A System for Learning GCI 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
`francesca.lisi@uniba.it`
[2] ISTI - CNR, Pisa, Italy
`umberto.straccia@isti.cnr.it`

Abstract. Vagueness is inherent to several real world domains and is particularly pervading in those domains where entities could be better described in natural language. In order to deal with vague knowledge, several fuzzy extensions of DLs have been proposed. In this paper, we face the problem of supporting the evolution of DL ontologies under vagueness. Here, we present a system for learning fuzzy GCI axioms from crisp assertions and discuss preliminary experimental results obtained in the tourism application domain.

1 Introduction

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 [21]. 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 [5]. Specifically, under *uncertainty theory* fall all those approaches in which statements 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 [15]).

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 relatively unaddressed. In this paper, we describe a method, named FOIL-$\mathcal{DL}$, for the automated induction of fuzzy *General Concept Inclusion* (GCI) axioms. 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 [17]. 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. Foil-$\mathcal{DL}$ adapts known results in ILP concerning crisp rules to the novel case of fuzzy DL GCIs. More precisely, it adapts the popular rule induction method Foil [18].

The paper is structured as follows. Section 2 is devoted to preliminaries on fuzzy DLs. Section 3 describes the learning problem, the solution strategy and the implementation of Foil-$\mathcal{DL}$. Section 4 introduces a case study in the tourism application domain. Section 5 concludes the paper by discussing limits of the current work, related work and possible directions of future work.

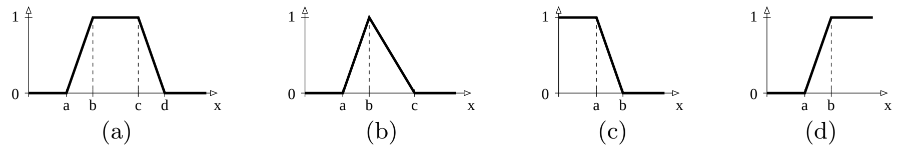


Fig. 1. (a) Trapezoidal function $trz(a, b, c, d)$, (b) triangular function $tri(a, b, c)$, (c) left shoulder function $ls(a, b)$, and (d) right shoulder function $rs(a, b)$.

2 Basics of Fuzzy DLs

We recap here some basic definitions we rely on. We refer the reader to *e.g.* [15], for a more in depth presentation.

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 standard fuzzy set operations conform to $(A \cap B)(x) = \min(A(x), B(x))$, $(A \cup B)(x) = \max(A(x), B(x))$ and $\bar{A}(x) = 1 - A(x)$, while the *inclusion degree* between A and B is defined typically as $deg(A, B) = \frac{\sum_{x \in X}(A \cap B)(x)}{\sum_{x \in X} A(x)}$. 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 functions of fuzzy sets. Although fuzzy sets have a greater expressive power than classical crisp sets, its usefulness depends critically on the 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.* [10, 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 minimum 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 and is also the approach we adopt.

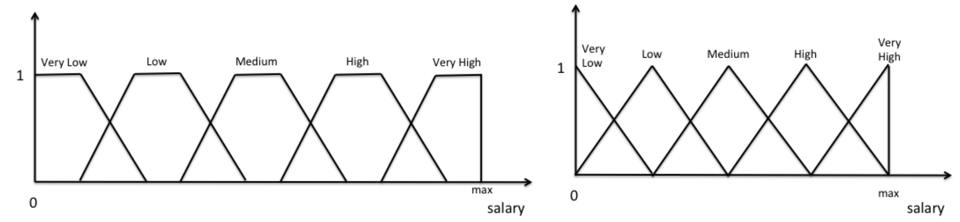


Fig. 2. Fuzzy sets over salaries using trapezoidal or triangular functions.

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 *e.g.* $[0,1]$. This degree is called *degree of truth* of the logical statement ϕ in the interpretation $\mathcal{I}$. For us, *fuzzy statements* have the form $\langle \phi, \alpha \rangle$, where $\alpha \in (0,1]$ and ϕ is a statement, encoding that the degree of truth of ϕ is *greater or equal* α.

A *fuzzy interpretation* $\mathcal{I}$ maps each atomic statement p_i into $[0,1]$ and is then extended inductively to all statements: $\mathcal{I}(\phi \wedge \psi) = \mathcal{I}(\phi) \otimes \mathcal{I}(\psi)$, $\mathcal{I}(\phi \vee \psi) = \mathcal{I}(\phi) \oplus \mathcal{I}(\psi)$, $\mathcal{I}(\phi \rightarrow \psi) = \mathcal{I}(\phi) \Rightarrow \mathcal{I}(\psi)$, $\mathcal{I}(\neg \phi) = \ominus \mathcal{I}(\phi)$, $\mathcal{I}(\exists x.\phi(x)) = \sup_{y \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(y))$, $\mathcal{I}(\forall x.\phi(x)) = \inf_{y \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(y))$, 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][3], whose combination functions are reported in Table 1.

Table 1. Combination functions of various fuzzy logics.

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

Note that the operators for Zadeh logic, namely $\alpha \otimes \beta = \min(\alpha, \beta)$, $\alpha \oplus \beta = \max(\alpha, \beta)$, $\ominus \alpha = 1 - \alpha$ and $\alpha \Rightarrow \beta = \max(1 - \alpha, \beta)$, can be expressed in Łukasiewicz logic. More precisely, $\min(\alpha, \beta) = \alpha \otimes_l (\alpha \Rightarrow_l \beta), \max(\alpha, \beta) = 1 - \min(1 - \alpha, 1 - \beta)$. Furthermore, the implication $\alpha \Rightarrow_{kd} \beta = \max(1 - \alpha, b)$ is called *Kleene-Dienes implication* (denoted $\Rightarrow_{kd}$), while *Zadeh implication* (denoted $\Rightarrow_z$) is the implication $\alpha \Rightarrow_z \beta = 1$ if $\alpha \leq \beta$; 0 otherwise.

[3] Any other continuos t-norm can be obtained as a combination of them.

An *r-implication* is an implication function obtained as the residuum of a continuous t-norm $\otimes$[4], *i.e.*$\alpha \Rightarrow \beta = \max\{\gamma \mid \alpha \otimes \gamma \leq \beta\}$. Note also, that given an r-implication $\Rightarrow_r$, we may also define its related negation $\ominus_r \alpha$ by means of $\alpha \Rightarrow_r 0$ for every $\alpha \in [0,1]$.

The notions of satisfiability and logical consequence are defined in the standard way, where a fuzzy interpretation $\mathcal{I}$ *satisfies* a fuzzy statement $\langle \phi, \alpha \rangle$ or $\mathcal{I}$ is a *model* of $\langle \phi, \alpha \rangle$, denoted as $\mathcal{I} \models \langle \phi, \alpha \rangle$, iff $\mathcal{I}(\phi) \geq \alpha$.

Fuzzy $\mathcal{ALC}(\mathbf{D})$ basics. We recap here the fuzzy variant of the DL $\mathcal{ALC}(\mathbf{D})$ [22].

A *fuzzy concrete domain* or *fuzzy datatype theory* $\mathbf{D} = \langle \Delta^{\mathbf{D}}, \cdot^{\mathbf{D}} \rangle$ consists of a datatype domain $\Delta^{\mathbf{D}}$ and a mapping $\cdot^{\mathbf{D}}$ that assigns to each data value an element of $\Delta^{\mathbf{D}}$, and to every n-ary datatype predicate $\mathbf{d}$ an n-ary fuzzy relation over $\Delta_{\mathbf{D}}$. We will restrict to unary datatypes as usual in fuzzy DLs. Therefore, $\cdot^{\mathbf{D}}$ maps indeed each datatype predicate into a function from $\Delta^{\mathbf{D}}$ to $[0,1]$. Typical examples of datatype predicates $\mathbf{d}$ are the well known membership functions

$$\mathbf{d} := ls(a,b) \mid rs(a,b) \mid tri(a,b,c) \mid trz(a,b,c,d) \mid \ \geq_v \ \mid \ \leq_v \ \mid \ =_v \ ,$$

where *e.g.* $ls(a,b)$ is the left-shoulder membership function and $\geq_v$ corresponds to the crisp set of data values that are greater or equal than the value v.

Now, let $\mathbf{A}$ be a set of *concept names* (also called atomic concepts), $\mathbf{R}$ be a set of *role names*. Each role is either an *object property* or a *datatype property*. The set of *concepts* are built from concept names A using connectives and quantification constructs over object properties R and datatype properties T, as described by the following syntactic rules:

$$C \rightarrow \top \mid \ \bot \ \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid C_1 \rightarrow C_2 \mid \exists R.C \mid \forall R.C \mid \exists T.\mathbf{d} \mid \forall T.\mathbf{d} \ .$$

An *ABox* $\mathcal{A}$ consists of a finite set of assertion axioms. An *assertion* axiom is an expression of the form $\langle a{:}C, \alpha \rangle$ (*concept assertion*, a is an instance of concept C to degree at least α) or of the form $\langle (a_1,a_2){:}R, \alpha \rangle$ (*role assertion*, (a_1,a_2) is an instance of role R to degree at least α), where a, a_1, a_2 are individual names, C is a concept, R is a role name and $\alpha \in (0,1]$ is a truth value.

A *Terminological Box* or *TBox* $\mathcal{T}$ is a finite set of *General Concept Inclusion* (GCI) axioms, where a GCI is of the form $\langle C_1 \sqsubseteq C_2, \alpha \rangle$ (C_1 is a sub-concept of C_2 to degree at least α), where C_i is a concept and $\alpha \in (0,1]$.

We may omit the truth degree α of an axiom; in this case $\alpha = 1$ is assumed.

A *Knowledge Base* (KB) is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$.

Concerning the semantics, let us fix a fuzzy logic. Unlike classical DLs in which an interpretation $\mathcal{I}$ maps *e.g.* a concept C into a set of individuals $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, *i.e.* $\mathcal{I}$ maps C into a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \{0,1\}$ (either an individual belongs to the extension of C or does not belong to it), in fuzzy DLs, $\mathcal{I}$ maps C into a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow [0,1]$ and, thus, an individual belongs to the extension of C to some degree in $[0,1]$, *i.e.* $C^{\mathcal{I}}$ is a fuzzy set. Specifically, a *fuzzy interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a nonempty (crisp) set $\Delta^{\mathcal{I}}$ (the *domain*) and of a *fuzzy interpretation function* $\cdot^{\mathcal{I}}$ that assigns: *(i)* to each atomic concept A a function $A^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \rightarrow [0,1]$; *(ii)* to each object property R a function $R^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0,1]$; *(iii)* to each data type property T a function $T^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}} \rightarrow [0,1]$; *(iv)* to each individual a an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$; and *(v)* to each concrete value v an

[4] Note that Łukasiewicz, Gödel and Product implications are r-implications, while Kleene-Dienes implication is not.

element $v^{\mathcal{I}} \in \Delta^{\mathbf{D}}$. Now, a fuzzy interpretation function is extended to concepts as specified below (where $x \in \Delta^{\mathcal{I}}$):

$$
\begin{array}{l}
\bot^{\mathcal{I}}(x) = 0, \quad \top^{\mathcal{I}}(x) = 1, \\
(C \sqcap D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \otimes D^{\mathcal{I}}(x), \quad (C \sqcup D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \oplus D^{\mathcal{I}}(x), \\
(\neg C)^{\mathcal{I}}(x) = \ominus C^{\mathcal{I}}(x), \quad (C \to D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x), \\
(\forall R.C)^{\mathcal{I}}(x) = \inf_{y \in \Delta^{\mathcal{I}}}\{R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)\}, \quad (\exists R.C)^{\mathcal{I}}(x) = \sup_{y \in \Delta^{\mathcal{I}}}\{R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)\}, \\
(\forall T.\mathbf{d})^{\mathcal{I}}(x) = \inf_{y \in \Delta^{\mathbf{D}}}\{T^{\mathcal{I}}(x,y) \Rightarrow \mathbf{d}^{\mathbf{D}}(y)\}, \quad (\exists T.\mathbf{d})^{\mathcal{I}}(x) = \sup_{y \in \Delta^{\mathbf{D}}}\{T^{\mathcal{I}}(x,y) \otimes \mathbf{d}^{\mathbf{D}}(y)\} .
\end{array}
$$

Hence, for every concept C we get a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$.

The *satisfiability of axioms* is then defined by the following conditions: *(i)* $\mathcal{I}$ satisfies an axiom $\langle a{:}C, \alpha\rangle$ if $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq \alpha$; *(ii)* $\mathcal{I}$ satisfies an axiom $\langle (a,b){:}R, \alpha\rangle$ if $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \geq \alpha$; *(iii)* $\mathcal{I}$ satisfies an axiom $\langle C \sqsubseteq D, \alpha\rangle$ if $(C \sqsubseteq D)^{\mathcal{I}} \geq \alpha$ where[5] $(C \sqsubseteq D)^{\mathcal{I}} = \inf_{x \in \Delta^{\mathcal{I}}}\{C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)\}$. $\mathcal{I}$ is a model of $\mathcal{K} = \langle \mathcal{A}, \mathcal{T}\rangle$ iff $\mathcal{I}$ satisfies each axiom in $\mathcal{K}$. We say that $\mathcal{K}$ *entails* axiom τ, denoted $\mathcal{K} \models \tau$, if any model of $\mathcal{K}$ satisfies τ. The *best entailment degree* of τ of the form $C \sqsubseteq D$, $a{:}C$ or $(a,b){:}R$, denoted $bed(\mathcal{K}, \tau)$, is defined as $bed(\mathcal{K}, \tau) = \sup\{\alpha \mid \mathcal{K} \models \langle \tau, \alpha\rangle\}$.

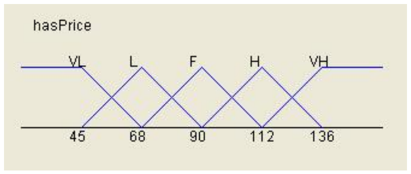


Fig. 3. Fuzzy concepts derived from the datatype property `hasPrice`.

Example 1. Let us consider the following axiom

$$\langle \exists hasPrice.High \sqsubseteq GoodHotel, 0.569\rangle ,$$

where *hasPrice* is a datatype property whose values are measured in euros and the price concrete domain has been automatically fuzzified as illustrated in Figure 3. Now, it can be verified that for hotel *verdi*, whose room price is 105 euro, *i.e.* we have the assertion $verdi{:}\exists hasPrice. =_{105}$ in the KB, we infer under Product logic that[6] $\mathcal{K} \models \langle verdi{:}GoodHotel, 0.18\rangle$.

3 Learning fuzzy DL axioms with Foil-$\mathcal{DL}$

The problem statement. The problem considered in this paper concerns the automated induction of fuzzy DL GCI axioms providing a sufficient condition for a given atomic concept H. It can be cast as a rule learning problem, provided that positive and negative examples of H are available. This problem can be formalized as follows.

Given:

– a consistent $\mathcal{DL}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ (the *background theory*);

[5] However, note that under Zadeh logic $\sqsubseteq$ is interpreted as $\Rightarrow_z$ and not as $\Rightarrow_{kd}$.
[6] $0.18 = 0.318 \cdot 0.569$, where $0.318 = tri(90, 112, 136)(105)$.

– an atomic concept H (the *target concept*);
– a set $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ of crisp concept assertions labelled as either positive or negative examples for H (the *training set*);
– a set $\mathcal{L}_\mathcal{H}$ of fuzzy GCIs (the *language of hypotheses*)

the goal is to find a set $\mathcal{H} \subset \mathcal{L}_\mathcal{H}$ (a *hypothesis*) such that:

Completeness. $\forall e \in \mathcal{E}^+, \mathcal{K} \cup \mathcal{H} \models e$, and
Consistency. $\forall e \in \mathcal{E}^-, \mathcal{K} \cup \mathcal{H} \not\models e$.

Here we assume that $\mathcal{K} \cap \mathcal{E} = \emptyset$. Also, the language $\mathcal{L}_\mathcal{H}$ is given implicitly by means of syntactic restrictions over a given alphabet. In particular, the alphabet underlying $\mathcal{L}_\mathcal{H}$ is a subset of the alphabet for the language $\mathcal{L}_\mathcal{K}$ of the background theory. However, $\mathcal{L}_\mathcal{H}$ differs from $\mathcal{L}_\mathcal{K}$ as for the form of axioms. 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, *i.e.* has a model. An axiom $\phi \in \mathcal{L}_\mathcal{H}$ *covers* an example $e \in \mathcal{E}$ iff $\mathcal{K} \cup \{\phi\} \models e$.

The training examples. Given the target concept H, the training set $\mathcal{E}$ consists of concept assertions of the form

$$H(a) \tag{1}$$

where a is an individual occurring in $\mathcal{K}$. In this paper, the training examples are crisp. Also, $\mathcal{E}$ is split into $\mathcal{E}^+$ and $\mathcal{E}^-$. Note that, under OWA, $\mathcal{E}^-$ consists of all those individuals which can be proved to be instance of $\neg H$. However, $\mathcal{E}^-$ can be deduced under CWA by collecting the individuals which cannot be proved to be instance of H.

The language of hypotheses. Given the target concept H, the hypotheses to be induced are fuzzy GCIs of the form

$$B \sqsubseteq H \ , \tag{2}$$

where the left-hand side is defined according to the following syntax

$$B \longrightarrow \top \mid A \mid \exists R.B \mid \exists T.\mathbf{d} \mid B_1 \sqcap B_2 \ . \tag{3}$$

Note that the language $\mathcal{L}_\mathcal{H}$ generated by this $\mathcal{EL}(\mathbf{D})$ syntax is potentially infinite due, *e.g.*, to the nesting of existential restrictions yielding to complex concept expressions such as $\exists R_1.(\exists R_2 \ldots .(\exists R_n.(C))\ldots)$. The language can be made finite by imposing further restrictions on the generation process such as the maximal number of conjuncts and the depth of existential nestings allowed in the left-hand side. Also, note that the learnable GCIs do not have an explicit truth degree. However, even if $\mathcal{K}$ is a crisp $\mathcal{DL}$ KB, the possible occurrence of fuzzy concrete domains in expressions of the form $\exists T.\mathbf{d}$ in the left-hand side of a fuzzy GCI of the form (2) may imply both that $bed(\mathcal{K}, B \sqsubseteq H) \not\in \{0,1\}$ and $bed(\mathcal{K}, a{:}B) \not\in \{0,1\}$. Furthermore, as we shall see later on, once we have learned a fuzzy GCI $B \sqsubseteq H$, we attach to it a truth degree that is obtained by computing the confidence degree by means of the cf function (see Eq (5)), which may be seen as the fuzzy set inclusion degree (see Section 2) between the fuzzy set represented by concept

```
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.     $\phi :=$ LEARN-ONE-AXIOM($\mathcal{K}$, $H$, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{L}_\mathcal{H}$);
4.     $\mathcal{H} := \mathcal{H} \cup \{\phi\}$;
5.     $\mathcal{E}^+_\phi := \{e \in \mathcal{E}^+ | \mathcal{K} \cup \phi \models e\}$;
6.     $\mathcal{E}^+ := \mathcal{E}^+ \setminus \mathcal{E}^+_\phi$;
7. endwhile
8. return $\mathcal{H}$
end
```

Fig. 4. FOIL-$\mathcal{DL}$: Learning a set of GCI axioms.

B and the (crisp) set represented by concept H. Finally, note that the syntactic restrictions of $\mathcal{L}_\mathcal{H}$ allow for a straightforward translation of the inducible axioms into rules of the kind "if x is a C_1 and ... and x is a C_n then x is an H", which corresponds to the usual pattern in fuzzy rule induction (in our case, $B \sqsubseteq H$ is seen as a rule "if B then H") .

The solution strategy. The solution proposed for the learning problem defined in Section 3 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 [18].

In FOIL-$\mathcal{DL}$, the learning strategy of FOIL (*i.e.* the so-called *sequential covering* approach) is kept. The function LEARN-SETS-OF-AXIOMS (reported in Figure 4) carries on inducing axioms until all positive examples are covered. When an axiom is induced, the positive examples covered by the axiom are removed from $\mathcal{E}$. In order to induce an axiom, the function LEARN-ONE-AXIOM (reported in Figure 5) starts with the most general axiom (*i.e.* $\top \sqsubseteq H$) and specializes it by applying the refinement rules implemented in the function REFINE (step 7.). The iterated specialization of the axiom continues until the axiom does not cover any negative example and its *confidence degree* is greater than a fixed threshold (θ). The confidence degree of axioms being generated with REFINE allows for evaluating the *information gain* obtained on each refinement step by calling the function GAIN (step 9.).

Of course, we need to adapt the functions REFINE and GAIN developed for FOIL to FOIL-$\mathcal{DL}$, which we address in the next two subsections.

The refinement operator. The function REFINE implements a specialization operator, *i.e.* an operator for traversing the hypotheses space top down, with the following refinement rules:

Add_A adds an atomic concept A
$Add_{\exists R.\top}$ adds a complex concept $\exists R.\top$ by existential role restriction
$Add_{\exists T.\mathbf{d}}$ adds a complex concept $\exists T.\mathbf{d}$ by existential role restriction
$Subst_A$ replaces an atomic concept A with another atomic concept A' s.t. $A' \sqsubseteq A$

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

Fig. 5. FOIL-$\mathcal{DL}$: Learning one GCI axiom.

At each refinement step (*i.e.* at each call of REFINE), the rules are applied first to the left-hand side of the axiom being specialized and then recursively to the range of all the conjuncts defined with existential role restriction. For example, let us consider that H is the target concept, A, A', B, R, R', T are concepts and roles occurring in $\mathcal{K}$, and $A' \sqsubseteq A$ holds in $\mathcal{K}$. Under these assumptions, the axiom $\exists R.B \sqsubseteq H$ is specialized into the following axioms:

– $A \sqcap \exists R.B \sqsubseteq H$, $B \sqcap \exists R.B \sqsubseteq H$, $A' \sqcap \exists R.B \sqsubseteq H$;
– $\exists R'.\top \sqcap \exists R.B \sqsubseteq H$, $\exists T.\mathbf{d} \sqcap \exists R.B \sqsubseteq H$;
– $\exists R.(B \sqcap A) \sqsubseteq H$, $\exists R.(B \sqcap A') \sqsubseteq H$;
– $\exists R.(B \sqcap \exists R.\top) \sqsubseteq H$, $\exists R.(B \sqcap \exists R'.\top) \sqsubseteq H$, $\exists R.(B \sqcap \exists T.\mathbf{d}) \sqsubseteq H$.

Note that a specialization operator reduces the number of examples covered by a GCI. The aim of a refinement is to reduce the number of covered negative examples, while still keeping some covered positive examples.

The heuristic. The function GAIN implements the criterion for selecting the best candidate at each refinement step according to the following formula:

$$\text{GAIN}(\phi', \phi) = p * (log_2(cf(\phi')) - log_2(cf(\phi))) \;, \tag{4}$$

where p is the number of positive examples covered by the axiom ϕ that are still covered by ϕ'. Thus, the gain is positive iff ϕ' is more informative in the sense of Shannon's information theory (*i.e.* iff the confidence degree increases). If there are some refinements, which increase the confidence degree, the function GAIN tends to favour those that offer the best compromise between the confidence degree and the number of examples covered.

The confidence degree of an axiom ϕ is computed as a sort of fuzzy set inclusion degree (see Section 2) between the fuzzy set represented by concept B and the (crisp) set represented by concept H and is as follows:

$$cf(\phi) = cf(B \sqsubseteq H) = \frac{\sum_{a \in P} bed(\mathcal{K}, a{:}B)}{|D|} \tag{5}$$

where

- D is the set of individuals occurring in $\mathcal{E}^+_\phi \cup \mathcal{E}^-_\phi$ such that $bed(\mathcal{K}, a{:}B) > 0$.
- P is the set of individuals occurring in $\mathcal{E}^+_\phi$ such that $bed(\mathcal{K}, a{:}B) > 0$ [7].

We remind the reader that $bed(\mathcal{K}, a{:}B)$ denotes the best entailment degree of the concept assertion $a{:}B$ w.r.t. $\mathcal{K}$ (see Section 2). Also, note that, as pointed out in Section 3, the possible occurrence of expressions of the form $\exists T.\mathbf{d}$ in B may imply that $bed(\mathcal{K}, a{:}B) \notin \{0, 1\}$.

The implementation. A variant of Foil-$\mathcal{DL}$ has been implemented in the *fuzzyDL-Learner*[8] system. Notably, fuzzy GCIs in $\mathcal{L}_\mathcal{H}$ are interpreted under Gödel semantics, while the background theory and the training set are represented in crisp DLs. As $\mathcal{K}$ is crisp, we have used a classical DL reasoner, together with a specialised code, to compute the confidence degree of fuzzy GCI. For instance, $bed(\mathcal{K}, a{:}\exists T.\mathbf{d})$, can be computed from the derived T-fillers v of a, and applying v to the fuzzy membership function of $\mathbf{d}$. The examples covered by a GCI, and, thus, the entailment tests in Learn-Sets-of-Axioms and Learn-One-Axiom, have been determined in a similar way.

The implementation of Foil-$\mathcal{DL}$ features several optimizations wrt the solution strategy presented in Section 3. Notably, the search in the hypothesis space can be optimized by enabling a backtracking mode. This option allows to overcome one of the main limits of Foil, *i.e.* 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 can be replaced in Foil-$\mathcal{DL}$ by a *beam search* which maintains a list of k best candidates at each step instead of a single best candidate.

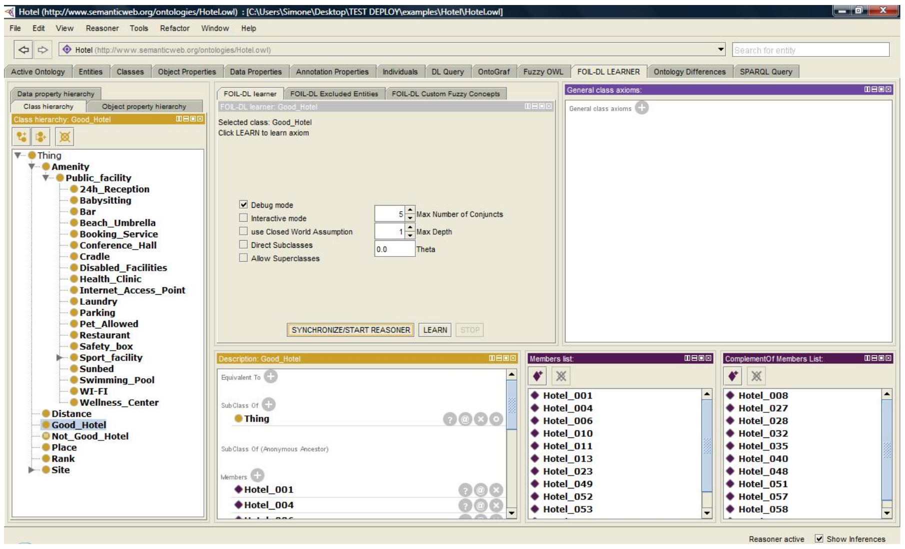
Additionally, to guarantee termination, we provide two parameters to limit the search space: namely, the maximal number of conjuncts and the maximal depth of existential nestings allowed in a fuzzy GCI. In fact, the computation may end without covering all positive examples.

Two graphical user interfaces are available for Foil-$\mathcal{DL}$: One is a stand-alone Java application, the other is a tab widget plug-in for the ontology editor Protégé[9] (release 4.2). The latter is shown in Figure 6.

[7] Note that for individuals $a \in P$, $\mathcal{K} \models a{:}H$ holds and, thus, $bed(\mathcal{K}, a{:}B \sqcap H) = bed(\mathcal{K}, a{:}B)$.

[8] `http://straccia.info/software/FuzzyDL-Learner`

[9] `http://protege.stanford.edu/`


Hotel (http://www.semanticweb.org/ontologies/Hotel.owl) : [C:\Users\Simone\Desktop\TEST DEPLOY\examples\Hotel\Hotel.owl]
File Edit View Reasoner Tools Refactor Window Help
Active Ontology | Entities | Classes | Object Properties | Data Properties | Annotation Properties | Individuals | DL Query | OntoGraf | Fuzzy OWL | FOIL-DL LEARNER | Ontology Differences | SPARQL Query
Data property hierarchy | Class hierarchy | Object property hierarchy
Class hierarchy: Good_Hotel
Thing
Amenity
Public_facility
24h_Reception
Babysitting
Bar
Beach_Umbrella
Booking_Service
Conference_Hall
Cradle
Disabled_Facilities
Health_Clinic
Internet_Access_Point
Laundry
Parking
Pet_Allowed
Restaurant
Safety_box
Sport_facility
Sunbed
Swimming_Pool
WI-FI
Wellness_Center
Distance
Good_Hotel
Not_Good_Hotel
Place
Rank
Site
FOIL-DL learner | FOIL-DL Excluded Entities | FOIL-DL Custom Fuzzy Concepts
General class axioms
SYNCHRONIZE/START REASONER | LEARN | STOP
Description: Good_Hotel
Equivalent To
SubClass Of
Thing
SubClass Of (Anonymous Ancestor)
Members
Hotel_001
Hotel_004
Members list:
Hotel_001
Hotel_004
Hotel_006
Hotel_010
Hotel_011
Hotel_013
Hotel_023
Hotel_049
Hotel_052
Hotel_053
ComplementOf Members List:
Hotel_008
Hotel_027
Hotel_028
Hotel_032
Hotel_035
Hotel_040
Hotel_048
Hotel_051
Hotel_057
Hotel_058
Reasoner active
Show Inferences

FOIL-DL learner: Good_Hotel

Selected class: Good_Hotel
Click LEARN to learn axiom

- [✓] Debug mode
- [ ] Interactive mode
- [ ] use Closed World Assumption
- [ ] Direct Subclasses
- [ ] Allow Superclasses

Max Number of Conjuncts: 5
Max Depth: 1
Theta: 0.0

Fig. 6. User interface of FOIL-$\mathcal{DL}$ as tab widget plug-in for Protégé.

4 A Case Study in the Tourism Domain

In order to test the validity and the usefulness of FOIL-$\mathcal{DL}$ on a real-world application, we have considered a case study in the tourism domain. More precisely, we have focused on the task of hotel finding because it can be reformulated as a classification problem solvable with FOIL-like algorithms. To the purpose, we have built an ontology, named `Hotel.owl`[10], which models the meaningful entities of the domain in hand (see Figure 6, further details can be found in the appendix). It has the DL expressivity of $\mathcal{ALCHOF}(\mathbf{D})$ and consists of 8000 axioms, 74 classes, 4 object properties, and 2 data properties. The ontology has been populated with 1504 individuals concerning tourism in the city of Pisa. In particular, data about hotels as well as graded hotel judgements from users have been automatically extracted from the web site of Trip Advisor[11] whereas the distances of hotels from sites of interest have been computed by means of Google Maps[12] API. Then, one may set a learning problem with the class `Good_Hotel` as target concept and ask FOIL-$\mathcal{DL}$ to induce axioms - such as the graded GCI reported in Example 1 - from positive and negative examples of `Good_Hotel`, which allow for discriminating good hotels from bad ones.

For more details on this case study we refer the reader to the Appendix.

[10] `http://gaia.isti.cnr.it/~straccia/software/FOIL-DL/download/FOIL-DL/examples/Hotel/Hotel.owl`
[11] `http://www.tripadvisor.com`
[12] `http://maps.google.com/`

5 Conclusions and future work

In this paper, we have described a method, named FOIL-$\mathcal{DL}$, which solves the problem of automatically inducing fuzzy $\mathcal{EL}(\mathbf{D})$ GCI axioms from crisp DL assertions. 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 (5). Here, the different truth degrees of the variable bindings with which an axiom covers a positive example are taken into account. Also, thanks to the variable-free syntax of DLs, the learnable GCIs are highly understandable by humans, *e.g.* the axiom reported in Example 1 translates into the natural language sentence "a *hotel* having a *high price* is a *good hotel*."

Related FOIL-like algorithms for the fuzzy case are reported in the literature [4,19,20] but they are not conceived for DL ontologies. In the context of DL ontologies, DL-FOIL adapts FOIL to learn crisp OWL DL equivalence axioms under OWA [6]. DL-Learner supports the same learning problem as in DL-FOIL but implements algorithms that are not based on FOIL [8]. Likewise, Lehmann and Haase [12] propose a refinement operator for concept learning in $\mathcal{EL}$ (implemented in the ELTL algorithm within DL-Learner) whereas Chitsaz *et al.* [3] combine a refinement operator for $\mathcal{EL}^{++}$ with a reinforcement learning algorithm. However, both works deal only with crip ontologies. Very recently, an extension of DL-Learner with some of the most up-to-date fuzzy ontology tools has been proposed [9]. Notably, it can learn fuzzy OWL DL equivalence axioms from FuzzyOWL 2 ontologies[13] by interfacing the *fuzzyDL* reasoner [1]. However, it has been tested only on a toy ontology[14] with crisp training examples. 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 mapping from fuzzy DLs to LP is incomplete [16] and entailment in Łukasiewicz $\mathcal{ALC}$ is undecidable [2]. A previous investigation of the problem considered in the present paper can be found in [13,14]. However, the resulting method (named *Soft*FOIL) provides a different solution from FOIL-$\mathcal{DL}$ as for the knowledge representation language, the confidence degree computation, the refinement operator and the heuristic. Finally, unlike FOIL-$\mathcal{DL}$, *Soft*FOIL has not been implemented.

For the future, we intend to conduct a more extensive empirical validation of the system which 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, the choice of the t-norm, so as to investigate other fuzzy GCI learning algorithms based on *e.g.* fuzzy Decision Trees [23]. Eventually, we will investigate about learning fuzzy GCI axioms from FuzzyOWL 2 ontologies, by coupling the learning algorithm to the *fuzzyDL* reasoner, instead of learning from crisp OWL 2 data by using a classical DL reasoner. Only then, a direct comparative evaluation with DL-Learner will be possible.

---

[13] `http://www.straccia.info/software/FuzzyOWL`

[14] `http://wiki.aksw.org/Projects/DLLearner/fuzzyTrains`

A Evaluation Data

A.1 The ontology `Hotel.owl`

According to Protégé, the ontology metrics for `Hotel.owl` are the following:

- metrics:
  - axioms: 8000;
  - logical axiom count: 6309;
  - class count: 74
  - object property count: 4;
  - data property count: 2;
  - individual count: 1504;
  - DL expressivity: ALCHOF(D).
- class axioms:
  - subClassOf axioms count: 71;
  - equivalentClasses axioms count: 6;
  - disjointClasses axioms count: 1;
  - hidden GCI count:6.
- object property axioms:
  - objectPropertyDomain axioms count: 4;
  - objectPropertyRange axioms count:4.
- data property axioms:
  - functionalDataProperty axioms count: 2;
  - dataPropertyDomain axioms count: 2;
  - dataPropertyRange axioms count: 2.
- individual axioms:
  - classAssertion axioms count: 1866;
  - objectPropertyAssertion axioms count: 2876;
  - dataPropertyAssertion axioms count: 1475.

Classes. The classes forming the terminology of `Hotel.owl` are shown in Figure 7, 8 and 9. The main concepts model the sites of interest (class `Site`), the places where sites are located (class `Place`), the services offered by hotels (class `Amenity`), the ranks assigned to hotels in the star classification (class `Rank`), and the distances between sites (class `Distance`). The classes `Good_Hotel` and `Not_Good_Hotel` represent the target concept and its complement. Sites of interest in the tourism application domain include accommodations such as hotels (class `Accomodation`), attractions such as parks (class `Attraction`), stations of transportation means such as airports (class `Station`), and civic facilities such as hospitals (class `Civic`).

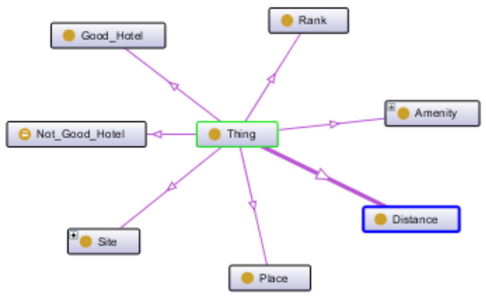


Fig. 7. Classes modeling the main entities in `Hotel.owl`.

Object and data properties. The object properties in `Hotel.owl` are:

– `hasAmenity` (with domain `Hotel` and range `Amenity`) models the relationship between hotels and the services offered;
– `hasRank` (with domain `Hotel` and range `Rank`) models the rank assigned to a hotel;
– `hasDistance` (with domain `Site` and range `Distance`) models the relationship between a site and a distance;
– `isDistanceFor` (with domain `Distance` and range `Site`) models the relationship between a distance and the two sites.

The data properties are:

– `hasPrice` (with domain `Accommodation` and range `integer`) is the average price of a room;
– `hasValue` (with domain `Distance` and range `double`) is the numerical value of the distance between two sites.

Note that the numerical value of the distance between two sites would be better modeled as attribute of a ternary relation. However, only binary relations can be represented in OWL. The concept `Distance` and the properties `hasDistance`, `isDistanceFor` and `hasValue` are necessary to simulate a ternary relation by means of binary relations.

Individuals. The individuals occurring in `Hotel.owl` refer to the case of Pisa. In particular, 59 instances of `Hotel` have been extracted from TripAdvisor. Information about the rank, the amenities and the average room price has been added in the ontology for each of these instances. Out of the 59 hotels, 12 with a higher percentage of positive feedback have been classified as instances of `Good_Hotel` whereas 11 with a lower percentage have been classified as instances of `Not_Good_Hotel`. Also, further 24 instances of `Site` have been created and distributed among the classes under `Attraction`, `Civic` and `Station`. Finally, 1416 distances (instances of `Distance`) between the accommodations and the sites of interest have been measured in km and computed by means of Google Maps API.

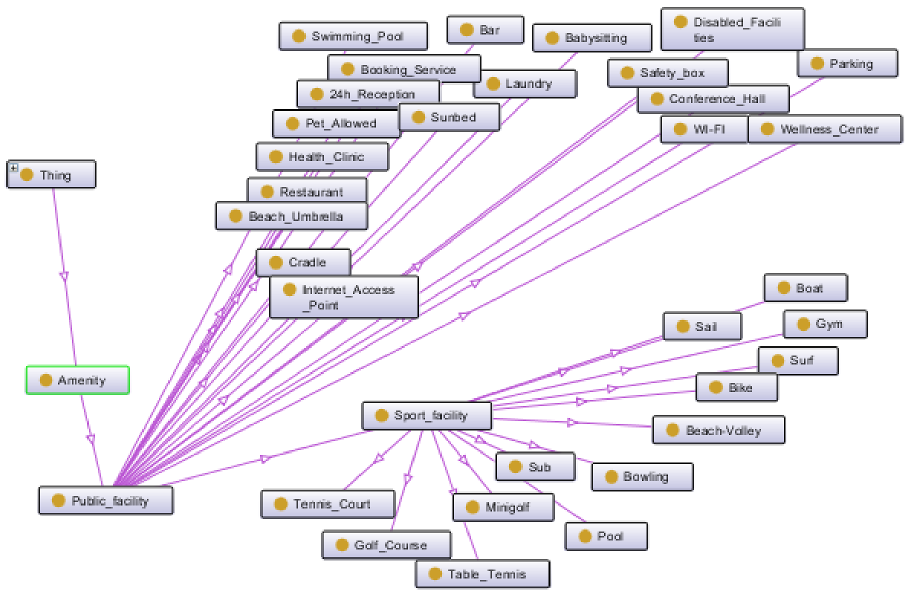


Fig. 8. Classes modeling hotel amenities in `Hotel.owl`.

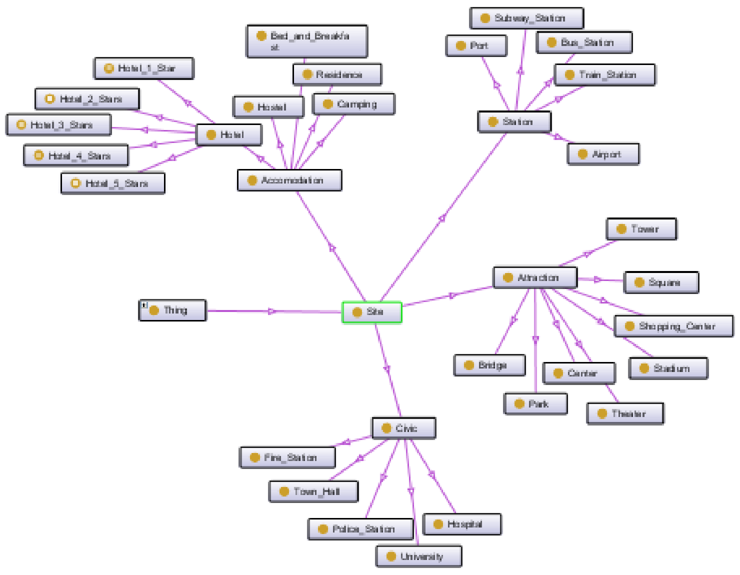


Fig. 9. Classes modeling sites in `Hotel.owl`.

Two further remarks concern the modeling of the instance level of `Hotel.owl`. First, no instance of the class `Amenity` has been created because it is not necessary for our purposes. However, hotel amenities can be implicitely declared. For instance, the fact that a hotel, say *hotel_10*, offers a laundry service is modeled by means of an axiom stating that *hotel_10* is instance of the class $\exists$`hasAmenity.Laundry`. Second, ranks are modeled as individuals (`1_Star`, ..., `5_Stars`). Further classes have been created in order to classify hotels according to the star ranking system. For instance, 3-star hotels are modeled with the class `Hotel_3_Stars` $\equiv$ $\exists$`hasRank`.{`3_stars`} defined by means of an equivalence axiom. This modeling choice allows to overcome the limit of FOIL-$\mathcal{DL}$ in dealing directly with expressions like $\exists$`hasRank`.{`3_stars`}.

A.2 The experiments on `Hotel.owl`

First trial. The first experiment conducted by running FOIL-$\mathcal{DL}$ on `Hotel.owl` is summarized in the log reported in Figure 10. Here, the target concept is `Good_Hotel` and the search for hypotheses is conducted under OWA by using Pellet as a DL reasoner. All the classes, object properties and data properties occurring in `Hotel.owl` are considered to be part of the alphabet of the language of hypotheses. However, syntactic restrictions are imposed on the form of the learnable GCI axioms. More precisely, conjunctions can have at most 5 conjuncts and at most 2 levels of nesting are allowed in existential role restrictions. The membership functions for fuzzy concepts derived from the data properties `hasPrice` and `hasValue` in this trial are shown in Figure 3 and Figure 11(a), respectively. The results obtained for this configuration of FOIL-$\mathcal{DL}$ suggest the existence of user profiles, *e.g.* families and disabled people. Note that no axiom has been induced which encompasses knowledge about the distance of the accommodation from sites of interest.

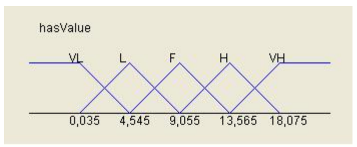
Second trial. In the second experiment, the configuration of FOIL-$\mathcal{DL}$ remains unchanged except for the alphabet underlying the language of hypotheses and the definition of membership functions for the fuzzy concepts (see log reported in Figure 12). More precisely, the use of the object property `hasAmenity` is forbidden. Also, the fuzzification of the data property `hasValue` is more reasonable for a foot distance. Here, a very low distance does not exceed 900 meters, an average distance is about 1500 meters, and so on, as illustrated in Figure 11(b). The axioms learned by FOIL-$\mathcal{DL}$ suggest that closeness to stations of transportation means is a desirable feature when choosing a hotel.

References

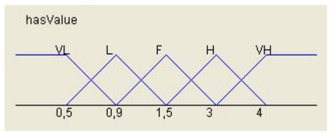
1. Bobillo, F., Straccia, U.: fuzzyDL: An expressive fuzzy description logic reasoner. In: 2008 International Conference on Fuzzy Systems (FUZZ-08). pp. 923–930. IEEE Computer Society (2008)
2. Cerami, M., Straccia, U.: On the (un)decidability of fuzzy description logics under Łukasiewicz t-norm. Information Sciences 227, 1–21 (2013)

```
----------------------------------------------------

http://www.semanticweb.org/ontologies/Hotel.owl

Parameters:
 - ontology: .\examples\Hotel\Hotel.owl
 - concept: Good_Hotel
 - debug_mode: true
 - cwa_mode: false
 - direct_subclasses: false
 - backtrack_mode: false
 - max_conjuncts: 5
 - max_depth: 2
 - theta: 0.0
 - reasoner: PELLET
 - skip_classes: []
 - skip_properties: []
 - skip_data_properties: []

FuzzyConcepts:
 - hasPrice_veryhigh: hasPrice, rightShoulder(112,136)
 - hasPrice_low: hasPrice, triangular(45,68,90)
 - hasPrice_fair: hasPrice, triangular(68,90,112)
 - hasPrice_high: hasPrice, triangular(90,112,136)
 - hasPrice_verylow: hasPrice, leftShoulder(45,68)
 - hasValue_low: hasValue, triangular(0.035,4.545,9.055)
 - hasValue_veryhigh: hasValue, rightShoulder(13.565,18.075)
 - hasValue_fair: hasValue, triangular(4.545,9.055,13.565)
 - hasValue_high: hasValue, triangular(9.055,13.565,18.075)
 - hasValue_verylow: hasValue, leftShoulder(0.035,4.545)


Confidence         Axiom
   1,000      Hostel subclass of Good_Hotel
   1,000      hasPrice_veryhigh subclass of Good_Hotel
   0,569      hasPrice_high subclass of Good_Hotel
   0,286      hasAmenity some (24h_Reception) and hasAmenity some (Disabled_Facilities)
                  and hasPrice_low subclass of Good_Hotel
   0,282      hasAmenity some (Babysitting) and hasRank some (Rank)
                  and hasPrice_fair subclass of Good_Hotel
   0,200      Hotel_1_Star subclass of Good_Hotel

Running time 76 sec and 786 msec of which:
287 msec for subclasses retrieval
 61  sec  for individuals retrieval
 14  sec  for instance checking
  1  sec  for the algorithm running

----------------------------------------------------
```

Fig. 10. Log of the first trial of Foil-$\mathcal{DL}$ on `Hotel.owl` (target concept: `Good_Hotel`).



(a)



(b)

Fig. 11. Membership functions for fuzzy concepts derived from the data property `hasValue` in (a) the first trial and (b) the second trial of Foil-$\mathcal{DL}$ on `Hotel.owl`.

```
----------------------------------------------------

http://www.semanticweb.org/ontologies/Hotel.owl

Parameters:
 - ontology: .\examples\Hotel\Hotel.owl
 - concept: Good_Hotel
 - debug_mode: true
 - cwa_mode: false
 - direct_subclasses: false
 - backtrack_mode: false
 - max_conjuncts: 5
 - max_depth: 2
 - theta: 0.0
 - reasoner: PELLET
 - skip_classes: []
 - skip_properties: [hasAmenity]
 - skip_data_properties: []

FuzzyConcepts:
 - hasPrice_veryhigh: hasPrice, rightShoulder(112,136)
 - hasPrice_low: hasPrice, triangular(45,68,90)
 - hasPrice_fair: hasPrice, triangular(68,90,112)
 - hasPrice_high: hasPrice, triangular(90,112,136)
 - hasPrice_verylow: hasPrice, leftShoulder(45,68)
 - hasValue_low: hasValue, triangular(0.5,0.9,1.5)
 - hasValue_fair: hasValue, triangular(0.9,1.5,3.0)
 - hasValue_high: hasValue, triangular(1.5,3.0,4.0)
 - hasValue_veryhigh: hasValue, rightShoulder(3.0,4.0)
 - hasValue_verylow: hasValue, leftShoulder(0.5,0.9)


Confidence      Axiom
   1,000      Hostel subclass of Good_Hotel
   1,000      hasPrice_veryhigh subclass of Good_Hotel
   0,739      hasDistance some (isDistanceFor some (Bus_Station) and hasValue_low)
                  and hasDistance some (isDistanceFor some (Town_Hall) and hasValue_fair)
                  and hasRank some (Rank) and hasPrice_verylow subclass of Good_Hotel
   0,569      hasPrice_high subclass of Good_Hotel
   0,289      Hotel_3_Stars and hasDistance some (isDistanceFor some (Train_Station)
                  and hasValue_verylow) and hasPrice_fair subclass of Good_Hotel
   0,198      Hotel_4_Stars and hasDistance some (isDistanceFor some (Square) and hasValue_high)
                  and hasRank some (Rank) and hasPrice_fair subclass of Good_Hotel

Running time 655 sec and 905 msec of which:
573 msec for subclasses retrieval
226  sec  for individuals retrieval
424  sec  for instance checking
  4  sec  for the algorithm running

----------------------------------------------------
```

Fig. 12. Log of the second trial of FOIL-$\mathcal{DL}$ on `Hotel.owl` (target concept: `Good_Hotel`).

3. Chitsaz, M., Wang, K., Blumenstein, M., Qi, G.: Concept learning for $\mathcal{EL}^{++}$ by refinement and reinforcement. In: Anthony, P., Ishizuka, M., Lukose, D. (eds.) PRICAI 2012: Trends in Artificial Intelligence - 12th Pacific Rim International Conference on Artificial Intelligence, Kuching, Malaysia, September 3-7, 2012. Proceedings. pp. 15–26. Lecture Notes in Artificial Intelligence, Springer-Verlag (2012)
4. Drobics, M., Bodenhofer, U., Klement, E.P.: FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions. Int. J. Approximate Reasoning 32(2-3), 131–152 (2003)
5. Dubois, D., Prade, H.: Possibility theory, probability theory and multiple-valued logics: A clarification. Annals of Mathematics and Artificial Intelligence 32(1-4), 35–66 (2001)
6. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Zelezný, F., Lavrač, N. (eds.) Inductive Logic Programming, 18th International Conference, ILP 2008, Prague, Czech Republic, September 10-12, 2008, Proceedings. Lecture Notes in Computer Science, vol. 5194, pp. 107–121. Springer (2008)
7. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer (1998)
8. 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), 25–48 (2009)
9. Iglesias, J., Lehmann, J.: Towards integrating fuzzy logic capabilities into an ontology-based inductive logic programming framework. In: Proc. of the 11th Int. Conf. on Intelligent Systems Design and Applications. IEEE Press (2011)
10. Klir, G.J., Yuan, B.: Fuzzy sets and fuzzy logic: theory and applications. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1995)
11. Konstantopoulos, S., Charalambidis, A.: Formulating description logic learning as an inductive logic programming task. In: Proc. of the 19th IEEE Int. Conf. on Fuzzy Systems. pp. 1–7. IEEE Press (2010)
12. Lehmann, J., Haase, C.: Ideal Downward Refinement in the $\mathcal{EL}$ Description Logic. In: De Raedt, L. (ed.) Inductive Logic Programming, 19th International Conference, ILP 2009, Leuven, Belgium, July 02-04, 2009. Revised Papers. Lecture Notes in Computer Science, vol. 5989, pp. 73–87 (2010)
13. Lisi, F.A., Straccia, U.: Towards learning fuzzy DL inclusion axioms. In: Fanelli, A.M., Pedrycz, W., Petrosino, A. (eds.) Fuzzy Logic and Applications - 9th International Workshop, WILF 2011, Trani, Italy, August 29-31,2011. Proceedings. Lecture Notes in Computer Science, vol. 6857, pp. 58–66. Springer (2011)
14. Lisi, F.A., Straccia, U.: A logic-based computational method for the automated induction of fuzzy ontology axioms. Fundamenta Informaticae 124, 1–17 (2013)
15. Lukasiewicz, T., Straccia, U.: Managing Uncertainty and Vagueness in Description Logics for the Semantic Web. Journal of Web Semantics 6, 291–308 (2008)
16. Motik, B., Rosati, R.: A faithful integration of description logics with logic programming. In: Veloso, M. (ed.) IJCAI 2007, Proc. of the 20th Int. Joint Conf. on Artificial Intelligence. pp. 477–482 (2007)
17. Nienhuys-Cheng, S.H., de Wolf, R.: Foundations of Inductive Logic Programming, Lecture Notes in Artificial Intelligence, vol. 1228. Springer (1997)
18. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning 5, 239–266 (1990)
19. Serrurier, M., Prade, H.: Improving expressivity of inductive logic programming by learning different kinds of fuzzy rules. Soft Computing 11(5), 459–466 (2007)
20. Shibata, D., Inuzuka, N., Kato, S., Matsui, T., Itoh, H.: An induction algorithm based on fuzzy logic programming. In: Zhong, N., Zhou, L. (eds.) Methodologies for

Knowledge Discovery and Data Mining, Third Pacific-Asia Conference, PAKDD-99, Beijing, China, April 26-28, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1574, pp. 268–273. Springer (1999)

21. Straccia, U.: Reasoning within fuzzy description logics. Journal of Artificial Intelligence Research 14, 137–166 (2001)
22. Straccia, U.: Description logics with fuzzy concrete domains. In: Bachus, F., Jaakkola, T. (eds.) 21st Conference on Uncertainty in Artificial Intelligence (UAI-05). pp. 559–567. AUAI Press, Edinburgh, Scotland (2005)
23. Wang, T., Li, Z., Yan, Y., Chen, H.: A survey of fuzzy decision tree classifier methodology. In: Proceedings of the Second International Conference of Fuzzy Information and Engineering, (ICFIE-07). pp. 959–968. No. 40 in Advances in Soft Computing, Springer Verlag (2007)