pFOIL-DL: Learning (Fuzzy) EL Concept Descriptions from Crisp OWL Data Using a Probabilistic Ensemble Estimation

Umberto Straccia  
ISTI - CNR  
Via G. Moruzzi 1  
Pisa, Italy

Matteo Mucci  
ISTI - CNR  
Via G. Moruzzi 1  
Pisa, Italy

ABSTRACT

OWL ontologies are nowadays a quite popular way to describe structured knowledge in terms of classes, relations among classes and class instances.

In this paper, given an OWL target class $T$, we address the problem of inducing $\mathcal{EL}(D)$ concept descriptions that describe sufficient conditions for being an individual instance of $T$. To do so, we use a FOIL-based method with a probabilistic candidate ensemble estimation. We illustrate its effectiveness by means of an experiment.

1. INTRODUCTION

OWL 2 ontologies [47] are nowadays a popular means to represent structured knowledge and its formal semantics is based on Description Logics (DLs) [3]. The basic ingredients of DLs are concept descriptions (in First-Order Logic terminology, unary predicates), inheritance relationships among them and instances of them.

Although an important amount of work has been carried about DLs, the application of machine learning techniques to OWL 2 data, is relatively marginally addressed [6, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42, 43, 44, 46, 50, 51, 52, 53, 54]. We refer the reader to [50] for a recent overview.

In this work, we focus on the problem of automatically learn concept descriptions from OWL 2 data. More specifically, given an OWL target class $T$, we address the problem of inducing $\mathcal{EL}(D)$ [2] concept descriptions that describe sufficient conditions for being an individual instance of $T$.

Example 1.1 ([39]). Consider an ontology that describes the meaningful entities of a city. 1 Now, one may fix a city, say Pisa, extract the properties of the hotels from Web sites, such as location, price, etc., and the hotel judgements of the users, e.g., from Trip Advisor. 2 Now, using the terminalogy of the ontology, one may ask about what are sufficient conditions characterising good hotels in Pisa according the user feedback. Then one may learn from the user feedback that, for instance, ‘An expensive Bed and Breakfast is a good hotel’. The goal here is to have both a human understandable characterisation as well as a good classifier for new, unclassified hotels.

To improve the readability of such characterisations, we further allow to use so-called fuzzy concept descriptions [40, 58] such as ‘an expensive Bed and Breakfast is a good hotel’. Here, the concept expensive is a so-called fuzzy concept [59], i.e., a concept for which the belonging of an individual to the class is not necessarily a binary yes/no question, but rather a matter of degree. For instance, in our example, the degree of expensiveness of a hotel may depend on the price of the hotel: the higher the price the more expensive is the hotel.

More specifically, we describe pFOIL-DL, a method for the automated induction of fuzzy $\mathcal{EL}(D)$ concept descriptions 3 by adapting the popular rule induction method FOIL [49]. But, unlike FOIL, in which the goodness of an induced rule is evaluated independently of previously learned rules, here we define a method that evaluates the goodness of a candidate in terms of the ensemble of so far learned rules, in a similar way as it happens in nFOIL [29].

Related Work. Related FOIL-like algorithms for the fuzzy case are reported in the literature [13, 55, 56] 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 [15]. DL-Learner supports the same learning problem as in DL-FOIL but implements algorithms that are not based on FOIL [24]. Likewise, Lehmann and Haase [33] propose a refinement operator for concept learning in $\mathcal{EL}$ (implemented in the ELTL algorithm within DL-Learner) whereas Chitsaz et al. [9] combine a refinement operator for $\mathcal{EL}++$ with a reinforcement learning algorithm. Both works deal with crisp ontologies. [19, 53] uses crisp terminological decision trees [5], while [25] is tailored to learn crisp $\mathcal{ALC}$ definitions. Very recently, an extension of DL-Learner with some of the most up-to-date fuzzy ontology tools has been proposed [26]. Notably, it can learn fuzzy OWL DL equivalence axioms from FuzzyOWL 2 ontologies 4 by interfacing the fuzzyDL reasoner [7]. However, it has been tested only on a toy ontology with crisp training

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

$\odot$ ACM 2015 April 13-17, 2015, Salamanca, Spain.  
Copyright 2015 ACM 978-1-4503-3196-8/15/04..S15.00.  
http://dx.doi.org/10.1145/2695864.2695907

\footnote{1For instance, \url{http://donghee.info/research/SHSS/ObjectiveConceptsOntology(OCO).html}}  
\footnote{2\url{http://www.tripadvisor.com}}  
\footnote{3$\mathcal{EL}$ is often used as learning language as illustrated in the related work section, and expressions are easily interpretable by humans.}

\footnote{4\url{http://www.straccia.info/software/FuzzyOWL}}
examples. The work reported in [28] is based on an ad-hoc translation of fuzzy Łukasiewicz \( \mathcal{ALC} \) DL constructs into Logic Programming (LP) and then uses a conventional Inductive Logic Programming ILP method to learn rules. The method is not sound as it has been recently shown that the mapping from fuzzy DLs to LP is incomplete [45] and entailment in Łukasiewicz \( \mathcal{ACC} \) is undecidable [8]. A similar investigation of the problem considered in the present paper can be found in [36, 38, 39]. In [36, 38], the resulting method (named SoftFOIL) provides a different solution from pFOIL-DC as for the weaker knowledge representation language DL-Lite [1], the confidence degree computation, the refinement operator and the heuristic. Moreover, unlike pFOIL-DC, SoftFOIL has not been implemented. Finally, [39] is the closest to our work and presents a FOIL variant for fuzzy and crisp DLs. Unlike [39], however, we rely here on a novel (fuzzy) probabilistic ensemble evaluation of the fuzzy concept description candidates that seems to provide better effectiveness on our test set.

The paper is structured as follows. Section 2 is devoted to the introduction of basic definitions to make the paper reasonably self-contained. Section 3 describes the learning problem and our algorithm. Section 4 addresses an experimental evaluation and Section 5 concludes the paper.

2. PRELIMINARIES

At first, we address Fuzzy Logics and then Fuzzy Description Logics basics (see [58] for an introduction to both).

Mathematical Fuzzy Logic. Fuzzy Logic is the logic of fuzzy sets [59]. A fuzzy set \( A \) over a countable crisp set \( X \) is a function \( A : X \rightarrow [0,1] \) called fuzzy membership function of \( A \). A crisp set \( B \) is characterised by a membership function \( B : X \rightarrow \{0,1\} \) instead. The standard fuzzy set operations conform to (named set operations) in [36, 38, 39]. In [36, 38], the resulting method (named SoftFOIL) provides a different solution from pFOIL-DC as for the weaker knowledge representation language DL-Lite [1], the confidence degree computation, the refinement operator and the heuristic. Moreover, unlike pFOIL-DC, SoftFOIL has not been implemented. Finally, [39] is the closest to our work and presents a FOIL variant for fuzzy and crisp DLs. Unlike [39], however, we rely here on a novel (fuzzy) probabilistic ensemble evaluation of the fuzzy concept description candidates that seems to provide better effectiveness on our test set.

The paper is structured as follows. Section 2 is devoted to the introduction of basic definitions to make the paper reasonably self-contained. Section 3 describes the learning problem and our algorithm. Section 4 addresses an experimental evaluation and Section 5 concludes the paper.

Table 1: Combination functions for fuzzy logics.

<table>
<thead>
<tr>
<th></th>
<th>( \text{Łukasiewicz} )</th>
<th>( \text{Gödel} )</th>
<th>( \text{Product} )</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha \oplus \beta )</td>
<td>( \min(\alpha + \beta,1) )</td>
<td>( \max(\alpha,\beta) )</td>
<td>( \alpha + \beta )</td>
<td>( \max(\alpha,\beta) )</td>
</tr>
<tr>
<td>( \alpha \odot \beta )</td>
<td>( \min(1 - \alpha + \beta,1) )</td>
<td>( \max(1,\beta/\alpha) )</td>
<td>( 1 )</td>
<td>( \max(1,\alpha) )</td>
</tr>
<tr>
<td>( \odot )</td>
<td>( 1 - \alpha )</td>
<td>( 1 ) if ( \alpha = 0 )</td>
<td>( 0 ) otherwise</td>
<td>( 0 ) otherwise</td>
</tr>
</tbody>
</table>

Table 2: Fuzzy sets over salaries.

<table>
<thead>
<tr>
<th>Salary</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 1: (a) Trapezoidal function \( \operatorname{trz}(a,b,c,d) \), (b) triangular function \( \operatorname{tri}(a,b,c) \), (c) left shoulder function \( \operatorname{ls}(a,b) \), and (d) right shoulder function \( \operatorname{rs}(a,b) \).

Figure 2: Fuzzy sets over salaries.
An *r-implication* is an implication function obtained as the residuum of a continuous t-norm $\otimes^\delta$; i.e., $\alpha \Rightarrow \beta = \max \{ \gamma | \alpha \otimes \gamma \leq \beta \}$. Note also, that given an r-implication $\Rightarrow$, we may also define its related negation $\ominus \alpha$ by means of $\alpha \Rightarrow 0$ for every $\alpha \in [0,1]$.

The notions of satisfiability and logical consequence are defined in the standard way, where a fuzzy interpretation $I$ satisfies a fuzzy statement $(\phi, \alpha)$, or $I$ is a model of $(\phi, \alpha)$, denoted as $I \models (\phi, \alpha)$, iff $I(\phi) \geq \alpha$. Notably, from $(\phi, \alpha)$ and $(\phi \rightarrow \psi, \beta)$ one may conclude (if $\models$ is an r-implication) $(\psi, \alpha \otimes \beta)$ (called fuzzy modus ponens).

**Fuzzy Description Logics basics.** We recap here the fuzzy variant of the DL ALC(D) [57], which is expressive enough to capture the main ingredients of fuzzy DLs.

We start with the notion of fuzzy concrete domain, that is a tuple $D = (\Delta^D, \cdot^D)$ with data type domain $\Delta^D$ and a mapping $\cdot^D$ that assigns to each data value an element of $\Delta^D$, and to every 1-ary data type predicate $d$ a 1-ary fuzzy relation over $\Delta^D$. Therefore, $\cdot^D$ maps indeed each data type predicate into a function from $\Delta^D$ to $[0,1]$. Typical examples of data type predicates $d$ are the well known membership functions

$$d \rightarrow \text{ls}(a, b) | \text{rs}(a, b) | \text{tri}(a, b, c) | \text{trz}(a, b, c, d)$$

where e.g. $\text{ls}(a, b)$ 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 $A$ be a set of concept names (also called atomic concepts), $R$ be a set of role names. Each role is either an object property or a data type property. The set of concepts are built from concept names $A$ using connectives and quantification constructs over object properties $R$ and data type properties $S$, as described by the following syntactic rule:

$$C \rightarrow T | \bot | A | C_1 \sqcap C_2 | C_1 \sqcup C_2 | \neg C | C_1 \rightarrow C_2$$

$$| \exists R.C | \forall R.C | \exists S.d | \forall S.d$$

An ABox $A$ consists of a finite set of assertion axioms. An assertion axiom is an expression of the form $(a : C, \alpha)$ (concept assertion, $a$ is an instance of concept $C$ to degree at least $\alpha$) or of the form $(\tau(a_1, a_2): R, \alpha)$ (role assertion, $(a_1, a_2)$ is an instance of object property $R$ to degree at least $\alpha$), where $a, a_1, a_2$ are individual names, $C$ is a concept, $R$ is an object property and $\alpha \in (0,1]$ is a truth value. A Terminological Box or TBox $T$ is a finite set of General Concept Inclusion (GCI) axioms, where a GCI is of the form $(C_1 \sqsubseteq C_2, \alpha)$ ($C_1$ is a sub-concept of $C_2$ to degree at least $\alpha$), where $C_1$ is a concept and $\alpha \in (0,1]$. We may omit the truth degree $\alpha$ of an axiom; in this case $\alpha = 1$ is assumed. A Knowledge Base (KB) is a pair $K = (T, A)$. With $I_K$ we denote the set of individuals occurring in $K$.

Concerning the semantics, let us fix a fuzzy logic. Unlike classical DLs in which an interpretation $I$ maps e.g. a concept $C$ into a set of individuals $C^I \subseteq \Delta^I$, i.e. $I$ maps $C$ into a function $C^I : \Delta^I \rightarrow [0,1]$ (either an individual belongs to the extension of $C$ or does not belong to it), in fuzzy DLs, $I$ maps $C$ into a function $C^I : \Delta^I \rightarrow [0,1]$ and, thus, an individual belongs to the extension of $C$ to some degree in $[0,1]$, i.e. $C^I$ is a fuzzy set. Specifically, a fuzzy interpretation is a pair $I = (\Delta^I, C^I)$ consisting of a nonempty (crisp) set $\Delta^I$ (the domain) and of a fuzzy interpretation function $\cdot^I$ that assigns: (i) to each atomic concept $A$ a function $A^I : \Delta^I \rightarrow [0,1]$; (ii) to each object property $R$ a function $R^I : \Delta^I \times \Delta^I \rightarrow [0,1]$; (iii) to each data type property $S$ a function $S^I : \Delta^I \times \Delta^I \rightarrow [0,1]$; (iv) to each individual $a$ an element $a^I \in \Delta^I$ such that $a^I \neq a^I$ if $a \neq b$ (Unique Name Assumption); and (v) to each data value $v$ an element $v^I \in \Delta^I$. Now, a fuzzy interpretation function is extended to concepts as specified below (where $x \in \Delta^I$):

$$\bot^I(x) = 0 \quad \top^I(x) = 1$$

$$(C \sqcap D)^I(x) = C^I(x) \otimes D^I(x)$$

$$(C \sqcup D)^I(x) = C^I(x) \oplus D^I(x)$$

$$(-C)^I(x) = \ominus C^I(x)$$

$$(C \rightarrow D)^I(x) = C^I(x) \rightarrow D^I(x)$$

$$(\forall R.C)^I(x) = \inf\{ R^I(x, y) \Rightarrow C^I(y) \}_{y \in \Delta^I}$$

$$(\exists R.C)^I(x) = \sup\{ R^I(x, y) \cap C^I(y) \}_{y \in \Delta^I}$$

$$(\forall S.d)^I(x) = \inf\{ S^I(x, y) \Rightarrow d^I(y) \}_{y \in \Delta^D}$$

$$(\exists S.d)^I(x) = \sup\{ S^I(x, y) \cap d^I(y) \}_{y \in \Delta^D}$$

The satisfiability of axioms is then defined by the following conditions: (i) $I$ satisfies an axiom $(a : C, \alpha)$ if $C^I(a^I) \geq \alpha$; (ii) $I$ satisfies an axiom $(a_1, a_2 : R, \alpha)$ if $R^I(a_1^I, a_2^I) \geq \alpha$; (iii) $I$ satisfies an axiom $(C \sqsubseteq D, \alpha)$ if $(C \sqsubseteq D)^I \geq \alpha$ where $\tau \in \Delta^I$.

A model of $K = (A, T)$ is a model of $K$ if $I_K \models \tau$. The best entailment degree of $\tau$ of the form $C \sqsubseteq D, a : C$ or $(a, b) : R$, denoted $\beta(K, \tau)$, is defined as

$$\beta(K, \tau) = \sup\{ \alpha | I_K \models (\tau, \alpha) \}$$

The cardinality of a concept $C$ w.r.t. a KB $K$ and a set of individuals $I$, denoted $|C|_K$, is defined as

$$|C|_K = \sum_{a \in I} \beta(K, a : C)$$

**Example 2.1.** Let us consider the following axiom

$$\exists \text{hasPrice. High } \sqsubseteq \text{GoodHotel, 0.569}$$

where hasPrice is a data type property whose values are measured in euros and the price concrete domain has been

\[\text{Note that Łukasiewicz, Gödel and Product implications are r-implications, while Kleene-Dienes implication is not.}\]
automatically fuzzified as illustrated in Figure 2. Now, it can be verified that for hotel verdi, whose room price is 105 euro, i.e. we have the assertion verdi:hasPrice. \(= 105 \) in the KB, we infer under Product logic that

\[\mathcal{K} \models (\text{verdi}:\text{GoodHotel}, 0.18).\]

3. THE LEARNING ALGORITHM

The problem statement. Given the target concept name \( T \), the training set \( \mathcal{E} \) consists of crisp concept assertions of the form \( a: T \) where \( a \) is an individual occurring in \( \mathcal{K} \). Also, \( \mathcal{E} \) is partitioned into \( \mathcal{E}^+ \) and \( \mathcal{E}^- \) (the positive and negative examples, respectively). Now, the learning problem is defined as follows. Given:

- a consistent KB \( \mathcal{K} = \langle T, A \rangle \) (the background theory);
- a concept name \( T \) (the target concept);
- a training set \( \mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^- \) such that \( \forall e \in \mathcal{E}, \mathcal{K} \not\models e \);
- a set \( \mathcal{H} \) of fuzzy GCIs (the language of hypotheses)

the goal is to find a finite set \( \mathcal{H} \subset \mathcal{L}_H \) (a hypothesis) of GCIs \( \mathcal{H} = \{ C_1 \subseteq T, \ldots, C_n \subseteq T \} \) such that:

- Consistency. \( \mathcal{K} \cup \mathcal{H} \) is a consistent;
- Non-Redundancy. \( \forall \tau \in \mathcal{H}, \mathcal{K} \not\models \tau \);
- Soundness. \( \forall e \in \mathcal{E}^-, \mathcal{K} \cup \mathcal{H} \not\models e \);
- Completeness. \( \forall a: T \in \mathcal{E}^+ \), \( \mathcal{K} \models a:C_i \), for some \( i \in \{1, \ldots, n\} \).

We say that an axiom \( \tau \in \mathcal{L}_H \) covers an example \( e \in \mathcal{E} \) iff \( \mathcal{K} \cup \{ \tau \} \models e \). Note that the way completeness is stated guarantees that all positive examples are covered by at least one axiom. Also, in practice one may allow a hypothesis that does not necessarily cover all positive examples, which is what we allow in our experiments.

The language of hypotheses. Given the target concept name \( T \), the hypotheses to be induced are fuzzy \( \mathcal{E} \mathcal{L}(D) \) GCIs, \(^9\) which are of the form

\[ C \subseteq T, \]  

(1)

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

\[
C \rightarrow T \mid A \mid \exists R.C \mid \exists S.d \mid C_1 \cap C_2 \\
d \rightarrow ls(a, b) \mid rs(a, b) \mid tri(a, b, c) \mid trz(a, b, c, d).
\]

Note that the language \( \mathcal{L}_H \) generated by this \( \mathcal{E} \mathcal{L}(D) \) syntax is potentially infinite due, e.g., to the nesting of existential restrictions yielding to complex concept expressions such as \( \exists R_1.C \exists R_2.\ldots \exists R_n.C(C) \ldots \). The language is 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 \( \mathcal{D} \mathcal{C} \) KB, the possible occurrence of fuzzy concrete domains in expressions of the form \( \exists S.d \) in the left-hand side of a fuzzy GCI of the form (1) may imply both that \( \text{bed}(\mathcal{K}, C \subseteq T) \notin \{0, 1\} \) and \( \text{bed}(\mathcal{K}, a: C) \notin \{0, 1\} \). Furthermore, as we shall see later on, once we have learned a fuzzy GCI \( C \subseteq T \), we may attach to it a degree (see Eq. 2), which may be seen as a kind of fuzzy set inclusion degree (see also Section 2) between the fuzzy concept \( C \) and \( T \).

Finally, note that the syntactic restrictions of \( \mathcal{L}_H \) allow for a straightforward translation of the inducible axioms of the form \( C_1 \cap \ldots \cap C_n \subseteq T \) as a rule "if \( C_1 \) and \ldots and \( C_n \) then \( T \)."

pFoil-\( DL \). Overall, pFoil-\( DL \) is inspired on Foil, a popular ILP algorithm for learning sets of rules which performs a greedy search in order to maximise a gain function \([49]\). More specifically, pFoil-\( DL \) has many commonalities with Foil-\( DL \) \([39]\), a Foil variant for fuzzy and crisp DLs. For reasons of space, we do not present Foil-\( DL \) here. Let us remark, however, that pFoil-\( DL \) is a probabilistic variant of Foil-\( DL \). It has three main differences from Foil-\( DL \):

- it uses a probabilistic measure to evaluate concept expressions;
- instead of removing positive examples covered from the training set as Foil-\( DL \) does, once an axiom has been learnt, positive examples are left unchanged after each learned rule;
- concept expressions are no longer evaluated one at a time, but the whole set of learnt expressions is evaluated as an ensemble.

Roughly, our learning algorithm proceeds as follows:

1. start from concept \( T \);
2. apply a refinement operator to find more specific concept description candidates;
3. exploit a scoring function to choose the best candidate;
4. re-apply the refinement operator until a good candidate is found;
5. iterate the whole procedure until a satisfactory coverage of the positive examples is achieved.

We are going now to detail the steps.

Computing fuzzy data types. For numerical data types, we adopt a discretisation method to partition it into a finite number of sets. After such a discretisation has been obtained we can define a fuzzy function on each set and obtain a fuzzy partition. In this work, we chose equal width triangular partition (see Figure 2). Specifically, for the triangular case, we look for the minimum and maximum value of a data property \( S \) in the ontology \( \mathcal{K} \), specifically \( \text{min}_S = \min\{v \mid \mathcal{K} \models a:3S. = v \text{ for some } a \in \mathcal{K}\} \) and \( \text{max}_S = \max\{v \mid \mathcal{K} \models a:3S. = v \text{ for some } a \in \mathcal{K}\} \). Then the interval \([\text{min}_S, \text{max}_S]\) is split into \( n > 1 \) intervals of same width. Each of the \( n \) intervals, \( I_1, \ldots, I_n \), has a fuzzy membership function associated. Let \( \Delta_S = \frac{\text{max}_S - \text{min}_S}{n - 1} \), then \( I_1 \) has associated a left shoulder \( \text{ls}(\text{min}_S, \text{min}_S + \Delta_S) \). \( I_n \) instead has associated a right shoulder \( \text{rs}(\text{max}_S - \Delta_S, \text{max}_S) \) and \( I_i \), with \( i = 2, \ldots, n - 1 \), has associated a triangular function \( \text{tri}(\text{min}_S + (i - 2) \cdot \Delta, \text{min}_S + (i - 1) \cdot \Delta, \text{min}_S + i \cdot \Delta) \).

The partition with \( n = 5 \) is the typical approach and indeed is also what we have chosen in our experiments.
The refinement operator. The refinement operator we employ follows [37, 38, 39]. Essentially, this operator takes as input a concept C and generates new, more specific concept description candidates D (i.e., K ⪰ D ⊆ C) and is defined as follows. Let K be an ontology, A be the set of all atomic concepts in K, R the set of all object properties in K, S the set of all data type properties in K and D a set of (fuzzy) datatypes. The refinement operator ρ is defined in Table 2. An example of refinement is the following.

Example 3.1. Consider the KB in Example 1.1. Assume that the target concept is Good_Hotel, that

K ⪰ Hotel_A_Stars ⊆ Accommodation

and also assume that the current concept description candidate C is

Accommodation ∩ ∃hasPrice.VeryHigh.

Then the following concept description belongs to the refinement of C:

Hotel_A_Stars ∩ ∃hasPrice.VeryHigh.

Note that the aim of a refinement is to reduce the number of covered negative examples, while still keeping some covered positive examples.

The scoring function. pFoil-\mathcal{DL} uses a scoring function to select the best candidate at each refinement step. For ease of presentation, let us assume that the hypothesis contains one axiom only, i.e., H = \{ C ⊆ T \}. Now, we define

\begin{align*}
T^+ &= |T|_K^E \\
C^+ &= |C|_K^E \\
\frac{T^+ \cap C^+}{T^+} &= |T \cap C|_K^E.
\end{align*}

So, for instance, C^+ is the cardinality of the fuzzy concept C w.r.t. the KB K and the set of individuals occurring in the training set E. Now, we define

\begin{align*}
P(C^+ | T^+) &= \frac{C^+ \cap T^+}{T^+}, \\
P(T^+ | C^+) &= \frac{T^+ \cap C^+}{C^+},
\end{align*}

as the probability of correctly classifying a positive example, and

\begin{align*}
\text{score}_\beta(C) &= (1 + \beta^2) \cdot \frac{P(T^+ | C^+) \cdot P(C^+ | T^+)}{\beta^2 \cdot P(T^+ | C^+) + P(C^+ | T^+)}
\end{align*}

to compute the confidence degree of concept C.

For the general case in which we have a set H = \{ C_1 ⊆ T, ..., C_n ⊆ T \}, we define

\begin{align*}
H^+ &= |C_1 \cup ... \cup C_n|_K^E, \\
H^+ \cap T^+ &= |(C_1 \cup ... \cup C_n) \cap T|_K^E,
\end{align*}

which derives from viewing the axioms in H as a GCI C_1 \cup ... \cup C_n ⊆ T and, thus, we have:

\begin{align*}
P(T^+ | H^+) &= \frac{H^+ \cap T^+}{H^+}, \\
P(H^+ | T^+) &= \frac{H^+ \cap T^+}{T^+},
\end{align*}

\begin{align*}
\text{score}_\beta(H) &= (1 + \beta^2) \cdot \frac{P(T^+ | H^+) \cdot P(H^+ | T^+)}{\beta^2 \cdot P(T^+ | H^+) + P(H^+ | T^+)},
\text{score}_\beta(C, H) &= \text{score}_\beta(H \cup \{ C \subseteq T \}).
\end{align*}

Stop Criterion. Foil-\mathcal{DL} stops when all positive samples are covered or when it becomes impossible to find a concept without negative coverage [39]. In pFoil-\mathcal{DL}, instead, we impose that no axioms are learnt unless they improve the ensemble score. So, if adding a new axiom the score of the ensemble does not vary (or decreases) the axiom is not learnt. Moreover, we use a threshold \( \delta \) and stop a soon as the score improvement is below \( \delta \).

Once an axiom has been learnt, the ensemble performance is evaluated using Eq. 6, while a candidate concept description is evaluated according to Eq. 7. In our algorithm, we allow to specify the \( \beta \) used during concept evaluation as \( \beta_1 \), while the \( \beta \) used during ensemble evaluation is specified as \( \beta_2 \).

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 (therefore, the computation may end without covering all positive examples).

Backtracking. The implementation of pFoil-\mathcal{DL} features, as for Foil-\mathcal{DL} [39], also a backtracking mode. This option allows to overcome one of the main limits of Foil. Indeed, as it performs a greedy search, formulating a sequence of axioms without backtracking, Foil does not guarantee to find the smallest or best set of rules that explain the training examples. Also, learning axioms 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 pFoil-\mathcal{DL} by a beam search which maintains a list of \( k \) best candidates at each step instead of a single best candidate.
The Algorithm. The pFoil-DL algorithm is defined in Algorithm 1. The algorithm learnSetOfAxioms loops to learn one axiom at a time, invoking learnOneAxiom. This latter procedure is detailed in Algorithm 2. Note that we do not allow the covering of negative examples due to the condition $|C|_{K}^{e^-} \neq 0$ at step 3. Its inner loop (steps 7 - 11) just determines the best candidate concept among all refinements. For completeness, in Algorithm 3 we report the variant of learnOneAxiom with backtracking using Top-k backtracking. Top-k backtracking saves the k best concept expressions obtained from refinements. We use a stack of k concepts ordered in decreasing order of the score. The ADD function, called at step 10, adds a concept to the stack if it makes into the top-k ones (and removes the $k + 1$ ranked one), while the POP function, called in step 16, instead removes from the stack the concept expression that has the highest score. The removed concept is returned as result.

Algorithm 1 pFoil-DL: learnSetOfAxioms

```plaintext
1: function learnSetOfAxioms(K, T, $e^+, e^-$, $\theta$, $\beta_1$, $\beta_2$)
2:     $H \leftarrow \emptyset$;
3:     oldScore $\leftarrow 0$;
4:     newScore $\leftarrow \text{score}_{\beta_2}([T \subseteq T])$;
5:     while newScore $\neq$ oldScore and newScore $\geq 0$ do
6:         $C \leftarrow \text{LEARNONEAXIOM}(K, T, H, e^+, e^-, \beta_1)$;
7:         if($C = \top$ or $C = \emptyset$) then
8:             return $H$;
9:         $H \leftarrow H \cup \{C \subseteq T\}$;
10:        oldScore $\leftarrow$ newScore;
11:        newScore $\leftarrow \text{score}_{\beta_2}(H)$;
12:    return $H$;
13: end
```

Algorithm 2 pFoil-DL: learnOneAxiom

```plaintext
1: function learnOneAxiom(K, T, $H$, $e^+$, $e^-$, $\beta_1$)
2:     $C \leftarrow \top$;
3:     while $|C|_{K}^{e^-} \neq 0$ do
4:         $C_{best} \leftarrow C$;
5:         maxscore $\leftarrow \text{score}_{\beta_2}(C_{best}, H)$;
6:         $C \leftarrow \text{REFINE}(C)$;
7:         for all $C' \in C$ do
8:             $\text{score} \leftarrow \text{score}_{\beta_2}(C', H)$;
9:             if $\text{score} > \text{maxscore}$ then
10:                 maxscore $\leftarrow \text{score}$;
11:                 $C_{best} \leftarrow C'$;
12:         if($C_{best} = C$) then
13:             return null;
14:         $C \leftarrow C_{best}$;
15: end
```

Algorithm 3 pFoil-DL: learnOneAxiom (Top-k backtracking)

```plaintext
1: function learnOneAxiom(K, T, $H$, $e^+$, $e^-$, $\beta_1$)
2:     $C \leftarrow \top$;
3:     stack $\leftarrow \emptyset$;
4:     while $|C|_{K}^{e^-} \neq 0$ do
5:         $C_{best} \leftarrow C$;
6:         maxscore $\leftarrow \text{score}_{\beta_2}(C_{best}, H)$;
7:         $C \leftarrow \text{REFINE}(C)$;
8:         for all $C' \in C$ do
9:             $\text{score} \leftarrow \text{score}_{\beta_2}(C', H)$;
10:        if $\text{score} > \text{maxscore}$ then
11:            $C_{best} \leftarrow C'$;
12:        if $C_{best} = C$ then
13:            return $C$;
14:        $C \leftarrow C_{best}$;
15: end
```

4. EVALUATION

Setup. We conducted a preliminary experimentation. A prototype has been implemented (on top of the owl-api and a classical DL reasoner), supporting both FoIL-DL as well as pFoil-DL and applied them to test a classification case. To this purpose, a number of OWL ontologies have been considered as illustrated in Table 3. For each ontology $K$, a meaning full target concept $T$ has manually been selected such that the conditions of the learning problem statement (see Section 3) are satisfied. Then we learned inclusion axioms of the form $C \subseteq T$ and measured how good these inclusion axioms are to classify individuals of being instances of $T$. We report the DL the ontology refers to, the number of concept names, object properties, datatype properties and concept names, object properties, datatype properties and

Consider as illustrated in Table 3. For each ontology $K$, a meaning full target concept $T$ has manually been selected such that the conditions of the learning problem statement (see Section 3) are satisfied. Then we learned inclusion axioms of the form $C \subseteq T$ and measured how good these inclusion axioms are to classify individuals of being instances of $T$. We report the DL the ontology refers to, the number of concept names, object properties, datatype properties and

<table>
<thead>
<tr>
<th>DL</th>
<th>Concept Names</th>
<th>Object Properties</th>
<th>Datatype Properties</th>
<th>Total Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWL</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>RDFS</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>SHOIN</td>
<td>12</td>
<td>7</td>
<td>4</td>
<td>23</td>
</tr>
</tbody>
</table>

For each measure, the average value over the various folds is reported in the tables.

Results. For illustrative purposes, example expressions learnt by pFoil-DL during the experiments are reported in Table 4. Note that both in the Hotel and the UBA ontology case a fuzzy GCI has been induced.

Table 5 summarises the results of the tests for each ontology and measure. The $t$ column reports the average time (in

10We recall that concept descriptions may be fuzzy.


Tests.zip.
Table 3: Classification results.

<table>
<thead>
<tr>
<th>ontology</th>
<th>DL classes</th>
<th>obj. prop.</th>
<th>data. prop.</th>
<th>individuals</th>
<th>target</th>
<th>max d./c.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family-tree</td>
<td>SHOIN(D)</td>
<td>22</td>
<td>52</td>
<td>6</td>
<td>368</td>
<td>Uncle</td>
</tr>
<tr>
<td>Hotel</td>
<td>ALCOF(D)</td>
<td>89</td>
<td>3</td>
<td>1</td>
<td>88</td>
<td>Good_Hotel</td>
</tr>
<tr>
<td>Moral</td>
<td>ACC</td>
<td>46</td>
<td>0</td>
<td>0</td>
<td>202</td>
<td>Guilty</td>
</tr>
<tr>
<td>SemanticBible</td>
<td>SHOTIN(D)</td>
<td>51</td>
<td>29</td>
<td>9</td>
<td>723</td>
<td>Woman</td>
</tr>
<tr>
<td>UBA</td>
<td>SH2(D)</td>
<td>44</td>
<td>26</td>
<td>8</td>
<td>1268</td>
<td>Good_Researcher</td>
</tr>
<tr>
<td>FuzzyWine</td>
<td>SHI(D)</td>
<td>178</td>
<td>15</td>
<td>7</td>
<td>138</td>
<td>DryWine</td>
</tr>
</tbody>
</table>

Table 4: Examples of learned axioms by pFOIL-DL.

<table>
<thead>
<tr>
<th>ontology</th>
<th>induced axiom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family-tree</td>
<td>(Person ⊓ (HasNumberOfPublications,VeryHigh) ⊑ Good_Hotel)</td>
</tr>
<tr>
<td>Hotel</td>
<td>(Bed ⊓ Good_Hotel,Low) ⊓ (Bed ⊑ Good_Hotel,Low)</td>
</tr>
<tr>
<td>Moral</td>
<td>(blameworthy ⊑ Guilty)</td>
</tr>
<tr>
<td>SemanticBible</td>
<td>(spouse,Male,Man) ⊑ Woman</td>
</tr>
<tr>
<td>UBA</td>
<td>:hasNumberOfPublications,VeryHigh ⊑ Good_Researcher</td>
</tr>
<tr>
<td>FuzzyWine</td>
<td>:isDryWine, :isSweetWine ⊑ DryWine</td>
</tr>
</tbody>
</table>

Table 5: Facts about the ontologies of the experiment.

<table>
<thead>
<tr>
<th>ontology</th>
<th>DL classes</th>
<th>obj. prop.</th>
<th>data. prop.</th>
<th>individuals</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family-tree</td>
<td>SHOIN(D)</td>
<td>22</td>
<td>52</td>
<td>6</td>
<td>368</td>
</tr>
<tr>
<td>Hotel</td>
<td>ALCOF(D)</td>
<td>89</td>
<td>3</td>
<td>1</td>
<td>88</td>
</tr>
<tr>
<td>Moral</td>
<td>ACC</td>
<td>46</td>
<td>0</td>
<td>0</td>
<td>202</td>
</tr>
<tr>
<td>SemanticBible</td>
<td>SHOTIN(D)</td>
<td>51</td>
<td>29</td>
<td>9</td>
<td>723</td>
</tr>
<tr>
<td>UBA</td>
<td>SH2(D)</td>
<td>44</td>
<td>26</td>
<td>8</td>
<td>1268</td>
</tr>
<tr>
<td>FuzzyWine</td>
<td>SHI(D)</td>
<td>178</td>
<td>15</td>
<td>7</td>
<td>138</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS

In this work we addressed the problem to induce (possibly fuzzy) $\cal{EL}(D)$ concept descriptions that describe sufficient conditions of being an individual instance of a target class. To do so, we described pFOIL-DL, which stems from a previous method, called FOIL-DL [39]. But unlike FOIL-DL, in pFOIL-DL the goodness of a candidate is evaluated in terms of the ensemble of so far learned candidates. Both, FOIL-DL and pFOIL-DL have been submitted to a preliminary experimentation to measure their effectiveness. Our experiments seem to indicate that pFOIL-DL performs better so far, but comes at a price of a higher execution time. We plan to conduct a more extensive experimentation and look at various other learning algorithms as well.

6. REFERENCES


