An Inductive Logic Programming Approach to Learning Inclusion Axioms in Fuzzy Description Logics

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 very little 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* [3], the current standard language to represent ontologies, and its profiles. [4] E.g., DL-Lite [2] is the DL behind the *OWL 2 QL* profile and is especially aimed at applications that use very large volumes of instance data, and where query answering is the most important reasoning task.

On the other hand, it is well-known that "classical" ontology languages are not appropriate to deal with *vague knowledge*, which is inherent to several real world domains [21]. So far, several fuzzy extensions of DLs can be found in the literature (see the survey in [14]), which includes, among others a fuzzy DL-Lite like DL [23] which has been implemented in the SoftFacts system [23] [5].

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 very little attention so far. In this work, we report our preliminary investigation on this issue by describing

[3] `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`
[4] `http://www.w3.org/TR/owl2-profiles/`.
[5] See, `http://www.straccia.info/software/SoftFacts/SoftFacts.html`

	Łukasiewicz logic	Gödel logic	Product logic
$a \otimes b$	$\max(a+b-1,0)$	$\min(a,b)$	$a \cdot b$
$a \oplus b$	$\min(a+b,1)$	$\max(a,b)$	$a+b-a\cdot b$
$a \Rightarrow b$	$\min(1-a+b,1)$	$\begin{cases}1 & \text{if } a \leqslant b\\ b & \text{otherwise}\end{cases}$	$\min(1,b/a)$
$\ominus\, a$	$1-a$	$\begin{cases}1 & \text{if } a=0\\ 0 & \text{otherwise}\end{cases}$	$\begin{cases}1 & \text{if } a=0\\ 0 & \text{otherwise}\end{cases}$

Table 1. Combination functions of various fuzzy logics.

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 Mathematical Fuzzy Logic, Fuzzy DLs and ILP. 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.

2 Background

2.1 Mathematical Fuzzy Logic Basics

In *Mathematical Fuzzy Logic* [7], the convention prescribing that a statement is either true or false is changed and is a matter of degree measured on an ordered scale that is no longer $\{0,1\}$, but the $[0,1]$. This degree is called *degree of truth* (or *score*) of the logical statement ϕ in the interpretation $\mathcal{I}$. In this section, *fuzzy statements* have the form $\phi[r]$, where $r \in [0,1]$ (see, *e.g.* [6,7]) and ϕ is a statement, which encode that the degree of truth of ϕ is *greater or equal* r.

A *fuzzy interpretation* $\mathcal{I}$ maps each basic statement p_i into $[0,1]$ and is then extended inductively to all statements: $\mathcal{I}(\phi \wedge \psi) = \mathcal{I}(\phi) \otimes \mathcal{I}(\psi)$, $\mathcal{I}(\phi \vee \psi) = \mathcal{I}(\phi) \oplus \mathcal{I}(\psi)$, $\mathcal{I}(\phi \rightarrow \psi) = \mathcal{I}(\phi) \Rightarrow \mathcal{I}(\psi)$, $\mathcal{I}(\neg\phi) = \ominus\mathcal{I}(\phi)$, $\mathcal{I}(\exists x.\phi(x)) = \sup_{a \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(a))$, $\mathcal{I}(\forall x.\phi(x)) = \inf_{a \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(a))$, where $\Delta^{\mathcal{I}}$ is the domain of $\mathcal{I}$, and $\otimes$, $\oplus$, $\Rightarrow$, and $\ominus$ are so-called *t-norms*, *t-conorms*, *implication functions*, and *negation functions*, respectively, which extend the Boolean conjunction, disjunction, implication, and negation, respectively, to the fuzzy case.

One usually distinguishes three different logics, namely Łukasiewicz, Gödel, and Product logics [7], whose combination functions are reported in Table 1. The operators for Zadeh logic, namely $a \otimes b = \min(a,b)$, $a \oplus b = \max(a,b)$, $\ominus\, a = 1-a$ and $a \Rightarrow b = \max(1-a,b)$, can be expressed in Łukasiewicz logic[6].

[6] More precisely, $\min(a,b) = a \otimes_L (a \Rightarrow_L b), \max(a,b) = 1 - \min(1-a, 1-b)$.

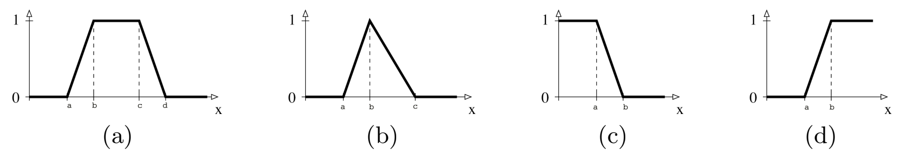

Fig. 1. (a) Trapezoidal function $trz(x; a, b, c, d)$, (b) triangular function $tri(x; a, b, c)$, (c) left shoulder function $ls(x; a, b)$, and (d) right shoulder function $rs(x; a, b)$.

A *fuzzy set* R over a countable crisp set X is a function $R \colon X \to [0, 1]$. The trapezoidal (Fig. 1 (a)), the triangular (Fig. 1 (b)), the L-function (left-shoulder function, Fig. 1 (c)), and the R-function (right-shoulder function, Fig. 1 (d)) are frequently used to specify membership degrees. In particular, the left-shoulder function is defined as

$$ls(x; a, b) = \begin{cases} 1 & \text{if } x \leqslant a \\ 0 & \text{if } x \geqslant b \\ (b - x)/(b - a) & \text{if } x \in [a, b] \end{cases} \tag{1}$$

The notions of satisfiability and logical consequence are defined in the standard way. A fuzzy interpretation $\mathcal{I}$ *satisfies* a fuzzy statement $\phi[r]$ or $\mathcal{I}$ is a *model* of $\phi[r]$, denoted $\mathcal{I} \models \phi[r]$ iff $\mathcal{I}(\phi) \geqslant r$.

2.2 DL-Lite like description logic and its fuzzy extensions

For computational reasons, the logic we adopt is based on a fuzzy extension of the DL-Lite DL without negation [23]. It supports at the intensional level unary relations (called *concepts*) and binary relations (called *roles*), while supports n-ary relations (relational tables) at the extensional level.

Formally, 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 [22]). Information can be retrieved from the knowledge base by means of an appropriate *query language* discussed later.

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

$$R(c_1, \ldots, c_n)[s] \ , \tag{2}$$

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.[7] Facts are stored in a relational database. We may omit the score component and in such case the value 1 is assumed.

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

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

$$Rl_1 \sqcap \ldots \sqcap Rl_m \sqsubseteq Rr \ , \tag{3}$$

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*[8]. 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 $\mathcal{O}$). Also we recall that from a semantics point of view, Gödel logic is adopted. The intuition for one such 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:

$$\begin{array}{l} Rl \longrightarrow A \mid R[i_1, i_2] \\ Rr \longrightarrow A \mid R[i_1, i_2] \mid \exists R.A \end{array} \tag{4}$$

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 t-norm in the Gödel logic (see Table 1).

Abstraction Component. $\mathcal{A}$ (similarly to [3,17]) is a set of "abstraction statements" that allow to connect atomic concepts and roles to physical relational tables. Essentially, this component 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

$$R \mapsto (c_1, \ldots, c_n)[c_{score}].sql \ , \tag{5}$$

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 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.

[8] Note that recursive inclusion axioms are allowed.

Query Language. The query language enables the formulation of conjunctive queries with a scoring function to rank the answers. More precisely, a *ranking query* [13] is of the form

$$\begin{aligned} q(\mathbf{x})[s] \leftarrow & \exists \mathbf{y}\ R_1(\mathbf{z}_1)[s_1], \ldots, R_l(\mathbf{z}_l)[s_l], \\ & \mathsf{OrderBy}(s = f(s_1, \ldots, s_l, p_1(\mathbf{z}'_1), \ldots, p_h(\mathbf{z}'_h)), \mathsf{Limit}(k) \end{aligned} \tag{6}$$

where

1. q is an n-ary relation, every R_i is a n_i-ary relation ($1 \leqslant n_i \leqslant 2$). $R_i(\mathbf{z_i})$ 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 $f\colon ([0,1])^{l+h} \to [0,1]$, 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 s to be assigned to $q(\mathbf{c})$. 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;
8. $\mathsf{Limit}(k)$ indicates the number of answers to retrieve and is optional. If omitted, all answers are retrieved.

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) and the score s is as high as possible, *i.e.* if $\langle \mathbf{t}, s \rangle \in ans_{\mathcal{K}}(q)$ then (i) $\mathcal{K} \not\models q(\mathbf{t})[s']$ for any $s' > s$; and (ii) there cannot be another $\langle \mathbf{t}, s' \rangle \in ans_{\mathcal{K}}(q)$ with $s > s'$.

2.3 Learning rules with ILP

Inductive Logic Programming (ILP) was born at the intersection between Concept Learning and Logic Programming [16].

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

$$B(\mathbf{x}) \rightarrow H(\mathbf{x}) \tag{7}$$

where $\mathbf{x}$ is the vector of the n variables that appear in the rule, $B(\mathbf{x}) = B_0(\mathbf{x}) \wedge \ldots \wedge B_q(\mathbf{x})$ 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 with respect to other forms of Concept Learning is the use of prior domain knowledge in the background during the induction process. The classical ILP problem is described by means of two logic programs: (i) the *background theory* $\mathcal{K}$ 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 predicate to be learnt. 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). A rule r *covers* an example $e \in \mathcal{E}$ iff $\mathcal{K} \cup \{r\} \models e$. The task of induction is to find, given $\mathcal{K}$ and $\mathcal{E}$, a set $\mathcal{H}$ of rules such that: (i) $\forall e \in \mathcal{E}^+, \mathcal{K} \cup \mathcal{H} \models e$ (completeness of $\mathcal{H}$) and (ii) $\forall e \in \mathcal{E}^-, \mathcal{K} \cup \mathcal{H} \not\models e$ (consistency of $\mathcal{H}$). Two further restrictions hold naturally. One is that $\mathcal{K} \not\models \mathcal{E}^+$ since, in such a case, $\mathcal{H}$ would not be necessary to explain $\mathcal{E}^+$. The other is $\mathcal{K} \cup \mathcal{H} \not\models \bot$, which means that $\mathcal{K} \cup \mathcal{H}$ is a consistent theory. Usually, rule induction fits with the idea of providing a compression of the information contained in $\mathcal{E}$.

A popular ILP algorithm for learning sets of rules is FOIL [18]. It performs a greedy search in order to maximise a gain function. The rules are induced until all 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 $\mathcal{E}$. This is the *sequential covering* approach underlying the function FOIL-Learn-Sets-of-Rules shown in Figure 2. For inducing a rule, the function FOIL-Learn-One-Rule reported in Figure 3 starts with the most general clause ($\top \rightarrow H(\mathbf{x})$) and specialises it step by step by adding literals to 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:

$$\textsc{Gain}(r', r) = p * (log_2(cf(r')) - log_2(cf(r))) \ , \tag{8}$$

where p is the number of distinct positive examples covered by the rule r that are still covered by r'. Thus, the gain is positive iff r' is more informative 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.

```
function FOIL-Learn-Sets-of-Rules($H$, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{K}$): $\mathcal{H}$
begin
1. $\mathcal{H} \leftarrow \emptyset$;
2. while $\mathcal{E}^+ \neq \emptyset$ do
3.       $r \leftarrow$ FOIL-Learn-One-Rule($H$, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{K}$);
4.       $\mathcal{H} \leftarrow \mathcal{H} \cup \{r\}$;
5.       $\mathcal{E}^+_r \leftarrow \{e \in \mathcal{E}^+ | \mathcal{K} \cup r \models e\}$;
6.       $\mathcal{E}^+ \leftarrow \mathcal{E}^+ \setminus \mathcal{E}^+_r$;
7. endwhile
8. return $\mathcal{H}$
end
```

Fig. 2. Algorithm for learning sets of rules in FOIL

Given a Horn clause $B(\mathbf{x}) \rightarrow H(\mathbf{x})$, the confidence degree is given by:

$$cf(B(\mathbf{x}) \rightarrow H(\mathbf{x})) = P(B(\mathbf{x}) \wedge H(\mathbf{x}))/P(B(\mathbf{x})) \ . \tag{9}$$

Confidence degrees are computed in the spirit of domain probabilities [8]. Input data in ILP are supposed to describe one interpretation under CWA. We call $\mathcal{I}_{ILP}$ this interpretation. So, given a fact f, we define:

$$\mathcal{I}_{ILP} \models f \text{ iff } \mathcal{K} \cup \mathcal{E} \models f \ . \tag{10}$$

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

$$cf(B(\mathbf{x}) \rightarrow H(\mathbf{x})) = \frac{|\mathbf{t} \in \mathcal{D}^n \mid \mathcal{I}_{ILP} \models B(\mathbf{t}) \text{ and } H(\mathbf{t}) \in \mathcal{E}^+|}{|\mathbf{t} \in \mathcal{D}^n \mid \mathcal{I}_{ILP} \models B(\mathbf{t}) \text{ and } H(\mathbf{t}) \in \mathcal{E}|} \tag{11}$$

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.

3 Towards Learning Fuzzy DL-Lite like Inclusion Axioms

In this section we consider a learning problem where:

- the target concept H is a DL-Lite atomic concept;
- the background theory $\mathcal{K}$ is a DL-Lite like knowledge base $\langle \mathcal{F}, \mathcal{O}, \mathcal{A} \rangle$ of the form described in Section 2.2;
- the training set $\mathcal{E}$ is a collection of fuzzy DL-Lite like facts of the form (2) and labeled as either positive or negative examples for H. We assume that $\mathcal{F} \cap \mathcal{E} = \emptyset$;

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

Fig. 3. Algorithm for learning one rule in FOIL

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

$$B \sqsubseteq H \qquad (12)$$

where H is an atomic concept, $B = C_1 \sqcap \ldots \sqcap C_m$, and each concept C_i has syntax

$$C \longrightarrow A \mid \exists R.A \mid \exists R.\top \ . \qquad (13)$$

Note that the language of hypotheses $\mathcal{L}_{\mathcal{H}}$ differs from the language of the background theory $\mathcal{L}_{\mathcal{K}}$ as for the form of axioms. Yet the alphabet underlying $\mathcal{L}_{\mathcal{H}}$ is a subset of the alphabet for $\mathcal{L}_{\mathcal{K}}$. Note also that $\mathcal{H}$, in order to be acceptable, must be complete and consistent w.r.t. $\mathcal{E}$, *i.e.* it must cover all the positive examples and none of the negative examples.

3.1 The FOIL-like algorithm

We now show how we may learn inclusion axioms of the form (12). To this aim, we adapt (10) to our case and define for $C \neq H$

$$\mathcal{I}_{ILP} \models C(t) \text{ iff } \mathcal{K} \cup \mathcal{E} \models C(t)[s] \text{ and } s > 0 \ . \qquad (14)$$

That is, we write $\mathcal{I}_{ILP} \models C(t)$ iff it can be inferred from $\mathcal{K}$ and $\mathcal{E}$ that t is an instance of concept C to a non-zero degree. Note that $\mathcal{E}$ is split into $\mathcal{E}^+$ and $\mathcal{E}^-$. In order to distinguish between the two sets while using a uniform representation with $\mathcal{K}$, we introduce two additional concepts, H^+ and H^-, whose intension coincide with the sets $\mathcal{E}^+$ and $\mathcal{E}^-$, respectively, as well as the axioms $H^+ \sqsubseteq H$ and $H^- \sqsubseteq H$. We call $\mathcal{K}'$ the background theory augmented with the training set represented this way, *i.e.* $\mathcal{K}' = \mathcal{K} \cup \mathcal{E}$.

Now, in order to account for multiple fuzzy instantiations of fuzzy predicates occurring in the inclusion axioms of interest to us, we customise (11) into the following formula for computing the confidence degree:

$$cf(B \sqsubseteq H) = \frac{\sum_{t \in P} B(t) \Rightarrow H(t)}{|D|} \tag{15}$$

where

- $P = \{t \mid \mathcal{I}_{ILP} \models C_i(t) \text{ and } H(t)[s] \in \mathcal{E}^+\}$, *i.e.* P is the set of instances for which the implication covers a positive example;
- $D = \{t \mid \mathcal{I}_{ILP} \models C_i(t) \text{ and } H(t)[s] \in \mathcal{E}\}$, *i.e.* D is the set of instances for which the implication covers an example (either positive or negative);
- $B(t) \Rightarrow H(t)$ denotes the degree to which the implication holds for a certain instance t;
- $B(t) = \min(s_1, \ldots, s_n)$, with $\mathcal{K} \cup \mathcal{E} \models C_i(t)[s_i]$;
- $H(t) = s$ with $H(t)[s] \in \mathcal{E}$.

Clearly, the more positive instances supporting the inclusion axiom, the higher the confidence degree of the axiom.

Note that the confidence score can be determined easily by submitting appropriate queries via the query language described in Section 2.2. More precisely, proving the fuzzy entailment in (14) for each C_i is equivalent to answering a unique ranking query whose body is the conjunction of the relations R_l resulting from the transformation of C_i's into FOL predicates and whose score s is given by the minimum between s_l's.

From an algorithm point of view, it suffices to change FOIL-Learn-One-Rule at step 7., where now l may be of any of the forms allowed in (13). More precisely, in line with the tradition in ILP and in conformance with the search direction in FOIL, we devise a specialization operator, *i.e.* an operator for traversing the hypotheses space top down, with the following refinement rules:

1. Add atomic concept (A)
2. Add complex concept by existential role restriction ($\exists R.\top$)
3. Add complex concept by qualified existential role restriction ($\exists R.A$)
4. Replace atomic concept (A replaced by A' if $A' \sqsubseteq A$)
5. Replace complex concept ($\exists R.A$ replaced by $\exists R.A'$ if $A' \sqsubseteq A$)

The rules are numbered according to an order of precedence, *e.g.* the addition of an atomic concept has priority over the addition of a complex concept obtained by existential role restriction. A rule can be applied when the preceding one in the list can not be applied anymore. Concept and role names in the alphabet underlying $\mathcal{L}_{\mathcal{H}}$ are themselves ordered. This implies that, *e.g.*, the addition of an atomic concept is not possible anymore when all the atomic concepts in the alphabet have been already used in preceding applications of the rule.

3.2 An illustrative example

For illustrative purposes we consider the following case involving the classification of hotels as good ones. We assume to have a background theory $\mathcal{K}$ with a relational database $\mathcal{F}$ (reported in Figure 4), an ontology $\mathcal{O}$ [9] (illustrated in Figure 5) which encompasses the following inclusion axioms

$Park \sqsubseteq Attraction$
$Tower \sqsubseteq Attraction$
$Attraction \sqsubseteq Site$
$Hotel \sqsubseteq Site$

and the following set $\mathcal{A}$ of 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.id = r.hotel
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 $closedistance(d) = ls(d; 5, 25)$.

Assume now that:

- $H = GoodHotel$;
- $\mathcal{E}^+ = \{GoodHotel(h1)[0.6], GoodHotel(h2)[0.8]\}$;
- $\mathcal{E}^- = \{GoodHotel(h3)[0.4]\}$.

In order to have a uniform representation of the examples w.r.t. the background theory, we transform $\mathcal{E}$ as follows:

$GoodHotel^+ \sqsubseteq GoodHotel$
$GoodHotel^- \sqsubseteq GoodHotel$
$GoodHotel^+(h1)[0.6]$
$GoodHotel^+(h2)[0.8]$
$GoodHotel^-(h3)[0.4]$

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

HotelTable		
id	rank	noRooms
h1	3	21
h2	5	123
h3	4	95

RoomTable			
id	price	roomType	hotel
r1	60	single	h1
r2	90	double	h1
r3	80	single	h2
r4	120	double	h2
r5	70	single	h3
r6	90	double	h3

Tower
id
t1

Park
id
p1
p2

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

Fig. 4. Hotel database

and call $\mathcal{K}'$ the background theory augmented with the training set represented this way.

The following inclusion axioms:

$r_0 : \top \sqsubseteq \mathit{GoodHotel}$
$r_1 : \mathit{Hotel} \sqsubseteq \mathit{GoodHotel}$
$r_2 : \mathit{Hotel} \sqcap \exists \mathit{cheapPrice}.\top \sqsubseteq \mathit{GoodHotel}$
$r_3 : \mathit{Hotel} \sqcap \exists \mathit{cheapPrice}.\top \sqcap \exists \mathit{closeTo}.\mathit{Attraction} \sqsubseteq \mathit{GoodHotel}$
$r_4 : \mathit{Hotel} \sqcap \exists \mathit{cheapPrice}.\top \sqcap \exists \mathit{closeTo}.\mathit{Park} \sqsubseteq \mathit{GoodHotel}$
$r_5 : \mathit{Hotel} \sqcap \exists \mathit{cheapPrice}.\top \sqcap \exists \mathit{closeTo}.\mathit{Tower} \sqsubseteq \mathit{GoodHotel}$

belong to $\mathcal{L}_{\mathcal{H}} = \{r | C_i \in \{\top, \mathit{Hotel}, \exists \mathit{cheapPrice}, \exists \mathit{closeTo}\}\} \subset \mathcal{L}_{\mathcal{K}}$. They can be read as:

r_0 : Everything is a good hotel
r_1 : Every hotel is a good hotel
r_2 : Hotels having a cheap price are good hotels
r_3 : Hotels having a cheap price and close to an attraction are good hotels
r_4 : Hotels having a cheap price and close to a park are good hotels
r_5 : Hotels having a cheap price and close to a tower are good hotels

thus highlighting the possibility of generating "extreme" hypotheses about good hotels such as r_0 and r_1. Of course, some of them will be discarded on the basis of their confidence degree.

Before showing how hypotheses evaluation is performed in our adaptation of FOIL, we illustrate the computation of the confidence degree for r_3. It can be verified that for $\mathcal{K}'$

1. The query
$$\begin{aligned} q_P(h)[s] \leftarrow\ & \mathit{GoodHotel}^+(h), \\ & \mathit{cheapPrice}(h,p)[s_1], \\ & \mathit{closeTo}(h,a)[s_2], \mathit{Attraction}(a), \\ & s = \min(s_1, s_2) \end{aligned}$$
has answer set $\mathit{ans}_{\mathcal{K}'}(q_P) = \{\langle h1, 0.75\rangle, \langle h2, 0.4\rangle\}$ over $\mathcal{K}'$;

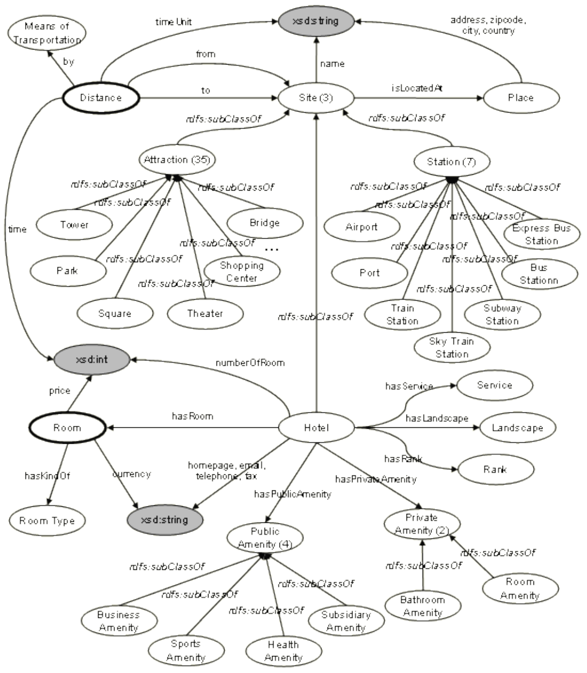

Fig. 5. Hotel ontology.

2. The query

$$\begin{aligned} q_D(h)[s] \leftarrow\ & GoodHotel(h), \\ & cheapPrice(h,p)[s_1], \\ & closeTo(h,a)[s_2], Attraction(a), \\ & s = \min(s_1, s_2) \end{aligned}$$

   has answer set $ans_{\mathcal{K}'}(q_D) = \{\langle h1, 0.75\rangle, \langle h2, 0.4\rangle, \langle h3, 0.6\rangle\}$ over $\mathcal{K}'$;
3. Therefore, according to (15), $P = \{h1, h2\}$, while $D = \{h1, h2, h3\}$;
4. As a consequence,

$$cf(r_3) = \frac{0.75 \Rightarrow 0.6 + 0.4 \Rightarrow 0.8}{3} = \frac{0.6 + 1.0}{3} = 0.5333\ .$$

Note that in q_P the literals $Hotel(h)$ and $GoodHotel(h)$ are removed from the body in favour of $GoodHotel^+(h)$ because the concepts $Hotel$ and $GoodHotel$ subsume $GoodHotel$ (due to a derived axiom) and $GoodHotel^+$ (due to an asserted axiom) respectively. Analogously, in q_D, the literal $Hotel(h)$ is superseded by $GoodHotel(h)$.

Analogously, we can obtain:

$$cf(r_2) = \frac{0.8 \Rightarrow 0.6 + 0.4 \Rightarrow 0.8}{3} = \frac{0.6 + 0.4}{3} = 0.3333\ .$$

$$cf(r_4) = \frac{0.4 \Rightarrow 0.8}{2} = \frac{0.4}{2} = 0.2\ .$$

$$cf(r_5) = \frac{0.8 \Rightarrow 0.6}{2} = \frac{0.6}{2} = 0.3 \ .$$

The function FOIL-Learn-One-Rule starts from r_0 which is then specialized into r_1 by applying the refinement rule which adds an atomic concept, *Hotel*, to the left-hand side of the axiom. As aforementioned, r_0 and r_1 are trivial hypotheses, therefore we can skip the computation steps for them and go ahead. In particular, the algorithm generates r_2 from r_1 by adding a complex concept obtained as existential restriction of the role *cheapPrice*. This hypothesis is not consistent with the training set, therefore it must be specialized in order not to cover the negative example. Considering that r_3, r_4 and r_5 are both possible specializations of r_2, we can now compute the information gain for each of them according to (8):

$$\text{Gain}(r_3, r_2) = 2*(log_2(0.5333) - log_2(0.3333)) = 2*(-0.907+1.5851) = 1.3562 \ ,$$

$$\text{Gain}(r_4, r_2) = 1*(log_2(0.2) - log_2(0.3333)) = (-2.3219 + 1.5851) = -0.7368 \ ,$$

$$\text{Gain}(r_5, r_2) = 1*(log_2(0.3) - log_2(0.3333)) = (-1.7369 + 1.5851) = -0.1518 \ ,$$

The algorithm will prefer r_3. Yet r_3 still covers the negative example, therefore it must be further refined, e.g. by strengthening the qualified restriction of *closeTo*. The algorithm then generates once again the axioms r_4 and r_5 which have the following values of information gain over r_3:

$$\text{Gain}(r_4, r_3) = 1*(log_2(0.2) - log_2(0.5333)) = (-2.3219 + 0.907) = -1.4149 \ ,$$

$$\text{Gain}(r_5, r_3) = 1*(log_2(0.3) - log_2(0.5333)) = (-1.7369 + 0.907) = -0.8299 \ ,$$

The axiom r_5 is more informative than r_4, therefore it is preferred to r_4. Also it does not cover the negative example. Indeed, the literal $\exists closeTo.Tower$ is a discriminant feature. Therefore, r_5 becomes part of the target theory. Since one positive example is still uncovered, the computation continues within the function FOIL-Learn-Sets-of-Rules aiming at finding a complete theory, *i.e.* a theory which explains all the positive examples.

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 where vagueness is dealt with the Gödel logic. 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 [20,5,19] but they can only learn fuzzy rules. Another relevant work is the formal study of fuzzy ILP contributed by [10]. Yet, it is less promising than our proposal from the practical side. Close to our application domain, [9] faces the problem of inducing equivalence axioms in a fragment of OWL corresponding to the $\mathcal{ALC}$ DL. Last, the work reported in [11] is based on an ad-hoc translation of fuzzy Łukasiewicz $\mathcal{ALC}$ DL constructs into LP and then uses a conventional ILP method to lean rules. The method is not sound as it has been recently shown that the traduction from fuzzy DLs to LP is incomplete [15] and entailment in Łukasiewicz $\mathcal{ALC}$ is undecidable [4].

For the future we intend to study more formally the proposed specialization operator 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. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.
2. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proc. of the Tenth Int. Conf. on Principles of Knowledge Representation and Reasoning (KR-06)*, pages 260–270, 2006.
3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, and M. Ruzzi. Data integration through DL-Lite$_{\mathrm{A}}$ ontologies. In K.-D. Schewe and B. Thalheim, editors, *Semantics in Data and Knowledge Bases*, number 4925 in Lecture Notes in Computer Science, pages 26–47. Springer Verlag, 2008.
4. M. Cerami and U. Straccia. On the Undecidability of Fuzzy Description Logics with GCIs with Lukasiewicz t-norm. Technical report, Computing Research Repository, 2011. Available as CoRR technical report at http://arxiv.org/abs/1107.4212.
5. M. Drobics, U. Bodenhofer, and E.-P. Klement. FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions. *Int. J. Approximate Reasoning*, 32(2-3):131–152, 2003.
6. R. Hähnle. Advanced many-valued logics. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, 2nd Edition*, volume 2. Kluwer, 2001.
7. P. Hájek. *Metamathematics of Fuzzy Logic.* Kluwer, 1998.
8. J.Y. Halpern. An Analysis of First-Order Logics of Probability. *Artificial Intelligence*, 46(3):311–350, 1990.
9. S. Hellmann, J. Lehmann, and S. Auer. Learning of OWL Class Descriptions on Very Large Knowledge Bases. *International Journal on Semantic Web and Information Systems*, 5(2):25–48, 2009.
10. T. Horváth and P. Vojtás. Induction of fuzzy and annotated logic programs. In S. Muggleton, R. P. Otero, and A. Tamaddoni-Nezhad, editors, *Inductive Logic Programming*, volume 4455 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2007.
11. S. Konstantopoulos and A. Charalambidis. Formulating description logic learning as an inductive logic programming task. In *Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010)*, pages 1–7. IEEE Press, 2010.
12. G.J. Klir and Bo Yuan. *Fuzzy sets and fuzzy logic: theory and applications.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

13. T. Lukasiewicz and U. Straccia. Top-k retrieval in description logic programs under vagueness for the semantic web. In H. Prade, V.S. Subrahmanian, editors, *Scalable Uncertainty Management*, number 4772 in Lecture Notes in Computer Science, pages 16–30. Springer Verlag, 2007.
14. T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6:291–308, 2008.
15. B. Motik and R. Rosati. A faithful integration of description logics with logic programming. In M.M. Veloso, editor, *IJCAI 2007, Proc. of the 20th Int. Joint Conf. on Artificial Intelligence*, pages 477–482, 2007.
16. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.
17. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *Journal of Data Semantics*, 10:133–173, 2008.
18. J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
19. M. Serrurier and H. Prade. Improving expressivity of inductive logic programming by learning different kinds of fuzzy rules. *Soft Computing*, 11(5):459–466, 2007.
20. D. Shibata, N. Inuzuka, S. Kato, T. Matsui, and H. Itoh. An induction algorithm based on fuzzy logic programming. In N. Zhong and L. Zhou, editors, *Methodologies for Knowledge Discovery and Data Mining*, volume 1574 of *Lecture Notes in Computer Science*, pages 268–273. Springer, 1999.
21. U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.
22. U. Straccia. Softfacts: a top-k retrieval engine for a tractable description logic accessing relational databases. Technical report, 2009.
23. U. Straccia. Softfacts: A top-k retrieval engine for ontology mediated access to relational databases. In *Proceedings of the 2010 IEEE International Conference on Systems, Man and Cybernetics (SMC-10)*, pages 4115–4122. IEEE Press, 2010.