The Limitations of Description Logic for Mathematical Ontologies: An Example on Neural Networks

Fred Freitas, Fernando Lins

Informatics Center - Federal University of Pernambuco (CIn - UFPE)
Av. Prof. Luis Freire, s/n, Cidade Universitária, 50740-540, Recife – PE – Brazil

{fred,fval}@cin.ufpe.br

Abstract. *In this work, we discuss appropriate formalisms to account for the representation of mathematical ontologies, and, particularly, the problems for employing Description Logics (DL) for the task.DL has proven to be not expressive enough to such tasks, because it cannot represent, for instance, simple numerical constraints, except of cardinality constraints over instances of individuals and relation instances. Therefore, for such representations, we advocate the use of a more expressive formalism, based at least on first-order logic using a top ontology as support vocabulary. We also provide a thorough example on the representation of an Artificial Neural Network (ANN) defined in KIF (Knowledge Interchange Format), along with a discussion on the requirements needed for such a representation and the results achieved.*

1 Introduction

All In this work, we introduce a discussion on the appropriate formalism to carry out the task of representing mathematical knowledge, and particularly the problems for doing it with Description Logics (BAADER ET AL, 2003). It departed from an attempt to develop an Ontology of Multilayer Perceptron (MLP) Artificial Neural Networks (ANN). Our initial intent was to define it using the Semantic Web standardized description logic language OWL (Ontology Web Language) (WELTY ET AL, 2004), which would offer us a lot of benefits, like mature development tools, availability of reasoners, large community of users, etc.

On creating such ontology, our work was focused in developing a conceptual model that would include the necessary mathematics to enable the future development of knowledge-based applications that can reason over artificial neural networks, such as an intelligent agent capable of interacting with users to answer questions about neural networks that can even create and run them.

During the development of the ontology, we faced several representational problems while trying to represent the mathematical expressions inherent to neural networks in Description Logic (DL). Since this formalism is based on set theoretic semantics, it fits properly to representing domains in which relationships among sets suffice. For such domains, DL terminologies can accurately describe the domains' classes. On the one hand, the choice of this formalism for the Semantic Web lies actually on these grounds. On the other hand, it is not endowed with any apparatus to deal with many types of mathematical operations and constraints that would turn it into a more generic formalism, like what can be represented by Prolog, for instance. Moreover, a core mathematical ontology is required as a vocabulary to make for the needed complex well-defined mathematical meanings.

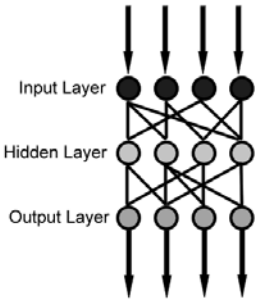
Therefore, for such representations, we advocate the use of a more expressive formalism, based at least on full first-order logic, and using a rich, well-defined support vocabulary. In order to illustrate the discussion, we first introduce the required basic mathematical definitions about MLP neural networks and the expressiveness requirements that a language should fulfill to represent the ontology. Then, we discuss the practical expressiveness limitations that hamper the use of Description Logic for such representations. Finally, we present our MLP ANN ontology and some of its more important definitions. We developed it in the Standard Upper Ontology Knowledge Interchange Format (SUO-KIF) language (PEASE 2009) and used the top ontology Suggested Upper Merged Ontology (SUMO) (NILES & PEASE, 2001) as support vocabulary, which proved well-tailored for our mathematical representations. We complete the article giving a very brief introduction of the SUO-KIF language and the top ontology SUMO, discussing some related work, future work and conclusions.Full papers must respect the page limits defined by the conference. Conferences that publish just abstracts ask for **one**-page texts.

2 An Example: Representing the MLP ANN in an Ontology

Artificial Neural Networks (ANNs) were developed as a rough abstraction of biological neural networks. An ANN consists of a number of nodes that perform simple numerical processing. Its nodes are highly interlinked and grouped in layers. Mathematically speaking, it implements complex function approximators (HAYKIN, 1994). ANNs are used in the computer mainstream to perform pattern learning and recognition.

The most commonly used ANN is the Multilayer Perceptron (MLP) (HAYKIN, 1994). It has three layers (entry, hidden and output) or more (when more than one hidden layer is used). Each layer consists of a set of "artificial neurons" connected with the neighbor layers, as displayed in the figure 1 below.

Figure 1– A MLP Artificial Neural Network, its layers and neurons.



A MLP ANN "learns" the implicit function it should approximate by computing, correcting and propagating classification errors throughout all its neurons, as follows. In the training phase, the MLP ANN is given a set of problem instances, usually represented as numerical values, along with their solutions, and process the data many times to assign correctly the neurons' parameters. Synapses among artificial neurons are modeled via activation functions, which trigger the delivery of numerical stimuli as inputs to the next layer's neurons. At the end of the learning phase, the network is tuned with the parameters so as to reflect as closely as possible the classification function implicit in the training set.

The Backpropagation algorithm (HAYKIN, 1994) is used to train the MLP ANN. It is divided into two phases. In the *forward phase*, the network is run and a classification output is calculated. In the backward phase, it compares the output with the correct result and, if wrong, it calculates and propagates the error through the weights of the connections among all the neurons of the network. Therefore, the learning ability of MLP ANNs lies on the neurons. The algorithm iterates between these two phases until it reaches a stopping criteria. These two phases are explained in detail in the next subsections.

2.1 Forward Phase

Here, the ANN basically computes summations of a product. At first, each neuron receives its entry and propagates it to the first hidden layer's neurons, if its activation function "triggers". For each neuron of the hidden layer, the potential of activation of the neuron j ($net_j^p, in\ the\ formula$) is calculated by summating all synapses weights (w_{ji}) from neurons i that reach neuron j, multiplied by the outputs of neurons ii:

$$net_j^p = \sum_{i=1}^{n} x_i^p w_{ji} \tag{1}$$

Then, the activation potential is applied to an activation function that will return the output of the neuron. One of the most used activation functions is the logistic function (also called sigmoid), that can be seen below:

$$P\left(net_j^p\right) = \frac{1}{1 + e^{-net_j^p}} \tag{2}$$

The output of the neuron, given by the result of the activation function, typically consists of a real value between -1 and 1. A synapse is defined as the propagation of the output as an entry to the next layer, or, if it is the last layer, one of the outputs belonging to the output vector of the network. Synapses in a neuron occur if its output is higher than a given threshold, usually set to 0. Synapses are then propagated (or not) through the network until they reach the output layer, finishing the forward phase. This phase is executed when the network is in the execution mode too.

2.2 Backward Phase

The first step of this phase consists in calculating the error between the output of the neuron j from the forward phase and the expected output from the instance of the training set, as in the following formula

$$\delta_j = \left(d_j - y_j\right) f'(net_j) \tag{3}$$

where d_j is the desired output, y_j the actual output, and $f'(net_j)$ the activation function of the neuron's derivate. Hidden nodes' errors are computed as follows:

$$\delta_j = f'(net_j) \sum_{i=1}^{n} \delta_l w_{lj} \tag{4}$$

They are the product of the activation function derivate of the current neuron and the summation of the products between the error (δ_j) and each of the last layers' weights (w_{lj}). Finally, the synapse weights that connect current neuron j with its antecessors (neurons i) are given by:

$$w_{ji}(t+1) = w_{ji}(t) + \eta\delta_j(t)x_i(t) \quad (5)$$

where $w_{ji}(t)$ stands for the old weight, $w_{ji}(t+1)$ means the new weight, η is the learning rate that regulates learning speed, $\delta_j(t)$ is the error and $x_i(t)$ is the input signal of the neuron with the old weight. The rough idea is slowly (as tuned by η) distributing the error issued by the network through all the layers and neurons.

In the next section, we enlist the requirements for representing such knowledge.

3 Requirements of Candidate Formalisms and Languages

To address the problem of choosing the knowledge representation formalism and language to represent the MLP ANN, we had to consider a number of representation requirements. We needed enough expressive power to represent the mathematical concepts on the ontology. Potential candidate formalisms and languages should, at the same time, allow us to do it in an easy way, as well as give us the possibility of computing results afterwards. Below, we list the most important requirements for that:

- **Ability to represent numerical constraints** - as seen above, MLP encompasses plenty of calculations. The ontology must be capable of representing them in an straight-forward way, as well as the constraints over them involving mathematical operations like summations, exponentiations or function derivations.
- **Availability of a rich vocabulary of abstract definitions** - the MLP ANN Ontology makes use of a lot of abstract concepts that are fulfilled either by a core mathematical ontology or by a top ontology which must include mathematical expressions, set theory, graph theory, algorithms, sequences, etc.
- **Ability to represent n-ary relations** - the freedom to use unlimited arities in relations leaves ontologists free to use some advanced constructs. For instance, a synapse connects two neurons, thus possessing a ternary relation named `connects` that has, as arguments, the synapse and the two neurons.
- **Ability to represent functions** - when dealing with mathematics concepts, we often need to represent functions. In the MLP ANN, we had to represent the activation function and the stopping criteria based on functions.

In the next section, we discuss the difficulties in fulfilling such criteria with DL.

4 Description Logics' Representational Problems regarding Mathematical Ontologies

DL and its Ontology Web Language OWL present in practice a number of limitations in expressiveness that prevented us from using it for representing the knowledge presented in the section 2. We discuss the main issues in the next subsections.

4.1 Description Logic Ontological Engagement

The first aspect to remark when embarking in such a discussion relates to the conceptual and epistemological commitment (BRACHMAN 1978) of DL as a representation formalism. Its representation purposes and consequent grounding on precise and unambiguous set theoretic semantics (BAADER ET AL, 2003) allows the language mainly to describe quite precisely domains that involve sets (as concepts or classes according to the formalism's jargon), elements (as class and relation instances), their relations (also called properties or roles) and simple constraints that can be expressed in axioms that use a limited set of constructs like relations' cardinality, type checking of relations' domain and range, classes' subsumption, complement, disjointness, equivalence, instance membership to a class and instance equality/inequality.. This is enough to make for the main reasoning task of concept classification, which means to entail class subsumption, when this relation is not declared explicitly[1]. Class subsumption, equivalence, disjointness and inconsistency (standing for a class that according to its definition cannot bear any instances) can be inferred by the reasoned, also called *classifier*.

In the next subsection we discuss the use of numerical constraints in DL.

4.2 Numerical Representations and Constraints

Despite consisting of a powerful family of semantically well-founded formalisms', DLs are not endowed with the basic mathematical resources necessary for representing neural networks, for instance the arithmetical operations. The only type of arithmetical processing is counting and comparing the number of instances that participate in relations for solving queries, like with the following axiom:

$$Worried\text{-}Woman \equiv Woman \sqcap (\geq 3\ child.Man) \qquad (6)$$

It states that a worried woman is one who has three or more male children. The classifier is able to solve queries that searches for the instances of the class *Worried-Woman* by counting the instances of *Woman* which are related to instances of the class *Man* via the relation *child*.

Although it is possible to define relations with numerical values as datatype properties in OWL (types integer and float and special types date, time and dateTime), we did not find in the literature DL extensions that address the general problem of solving arithmetical constraints, ranging from the simplest ones (like the ones that use the four basic arithmetical operations) to the most complex (using exponential, differential, integral operators, differential equations, etc).

Description Logic consists indeed in a smart purely declarative formalism that performs optimized classical reasoning. Sticking to this stance, no ultimate solution is expected to the posed problem of including arbitrary arithmetical constraints, because classifier will not cope with them, ought to the fact that elementary number theory has been proved undecidable by Gödel in his classical incompleteness theorem (GÖDEL,

[1] Classification based on subsumption is also used for solving equivalence and disjointness between classes and class inconsistency. Even in assertional queries, i.e., the ones which include instances, subsumption checking is the basic processing behind the reasoning (BAADER ET AL, 2003).

1931) as well as rational number theory, since the elementary number theory could be defined in terms of the rationals, as proved by Julia Robinson (ROBINSON, 1949).

So, the hopes for proposing a general DL solution with arithmetical constraints probably lies both on restricting the constructs for a limited set of operations and solving them with built-in constructs that should be evaluated before classification (in the flavor of Prolog, for instance). Such extension, on the other hand, would change the formalism into a more procedural and less declarative one, and is not yet reported.

4.3 The Need for a Mathematical Background Knowledge in a Top Ontology

Due to the low expressiveness of OWL, any top-ontology developed using it would face a lot of problems in representing complex abstract mathematical concepts. This statement is corroborated by the fact that, despite the existence of several efforts for the translation the top ontology Supper Upper-Merged Ontology (SUMO) to OWL, none of them proved successful. Translations can be found in the SUMO site but they are in OWL Full (which is known to be undecidable (BAADER ET AL, 2003)). Besides this fact, any translations of full first order logic to DL will just prune the knowledge from the original ontology that cannot be expressed in DL.

We indeed have tried to work with these translations, encountering many practical problems. The two more relevant for us were the loss of a the rich set of mathematical axioms provided by SUMO upper and the necessary mathematical operations and constraints that could not be defined in DL, due to the lack of expressiveness.

5 Solution: More Expressive Languages and Top Ontologies

Since DL does not fulfill our expressivity needs, the solution lies on relying on employing a more expressive language, with first-order logic (FOL) expressivity at least and, if possible, capable of defining basic arithmetic operations, and a top ontology that includes some basic mathematical definitions. A Top Ontology defines a set of generic concepts to be shared by various domain-specific ontologies. Complying with it assures semantic interoperability among the ontologies and better ontological engagement (BRACHMAN, 1978). Thus, based on the requirements enlisted above, we decided to use the SUO-KIF (Standard Upper Ontology Knowledge Interchange Format) with SUMO as the background mathematical knowledge required.

The SUO-KIF language has declarative semantics and is quite comprehensive in terms of logical expressiveness, once it was intended primarily to implement the first-order logic. It is an extension of KIF (GENESERETH, 1991), created with the objective of supporting the development of SUMO. The goal was simplifying KIF, by maintaining its syntax, logical operators and quantifiers, but leaving to the ontology itself the declarations defining classes and instances, and thus eliminating the dependency from the Frame-Ontology of KIF, what happens to Eng-Math (GRUBER & OLSEN, 1994). Instead, one must create instances based on the concepts defined in SUMO as an external resource. Another relevant capability of SUO-KIF when compared to KIF's Eng-Math is the deployment of the Sigma reasoner (PEASE, 2003). Up to now, no reasoner can take KIF's rich expressivity on. As for the top ontology, we

opted for SUMO, since it endows us with the math we needed. In the next section, we present the MLP ANN ontology, with some examples using SUO-KIF and SUMO.

2. The Resulting MLP ANN Ontology in SUO-KIF

In this section, we present the most important concepts of the Multi-Layer Perceptron Artificial Neural Network (MLP ANN) Ontology. We deploy the basic taxonomy of the ontology in Figure 2. Note that it uses plenty of concepts from SUMO, like the classes Abstract, Physical, Relation, BinaryRelation, IrreflexiveRelation, etc. Next, we take a deeper look at the ontology, describing some of the concepts and axioms that required a higher level of expressiveness.

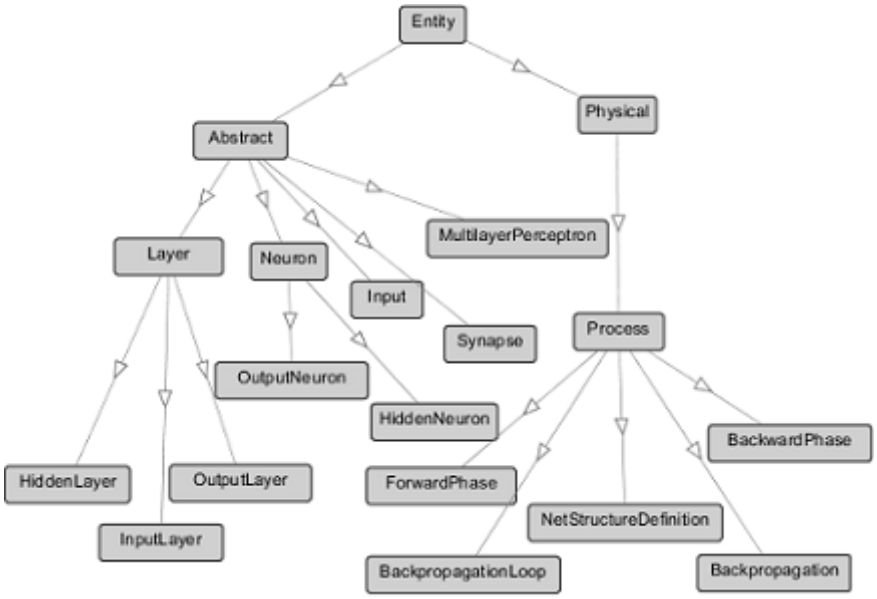


Figure 2. The ANN MLP Ontology basic taxonomy.

5.1 Activation function

The representation of the activation function constitutes a good example of a function definition that takes advantage of the SUO-KIF expressiveness and the richness of the mathematical vocabulary provided by SUMO. In the code below, we state the concept of `ActivationFunction` as a function (irreflexive, intransitive, assymetric relation) whose domain is the class Neuron.

```
(instance activationFunction BinaryPredicate)
(instance activationFunction IrreflexiveRelation)
(domain activationFunction 1 Neuron)
(domain activationFunction 2 UnaryFunction)
```

Then we represent the concept of a Logistic function, which is an unary function that receives a real number as the only argument and returns an `MLPRealNumber`.

```
(instance Logistic UnaryFunction)
(instance Logistic TotalValuedRelation)
(domain Logistic 1 RealNumber)
(range Logistic 1 RealNumber)
```

`MLPRealNumber` is defined as a real number ranging between -1 and 1:

```
(=>(instance ?NUMBER MLPRealNumber)
   (and (or (greaterThan ?NUMBER -1)
            (equal ?NUMBER -1))
        (or (lessThan ?NUMBER 1)
            (equal ?NUMBER 1))))
```

Finally we define the logistic function, as defined in subsection 2.1., def. (2):

```
(=>(equal (Logistic ?NUMBER1) ?NUMBER2)
   (equal ?NUMBER2 (DivisionFn 1
     (AdditionFn 1 (ExponentiationFn NumberE
            (SubtractionFn 0 ?NUMBER1))))))
```

Next, we describe the ANN learning process.

5.2 Backpropagation algorithm process

The process is an iteration alternating the backward and forward phases, until the halting criterion is met. We can represent it using four axioms. The first one states that the backpropagation algorithms starts defining its net structure

We can represent this process using four axioms. The first one states that for all backpropagation algorithms there exists a sub-process (the definition of the net structure) starting together with the backpropagation algorithm:

```
(=>(and(instance ?B Backpropagation)
        (instance ?NSD NetStructDefinition)
        (subProcess ?NSD ?B))
   (exists (?BL)
        (and(instance ?BL BackpropLoop)
            (subProcess ?BL ?B)
            (exactlyEarlier (WhenFN ?NSD)
 (WhenFn ?BL)))))
```

The second axiom states that right after the net structure definition, we start the iteration over the backpropagation loop. The `exactlyEarlier` relation means that the first argument finishes at the same instant that the second starts.

```
(=>(and(instance ?B Backpropagation)
        (instance ?BL Backproploop)
          (subProcess ?BL ?B)
          (not (StopCriteriaFn ?B)))
     (exists (?BL)
          (and(instance ?BL2 BackpropLoop)
            (subProcess ?BL2 ?B)
            (exactlyEarlier (WhenFN ?BL)(WhenFn?BL2)))))
```

Finally, the last axiom states that, if the halting criterion is met, the algorithm ends. The `finishes` relation means that both arguments finish at the same time:

```
(=>(and(instance ?B Backpropagation)
        (instance ?BL BackpropLoop)
          (subProcess ?BL ?B)
          (StopCriteriaFn ?B))
     (finishes (WhenFn ?B) (WhenFn ?BL)))
```

The weights are updated (subsection 2.2., definition 5) according to the axiom shown below. It states that if a weight is identified as one to be updated, then the old weight holds only until the loop cycle lasts, and the new weight is calculated and set.

```
(=>   (and
      (instance ?UPDATEWEIGHT UpdateSynapseWeight)
      (synapseToUpdate ?UPDATEWEIGHT ?SYNAPSE)
      (connects ?SYNAPSE ?NODEA ?NODEB)
      (hasOutput ?NODEB ?OUTPUT)
      (holdsDuring (BeginFn ?UPDATEWEIGHT)
      (hasWeight ?SYNAPSE ?OLDWEIGHT)))
      (holdsDuring (EndFn ?UPDATEWEIGHT)
      (and
         (hasWeight ?SYNAPSE ?NEWWEIGHT)
         (equal ?NEWWEIGHT (AdditionFn ?OLDWEIGHT
                (MultiplicationFn ?LEARNRATE
                (MultiplicationFn ?OUTPUT
                      (NeuronErrorFn ?SYNAPSE)))))))
```

5.3 An Example of Ternary Relation: The Synapse Definition

In the next encodings, we show an example of a ternary relation, the relation `connects`, that links two neurons through a synapse. The first stretch of code shows the basic characterizations of the relation in terms of inputs and outputs:

```
(instance connects TernaryPredicate)
(domain connects 1 Synapse)
(domain connects 2 Neuron)
(domain connects 3 Neuron)
```

Next, an axiom is defined, stating that all connections of a neuron are linked with a neuron from its next layer. Note that a synapse takes part of the ternary relation, thus sufficing for our representation needs.

```
(=>(and(instance ?N-A Neuron)
        (connects ?SYNAPSE ?N-A ?N-B))
   (exists(?LAYERA ?LAYERB)
          (and(instance ?LAYERB Layer)
               (instance ?LAYERA Layer)
               (neuronLayer ?N-A ?LAYERA)
               (neuronLayer ?N-B ?LAYERB)
(nextLayer ?LAYERA ?LAYERB))))
```

In the next section, we discuss related work regarding mathematical ontologies.

5.4 Discussion

The ontology defines the exact constraints that hold among the many elements of a MLP ANN, as well as the calculations and sequential operations needed to update it during the training phase. Once SUO-KIF is endowed with a reasoned able to deal with the mathematical constraints, an application that employs it – an intelligent agent, expert system, computer algebra system or theorem prover - is capable of creating a virtual MLP ANN, run it (including all the calculations and updates), and - more important in terms of declarativity – answer queries about each aspect of it, like stating how many layers at least a MLP ANN should possess, etc. With that features, even an intelligent tutor system can make a good use from its knowledge.

Note that, when an expressive formalism is used with a supportive top ontology, most, if not all, mathematical knowledge can be represented by a similar solution. Simple examples are other types of neural networks (even recurrent, constructive)

6 Related Work

In the field of Artificial Intelligence, the first successful systems to deal with mathematical contents that go beyond simple numerical calculi came from the branch known as Computer Algebra Systems (CAS) (BERTOLI ET AL 1998). They are capable of performing algebraic and symbolic computation in an abstract way A representative example of this trend was the REDUCE system (HEARN 2004), which is still in use nowadays. It is able to calculating algebraic derivates and integrals of complex functions among many other functionalities. Many of those systems employ declarative solutions for algebraic problems and some of them are able of using and presenting proofs indeed. A detailed comparison among computer algebra systems and their features can be found in [13]. Nevertheless, the integration between this type of system and the growing field of ontology is still lacking tough. The consequence is that some valuable mathematical knowledge available in mathematical ontologies, like the characterization of distinct relation types used here, are not being fully exploited in CAS systems.

As for the problem of mathematical knowledge in ontologies, we have searched the literature for possible solutions to our problems, before deciding for the SUO-KIF language. We found solutions ranging from simple syntactic agreements, like MathML (CARLISLE 2009) - a markup language without axioms -, to full-fledged languages and top ontologies like SUO-KIF and SUMO. Among them, we found a proposal to mix OWL with OpenMath to encode math contents (BUSWELL 2004). We indeed made attempts to employ this approach, by to representing the math needed in the MLP ANN ontology in OWL and processes using the Semantic Web Rule language (SWRL) built-ins. However, none of these solutions fitted our needs. All of them were limited, in the sense of depending upon external features (like the rules in SWRL), instead of using a mathematical theory. They mostly represent the expressions in a structured way and even the solutions using rules will produce fragmented pieces of knowledge that could pose problems of maintenance in the future.

Furthermore, we tried out a more consistent approach, the use of EngMath (GRUBER & OLSEN, 1994), a mathematical ontology for engineering. EngMath could indeed constitute a sound alternative, as it takes advantage of the Kif-Numbers ontology

(GRUBER & OLSEN, 1994), a KIF vocabulary focused on numbers, arithmetic operations and related definitions. We chose SUMO and SUO-KIF because, we conclude that SUMO covers all definitions comprised in Kif-Numbers and EngMath.

7 Future Work and Conclusions

Despite the popularity and usefulness of DL languages, we claim that this formalism suffers from a lack of expressiveness for the representation of mathematical knowledge. In our practical experience portrayed here, the most relevant lesson learned was that we could only overcome DL representational problems by using at least a full first-order logic formalism with a supportive top ontology like SUMO, that contains the mathematical background knowledge required to qualify and define properly the math definitions for the new ontologies. We presented a thorough case study in that direction, on the field of automatic learning, the MLP ANN ontology.

We envisage two types of future work. As for the use of the developed ontology, we are heading for practical applications of the ontology. The creation of these applications, such as an intelligent agent capable of interacting with users answering questions about artificial neural networks (making use of the ontology) are in our agenda. We also consider the translation of the ontology to other languages, such as Prolog, so as to enable us to reason with it and run concrete applications.

Another more general future work lies on the investigation of ways to embed ontologies into computer algebra systems. Devising a solution for this general problem will certainly endow the latter with more powerful reasoning techniques while solving mathematical problems. For instance, CAS systems could take advantage during inference of the applicable mathematical constraints defined as relations qualifiers in the ontology. These constraints need not be hardcoded in the systems, thus increasing knowledge reuse.

Acknowledgments. The authors would like to thank prof. Richard Banach, from the University of Manchester, England, who provided us with valuable input on the limitations of Description Logic for Mathematical representations, and prof. Jacques Calmet, from the University of Karlsruhe, Germany, who, as scientific ancestor, indirectly introduced us to the field of Computer Algebra.

References

BAADER, F., CALVANESE, D., MCGUINNESS, D.L., NARDI, D., PATEL-SCHNEIDER, P.F., eds.: **The Description Logic Handbook**. Cambridge Univ. Press (2003)

BERTOLI, P.G., CALMET, J., GIUNCHIGLIA, F., HOMANN, K.: **Specification and integration of theorem provers and computer algebra systems.** In Calmet, J., Plaza, J., eds.: AI and Symbolic Computation: Proceedings of AISC'98, Berlin, Germany, Springer (1998) 94-106

BRACHMAN, R.J.: **On the epistemological status of semantic networks.** BBN Report 3807, Bolt Beranek and Newman Inc., (apr 1978)

BUSWELL, S. CAPROTTI, O.,CARLISLE, D.,.DEWAR D., GAËTANO, M., KOHLHASE, M.: **Open math standard report**, Open Math Society (2004)

CARLISLE, D., MINER, R., ION, P.: **Mathematical markup language (MathML) version 3.0. Candidate recommendation**, W3C (dec 2009)

GENESERETH, M.R.: **Knowledge interchange format**. In Allen, J.F., Fikes, R., Sandewall, E., eds.: Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, San Mateo, California (1991) 599-600

GÖDEL, K.: **Über formal unentscheidbare sätze der principia mathematica und verwandter systeme**. Monatshefte für Mathematik und Physik 38 (1931) 173-198

GRUBER, T.R., OLSEN, G.R.: **An ontology for engineering mathematics**. In Doyle, J., Sandewall, E., Torasso, P., eds.: Principles of Knowledge Representation and Reasoning. Morgan Kaufmann, San Francisco

HAYKIN, S.: **Neural networks**. MacMillan, New York (1994)

HEARN, A.C.: **Reduce user's and contributed packages manual, version 3.8. Report**, The RAND Corporation, Berlin, Germany (2004)

NILES, I., PEASE, A.: **Towards a standard upper ontology.** In: Proc. of the int.conference on Formal Ontology in Information Systems, New York, NY, USA, ACM (2001) 2-9

PEASE, A. **The Sigma Ontology Development Environment.** In Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems, Acapulco, Mexico, 2003

PEASE,A. **A Standard Upper Ontology Knowledge Interchange Format**. 2009

ROBINSON, J.: Definability and decision problems in arithmetic. Journal of Symbolic Logic 14 (1949) 98-114

WELTY, C., MCGUINNESS, D.L., SMITH, M.K.: **OWL web ontology language guide. W3C recommendation**, W3C (2004) http://www.w3.org/TR/2004/RECowl-guide-20040210/.

WIKIPEDIA: **Comparison of computer algebra systems**. en.wikipedia.org/wiki/Comparison of computer algebra systems, 2010.