Towards Learning Fuzzy DL Inclusion Axioms

Francesca A. Lisi[1] and Umberto Straccia[2]

[1] Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro", Italy
lisi@di.uniba.it
[2] ISTI - CNR, Pisa, Italy
straccia@isti.cnr.it

Abstract. Fuzzy Description Logics (DLs) are logics that allow to deal with vague structured knowledge. Although a relatively important amount of work has been carried out in the last years concerning the use of fuzzy DLs as ontology languages, the problem of automatically managing fuzzy ontologies has received no attention so far. We report here our preliminary investigation on this issue by describing a method for inducing inclusion axioms in a fuzzy DL-Lite like DL.

1 Introduction

Description Logics (DLs) [1] play a key role in the design of *ontologies*. An ontology consists of a hierarchical description of important concepts in a particular domain, along with the description of the properties (of the instances) of each concept. In this context, DLs are important as they are essentially the theoretical counterpart of the *Web Ontology Language OWL 2*[1], the current standard language to represent ontologies, and its profiles.[2] E.g., DL-Lite [2] is the DL behind the *OWL 2 QL* profile.

It is well-known that "classical" ontology languages are not appropriate to deal with *vague knowledge*, which is inherent to several real world domains [13]. So far, several fuzzy extensions of DLs can be found in the literature (see the survey in [8]). Although a relatively important amount of work has been carried out in the last years concerning the use of fuzzy DLs as ontology languages, the problem of automatically managing fuzzy ontologies has received no attention so far. In this work, we report our preliminary investigation on this issue by describing a method for inducing inclusion axioms in a fuzzy DL-Lite like DL. The method follows the machine learning approach known as *Inductive Logic Programming* (ILP) by adapting known results in ILP concerning crisp rules to the novel case of fuzzy DL inclusion axioms.

The paper is structured as follows. Section 2 is devoted to preliminaries on ILP and fuzzy DLs. Section 3 describes our preliminary contribution to the problem in hand, also by means of an illustrative example. Section 4 concludes the paper with final remarks and comparison with related work.

[1] http://www.w3.org/TR/2009/REC-owl2-overview-20091027/
[2] http://www.w3.org/TR/owl2-profiles/

A. Petrosino et al. (Eds.): WILF 2011, LNAI 6857, pp. 58–66, 2011.

2 Preliminaries

2.1 Learning Rules with ILP

Inductive Logic Programming (ILP) was born at the intersection between Concept Learning and Logic Programming [9].

From Logic Programming it has borrowed the Knowledge Representation (KR) framework, i.e. the possibility of expressing facts and rules in the form of Horn clauses. In the following, rules are denoted by $A(\mathbf{x}) \rightarrow H(\mathbf{x})$ where $\mathbf{x}$ is the vector of the n variables that appear in the rule, $A(\mathbf{x}) = A_0(\mathbf{x}) \wedge \ldots \wedge A_q(\mathbf{x})$ represents the antecedent (called the *body*) of the rule, and $H(\mathbf{x})$ is the consequent (called *head*) of the rule. The predicate H pertains to the concept to be learnt (called *target*). Given an attribute domain $\mathcal{D}$ and a vector $\mathbf{t} \in \mathcal{D}_n$ of n values of the domain, we denote the ground substitution of the variable $\mathbf{x}$ with $\mathbf{t}$ by $H(\mathbf{t}) = \sigma[\mathbf{x}/\mathbf{t}]H(\mathbf{x})$. Then $H(\mathbf{t})$ is true or false in a given interpretation.

From Concept Learning it has inherited the inferential mechanisms for induction, the most prominent of which is *generalisation*. A distinguishing feature of ILP is the use of prior domain knowledge during the induction process. The classical ILP problem is described by means of two logic programs: (i) the *background theory* $\mathcal{B}$ which is a set of ground facts and rules; (ii) the *training set* $\mathcal{E}$ which is a set of ground facts, called *examples*, pertaining to the target predicate. It is often split in $\mathcal{E}^+$ and $\mathcal{E}^-$, which correspond respectively to positive and negative examples. If only $\mathcal{E}^+$ is given, $\mathcal{E}^-$ can be deduced by using the Closed World Assumption (CWA)[3]. The task of induction is to find, given $\mathcal{B}$ and $\mathcal{E}$, a set of rules $\mathcal{H}$ such that: (i) $\forall e \in \mathcal{E}^+, \mathcal{B} \cup \mathcal{H} \models e$ and (ii) $\forall e \in \mathcal{E}^-, \mathcal{B} \cup \mathcal{H} \not\models e$. Two further restrictions hold naturally. One is that $\mathcal{B} \not\models \mathcal{E}^+$ since, in such a case, $\mathcal{H}$ would not be necessary to explain $\mathcal{E}^+$. The other is $\mathcal{B} \cup \mathcal{H} \not\models \bot$, which means that $\mathcal{B} \cup \mathcal{H}$ is a consistent theory. Usually, rule induction fits with the idea of providing a compression of the information contained in $\mathcal{E}$. A rule r *covers* an example $e \in \mathcal{E}$ wrt $\mathcal{B}$ iff $\mathcal{B} \cup \{r\} \models e$.

A popular ILP algorithm for learning sets of rules from relational data is FOIL [10]. It performs a greedy search in order to maximise a gain function. The rules are induced until all positive examples are covered or no more rules are found that overcome the threshold. When a rule is induced, the positive examples covered by the rule are removed from $\mathcal{E}$. The function FOIL-Learn-One-Rule reported in Figure 1 starts with the most general clause ($\top \rightarrow H(\mathbf{x})$) and specialises it step by step by adding literals in the antecedent. The rule r is accepted when its confidence degree $cf(r)$ (see later on) overcomes a fixed threshold θ and it does not cover any negative example. The Gain function is computed by the formula:

$$\textsc{Gain}(cf(r_1), cf(r_2)) = p * (log_2(cf(r_1)) - log_2(cf(r_2))) \ , \tag{1}$$

where p is the number of distinct examples covered by the rule r_1, *i.e.* $p = |\{e \mid \mathcal{B} \cup \{r_1\} \models e\}|$. Thus, the gain is positive iff the new rule is more informative

[3] Anything that is not stated as true is assumed to be false.

function FOIL-LEARN-ONE-RULE(h, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{B}$): r
begin
1. $A(\mathbf{x}) \leftarrow \top$;
2. $r \leftarrow \{A(\mathbf{x}) \rightarrow H(\mathbf{x})\}$;
3. $\mathcal{E}_r^- \leftarrow \mathcal{E}^-$;
4. **while** $cf(r) < \theta$ **and** $\mathcal{E}_r^- \neq \emptyset$ **do**
5. $\quad A_{best}(\mathbf{x}) \leftarrow A(\mathbf{x})$;
6. $\quad maxgain \leftarrow 0$;
7. $\quad$ **foreach** $l \in \mathcal{B}$ **do**
8. $\quad\quad gain \leftarrow \text{GAIN}(cf(A(\mathbf{x}) \wedge l(\mathbf{x}) \rightarrow H(\mathbf{x})), cf(A(\mathbf{x}) \rightarrow H(\mathbf{x}))$;
9. $\quad\quad$ **if** $gain \geqslant maxgain$ **then**
10. $\quad\quad\quad maxgain \leftarrow gain$;
11. $\quad\quad\quad A_{best}(\mathbf{x}) \leftarrow A(\mathbf{x}) \wedge l(\mathbf{x})$;
12. $\quad\quad$ **endif**
13. $\quad$ **endforeach**
14. $\quad r \leftarrow \{A_{best}(\mathbf{x}) \rightarrow H(\mathbf{x})\}$;
15. $\quad \mathcal{E}_r^- \leftarrow \mathcal{E}_r^- \setminus \{e \in \mathcal{E}^- | \mathcal{B} \cup r \models e\}$;
16. **endwhile**
17. **return** r
end

Fig. 1. Algorithm for learning one rule in FOIL

in the sense of Shannon's information theory (i.e. iff the confidence degree increases). If there are some literals to add which increase the confidence degree, the gain tends to favor the literals that offer the best compromise between the confidence degree and the number of examples covered.

Given a Horn clause $A(\mathbf{x}) \rightarrow H(\mathbf{x})$, its confidence degree is given by: $cf(A(\mathbf{x}) \rightarrow H(\mathbf{x})) = P(A(\mathbf{x}) \wedge H(\mathbf{x}))/P(A(\mathbf{x}))$. Confidence degrees are computed in the spirit of domain probabilities. Input data in ILP problems are supposed to describe one interpretation under CWA. We call $\mathcal{I}_{ILP}$ this interpretation. So, given a fact f, we define

$$\mathcal{I}_{ILP} \models f \text{ iff } \mathcal{B} \cup \mathcal{E} \models f \ . \tag{2}$$

The domain $\mathcal{D}$ is the Herbrand domain described by $\mathcal{B}$ and $\mathcal{E}$. We take P as a uniform probability on $\mathcal{D}$. So the confidence degree in a clause $A(\mathbf{x}) \rightarrow H(\mathbf{x})$ is:

$$cf(A(\mathbf{x}) \rightarrow H(\mathbf{x})) = \frac{|\mathbf{t} \in \mathcal{D}^n \mid \mathcal{I}_{ILP} \models (A(\mathbf{t}) \wedge H(\mathbf{t})) \text{ and } H(\mathbf{t}) \in \mathcal{E}^+|}{|\mathbf{t} \in \mathcal{D}^n \mid \mathcal{I}_{ILP} \models A(\mathbf{t}) \text{ and } H(\mathbf{t}) \in \mathcal{E}|} \tag{3}$$

where $|\cdot|$ denotes set cardinality. Testing all possible $\mathbf{t} \in \mathcal{D}^n$ is not tractable in practice. However, we can equivalently restrict the computation to the substitutions that map variables to constants in their specific domains. In fact, this computation is equivalent to a database query and thus, we can also use some optimization strategy such as indexing or query ordering. This makes the computation tractable although it remains costly.

2.2 A DL-Lite Like Description Logic and Its Fuzzy Extensions

The logic we adopt is based on a fuzzy extension of the DL-Lite [2] DL without negation, and is supported by the SoftFacts system [15][4]. DL-Lite supports unary predicates (called *concepts*) and binary predicates (called *roles*).

[4] See, `http://www.straccia.info/software/SoftFacts/SoftFacts.html`

A *knowledge base* $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{A} \rangle$ consists of a *facts component* $\mathcal{F}$, an *ontology component* $\mathcal{O}$ and an *abstraction component* $\mathcal{A}$, which are defined as follows (for a detailed account of the semantics, see [14]).

Facts Component. $\mathcal{F}$ is a finite set of expressions of the form

$$R(c_1, \ldots, c_n)[s] \ , \tag{4}$$

where R is an n-ary relation, every c_i is a constant, and s is a degree of truth (or *score*) in $[0, 1]$ indicating to which extent the tuple $\langle c_1, \ldots, c_n \rangle$ is an instance of relation R.[5] Facts are stored in a relational database. We may omit the score component and in such case the value 1 is assumed.

Ontology Component. The ontology component is used to define the relevant abstract concepts and relations of the application domain by means of inclusion axioms. Specifically, $\mathcal{O}$ is a finite set of *inclusion axioms* having the form

$$Rl_1 \sqcap \ldots \sqcap Rl_m \sqsubseteq Rr \ , \tag{5}$$

where $m \geqslant 1$, all Rl_i and Rr have the same arity and each Rl_i is a so-called *left hand relation* and Rr is a *right hand relation*. We assume that relations occurring in $\mathcal{F}$ do not occur in inclusion axioms (so, we do not allow that database relation names occur in $\mathcal{O}$). The intuitive semantics is that if a tuple $\mathbf{c}$ is instance of each relation Rl_i to degree s_i then $\mathbf{c}$ is instance of Rr to degree $\min(s_1, \ldots, s_m)$.

The exact syntax of the relations appearing on the left-hand and right hand side of inclusion axioms is specified below:

$$\begin{array}{l} Rl \longrightarrow A \mid R[i_1, i_2] \\ Rr \longrightarrow A \mid R[i_1, i_2] \mid \exists R.A \end{array} \tag{6}$$

where A is an *atomic concept* and R is a role with $1 \leqslant i_1, i_2 \leqslant 2$. Here $R[i_1, i_2]$ is the projection of the relation R on the columns i_1, i_2 (the order of the indexes matters). Hence, $R[i_1, i_2]$ has arity 2. Additionally, $\exists R.A$ is a so-called qualified existential quantification on roles which corresponds to the FOL formula $\exists y.R(x, y) \wedge A(y)$ where $\wedge$ is interpreted as the min t-norm.

Abstraction Component. $\mathcal{A}$ is a set of statements that allow to connect atomic concepts and relations to physical relational tables. Essentially, it is used as a wrapper to the underlying database and, thus, prevents that relational table names occur in the ontology. Formally, an *abstraction statement* is of the form

$$R \mapsto (c_1, \ldots, c_n)[c_{score}].sql \ , \tag{7}$$

where sql is a SQL statement returning n-ary tuples $\langle c_1, \ldots, c_n \rangle$ ($n \leqslant 2$) with score determined by the c_{score} column. The tuples have to be ranked in decreasing order of score and, as for the fact component, we assume that there cannot be two records $\langle \mathbf{c}, s_1 \rangle$ and $\langle \mathbf{c}, s_2 \rangle$ in the result set of sql with $s_1 \neq s_2$ (if there are, then we remove the one with the lower score). The score c_{score} may be omitted

[5] The score s may have been computed by some external tool, such as a classifier, etc.

and in that case the score 1 is assumed for the tuples. We assume that R occurs in $\mathcal{O}$, while all of the relational tables occurring in the SQL statement occur in $\mathcal{F}$. Finally, we assume that there is at most one abstraction statement for each abstract relational symbol R.

Query Language. A *query* consists of a "conjunctive query", with a scoring function to rank the answers. A *ranking query* [7] is of the form

$$\begin{aligned} q(\mathbf{x})[s] \leftarrow \exists \mathbf{y}\ & R_1(\mathbf{z}_1)[s_1], \ldots, R_l(\mathbf{z}_l)[s_l], \\ & \mathsf{OrderBy}(s = f(s_1, \ldots, s_l, p_1(\mathbf{z}'_1), \ldots, p_h(\mathbf{z}'_h)) \end{aligned} \tag{8}$$

where

1. q is an n-ary relation, every R_i is a binary relation. $R_i(\mathbf{z_i})$ may also be of the form $(z \leqslant v), (z < v), (z \geqslant v), (z > v), (z = v), (z \neq v)$, where z is a variable, v is a value of the appropriate concrete domain;
2. $\mathbf{x}$ are the *distinguished variables*.
3. $\mathbf{y}$ are existentially quantified variables called the *non-distinguished variables*. We omit to write $\exists \mathbf{y}$ when $\mathbf{y}$ is clear from the context;
4. $\mathbf{z_i}, \mathbf{z'_j}$ are tuples of constants or variables in $\mathbf{x}$ or $\mathbf{y}$;
5. $s, s_1, \ldots, s_l$ are distinct variables and different from those in $\mathbf{x}$ and $\mathbf{y}$;
6. p_j is an n_j-ary *fuzzy predicate* assigning to each n_j-ary tuple $\mathbf{c}_j$ a *score* $p_j(\mathbf{c}_j) \in [0, 1]$. We require that an n-ary fuzzy predicate p is *safe*, that is, there is not an m-ary fuzzy predicate p' such that $m < n$ and $p = p'$. Informally, all parameters are needed in the definition of p.
7. f is a *scoring* function $f\colon ([0, 1])^{l+h} \rightarrow [0, 1]$, which combines the scores of the l relations $R_i(\mathbf{c}'_i)$ and the n fuzzy predicates $p_j(\mathbf{c}''_j)$ into an overall *score* to be assigned to the rule head $R(\mathbf{c})$. We assume that f is *monotone*, that is, for each $\mathbf{v}, \mathbf{v}' \in ([0, 1])^{l+h}$ such that $\mathbf{v} \leqslant \mathbf{v}'$, it holds $f(\mathbf{v}) \leqslant f(\mathbf{v}')$, where $(v_1, \ldots, v_{l+h}) \leqslant (v'_1, \ldots, v'_{l+h})$ iff $v_i \leqslant v'_i$ for all i. We also assume that the computational cost of f and all fuzzy predicates p_i is bounded by a constant.

We call $q(\mathbf{x})[s]$ its *head*, $\exists \mathbf{y}.R_1(\mathbf{z}_1)[s_1], \ldots, R_l(\mathbf{z}_l)[s_l]$ its *body* and $\mathsf{OrderBy}(s = f(s_1, \ldots, s_l, p_1(\mathbf{z}'_1), \ldots, p_h(\mathbf{z}'_h))$ the *scoring atom*. We also allow the scores $[s]$, $[s_1]$, $\ldots$, $[s_l]$ and the scoring atom to be omitted. In this case we assume the value 1 for s_i and s instead. The informal meaning of such a query is: if $\mathbf{z}_i$ is an instance of R_i to degree at least or equal to s_i, then $\mathbf{x}$ is an instance of q to degree at least or equal to s, where s has been determined by the scoring atom. The *answer set* $ans_{\mathcal{K}}(q)$ over $\mathcal{K}$ of a query q is the set of tuples $\langle \mathbf{t}, s \rangle$ such that $\mathcal{K} \models q(\mathbf{t})[s]$ with $s > 0$ (informally, $\mathbf{t}$ satisfies the query to non-zero degree s).

3 Towards Fuzzy DL-Lite Like Inclusion Axioms Learning

We now show how we may learn DL-Lite like inclusion axioms. We consider a learning problem where:

- the background theory $\mathcal{B}$ is a DL-Lite like knowledge base $\langle \mathcal{F}, \mathcal{O}, \mathcal{A} \rangle$;
- the training set $\mathcal{E}$ is a collection of fuzzy DL-Lite like facts of the form Eq. 4 and labeled as either positive or negative examples for the target atomic concept H. We assume that $\mathcal{F} \cap \mathcal{E} = \emptyset$;

– the target theory $\mathcal{H}$ is a set of inclusion axioms of the form

$$C_1 \sqcap \ldots \sqcap C_n \sqsubseteq H \tag{9}$$

where H is an atomic concept and each C_i has syntax

$$C \longrightarrow A \mid \exists R.A \mid \exists R.\top \ . \tag{10}$$

Now, we adapt Eq. 2 to our case and define

$$\mathcal{I}_{ILP} \models C(t) \text{ iff } \mathcal{B} \cup \mathcal{E} \models C(t)[s] \text{ and } s > 0 \ . \tag{11}$$

That is, we write $\mathcal{I}_{ILP} \models C(t)$ iff t can be inferred being instance of concept C to a non-zero degree.

Now, in order to account for multiple fuzzy instantiations of fuzzy predicates occurring in inclusion axioms, we customise Eq. 3 into the following formula for computing the confidence degree:

$$cf(RL \sqsubseteq H) = \frac{\sum_{t \in P} RL(t) \Rightarrow H(t)}{|D|} \ , \text{ where} \tag{12}$$

– $P = \{t \mid \mathcal{I}_{ILP} \models C_i(t), \mathcal{I}_{ILP} \models H(t) \text{ and } H(t)[s] \in \mathcal{E}^+\}$;
– $D = \{t \mid \mathcal{I}_{ILP} \models C_i(t) \text{ and } H(t)[s] \in \mathcal{E}\}$;
– $RL = C_1 \sqcap \ldots \sqcap C_n$;
– $\Rightarrow$ is an implication function (see, *e.g.* [6]);
– $RL(t) = \min(s_1, \ldots s_n)$, with $\mathcal{B} \cup \mathcal{E} \models C_i(t)[s_i]$;
– $H(t) = s$ with $H(t)[s] \in \mathcal{E}$.

Essentially, the numerator sums over all positive instances making the left hand and right hand side true, where $RL(t) \Rightarrow H(t)$ denotes the degree to which the implication holds. Clearly, the more positive instances supporting the inclusion axiom there are, the higher is the confidence degree of it. Note that the confidence score can be determined easily by submitting appropriate queries via our query language. From an algorithm point of view, it suffices to change the FOIL-Learn-One-Rule at step 7., where now l may be either an atomic concept A, or of the form $\exists R.A$ or $\exists R.\top$.

We illustrate our proposal via an example. For illustrative purposes consider the following case involving hotels. So, assume we have a background theory $\mathcal{B}$ with a relational database $\mathcal{F}$ storing facts such as

HotelTable			
id	rank	noRooms	hasRoomID
h1	3	21	r1
h2	5	123	r2
h3	4	95	r3

RoomTable		
id	price	roomType
r1	60	single
r1	90	double
r2	80	single
r2	120	double
r3	70	single
r3	90	double

Tower
id
t1

Park
id
p1
p2

DistanceTable			
id	from	to	time
d1	h1	t1	10
d2	h2	p1	15
d3	h3	p2	5

an ontology $\mathcal{O}$[6] encompassing the following inclusion axioms

$$Park \sqsubseteq Attraction \ , \ Tower \sqsubseteq Attraction \ , \ Attraction \sqsubseteq Site$$

[6] http://donghee.info/research/SHSS/ObjectiveConceptsOntology(OCO).html

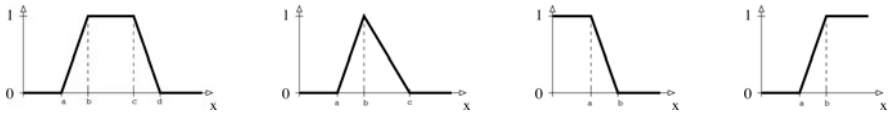

Fig. 2. (a) Trapezoidal function $trz(x; a, b, c, d)$, (b) triangular function $tri(x; a, b, c)$, (c) left shoulder function $ls(x; a, b)$, and (d) right shoulder function $rs(x; a, b)$

and abstraction statements:

$Hotel \mapsto (h.id)$. SELECT h.id
FROM HotelTable h

$hasRank \mapsto (h.id, h.rank)$. SELECT h.id, h.rank
FROM HotelTable h

$cheapPrice \mapsto (h.id, r.price)[score]$. SELECT h.id, r.price, $cheap$(r.price) AS score
FROM HotelTable h, RoomTable r
WHERE h.hasRoomID = r.id
ORDER BY score

$closeTo \mapsto (from, to)[score]$. SELECT d.from, d.to $closedistance$(d.time) AS score
FROM DistanceTable d
ORDER BY score

where $cheap(p)$ is a function determining how cheap a hotel room is given its price, modelled as *e.g.* a so-called left-shoulder function $cheap(p) = ls(p; 50, 100)$, while $closedistance(d) = ls(d; 5, 25)$. (see Figure 2 for typical fuzzy membership functions). Assume now that our target concept H is $GoodHotel$. As illustrative example, we compute the confidence score, according to Eq. 12, of

$$r : Hotel \sqcap \exists cheapPrice.\top \sqcap \exists closeTo.Attraction \sqsubseteq GoodHotel$$

i.e., a good hotel is one having a cheap price and close to an attraction. We assume that

- the implication function is Gödel (*i.e.*, it returns 1 if $x \leqslant y$, y if $x > y$);
- $\mathcal{E}^+ = \{GoodHotel^+(h1)[0.6], GoodHotel^+(h2)[0.8]\}$, while $\mathcal{E}^- = \{GoodHotel^-(h3)[0.4]\}$;
- $GoodHotel^+ \sqsubseteq GoodHotel$ and $GoodHotel^- \sqsubseteq GoodHotel$ occur in $\mathcal{B}$.

Now, it can be verified that for $\mathcal{T} = \mathcal{B} \cup \mathcal{E}$

1. The query

   $$q(h)[s] \leftarrow GoodHotel^+(h), cheapPrice(h, p)[s_1], closeTo(h, a)[s_2], Attraction(a), s{=}\min(s_1, s_2)$$

   has answer set over $\mathcal{T}$, $ans_{\mathcal{T}} = \{(h1, 0.75), (h2, 0.4)\}$;
2. The query

   $$q(h)[s] \leftarrow GoodHotel(h), cheapPrice(h, p)[s_1], closeTo(h, a)[s_2], Attraction(a), s = \min(s_1, s_2)$$

   has answer set over $\mathcal{T}$, $ans_{\mathcal{T}} = \{(h1, 0.75), (h2, 0.4), (h3, 0.6)\}$;
3. Therefore, according to Eq. 12, $P = \{h1, h2\}$, while $D = \{h1, h2, h3\}$;
4. As a consequence,

$$cf(r) = \frac{0.75 \Rightarrow 0.6 + 0.4 \Rightarrow 0.8}{3} = \frac{0.6 + 1.0}{3} = 0.5333 \ .$$

4 Final Remarks

In this paper we have proposed a method for inducing ontology inclusion axioms within the KR framework of a fuzzy DL-Lite like DL. The method extends FOIL, a popular ILP algorithm for learning sets of crisp rules, in a twofold direction: from crisp to fuzzy and from rules to inclusion axioms. Indeed, related FOIL-like algorithms are reported in the literature [12,3,11] but they can only learn fuzzy rules. Another relevant work is the formal study of fuzzy ILP contributed by [5]. Yet, it is less promising than our proposal from the practical side. Last, close to our application domain, [4] faces the problem of inducing equivalence axioms in a fragment of OWL corresponding to the $\mathcal{ALC}$ DL.

For the future we intend to define appropriate specialization operators for the fuzzy DL being considered. Also we would like to investigate in depth the impact of Open World Assumption (holding in DLs) on the proposed ILP setting, and implement and experiment our method. Finally, it can be interesting to analyze the effect of the different implication functions on the learning process.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning, pp. 260–270 (2006)
3. Drobics, M., Bodenhofer, U., Klement, E.-P.: FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions. Int. J. Approximate Reasoning 32(2-3), 131–152 (2003)
4. Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL Class Descriptions on Very Large Knowledge Bases. Int. J. on Semantic Web and Information Systems 5(2), 25–48 (2009)
5. Horváth, T., Vojtás, P.: Induction of fuzzy and annotated logic programs. In: Muggleton, S.H., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 260–274. Springer, Heidelberg (2007)
6. Klir, G.J., Yuan, B.: Fuzzy sets and fuzzy logic: theory and applications. Prentice-Hall, Inc., Upper Saddle River (1995)
7. Lukasiewicz, T., Straccia, U.: Top-k retrieval in description logic programs under vagueness for the semantic web. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 16–30. Springer, Heidelberg (2007)
8. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics 6, 291–308 (2008)
9. Nienhuys-Cheng, S.-H., de Wolf, R.: Foundations of Inductive Logic Programming. LNCS(LNAI), vol. 1228. Springer, Heidelberg (1997)
10. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning 5, 239–266 (1990)
11. Serrurier, M., Prade, H.: Improving expressivity of inductive logic programming by learning different kinds of fuzzy rules. Soft Computing 11(5), 459–466 (2007)

12. Shibata, D., Inuzuka, N., Kato, S., Matsui, T., Itoh, H.: An induction algorithm based on fuzzy logic programming. In: Zhong, N., Zhou, L. (eds.) PAKDD 1999. LNCS (LNAI), vol. 1574, pp. 268–274. Springer, Heidelberg (1999)
13. Straccia, U.: Reasoning within fuzzy description logics. Journal of Artificial Intelligence Research 14, 137–166 (2001)
14. Straccia, U.: SoftFacts: a top-k retrieval engine for a tractable description logic accessing relational databases. Technical report (2009)
15. Straccia, U.: SoftFacts: A top-k retrieval engine for ontology mediated access to relational databases. In: Proc. of the 2010 IEEE Int. Conf. on Systems, Man and Cybernetics, pp. 4115–4122. IEEE Press, Los Alamitos (2010)