Fundamenta Informaticae 124 (2013) 503–519
DOI 10.3233/FI-2013-846
IOS Press

A Logic-based Computational Method for the Automated Induction of Fuzzy Ontology Axioms

Francesca A. Lisi*
Dipartimento di Informatica
Università degli Studi di Bari "Aldo Moro", Italy
francesca.lisi@uniba.it

Umberto Straccia
ISTI - CNR, Pisa, Italy
umberto.straccia@isti.cnr.it

Abstract. Fuzzy Description Logics (DLs) are logics that allow to deal with structured vague 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 the evolution of fuzzy ontologies has received very little attention so far. We describe here a logic-based computational method for the automated induction of fuzzy ontology axioms which follows the machine learning approach of Inductive Logic Programming. The potential usefulness of the method is illustrated by means of an example taken from the tourism application domain.

Keywords: Inductive Logic Programming, Fuzzy Description Logics, Ontologies.

1. Introduction

In Artificial Intelligence, an *ontology* refers to an engineering artifact constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words. This set of assumptions has usually the form of a First Order Logic (FOL) theory, where vocabulary words appear as unary or binary predicate names, respectively called concepts and relations. In the simplest case, an ontology describes a hierarchy of concepts related by subsumption

*Address for correspondence: Campus Universitario "E. Quagliariello", Via E. Orabona 4, 70125 Bari, Italy

relationships; in more sophisticated cases, suitable axioms are added in order to express other relationships between concepts and to constrain their intended interpretation. The logical languages known as *Description Logics* (DLs) [1] play a key role in the design of ontologies 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] For instance, DL-Lite [5] 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.

Ontologies provide a major source of *structured* knowledge. However, it is well-known that "classical" ontology languages are not appropriate to deal with *vague* knowledge, which is inherent to several real world domains and is particularly pervading in those domains where objects could be better described in natural language [28]. We recall for the inexpert reader that there has been a long-lasting misunderstanding in the literature of artificial intelligence and uncertainty modelling, regarding the role of probability/possibility theory and vague/fuzzy theory. A clarifying paper is [9]. Specifically, under *uncertainty theory* fall all those approaches in which statements rather than being either true or false, are true or false to some *probability* or *possibility* (for example, "it will rain tomorrow"). That is, a statement is true or false in any world/interpretation, but we are "uncertain" about which world to consider as the right one, and thus we speak about, *e.g.*, a probability distribution or a possibility distribution over the worlds. On the other hand, under *fuzzy theory* fall all those approaches in which statements (for example, "the hotel is cheap") are true to some *degree*, which is taken from a truth space (usually $[0, 1]$). That is, an interpretation maps a statement to a truth degree, since we are unable to establish whether a statement is entirely true or false due to the involvement of vague concepts, such as "cheap" (we cannot always say whether a hotel is cheap or not). Here, we shall focus on fuzzy logic only. So far, several fuzzy extensions of DLs can be found in the literature (see the survey in [19]), which includes, among others, a fuzzy DL-Lite like DL which has been implemented in the *Soft*FACTS system [29] [3].

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 the evolution of fuzzy ontologies still remains little addressed. In this paper, we describe a method, named *Soft*FOIL, for the automated induction of axioms expressed in *Soft*FACTS. The method follows the machine learning approach known as *Inductive Logic Programming* (ILP). ILP was born at the intersection between Logic Programming and Concept Learning [21]. From Logic Programming (LP) it has borrowed the representation formalism for both data and hypotheses. From Concept Learning it has inherited the inferential mechanisms for induction. A distinguishing feature of ILP, also with respect to other forms of Concept Learning, is the use of prior domain knowledge available in the background during the induction process. ILP has been traditionally concerned with rule induction for classification purposes. *Soft*FOIL adapts known results in ILP concerning crisp rules to the novel case of fuzzy DL inclusion axioms. More precisely, it adapts the popular rule induction method FOIL [24] to deal with structured vague knowledge.

The paper is structured as follows. Section 2 is devoted to preliminaries on DLs and Mathematical Fuzzy Logic. We assume the reader is familiar with LP and its applications to databases. Section 3 briefly introduces *Soft*FACTS. Section 4 describes the learning problem and the algorithm in *Soft*FOIL also by means of an illustrative example taken from the tourism application domain. Section 5 concludes the paper by discussing limits of the current work, related work and possible directions of future work.

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

[2] `http://www.w3.org/TR/owl2-profiles/`.

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

Table 1. Syntax and semantics of some typical DL constructs.

bottom (resp. top) concept	$\bot$ (resp. $\top$)	$\emptyset$ (resp. $\Delta^{\mathcal{I}}$)
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
individual	a	$a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
concept intersection	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
concept union	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y\ (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
general concept inclusion	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

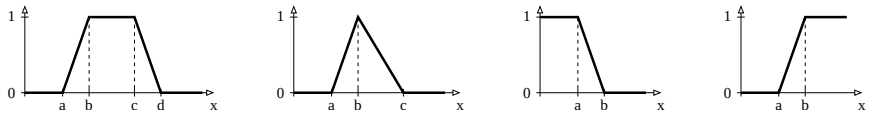
2. Preliminaries

2.1. Description Logics

DLs are a family of decidable FOL fragments that allow for the specification of structured knowledge in terms of classes (*concepts*), instances (*individuals*), and binary relations between instances (*roles*) [1]. Complex concepts can be defined from atomic concepts and roles by means of constructors. The syntax of some typical DL constructs is reported in Table 1. A DL knowledge base (KB) Σ consists of a so-called *terminological box* (TBox) $\mathcal{T}$ and a so-called *assertional box* (ABox) $\mathcal{A}$. The TBox is a finite set of axioms, called *general concept inclusions* (GCIs), which represent is-a relations between concepts, whereas the ABox is a finite set of *assertions* (or *facts*) that represent instance-of relations between individuals (resp. couples of individuals) and concepts (resp. roles). Thus, when a DL-based ontology language is adopted, an ontology is nothing else than a TBox, and a populated ontology corresponds to a whole DL KB (*i.e.*, encompassing also an ABox).

The semantics of DLs can be defined directly with set-theoretic formalizations as shown in Table 1 or through a mapping to FOL as shown in [2]. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ for a DL KB consists of a domain $\Delta^{\mathcal{I}}$ and a mapping function $\cdot^{\mathcal{I}}$. Under the *Unique Names Assumption* (UNA) [25], individuals are mapped to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$. However UNA does not hold by default in DLs. An interpretation $\mathcal{I}$ is a *model* of a KB $\Sigma = (\mathcal{T}, \mathcal{A})$ iff it satisfies all axioms and assertions in $\mathcal{T}$ and $\mathcal{A}$. In DLs a KB represents many different interpretations, *i.e.* all its models. This is coherent with the *Open World Assumption* (OWA) that holds in FOL semantics. A DL KB is *satisfiable* if it has at least one model. The main reasoning task for a DL KB Σ is the *consistency check* which tries to prove the satisfiability of Σ.

DLs have been recently used to provide access to large amounts of data through a high-level conceptual interface, which is of relevance to both data integration and ontology-based data access. In this setting, the TBox constitutes the conceptual, high-level view of the information managed by the system,



(a) (b) (c) (d)

Figure 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 $rs(x; a, b)$.

and the ABox is physically stored in an external relational database and accessed using the standard relational database technology [3, 23]. The fundamental inference service in this case is *query answering* by taking into account the constraints in the TBox and the data stored in the (external) ABox. The kind of queries that have most often been considered are first-order *conjunctive queries*, which correspond to the commonly used Select-Project-Join SQL queries. The key properties for such an approach to be viable in practice are (i) efficiency of query evaluation, with the ideal target being traditional database query processing, and (ii) that query evaluation can be done by leveraging the relational technology already used for storing the data. With these objectives in mind, the DL-Lite family of DLs has been proposed [5]. In DL-Lite, a distinction is made between the constructs allowed in the left-hand side and in the right-hand side of GCIs. Depending on the actual language used for the two sides, a different complexity is obtained. Variants of DL-Lite for which data complexity (*i.e.*, complexity w.r.t. the size of the ABox only) of query answering is in LOGSPACE are of high practical relevance w.r.t. the abovementioned properties [4].

2.2. Mathematical Fuzzy Logic

Fuzzy Logic is the logic of fuzzy sets. 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. For instance, the left-shoulder function is defined as

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

Although fuzzy sets have a far greater expressive power than classical crisp sets, its usefulness depends critically on our capability to construct appropriate membership functions for various given concepts in different contexts. The problem of constructing meaningful membership functions is a difficult one and we refer the interested reader to, *e.g.*, [15, Chapter 10]. However, one easy and typically satisfactory method to define the membership functions is to uniformly partition the range of, *e.g.*, salary values (bounded by a mimimum and maximum value), into 5 or 7 fuzzy sets using either trapezoidal functions (*e.g.* as illustrated on the left in Figure 2), or using triangular functions (as illustrated on the right in Figure 2). The latter is the more used one, as it has less parameters.

In *Mathematical Fuzzy Logic* [11], 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 $[0, 1]$.

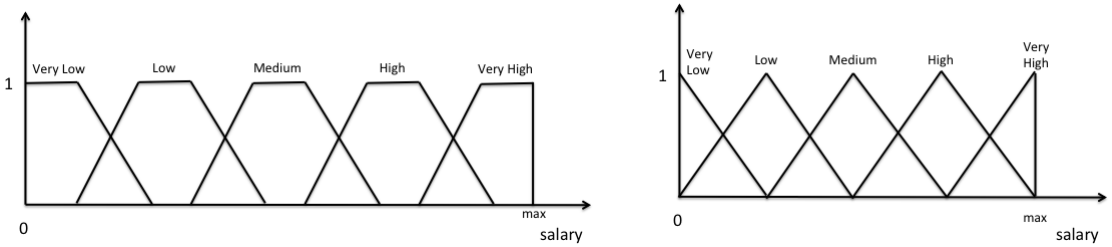


Figure 2. Fuzzy sets over salaries using trapezoidal or triangular functions.

	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 2. Combination functions of various fuzzy logics.

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 $r \in [0, 1]$ 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 [11], whose combination functions are reported in Table 2. 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 as $\mathcal{I} \models \phi[r]$, iff $\mathcal{I}(\phi) \geqslant r$.

3. Representing Fuzzy Ontologies in *Soft*FACTS

*Soft*FACTS is a top-k retrieval engine for ontology-mediated access to relational databases [29, 30]. It implements a logic based on a fuzzy extension of DL-Lite (more precisely, a variant without negation). Formally, a KB in *Soft*FACTS is the triple $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{A} \rangle$ which consists of a *facts component* $\mathcal{F}$, an *ontology component* $\mathcal{O}$ and an *abstraction component* $\mathcal{A}$. The ontology component is used to define the relevant abstract concepts and relations of the application domain by means of GCIs. The abstraction

component (similarly to [3, 23]) allows to connect atomic concepts and roles occurring in the ontology to database tables. Essentially, this component is used as a wrapper to the underlying database and, thus, prevents that table names occur in the ontology. Indeed, the semantics of a *Soft*FACTS KB is given by the possible transformation into positive DATALOG. Information can be retrieved from the KB by means of an appropriate *query language*. An important feature of *Soft*FACTS from the computational viewpoint is that the data complexity of query answering is within LOGSPACE.

Facts Component. The facts component $\mathcal{F}$ is a finite set of *assertions* 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 (typically, the score s has been computed by some external tool, such as a classifier, etc.). Facts are stored in a relational database. We may omit the score and in such case the value 1 is assumed.

Ontology Component. The ontology component $\mathcal{O}$ is a finite set of GCI *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*[4]. We assume that relations occurring in $\mathcal{F}$ do not occur in GCIs (so, we do not allow that database relation names occur in $\mathcal{O}$). 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 GCIs is specified below:

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

where A is an atomic concept and R is a role. Here $\exists[i_1, i_2]R$ is the binary relation obtained by projection of the n-ary relation R on the columns i_1, i_2 (the order of the indexes matters). Additionally, $\exists R.A$ 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 2).

Abstraction Component. The abstraction component $\mathcal{A}$ is a finite set of *statements* 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 the one with the lower score is removed). The score c_{score} may be omitted and in that case the score 1 is given to the tuples. We assume that R occurs in $\mathcal{O}$, while all the database tables occurring in the sql part occur in $\mathcal{F}$. Finally, we assume that there is at most one abstraction statement for each R.

[4]Note that recursive GCIs are allowed.

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

Figure 3. Hotel database

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

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

where $q(\mathbf{x})[s]$ is its *head*, and $\exists \mathbf{y}.R_1(\mathbf{z}_1)[s_1], \ldots, R_l(\mathbf{z}_l)[s_l]$ its *body*, and:

1. q is an n-ary relation, and 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. $\mathbf{x}$ is the tuple of the *distinguished variables*, and $\mathbf{y}$ is the tuple of the *non-distinguished variables*. We omit to write $\exists \mathbf{y}$ when $\mathbf{y}$ is clear from the context.
3. $\mathbf{z_i}, \mathbf{z'_j}$ are tuples of constants or variables in $\mathbf{x}$ or $\mathbf{y}$, and $s, s_1, \ldots, s_l$ are distinct variables and different from those in $\mathbf{x}$ and $\mathbf{y}$.
4. 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.
5. $\mathsf{OrderBy}(s = f(s_1, \ldots, s_l, p_1(\mathbf{z}'_1), \ldots, p_h(\mathbf{z}'_h))$ is the *scoring atom* where $f\colon ([0, 1])^{l+h} \to [0, 1]$ is a *scoring function* which combines the scores s_i 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 p_j is bounded by a constant. 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.
6. $\mathsf{Limit}(k)$ indicates the number of answers to retrieve. If omitted, all answers are retrieved.

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 $s > s'$.

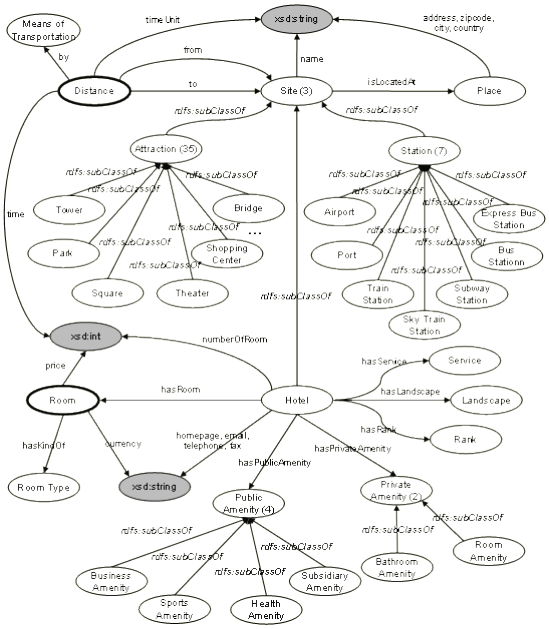


Figure 4. Hotel ontology.

Example 3.1. As a running example throughout the paper, we shall consider the case of hotel finding in a possible tourism application. We assume to have a *Soft*FACTS KB $\mathcal{K}$ with a relational database $\mathcal{F}$ (reported in Figure 3), an ontology $\mathcal{O}$ [5] (shown in Figure 4) which encompasses the following four GCIs

$\texttt{Park} \sqsubseteq \texttt{Attraction}$ $\texttt{Tower} \sqsubseteq \texttt{Attraction}$ $\texttt{Attraction} \sqsubseteq \texttt{Site}$ $\texttt{Hotel} \sqsubseteq \texttt{Site}$

and the following set $\mathcal{A}$ of abstraction statements:

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

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

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

$\texttt{closeTo} \mapsto (from, to)[score]$. 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 $closedistance(d) = ls(d; 5, 25)$.

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

4. Learning Fuzzy Ontology Axioms with *Soft*FOIL

4.1. The Learning Problem

The learning problem in hand concerns the automated induction of fuzzy ontology axioms within the framework of *Soft*FACTS. We would like to emphasise that reasoning in *Soft*FACTS works under Closed World Assumption (CWA) due to the link with databases highlighted in Section 3. Therefore, it is appropriate to resort to the methodological apparatus of ILP which we briefly recall before going into details of the problem of interest in this paper.

The classical ILP problem is described by means of two logic programs: (i) the *background theory* $\mathcal{K}$ which is a set of rules and ground facts; (ii) the *training set* $\mathcal{E}$ which is a set of ground facts, called *examples*, pertaining to the predicate (called *target concept*) whose rule-based definition is to be learnt. The set $\mathcal{E}$ 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 CWA valid in LP. 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) $\forall e \in \mathcal{E}^+, \mathcal{K} \cup \mathcal{H} \models e$ and (ii) $\forall e \in \mathcal{E}^-, \mathcal{K} \cup \mathcal{H} \not\models e$. These conditions are called *completeness* and *consistency*, respectively, of $\mathcal{H}$ w.r.t. $\mathcal{E}$. 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 that $\mathcal{K} \cup \mathcal{H} \not\models \bot$, which means that $\mathcal{K} \cup \mathcal{H}$ is a consistent theory. 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:

$$\mathcal{I}_{ILP} \models f \text{ iff } \mathcal{K} \cup \mathcal{E} \models f \ . \tag{7}$$

The domain $\mathcal{D}$ is the Herbrand domain described by $\mathcal{K}$ and $\mathcal{E}$.

In our **Fuzzy ILP (FILP)** problem, the hypotheses to be induced are fuzzy GCIs of the form

$$B \sqsubseteq H \tag{8}$$

where H is an atomic concept, $B = C_1 \sqcap \ldots \sqcap C_m$, and each concept C_i, $1 \leqslant i \leqslant m$, has syntax

$$C_i \quad \longrightarrow \quad A \mid \exists R.A \mid \exists R.\top \tag{9}$$

More formally, given:

- a consistent *Soft*FACTS KB $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{A} \rangle$ (the *background theory*);
- an atomic concept H for which no definition appears in $\mathcal{O}$ (the *target concept*);
- a collection $\mathcal{E}$ of facts of the form (2) and labeled as either positive or negative examples for H (the *training set*);
- a set $\mathcal{L}_{\mathcal{H}}$ of fuzzy GCIs of the form (8) (the *language of hypotheses*)

the goal is to find a set $\mathcal{H} \subset \mathcal{L}_{\mathcal{H}}$ (a *hypothesis*) such that $\mathcal{H}$ is complete and consistent w.r.t. $\mathcal{E}$. Here we assume that $\mathcal{F} \cap \mathcal{E} = \emptyset$. Also, the language $\mathcal{L}_{\mathcal{H}}$ is given implicitely by means of syntactic restrictions over a given alphabet as usual in ILP. In particular, the alphabet underlying $\mathcal{L}_{\mathcal{H}}$ is a subset of the alphabet for the language of the background theory $\mathcal{L}_{\mathcal{K}}$. However, $\mathcal{L}_{\mathcal{H}}$ differs from $\mathcal{L}_{\mathcal{K}}$ as for the form of axioms.

We adapt (7) to the FILP setting and define for $C \neq H$

$$\mathcal{I}_{FILP} \models C(t) \text{ iff } \mathcal{K} \cup \mathcal{E} \models C(t)[s] \text{ and } s > 0 \ . \tag{10}$$

That is, we write $\mathcal{I}_{FILP} \models C(t)$ iff it can be inferred from $\mathcal{K}$ and $\mathcal{E}$ that t is an instance of concept C to a non-zero degree. Here, $\mathcal{E}$ 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.* $\mathcal{K}' = \mathcal{K} \cup \mathcal{E}$.

Example 4.1. As an example of the kind of learning problems we can face with *Soft*FOIL let us consider the classification of hotels as good ones. To the purpose we assume to have $\mathcal{K}$ (see Example 3.1) as background theory and that:

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

The scores for the examples can be obtained from the five stars system for hotel classification used in TripAdvisor[6] as follows: `[0.0]` (no star), `[0.1]` (½ star), `[0.2]` (1 star), `[0.3]` (1 ½ stars), `[0.4]` (2 stars), `[0.5]` (2 ½ stars), `[0.6]` (3 stars), `[0.7]` (3 ½ stars), `[0.8]` (4 stars), `[0.9]` (4 ½ stars), and `[1.0]` (5 stars). Examples with score greater or equal to `[0.6]` are labeled as positive, the remaining ones as negative.

Also, in order to have a uniform representation of $\mathcal{E}$ w.r.t. $\mathcal{K}$, we transform $\mathcal{E}$ as follows:

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

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

The following GCIs:

$\phi_0 : \top \sqsubseteq \texttt{GoodHotel}$
i.e. Everything is a good hotel
$\phi_1 : \texttt{Hotel} \sqsubseteq \texttt{GoodHotel}$
i.e. Every hotel is a good hotel
$\phi_2 : \texttt{Hotel} \sqcap \exists\texttt{cheapPrice}.\top \sqsubseteq \texttt{GoodHotel}$
i.e. Every hotel having a cheap price is a good hotel
$\phi_3 : \texttt{Hotel} \sqcap \exists\texttt{cheapPrice}.\top \sqcap \exists\texttt{closeTo.Attraction} \sqsubseteq \texttt{GoodHotel}$
i.e. Every hotel having a cheap price and being close to an attraction is a good hotel
$\phi_4 : \texttt{Hotel} \sqcap \exists\texttt{cheapPrice}.\top \sqcap \exists\texttt{closeTo.Park} \sqsubseteq \texttt{GoodHotel}$
i.e. Every hotel having a cheap price and being close to a park is a good hotel
$\phi_5 : \texttt{Hotel} \sqcap \exists\texttt{cheapPrice}.\top \sqcap \exists\texttt{closeTo.Tower} \sqsubseteq \texttt{GoodHotel}$
i.e. Every hotel having a cheap price and being close to a tower is a good hotel

belong to $\mathcal{L}_{\mathcal{H}} = \{\phi | C_i \in \{\top, \texttt{Hotel}, \exists\texttt{cheapPrice}.\top, \exists\texttt{closeTo.Attraction}\}\}$.

[6] `http://www.tripadvisor.com/`

function LEARN-SETS-OF-AXIOMS($\mathcal{K}$, H, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{L}_\mathcal{H}$): $\mathcal{H}$
begin
1. $\mathcal{H} := \emptyset$;
2. **while** $\mathcal{E}^+ \neq \emptyset$ **do**
3. $\quad \phi :=$ LEARN-ONE-AXIOM($\mathcal{K}$, H, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{L}_\mathcal{H}$);
4. $\quad \mathcal{H} := \mathcal{H} \cup \{\phi\}$;
5. $\quad \mathcal{E}^+_\phi := \{e \in \mathcal{E}^+ | \mathcal{K} \cup \phi \models e\}$;
6. $\quad \mathcal{E}^+ := \mathcal{E}^+ \setminus \mathcal{E}^+_\phi$;
7. **endwhile**
8. **return** $\mathcal{H}$
end

Figure 5. *Soft*FOIL: Learning a set of GCI axioms.

4.2. The Algorithm

The solution proposed for the learning problem defined in Section 4.1 is inspired by FOIL. FOIL is a popular ILP algorithm for learning sets of rules which performs a greedy search in order to maximise a gain function [24]. 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 $\mathcal{E}$ (*sequential covering* approach). For inducing a rule, FOIL starts with the most general clause and specialises it step by step by adding literals to the antecedent. The rule is accepted when it does not cover any negative example. In FOIL confidence degrees are computed in the spirit of domain probabilities and allow for evaluating the gain obtained on each step during the search in the space of hypotheses according to the following formula:

$$\text{GAIN}(r', r) = p * (log_2(cf(r')) - log_2(cf(r))) \,, \tag{11}$$

where p is the number of distinct variable bindings with which positive examples covered by the rule r 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 *information gain* tends to favor the literals that offer the best compromise between the confidence degree and the number of examples covered.

In *Soft*FOIL, the learning strategy of FOIL is kept, while necessary changes are made to the technique employed during the candidate generation phase and the heuristics applied during the candidate evaluation phase. These novel features are described in the following.

The function REFINE (step 7. of LEARN-ONE-AXIOM) implements 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 $A' \sqsubseteq A$)

function LEARN-ONE-AXIOM($\mathcal{K}$, H, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{L}_\mathcal{H}$): ϕ
begin
1. $B := \top$;
2. $\phi := \{B \sqsubseteq H\}$;
3. $\mathcal{E}^-_\phi := \mathcal{E}^-$;
4. **while** $\mathcal{E}^-_\phi \neq \emptyset$ **do**
5. $\quad B_{best} := B$;
6. $\quad maxgain := 0$;
7. $\quad \Phi := \text{REFINE}(\phi, \mathcal{L}_\mathcal{H})$
8. $\quad$ **foreach** $\phi' \in \Phi$ **do**
9. $\quad\quad gain := \text{GAIN}(B' \sqsubseteq H, B \sqsubseteq H)$;
10. $\quad\quad$ **if** $gain \geqslant maxgain$ **then**
11. $\quad\quad\quad maxgain := gain$;
12. $\quad\quad\quad B_{best} := B'$;
13. $\quad\quad$ **endif**
14. $\quad$ **endforeach**
15. $\quad \phi := \{B_{best} \sqsubseteq H\}$;
16. $\quad \mathcal{E}^-_\phi := \{e \in \mathcal{E}^- | \mathcal{K} \cup \phi \models e\}$;
17. **endwhile**
18. **return** ϕ
end

Figure 6. *Soft*FOIL: Learning one GCI axiom.

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 so that the generation of redundancies can be limited. Also the language of hypotheses allows to generate "extreme" hypotheses about the target concept such as ϕ_0 and ϕ_1 for `GoodHotel` in Example 4.1. Of course, some of them will be discarded on the basis of their confidence degree during the subsequent candidate evaluation phase.

In the function GAIN (line 9. of LEARN-ONE-AXIOM), in order to account for multiple fuzzy instantiations of fuzzy predicates occurring in the induced GCIs, the confidence degree of an axiom is computed as follows:

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

where

- $P = \{t \mid \mathcal{I}_{FILP} \models C_i(t) \text{ and } H(t)[s] \in \mathcal{E}^+\}$, *i.e.* P is the set of instances for which the implication covers a positive example;
- $D = \{t \mid \mathcal{I}_{FILP} \models C_i(t) \text{ and } H(t)[s] \in \mathcal{E}\}$, *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 GCI, the higher the confidence degree of the axiom. Conversely, the truth degree of negative instances has a limited impact on the confidence degree, as it contributes to defining the set D only.

The confidence degree defined in (12) relies on the notion of entailment reported in (10) which in its turn boils down to the query language described in Section 2.1. In particular, the sets P and D appearing in (12) can be determined easily by submitting appropriate queries to the top-k query answering engine of *Soft*FACTS. More precisely, proving the entailment 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.

Example 4.2. Before showing how hypotheses evaluation is performed in *Soft*FOIL, we illustrate the computation of the confidence degree for ϕ_3. It can be verified that for $\mathcal{K}'$

1. The query

$$\begin{aligned} q_P(\mathtt{h})[\mathtt{s}] \leftarrow \quad & \mathtt{GoodHotel}^+\mathtt{(h)}, \\ & \mathtt{cheapPrice(h,p)[s1]}, \\ & \mathtt{closeTo(h,a)[s2]}, \mathtt{Attraction(a)}, \\ & \mathtt{s} = \min(\mathtt{s1}, \mathtt{s2}) \end{aligned}$$

   has answer set $ans_{\mathcal{K}'}(q_P) = \{\langle \mathtt{h1}, 0.75\rangle, \langle \mathtt{h2}, 0.4\rangle\}$ over $\mathcal{K}'$;

2. The query

$$\begin{aligned} q_D(\mathtt{h})[\mathtt{s}] \leftarrow \quad & \mathtt{GoodHotel(h)}, \\ & \mathtt{cheapPrice(h,p)[s1]}, \\ & \mathtt{closeTo(h,a)[s2]}, \mathtt{Attraction(a)}, \\ & \mathtt{s} = \min(\mathtt{s1}, \mathtt{s2}) \end{aligned}$$

   has answer set $ans_{\mathcal{K}'}(q_D) = \{\langle \mathtt{h1}, 0.75\rangle, \langle \mathtt{h2}, 0.4\rangle, \langle \mathtt{h3}, 0.6\rangle\}$ over $\mathcal{K}'$;

3. Therefore, according to (12), $P = \{\mathtt{h1}, \mathtt{h2}\}$, while $D = \{\mathtt{h1}, \mathtt{h2}, \mathtt{h3}\}$;

4. As a consequence,

$$cf(\phi_3) = \frac{0.75 \Rightarrow 0.6 + 0.4 \Rightarrow 0.8}{3} = \frac{0.6 + 1.0}{3} = 0.5333\,.$$

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

Analogously, we can obtain:

$$cf(\phi_2) = \frac{0.8 \Rightarrow 0.6 + 0.4 \Rightarrow 0.8}{3} = \frac{0.6 + 1.0}{3} = 0.5333 \, .$$

$$cf(\phi_4) = \frac{0.4 \Rightarrow 0.8}{2} = \frac{1.0}{2} = 0.5 \, .$$

$$cf(\phi_5) = \frac{0.8 \Rightarrow 0.6}{1} = 0.6 \, .$$

The function LEARN-ONE-AXIOM starts from ϕ_0 which is then specialized into ϕ_1 by applying the refinement rule which adds an atomic concept, `Hotel`, to the left-hand side of the axiom. As aforementioned, ϕ_0 and ϕ_1 are trivial hypotheses, therefore we can skip the computation steps for them and go ahead. In particular, the algorithm generates ϕ_2 from ϕ_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 ϕ_3, ϕ_4 and ϕ_5 are both possible specializations of ϕ_2, we can now compute the information gain for each of them according to (11):

$$\text{GAIN}(\phi_3, \phi_2) = 2 * (log_2(0.5333) - log_2(0.5333)) = 0.0 \, ,$$

$$\text{GAIN}(\phi_4, \phi_2) = 1 * (log_2(0.5) - log_2(0.5333)) = (-1.0 + 0.907) = -0.093 \, ,$$

$$\text{GAIN}(\phi_5, \phi_2) = 1 * (log_2(0.6) - log_2(0.5333)) = (-0.737 + 0.907) = 0.17 \, ,$$

Among the refinements of ϕ_2, the algorithm will prefer ϕ_5 because it is more informative than ϕ_3 and ϕ_4. Also it does not cover the negative example. Indeed, the literal $\exists$`closeTo.Tower` is a discriminant feature. Therefore, ϕ_5 becomes part of the target theory. Since one positive example is still uncovered, the computation continues within the function LEARN-SETS-OF-AXIOMS aiming at finding a complete theory, *i.e.* a theory which explains all the positive examples.

5. Final Remarks

In this paper, we have described a logic-based computational method, named *Soft*FOIL, which solves the problem (already investigated in [18, 17]) of automatically inducing fuzzy ontology inclusion axioms under CWA where the ontology language is a variant of DL-Lite and 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 GCIs. Notably, fuzziness is captured by the definition of confidence degree reported in (12). Here, the different truth degrees of the variable bindings with which an axiom covers a positive example are taken into account. Only non-zero truth degrees are considered thanks to the new form of entailment reported in (10). The learnable GCIs do not have an explicit score, meaning that the score value is 1. Thanks to the variable-free syntax of DLs, they are highly understandable by humans, *e.g.* ϕ_5 translates in the natural language sentence "a *hotel* having a *cheap price* and being *close to a tower* is a *good hotel*."

FOIL has been chosen as a starting point in our proposal for its simplicity and efficiency. However, it suffers from some limits. One of these limits is the so called "plateau problem" which might occur when adding one literal at a time. It corresponds to the case when two literals do not increase the information

gain when they are considered separately, but they can do it when they are considered together. In order to prevent the "plateau problem", the refinement operator could exploit in a more effective way the knowledge stored in the background theory, *e.g.* by adding more literals instead of a single one. Another limit derives from the sequential covering strategy. Because it performs a greedy search, formulating a sequence of rules without backtracking, FOIL does not guarantee to find the smallest or best set of rules that explain the training examples. Also, learning rules one by one could lead to less and less interesting rules. To reduce the risk of a suboptimal choice at any search step, the greedy search could be replaced by a beam search which maintains a list of k best candidates at each step instead of a single best candidate. Finally, there is a risk of getting trapped into the loop while searching for the best rule. In the case of noise-free training data, FOIL may continue adding new literals to the rule until it covers no negative examples. To handle noisy data, the search could be continued until some tradeoff occurs between rule accuracy, coverage, and complexity. FOIL uses a minimum description length approach to halt the growth of rules, in which new literals are added only when their description length is shorter than the description length of the training data they explain.

Related FOIL-like algorithms for the fuzzy case are reported in the literature [8, 26, 27] but they are not conceived for DL ontologies. A relevant work for the FILP setting is the formal study contributed by [13]. However, it is less promising than our proposal from the practical viewpoint. In the context of DL ontologies, $\mathcal{DL}$-FOIL adapts FOIL to learn crisp OWL DL equivalence axioms under OWA [10]. DL-Learner faces the same learning problem as in $\mathcal{DL}$-FOIL. However, it is not based on FOIL and it applies to ontologies expressed with a fragment of OWL corresponding to the $\mathcal{ALC}$ DL [12]. Note that $\mathcal{ALC}$ and DL-Lite (the DL supported in *Soft*FACTS) are two incomparable DLs. Very recently, an extension of DL-Learner with some of the most up-to-date fuzzy ontology tools has been proposed [14]. Last, the work reported in [16] 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 [20] and entailment in Łukasiewicz $\mathcal{ALC}$ is undecidable [6]. As a final remark concerning the aforementioned misunderstanding about the notion of vagueness, we would like to stress the fact that *Soft*FOIL deeply differs from so-called statistical relational learning algorithms such as [22, 7].

For the future we intend to implement and experiment our method. The empirical validation of the method could suggest directions of improvement of the method towards more effective formulations of, *e.g.*, the information gain function and the refinement operator as well as of the search strategy and the halt conditions employed in LEARN-ONE-AXIOM. Also, we would like to face the problem of computing the score for the induced GCIs. Finally, it can be interesting to analyze the impact of the different implication functions as well as of the OWA on the learning process.

References

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., Eds.: *The Description Logic Handbook: Theory, Implementation and Applications (2nd ed.)*, Cambridge University Press, 2007.

[2] Borgida, A.: On the Relative Expressiveness of Description Logics and Predicate Logics, *Artificial Intelligence*, **82**(1–2), 1996, 353–367.

[3] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R., Ruzzi, M.: Data Integration through DL-Lite$_A$ Ontologies, *Semantics in Data and Knowledge Bases, Third International Workshop,*

SDKB 2008, Nantes, France, March 29, 2008, Revised Selected Papers (K.-D. Schewe, B. Thalheim, Eds.), number 4925 in Lecture Notes in Computer Science, Springer Verlag, 2008.

[4] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data Complexity of Query Answering in Description Logics., *Proc. of the 2005 Int. Workshop on Description Logics* (I. Horrocks, U. Sattler, F. Wolter, Eds.), 147, CEUR-WS.org, 2005.

[5] Calvanese, D., Lenzerini, M., Rosati, R., Vetere, G.: DL-Lite: Practical Reasoning for Rich DLs, *Proc. of the 2004 Int. Workshop on Description Logics* (V. Haarslev, R. Möller, Eds.), 104, CEUR-WS.org, 2004.

[6] Cerami, M., Straccia, U.: *On the Undecidability of Fuzzy Description Logics with GCIs with Lukasiewicz t-norm*, Technical report, Computing Research Repository, 2011.

[7] De Raedt, L., Thon, I.: Probabilistic Rule Learning, *Inductive Logic Programming - 20th International Conference, ILP 2010, Florence, Italy, June 27-30, 2010. Revised Papers* (P. Frasconi, F. A. Lisi, Eds.), 6489, Springer, 2011, ISBN 978-3-642-21294-9.

[8] 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), 2003, 131–152.

[9] Dubois, D., Prade, H.: Possibility Theory, Probability Theory and Multiple-Valued Logics: A Clarification, *Annals of Mathematics and Artificial Intelligence*, **32**(1-4), 2001, 35–66, ISSN 1012-2443.

[10] Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL Concept Learning in Description Logics, *Inductive Logic Programming* (F. Zelezný, N. Lavrač, Eds.), 5194, Springer, 2008.

[11] Hájek, P.: *Metamathematics of Fuzzy Logic*, Kluwer, 1998.

[12] Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL Class Descriptions on Very Large Knowledge Bases, *International Journal on Semantic Web and Information Systems*, **5**(2), 2009, 25–48.

[13] Horváth, T., Vojtás, P.: Induction of Fuzzy and Annotated Logic Programs, *Inductive Logic Programming, 16th International Conference, ILP 2006, Santiago de Compostela, Spain, August 24-27, 2006, Revised Selected Papers* (S. Muggleton, R. P. Otero, A. Tamaddoni-Nezhad, Eds.), 4455, Springer, 2007.

[14] Iglesias, J., Lehmann, J.: Towards Integrating Fuzzy Logic Capabilities into an Ontology-based Inductive Logic Programming Framework, *Proc. of the 11th Int. Conf. on Intelligent Systems Design and Applications*, IEEE Press, 2011.

[15] Klir, G. J., Yuan, B.: *Fuzzy sets and fuzzy logic: theory and applications*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995, ISBN 0-13-101171-5.

[16] Konstantopoulos, S., Charalambidis, A.: Formulating description logic learning as an Inductive Logic Programming task, *Proc. of the 19th IEEE Int. Conf. on Fuzzy Systems*, IEEE Press, 2010.

[17] Lisi, F. A., Straccia, U.: An Inductive Logic Programming Approach to Learning Inclusion Axioms in Fuzzy Description Logics, *Proc. of the 26th Italian Conference on Computational Logic* (F. Fioravanti, Ed.), 810, CEUR-WS.org, 2011.

[18] Lisi, F. A., Straccia, U.: Towards Learning Fuzzy DL Inclusion Axioms, *Fuzzy Logic and Applications - 9th International Workshop, WILF 2011, Trani, Italy, August 29-31,2011. Proceedings* (A. M. Fanelli, W. Pedrycz, A. Petrosino, Eds.), 6857, Springer, 2011, ISBN 978-3-642-23712-6.

[19] Lukasiewicz, T., Straccia, U.: Managing Uncertainty and Vagueness in Description Logics for the Semantic Web, *Journal of Web Semantics*, **6**, 2008, 291–308.

[20] Motik, B., Rosati, R.: A Faithful Integration of Description Logics with Logic Programming, *IJCAI 2007, Proc. of the 20th Int. Joint Conf. on Artificial Intelligence* (M. Veloso, Ed.), 2007.

[21] Nienhuys-Cheng, S., de Wolf, R.: *Foundations of Inductive Logic Programming*, vol. 1228 of *Lecture Notes in Artificial Intelligence*, Springer, 1997.

[22] Ochoa Luna, J. E., Gagliardi Cozman, F.: An Algorithm for Learning with Probabilistic Description Logics, *Proceedings of the Fifth International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington DC, USA, October 26, 2009* (F. Bobillo, P. C. G. da Costa, C. d'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, T. Martin, M. Nickles, M. Pool, P. Smrz, Eds.), 527, CEUR-WS.org, 2009.

[23] Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking Data to Ontologies, *Journal of Data Semantics*, **10**, 2008, 133–173.

[24] Quinlan, J.: Learning Logical Definitions from Relations, *Machine Learning*, **5**, 1990, 239–266.

[25] Reiter, R.: Equality and Domain Closure in First Order Databases, *Journal of ACM*, **27**, 1980, 235–249.

[26] Serrurier, M., Prade, H.: Improving Expressivity of Inductive Logic Programming by Learning Different Kinds of Fuzzy Rules, *Soft Computing*, **11**(5), 2007, 459–466.

[27] Shibata, D., Inuzuka, N., Kato, S., Matsui, T., Itoh, H.: An Induction Algorithm Based on Fuzzy Logic Programming, *Methodologies for Knowledge Discovery and Data Mining, Third Pacific-Asia Conference, PAKDD-99, Beijing, China, April 26-28, 1999, Proceedings* (N. Zhong, L. Zhou, Eds.), 1574, Springer, 1999.

[28] Straccia, U.: Reasoning within Fuzzy Description Logics, *Journal of Artificial Intelligence Research*, **14**, 2001, 137–166.

[29] Straccia, U.: SoftFacts: A Top-k Retrieval Engine for Ontology Mediated Access to Relational Databases, *Proc. of the 2010 IEEE Int. Conf. on Systems, Man and Cybernetics*, IEEE Press, 2010.

[30] Straccia, U.: Top-k Retrieval for Ontology Mediated Access to Relational Databases, *Information Sciences*, **198**, 2012, 1–23.