating a potentially infinite number of ground terms, e.g. $s(s(s(\ldots)))$, were not tested. Although Amao is not likely to deal with it, another space orthogonal to all others seem to be one solution to deal with recursive loops on functions.
- a part from induction inference by recursive rules, which other kinds of deduction pattern can a self-trained NeMuS recognize?

References

1. Bader, S., Hitzler, P., Hölldlber, S.: Connectionist model generation: A first-order approach. Neurocomputing 1(71), 2420–2432 (2008)
2. van Emden, M.H.: An interpreting algorithm for Prolog programs, Ellis Horwood Series Artificial Intelligence, vol. 1, chap. 2, pp. 93–110. Ellis Horwood (1984)
3. d'A. Garcez, A., Besold, T.R., de Raedt, L., Földiak, P., Hitzler, P., Icard, T., Kühnberger, K.U., Lamb, L.C., Miikkkulainen, R., Silver, D.L.: Neural-symbolic learning and reasoning: Contributins and challenges. In: AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches - Dagstuhl (2014)
4. d'Avila Garcez, A.S., Broda, K., Gabbay, D.: Neural-Symbolic Learning Systems: Foundations and Applications, Perspectives in Neural Computing. Springer-Verlag (2002)
5. Guillame-Bert, M., Broda, K., d'Avila Garcez, A.: First-order logic learning in artificial neural networks. In: International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2010)
6. Kohonen, T.: Self-Organizing Maps. Springer, 3rd edn. (2001)
7. Komendantskaya, E.: Unification neural networks: unification by error-correction learning. Logic Journal of the IGPL 19(6), 821–847 (May 2010)
8. Konheim, A.G.: Hashing in Computer Science: Fifty Years of Slicing and Dicing. John Wiley & Sons (2010)
9. Mao, L.: An introduction to smarandache multi-spaces and mathematical combinatorics. Scientia Magna 3(1), 54–80 (2007)
10. McMCarthy, J.: Epistemological challenges for connectionism. Behavioral and Brain Sciences 11(1), 11–44 (1988)
11. Pinkas, G., Lima, P., Cohen, S.: Representing, binding, retrieving and unifying relatinal knowledge using pools of neural binders. Elsevier Biologically Inspired Cognitve Architectures 1(6), 87–95 (2013)
12. Robinson, A.: A machine-oriented logic based on the resolution principle. Journal of the ACM 12(1), 23–42 (1965)
13. R.S. Boyer, J.M.: The sharing of structure in theorem-proving programs. In: Bernadrd Meltzer, D.M. (ed.) Annual Machine Intelligence. vol. 7, pp. 101–116. Edinburgh University Press (1972)
14. Vieira, N.: Máquinas de Inferência para Sistemas Baseados em Conhecimento. Ph.D. thesis, Pontifícia Universidade Católica do Rio de Janeiro (1987), phD Thesis