Towards Integrating Fuzzy Logic Capabilities into an Ontology-based Inductive Logic Programming Framework

Josué Iglesias
Telecommunications Engineering School
Technical University of Madrid, Spain
Email: josue@grpss.ssr.upm.es

Jens Lehmann
Department of Computer Science
University of Leipzig, Germany
Email: lehmann@informatik.uni-leipzig.de

***Abstract*—Ontologies based on Description Logics (DLs) have proved to be useful in formally sharing knowledge across applications. Recently, several tools have extended ontologies with fuzzy logic capabilities in order to apply ontology-based reasoning to vague and imprecise domains. This paper first analyses the state of the art in tools for fuzzy ontologies management and then describes how some of the most significant ones have been integrated in order to extend an ontology-based Inductive Logic Programming (ILP) system with fuzzy logic capabilities. A fuzzy version of a well-known ILP test case has been developed in order to validate the approach. This research represents a first step towards fuzzy inductive reasoning for OWL ontologies.**

***Keywords*-ontologies; fuzzy logic; inductive learning programming**

I. Introduction and motivation

During the past 20 years, information systems have experienced significant improvements in intelligent information processing, thereby stimulating advances in the Knowledge Management area. The popularity of Internet technologies has led to combinations of Knowledge Management and web technologies, for instance in the area of Semantic Web. Since ontologies based on the OWL W3C standard form the backbone of a number of Semantic Web applications, the underlying Description Logics are now one of the most widely used knowledge representation formalisms in the web. Reasoning is one of the key features of these technologies, with a huge variety of possible application domains and different reasoning techniques for solving each particular problem.

Reasoning over imperfect information, which is inherent to most of the real world application domains, is one of the main research issues in Knowledge Management. In a broad sense, two approaches can be used to deal with non perfect information. The probabilistic approach is able to deal with the uncertain nature of the information (e.g., modelling sensor accuracy when acquiring data) whereas the fuzzy logic one is able to manage the vagueness of concepts arising from human perception and cognition processes (enabling to formally model, e.g., concepts as 'tall', 'cheap', 'easy', etc.).

Despite the success of ontologies, classical ontology languages are not appropriate to deal with imperfect knowledge [1], recently appearing several tools intended to complement this type of technology to support fuzzy and/or probabilistic knowledge representation and reasoning.

Within this vast world of Knowledge Management, Semantic Web and uncertainty support, the work presented in this paper focuses on adding fuzzy logic capabilities to DL-Learner [2], an already developed ontology-based Inductive Logic Programming (ILP) framework. The goal of DL-Learner is to provide a DL/OWL-based machine learning tool to solve supervised learning tasks, extending ILP to DL, OWL and the Semantic Web. The resulting tool integrates several third-party developments in order to be able to reason over fuzzy ontologies describing vague concepts, addressing previous works in DL-based ILP –e.g., [3]– from the perspective of Semantic Web standards and general purpose semantic tools. A complete fuzzy ontology test case, to validate this and future fuzzy ILP developments, is also presented.

The remainder of this paper is organized as follows. Next Section depicts the state of the art in tools for representing and reasoning over fuzzy ontologies. The particularities of each new component used to support the fuzzy extension of DL-Learner are detailed in Section III. The fuzzy ILP test case developed to validate this fuzzy extension of DL-Learner is presented in Section IV along with some experiments and results (Section V). Finally, Section VI offers some conclusions and future works.

II. Background

This section summarizes the state of the art in tools for fuzzy ontologies representation and reasoning. It concludes with an analysis and selection of the fuzzy tools finally used to extend DL-Learner.

A. Representing Fuzziness In Ontologies

Formalisms regarding *fuzzy ontologies* were introduced to represent semantic knowledge based on vague concepts and relations (e.g., [4]). Several approaches have emerged trying to implement those formalisms into OWL-based ontologies.

Some approaches focus on developing specific OWL ontologies formally defining the common elements of fuzzy set

theory to be later populated with instances representing the fuzzy axioms and elements of a particular domain ontology. A simple ontology was developed in [1] to demonstrate some basic functionality of exchanging uncertain information (not only for annotating vagueness), but it is still not mature enough to be applied to model real domains. In [5] an OWL ontology to extend relational databases with fuzzy information is proposed, but it only defines concepts based on the relational model thus lacking several expressions available in DL-based ontologies; besides, the ontology is not focused in solving reasoning problems but to act as interface to access fuzzy information stored in relational databases. *FuzzyOWL2Ontology* [6], a meta-ontology to represent fuzzy extensions of some OWL languages, is a further development in this line. Specific parsers can be developed translating fuzzy ontologies represented with *FuzzyOWL2Ontology* into the particular language used by a fuzzy DL reasoner.

Extending the OWL language to support fuzzy definitions is another strategy for building fuzzy ontologies. While some approaches propose extending the standard building blocks of the OWL language (e.g., [7]), others use the current OWL standard tools to represent such fuzzy information. As far as we are concerned, the work on *FuzzyOWL2* [8] is the most notable effort in this area. It uses OWL 2 annotation properties to encode fuzziness. The use of annotation properties makes fuzzy ontologies compatible with current OWL 2 management tools (editors, programmatic environments, etc.) and enable crisp OWL-based reasoners to compute inferences over this kind of ontologies discarding the fuzzy elements. *FuzzyOWL2* also offers a general Java parser as a base for building specific parsers for translating from *FuzzyOWL2* syntax to the syntax of any fuzzy DL reasoner.

B. Reasoning Over Fuzzy Ontologies

Although there exist several software tools for computationally managing fuzzy concepts (e.g., jFuzzyLogic or FuzzyJToolkit for general-purpose programming; FuzzyClips or FuzzyJess for handling fuzzy rule-based systems), only a few of them supporting DL-based reasoning can be found in the literature: FiRE[1], GURDL [9], GERDS[10], Yadlr[2] and *fuzzyDL* [11]. Most of them are outdated or not publicly available.

It is also worth mentioning those works developing *reduction procedures* translating fuzzy ontologies into its standard DLs equivalent, being then able to apply standard DLs reasoners for reasoning over them (notice that this kind of approaches have to consider a finite number of different membership degrees). DeLorean[3] combines a reduction procedure with a crisp DL reasoner (e.g., Pellet) to reason over the resulting ontology.

[1]http://www.image.ece.ntua.gr/~nsimou/FiRE/

[2]http://yadlr.sourceforge.net/

[3]http://webdiis.unizar.es/~fbobillo/delorean

C. Selection Of External Fuzzy Semantic Tools

As first step for extending DL-Learner we had to choose (i) a way for defining and managing fuzzy DL ontologies and (ii) a fuzzy DL reasoner to reason over them.

Regarding the formalism to represent fuzzy ontologies, each of the approaches previously presented has its advantages and disadvantages. Populating specific ontologies representing fuzzy concepts may maintain the semantics even when modelling the new fuzzy concepts, but the resulting ontologies are less human-understandable. Extending the building blocks of the semantic languages preserves semantics but none the available extensions is expected to be proposed as W3C standard in the near future [6]. On the other hand, although adding fuzzy concepts using the standard tools of the ontology languages (comments, annotations, etc.) provides a semantic-less way of representing fuzzy information, these approaches are highly compatibility with standard ontological tools (editors, programmatic environments, etc.), since these can simply ignore the fuzzy information encoded. Since reasoners often use their own languages, we also require an approach compatible with the fuzzy reasoners syntax or, at least, offering a straightforward way to be translated to it. *FuzzyOWL2Ontology* and *FuzzyOWL2* are the only mechanisms we are aware of, which satisfy those criteria. We finally selected *FuzzyOWL2* as it is a currently active project.

With regard to the selection of a fuzzy reasoner, the first parameter to consider has been its availability. Only 3 out of 6 fuzzy reasoners presented above are publicly available: FiRE, Yadlr and *fuzzyDL*. The fact that Yadlr does not offer a Java API focused attention in the other two reasoners, as a Java-based interface would simplify reasoner integration. While FiRE allows the use of some DLs constructs which *fuzzyDL* does not support (e.g., cardinality restrictions), *fuzzyDL* covers other interesting features (e.g., fuzzy concepts modifiers, fuzzy data types) [11]. Finally, taking into account that *fuzzyDL* seems to be the only actively developed reasoner (last bug was fixed in March 2011), it was finally selected to be integrated in DL-Learner.

III. Extending DL-Learner to support fuzzy concept management

To be flexible and easily extensible, DL-Learner uses a component-based model with four different types of components: knowledge source, reasoning service, learning problem, and learning algorithm. Although some parts of other already existing components were reused, four new components had to be designed and developed in order to extend DL-Learner with fuzzy logic capabilities. Figure 1 depicts the general components diagram of the resulting DL-Learner fuzzy extension. Particularities of each new component are detailed next.

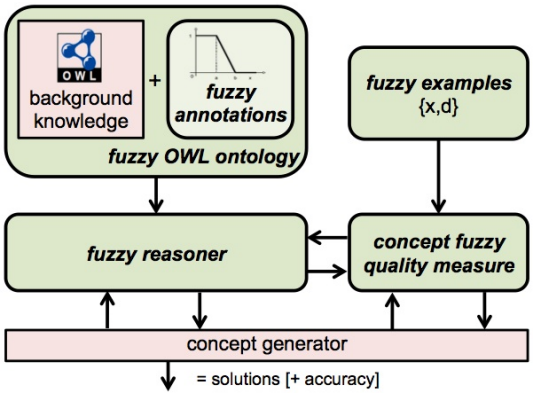

Figure 1. DL-Learner fuzzy extension general components diagram.

A. Fuzzy Knowledge Source Component

The DL-Learner component managing crisp OWL ontologies as input for the reasoning processes had to be updated in order to be able to manage fuzzy ontologies. This task was easily accomplished as it was decided to encode the fuzzy ontologies using *FuzzyOWL2*. It is worth remembering that *FuzzyOWL2* uses annotation properties to encode fuzzy ontologies, maintaining the original OWL 2 format and being then fully compatible with current OWL 2 management tools (review Section II-A for more details). DL-Learner already has support for OWL 2 ontologies (it uses OWLAPI).

This component had to be extended in order to host the processes for parsing *FuzzyOWL2* ontologies into the particular ontology format used by *fuzzyDL* reasoner. This task was achieved by adapting an already developed parser[4].

B. Fuzzy Reasoning Service Component

DL-Learner operation is supported by configuring external general-purpose ontology reasoners (e.g., Pellet, HermiT, etc.). The services these reasoners offer are mainly used in DL-Learner (i) to generate candidate concepts (class expressions) that may solve the ILP problem (see Section III-D) and (ii) to calculate the quality of each of those candidate concepts for a specific set of examples (see Section III-C). For this new DL-Learner extension, a new reasoning service component had to be developed in order to be able to reason over fuzzy ontologies. The *fuzzyDL* reasoner was encapsulated into an OWLAPI interface, in which some reasoning methods are performed by a crisp OWLAPI compatible reasoner and fuzziness-specific methods are performed by the fuzzy reasoner.

Instance checks, i.e., testing whether an individual is instance of a class, is, as in many ILP algorithms, a common reasoner call in most DL-Learner algorithms. This is a 'true or false' operation when applied to crisp ontologies but a degree value one (from 0 to 1) when dealing with fuzzy ontologies. In this new fuzzy component, instance checks are performed using *fuzzyDL*'s *minInstance* query. It determines the minimal degree to which individual a is an instance of concept C (being $\mathcal{K}$ the background knowledge base) [11]:

$$\text{minInstance}\left(\mathcal{K}, a, C\right) = \inf\left\{m \,|\, \mathcal{K} \models \langle a : C, m\rangle\right\} \quad (1)$$

C. Fuzzy Learning Problem Component

This new component was designed so DL-Learner could manage fuzzy examples. Within this new component, each example e is now composed of an URI a and a *truth degree* value d representing the desired class membership regarding the class expressions obtained as solution of the ILP problem ($e = (a, d)$). This new learning problem was developed generalizing a previous one only using positive and negative examples. Crisp positive and negative examples can now be represented as $e = (a, 1)$ and $e = (a, 0)$, respectively.

From the extensive number of performance measures for classification in machine learning, DL-Learner has been extended with the fuzzy version of *predictive accuracy* and *F-measure*.

• *Fuzzy predictive accuracy* (fpa) is obtained using the fuzzy reasoning service component (*minInstance*) and represents the membership degree of the complete set of examples $E = \{e_1, e_2, \ldots, e_N\}$ regarding a particular concept C. Considering N as the total number of examples, the *fuzzy predictive accuracy* is calculated as:

$$fpa'\left(\mathcal{K}, C, e\right) = 1 - |d - \text{minInstance}(\mathcal{K}, a, C)| \quad (2)$$

$$fpa\left(\mathcal{K}, C, E\right) = \frac{1}{N}\sum_{\forall i} fpa'\left(\mathcal{K}, C, e_i\right) \quad (3)$$

Given a particular concept C and a background knowledge base $\mathcal{K}$, fpa' (2) is the distance between the desired and actual truth degree for a particular individual a; fpa (3) is the average of this value for all examples E.

• *F-measure* is derived from *precision* and *recall*. In crisp classification, the terms true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn) are used to compare the given classification of a given example, being $Precision = tp/(tp + fp)$ and $Recall = tp/(tp + fn)$. According to previous equations, in fuzzy classification an example is no longer positive or negative but a degree ranging between 0 (totally negative example) and 1 (totally positive example). This range is determined by the example truth degree d. Similarly, fuzzy classification for a particular example e regarding a particular concept C is no longer totally true or false but, again, a degree between 0 (C does not cover e at all) and 1 (C totally covers e) that can be determined by fpa' (2). Then, based on previous works on fuzzy information retrieval, e.g., [12], *precision* and *recall* metrics can be expressed for the fuzzy ILP problem as:

$$fRecall(\mathcal{K}, C, E) = \frac{\sum_{\forall i} d_i \cdot fpa'\left(\mathcal{K}, C, e_i\right)}{\sum_{\forall i} d_i} \quad (4)$$

[4]http://gaia.isti.cnr.it/~straccia/software/FuzzyOWL

$$fPrecision(\mathcal{K}, C, E) = \\ = \frac{\sum_{\forall i} d_i \cdot fpa'(\mathcal{K}, C, e_i)}{\sum_{\forall i} d_i \cdot fpa'(\mathcal{K}, C, e_i) + \sum_{\forall i} (1-d_i) \cdot (1 - fpa'(\mathcal{K}, C, e_i))} \quad (5)$$

Then, *fuzzy F-measure* can be expressed as the harmonic mean of *fuzzy precision* and *fuzzy recall* as:

$$fF{-}measure = \frac{2 \cdot fPrecision \cdot fRecall}{fPrecision + fRecall} \quad (6)$$

Furthermore, this learning problem can be configured with a *noise factor* n (*fuzzy predictive accuracy* becomes $fpa\,(\mathcal{K}, C, E, n)$), i.e., a tolerance degree when discarding concepts not totally covering the set of examples. The *noise factor* can range between 0 and 1. A value of $n = 0$ (i.e., no noise) would discard those concepts not covering completely (with accuracy 100%) the set of examples. On the other hand, a value of $n = 1$ (i.e., maximum noise) would accept concepts even not covering at all any of the examples defined.

Figure 2 offers an overview of the fuzzy learning problem implementation, showing how *fuzzy predictive accuracy* and *noise factor* concepts are managed.

Input: $\mathcal{K}, C, E = \{e_1, e_2, \ldots, e_N\}, n$
1: $s \leftarrow \sum_{\forall i} d_i$; $t \leftarrow s$; $z \leftarrow 0$
2: **for all** $E = \{e_1, e_2, \ldots, e_N\}$ **do**
3: $m \leftarrow 1 - |d_i - \text{minInstance}\,(\mathcal{K}, a_i, C)\,|$
4: $s \leftarrow s - d_i$
5: **if** $[z + (m \cdot d_i) + s] < [(1 - n) \cdot t]$ **then**
6: **return** C too weak
7: **end if**
8: $z \leftarrow z + (m \cdot d_i)$
9: $g \leftarrow g + m$
10: **end for**
11: **return** g/N
Output: $fpa\,(\mathcal{K}, C, E, n)$ or too weak

Figure 2. DL-Learner fuzzy learning problem main algorithm.

D. Fuzzy Learning Algorithm Component

In DL-Learner, learning algorithm components are in charge of generating the candidate concepts (class expressions) that may solve the ILP problem. They may use the reasoning component services in order to obtain these candidate concepts. Several learning algorithms are implemented in DL-Learner. However, most algorithms use refinement operators, which have been analysed in [13]. Based on this analysis, several learning algorithms for $\mathcal{ALC}$ have been developed in [14] and later extended to more expressive Description Logics and adapted to the ontology engineering use case in [15]. The latter algorithm, called CELOE (Class Expression Learning for Ontology Engineering), was mainly used in our experiments. However, since the new fuzzy reasoning service component (see Section III-B) has been designed fulfilling (and extending) the OWLAPI interface, most other learning algorithms can in principle be used as well with only some changes to the heuristics of those algorithms required in order to adapt them to the fuzzy domain.

IV. Semantic fuzzy ILP test case development

In the machine learning literature there exist several well-known examples defining simple ILP problems to be solved. Michalski's train problem [16] is one of the most famous. It was invented around 20 years ago and proposes finding a common pattern in a group of trains (and their carriages). The solution must involve several concepts, e.g., size, number, position, contents and type of carriages. This is a standard problem which has been used to test and demonstrate many machine learning ILP implementations. However, there is a lack of such test cases for fuzzy ILP reasoners (like the one in [3]).

A. Semantic Fuzzy Trains Problem Design

A fuzzy and OWL-based version of Michalski's train problem has been designed in order to be used for testing general-purpose fuzzy ILP reasoners.

The fuzzy trains ontology design starts from defining a crisp ontology to be later 'fuzzified' (see Figure 3). In this ontology, each locomotive (*Train*) can have several carriages (*Car*) attached (*hasCar*). The order of the locomotive and the carriages is also stated (*isInFrontOf*). Each carriage has a certain length (*hasLength*) and may have some load (*hasLoad*). This load can have several shapes. Currently, triangular (*Triangle*) and square (*Square*) shapes are considered. Finally, there may exist three types of carriages depending on its length (*ShortCar*, *MediumCar*, *LongCar*). This crisp ontology is extended with fuzzy concepts by

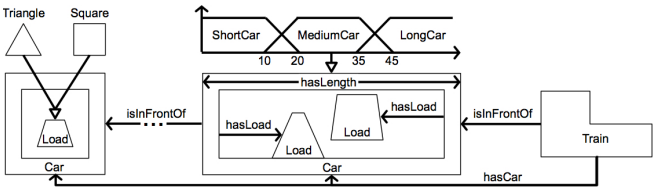

Figure 3. *fuzzy*Trains ontology design.

adding *FuzzyOWL2* annotations. Up to now, the semantic fuzzy trains ontology considers three kinds of 'fuzzification':

1) **Fuzzy classes definition:** *ShortCar*, *MediumCar* and *LongCar* classes become fuzzy classes when they are defined using the fuzzy datatypes *fuzzyShortCar*, *fuzzyMediumCar* and *fuzzyLongCar* respectively. These fuzzy datatypes are defined using the fuzzy number functions trapezoidal, triangular, left-shoulder and right-shoulder as specified in [17]. As an example, Figure 4 depicts the *ShortCar* fuzzy class definition.

2) **Fuzzy concept assertion:** Any individual a can belong to the *Train*, *Car*, *Triangle* or *Square* classes C with a certain truth degree $d : [0, 1]$. The closer this degree is to 1,

the more 'similar' to a perfect locomotive, carriage, triangle or square, respectively.

$$fuzzyShortCar(x) = leftShoulder(10, 15)$$

```
<rdfs:Datatype rdf:about="fuzzyShortCar">
  <fuzzyOwl2 fuzzyType="datatype">
    <Datatype type="leftshoulder" a="10" b="15" />
  </fuzzyOwl2>
</rdfs:Datatype>
```

$$ShortCar \equiv Car \sqcap (\exists hasCarLength.fuzzyShortCar)$$

```
<owl:Class rdf:about="ShortCar">
  <rdfs:subClassOf rdf:resource="Car"/>
  <owl:equivalentClass>
     <owl:Restriction>
        <owl:onProperty rdf:resource="hasCarLength"/>
        <owl:someValuesFrom rdf:resource="fuzzyShortCar"/>
     </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Figure 4. Example of fuzzy class definition (*ShortCar*).

3) **Fuzzy role assertion:** Relations between individuals can also have a truth degree value. This is the case for the load in carriages. The degree of *hasLoad* relation between a carriage and its load can range from 0 (falling off the car) to 1 (firmly fixed to the floor of the carriage).

An specific instantiation of the described design has been developed to verify the fuzzy learning capabilities of this new fuzzy extension of DL-Learner (see Figure 5). We have decided to make this contribution publicly available[5] to be used as a common test case for testing this kind of reasoners.

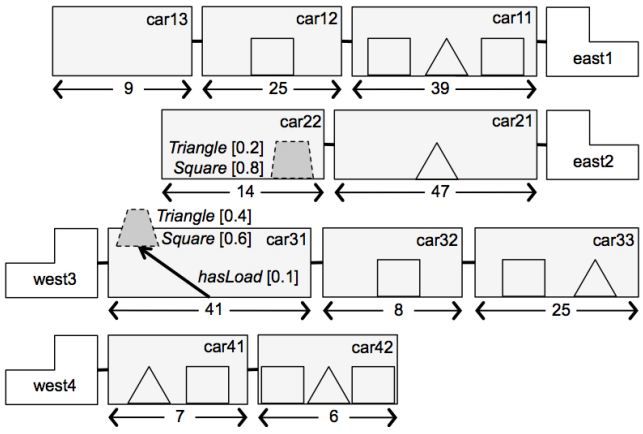

Figure 5. *fuzzy*Trains ontology development.

V. Experiments and Results

A. Comparing DL-Learner Fuzzy and Crisp Behavior

As previously mentioned, classical logic is covered by fuzzy logic as a special case where membership functions only take crisp values. So, in order to partially validate this new fuzzy version of DL-Learner, we first used it to solve a crisp ILP problem. A crisp version of the *fuzzyTrains* ontology was developed[5] to compare the solutions obtained by (i) DL-Learner configured to solve a crisp ILP problem and (ii) DL-Learner configured with the fuzzy components previously presented. Indeed, exactly the same solutions were obtained by both configurations. The only difference between them is the execution time (to be discussed at the end of the Section).

[5]http://dl-learner.org/Projects/DLLearner/fuzzyTrains

B. Fuzzy Trains Test Case Output and Performance

Table I shows some solutions to the particular ILP problems obtained defining east trains as positive examples and west trains as negatives (*east* column) or vice-versa (*west* column).

In the *east* case, after visiting 706 nodes during reduction, the following expression describing eastbound trains is discovered (in Manchester OWL syntax): '*isInFrontOf only (LongCar and hasLoad some Triangle)*'. The accuracy is 85% and not 100% as (i) *car11* is just considered to be *LongCar* at 40% according to the fuzzy data types definition and (ii) although *car31* is also *inFrontOf* a *Train*, its load is almost falling off the carriage ('*car31 hasLoad load31a* [0.1]') and is not a perfect triangle (only 40%).

Similarly, in the *west* case, after visiting 3623 nodes, the following expression is obtained: '*hasCar some (ShortCar and hasLoad some Rectangle)*'. In this case, accuracy is 90% and not 100% as *car22* is just considered to be *ShortCar* at 60% according to fuzzy data types definition and its load, *load22a*, is not a perfect rectangle (only 80%).

Table I
DL-Learner fuzzy ILP problem solutions

	east	**west**
time	1,369s (10,000 concepts)	1,325s (10,000 concepts)
accuracy	85% (fMeasure 82.35%)	90% (fMeasure 90.91%)
solution	$\forall isInFrontOf.(LongCar \sqcap \exists hasLoad.Triangle)$	$\exists hasCar.(ShortCar \sqcap \exists hasLoad.Rectangle)$

It has to be noted that there exist other solutions also valid for this particular ILP problem; Table I solutions are presented here as they involve several of the fuzzy axioms defined in the test case. Future updates should develop more comprehensive examples (e.g., just crisp examples were defined in the presented results, etc.).

Although the new fuzzy capabilities of DL-Learner have been validated from a functional perspective, it is important to remark that fuzzy reasoner execution times improvement is still an open issue [18] (*fuzzyDL* requires around 27.88 ms. to answer a *minInstance* query while, e.g., Pellet only employs around 0.997 ms. for solving an entailment). In this sense, it has to be pointed out that DL-Learner's fuzzy reasoning service component has been designed fulfilling (and extending) the OWLAPI interface, being ready to host any other kind of fuzzy reasoner offering this kind of interface with only minimal adaption effort required.

VI. Conclusions and future work

Despite the considerable research carried out towards extending standard ontology languages to support vague concept representation, current approaches have not been sufficiently applied to solve real world problems. The work presented in this paper represents a first step towards applying fuzzy OWL ontologies representation and reasoning to solve Inductive Logic Programming (ILP) problems. After reviewing the state of the art on these technologies, some of the most up-to-date fuzzy ontology tools were used to extend an already developed ontology-based ILP framework in order to be able to manage vague concepts. The resulting tool has been validated using a fuzzy ontology test case, also presented in this paper.

This work should be considered just as a first step for further development: more test cases should be implemented aiming at exploiting every new fuzzy feature of DL-Learner and applying DL-Learner to solve real world problems (e.g., we are thinking about developing a *shop assistant* based on fuzzy information regarding items purchased by a user). We are also collaborating with the developers of *fuzzyDL* as the current version is not fast enough to be applied to solve many real world problems; during the DL-Learner project, we already achieved significant improvements in this area. We also miss some functionality, e.g., cardinality restriction support [19].

What should be stressed, however, is that this fuzzy extension of DL-Learner has been developed following a component-based approach, being actually quite easy to update and extend.

Acknowledgments

This work has been supported by the Spanish Ministry for Science and Innovation (grant TIN2008-06742-C02-01). Josué Iglesias also acknowledges this Ministry for his grant and the support received by every member of AKSW group.

References

[1] K. J. Laskey, K. B. Laskey, P. C. G. Costa, M. M. Kokar, T. Martin, and T. Lukasiewicz, "W3C Incubator Group on Uncertainty Reasoning for the World Wide Web. Final Report," 2008.

[2] J. Lehmann, "DL-Learner: learning concepts in description logics," *Journal of Machine Learning Research (JMLR)*, vol. 10, pp. 2639–2642, 2009.

[3] S. Konstantopoulos and A. Charalambidis, "Formulating description logic learning as an inductive logic programming task," in *Proc. of FUZZ-IEEE, 2010 IEEE World Congress on Comp. Int., July 18–23, Barcelona*. IEEE, Jul. 2010.

[4] U. Straccia, "A Fuzzy Description Logic for the Semantic Web," in *Fuzzy logic and the Semantic Web, Capturing Int., Chapter 4*. Elsevier, 2005, pp. 167–181.

[5] I. J. Blanco, M. A. Vila, and C. Martinez-Cruz, "The use of ontologies for representing database schemas of fuzzy information," *Int. J. Intell. Syst.*, vol. 23, pp. 419–445, 2008.

[6] F. Bobillo and U. Straccia, "An OWL Ontology for Fuzzy OWL 2," in *Proc. of 18th Int. Symp. on Foundations of Int. Systems*, ser. ISMIS '09, 2009, pp. 151–160.

[7] G. Stoilos, G. Stamou, and J. Z. Pan, "Fuzzy extensions of OWL: Logical properties and reduction to fuzzy description logics," *Int. J. Approx. Reas.*, vol. 51, pp. 656–679, July 2010.

[8] F. Bobillo and U. Straccia, "Fuzzy ontology representation using OWL 2," *International Journal of Approximate Reasoning*, vol. 52, no. 7, pp. 1073 – 1094, 2011.

[9] H.-I. P. Volker Haarslev and N. Shiri, "Optimizing Tableau Reasoning in ALC Extended with Uncertainty," in *In Proc. of Description Logics*, 2007.

[10] H. Habiballa, "Resolution Strategies for Fuzzy Description Logic," in *European Society for Fuzzy Logic and Technology*, 2007, pp. 27–36.

[11] F. Bobillo and U. Straccia, "fuzzyDL: An expressive fuzzy description logic reasoner," in *Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Comp. Int.). IEEE Int. Conf. on*, june 2008, pp. 923 –930.

[12] V. Cross, "Fuzzy information retrieval," *Journal of Intelligent Information Systems*, vol. 3, pp. 29–56, 1994.

[13] J. Lehmann and P. Hitzler, "Foundations of refinement operators for description logics," in *Proc. of 17th Int. Conf. on Inductive Logic Programming*, vol. 4894. Springer, 2007, pp. 161–174.

[14] ——, "A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic," in *Proc. of 17th Int. Conf. on Inductive Logic Programming*, vol. 4894. Springer, 2007.

[15] J. Lehmann, S. Auer, L. Bühmann, and S. Tramp, "Class expression learning for ontology engineering," *Journal of Web Semantics*, vol. 9, pp. 71 – 81, 2011.

[16] J. Larson and R. S. Michalski, "Inductive inference of VL decision rules," *SIGART Bull.*, pp. 38–44, June 1977.

[17] D. Dubois and H. Prade, "Operations on fuzzy numbers," *Int. Journal of Systems Science*, vol. 9, pp. 613–626, 1978.

[18] N. Simou, T. P. Mailis, G. Stoilos, and G. B. Stamou, "Optimization techniques for fuzzy description logics." in *Description Logics'10*, 2010, pp. –1–1.

[19] H. Wang, Z. M. Ma, and J. Yin, "FRESG: A Kind of Fuzzy Description Logic Reasoner," in *Proc. of 20th Int. Conf. on Database and Expert Systems Applications*, ser. DEXA '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 443–450.