Chapter 21

Can ILP Deal with Incomplete and Vague Structured Knowledge?

Francesca A. Lisi

Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro", Italy

Umberto Straccia

ISTI — CNR, Pisa, Italy

21.1 Introduction

Ontologies are currently a prominent source of *structured* knowledge. The logical languages known as *Description Logics* (DLs) [Baader *et al.* (2003)] 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 example, DL-Lite [Calvanese *et al.* (2006)] 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.

Incompleteness and *vagueness* are inherent properties of knowledge in several real-world domains. DL-based ontology languages were born to address the former. Indeed, the Open World Assumption (OWA) holds in DLs. Fuzzy extensions of DLs have been more recently devised to address the latter (see the survey in [Lukasiewicz and Straccia (2008)]). They include, among others, a fuzzy DL-Lite like DL which has been implemented in the SoftFacts[3] system [Straccia (2010)].

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

In this chapter, we sketch the results of our preliminary investigation of the issue of whether ILP can deal with incomplete and vague structured knowledge. More precisely, we provide the ingredients for learning fuzzy DL inclusion axioms with ILP. The resulting method adapts known results in ILP concerning the induction of crisp rules, notably FOIL [Quinlan (1990)], to the novel context of ontologies.

The chapter is organized as follows. Section 21.2 introduces fuzzy DLs. Section 21.3 describes our preliminary contribution to the problem in hand, also by means of an illustrative example. Section 21.4 concludes the chapter with final remarks and comparison with related work.

21.2 Fuzzy Description Logics

For computational reasons, the logic we adopt is based on a fuzzy extension of the DL-Lite DL without negation [Straccia (2010)]. It supports at the intentional level unary relations (called *concepts*) and binary relations (called *roles*), while also supporting n-ary relations (relational tables) at the extensional level.

Formally, a *knowledge base* $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{A} \rangle$ consists of a *facts component* $\mathcal{F}$, an *ontology component* $\mathcal{O}$ and an *abstraction component* $\mathcal{A}$. Information can be retrieved from $\mathcal{K}$ by means of an appropriate *query language*.

In order to deal with vagueness, Gödel logic [Hájek (1998)] is adopted, where

$$a \otimes b = \min(a, b),$$
$$a \oplus b = \max(a, b),$$
$$a \Rightarrow b = \begin{cases} 1 & \text{if } a \leq b \\ b & \text{otherwise} \end{cases}, \quad \text{and} \quad \ominus a = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}.$$

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

$$R(c_1, \ldots, c_n)[s], \tag{21.1}$$

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

Ontology Component. $\mathcal{O}$ is a finite set of *inclusion axioms* having the form

$$Rl_1 \sqcap \ldots \sqcap Rl_m \sqsubseteq Rr, \tag{21.2}$$

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

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

$$\begin{array}{l} Rl \longrightarrow A \mid R[i_1, i_2] \\ Rr \longrightarrow A \mid R[i_1, i_2] \mid \exists R.A \end{array} \tag{21.3}$$

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

Abstraction Component. $\mathcal{A}$ is a finite set of *abstraction statements* of the form

$$R \mapsto (c_1, \ldots, c_n)[c_{score}].sql, \tag{21.4}$$

where sql is an SQL statement returning n-ary tuples $\langle c_1, \ldots, c_n \rangle$ $(n \leq 2)$ with score determined by the c_{score} column. The tuples have to be ranked in decreasing order of score and, as for the fact component, we assume that there cannot be two records $\langle \mathbf{c}, s_1 \rangle$ and $\langle \mathbf{c}, s_2 \rangle$ in the result set of sql with $s_1 \neq s_2$ (if there are, then we remove the one with the lower score). The score c_{score} may be omitted and in that case the score 1 is assumed for the tuples. We assume that R occurs in $\mathcal{O}$, while all of the relational tables occurring in the SQL statement occur in $\mathcal{F}$. Finally, we assume that there is at most one abstraction statement for each abstract relational symbol R.

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 of the form

$$\begin{array}{l} q(\mathbf{x})[s] \leftarrow \exists \mathbf{y}\ R_1(\mathbf{z}_1)[s_1], \ldots, R_l(\mathbf{z}_l)[s_l], \\ \qquad\qquad \mathsf{OrderBy}(s = f(s_1, \ldots, s_l, p_1(\mathbf{z}'_1), \ldots, p_h(\mathbf{z}'_h)) \end{array} \tag{21.5}$$

where

(1) q is an n-ary relation, every R_i is a n_i-ary relation $(1 \leq n_i \leq 2)$. $R_i(\mathbf{z_i})$ may also be of the form $(z \leq v), (z < v), (z \geq v), (z > v), (z = v)$,

($z \neq v$), where z is a variable, v is a value of the appropriate concrete domain;
(2) $\mathbf{x}$ are the *distinguished variables*;
(3) $\mathbf{y}$ are existentially quantified variables called the *non-distinguished variables*. We omit to write $\exists \mathbf{y}$ when $\mathbf{y}$ is clear from the context;
(4) $\mathbf{z_i}, \mathbf{z'_j}$ are tuples of constants or variables in $\mathbf{x}$ or $\mathbf{y}$;
(5) $s, s_1, \ldots, s_l$ are distinct variables and different from those in $\mathbf{x}$ and $\mathbf{y}$;
(6) p_j is an n_j-ary *fuzzy predicate* assigning a score $p_j(\mathbf{c}_j) \in [0,1]$ to each n_j-ary tuple $\mathbf{c}_j$ of constants;
(7) f is a *scoring function* $f\colon ([0,1])^{l+h} \to [0,1]$, which combines the scores of the l relations $R_i(\mathbf{c}'_i)$ and the n fuzzy predicates $p_j(\mathbf{c}''_j)$ into an overall score s to be assigned to $q(\mathbf{c})$.

We call $q(\mathbf{x})[s]$ its *head*, $\exists \mathbf{y}.R_1(\mathbf{z}_1)[s_1], \ldots, R_l(\mathbf{z}_l)[s_l]$ its *body* and $\mathsf{OrderBy}(s = f(s_1, \ldots, s_l, p_1(\mathbf{z}'_1), \ldots, p_h(\mathbf{z}'_h))$ the *scoring atom*. We also allow the scores $[s], [s_1], \ldots, [s_l]$ and the scoring atom to be omitted. In this case we assume the value 1 for s_i and s instead. The informal meaning of such a query is: if $\mathbf{z}_i$ is an instance of R_i to degree at least or equal to s_i, then $\mathbf{x}$ is an instance of q to degree at least or equal to s, where s has been determined by the scoring atom.

The answer set $ans_{\mathcal{K}}(q)$ over $\mathcal{K}$ of a query q is the set of tuples $\langle \mathbf{t}, s \rangle$ such that $\mathcal{K} \models q(\mathbf{t})[s]$ with $s > 0$ (informally, $\mathbf{t}$ satisfies the query to non-zero degree s) and the score s is as high as possible, *i.e.* if $\langle \mathbf{t}, s \rangle \in ans_{\mathcal{K}}(q)$ then (i) $\mathcal{K} \not\models q(\mathbf{t})[s']$ for any $s' > s$; and (ii) there cannot be another $\langle \mathbf{t}, s' \rangle \in ans_{\mathcal{K}}(q)$ with $s > s'$.

21.3 ILP for Learning Fuzzy DL Inclusion Axioms

In this section we consider a learning problem where:

- the target concept H is a DL-Lite atomic concept;
- the background theory $\mathcal{K}$ is a DL-Lite-like knowledge base $\langle \mathcal{F}, \mathcal{O}, \mathcal{A} \rangle$ of the form described in Section 21.2;
- the training set $\mathcal{E}$ is a collection of fuzzy DL-Lite-like facts of the form (21.1) and labeled as either positive or negative examples for H. We assume that $\mathcal{F} \cap \mathcal{E} = \emptyset$;
- the target theory $\mathcal{H}$ is a set of inclusion axioms of the form

$$B \sqsubseteq H \tag{21.6}$$

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

$$C \longrightarrow A \mid \exists R.A \mid \exists R.\top. \tag{21.7}$$

We now show how we may learn inclusion axioms of the form (21.6). To this aim, we define for $C \neq H$

$$\mathcal{I}_{ILP} \models C(t) \text{ iff } \mathcal{K} \cup \mathcal{E} \models C(t)[s] \text{ and } s > 0. \tag{21.8}$$

That is, we write $\mathcal{I}_{ILP} \models C(t)$ if it can be inferred from $\mathcal{K}$ and $\mathcal{E}$ that t is an instance of concept C to a non-zero degree.

Now, in order to account for multiple fuzzy instantiations of fuzzy predicates occurring in the inclusion axioms of interest to us, we propose the following formula for computing the *confidence degree* of an inclusion axiom:

$$cf(B \sqsubseteq H) = \frac{\sum_{t \in P} B(t) \Rightarrow H(t)}{|D|} \tag{21.9}$$

where

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

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

Note that the confidence score can be determined easily by submitting appropriate queries via the query language described in Section 21.2. More precisely, proving the fuzzy entailment in (21.8) 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_is into FOL predicates and whose score s is given by the minimum between s_ls.

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

For illustrative purposes we consider the case involving the classification of hotels as good ones. We assume to have a background theory $\mathcal{K}$ with a relational database $\mathcal{F}$ storing facts such as an ontology $\mathcal{O}$[4] encompassing the following inclusion axioms

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

and a set $\mathcal{A}$ of abstraction statements such as:

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

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

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

where $cheap(p)$ is a function determining how cheap a hotel room is given its price, modelled as e.g. a so-called left-shoulder function (defined in Fig. 21.1). We set $cheap(p) = ls(p; 50, 100)$, while $closedistance(d) = ls(d; 5, 25)$.

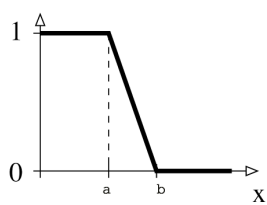
Assume now that our target concept H is $GoodHotel$, and that

- $\mathcal{E}^+ = \{GoodHotel^+(h1)[0.6], GoodHotel^+(h2)[0.8]\}$, while $\mathcal{E}^- = \{GoodHotel^-(h3)[0.4]\}$;
- $GoodHotel^+ \sqsubseteq GoodHotel$ and $GoodHotel^- \sqsubseteq GoodHotel$ occur in $\mathcal{K}$.

As an illustrative example, we compute the confidence degree of

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

i.e. a good hotel is one having a cheap price and close proximity to an attraction. Now, it can be verified that for $\mathcal{K}' = \mathcal{K} \cup \mathcal{E}$



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

Fig. 21.1 Left shoulder function $ls(x; a, b)$.

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

(1) The query

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

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

(2) The query

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

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

(3) Therefore, according to (21.9), $P = \{h1, h2\}$, while $D = \{h1, h2, h3\}$;

(4) As a consequence,

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

21.4 Final Remarks

In this chapter we have briefly presented the core ingredients for inducing ontology inclusion axioms within the KR framework of a fuzzy DL-Lite-like DL. These ingredients can be used to extend FOIL as shown in [Lisi and Straccia (2011)]. Related FOIL-like algorithms [Shibata *et al.* (1999); Drobics *et al.* (2003); Serrurier and Prade (2007)] can only learn fuzzy rules. The formal study of fuzzy ILP contributed by [Horváth and Vojtás (2007)] is also relevant but less promising than our proposal in practice. Closer to our application domain, [Iglesias and Lehmann (2011)] extends an existing ILP system for learning $\mathcal{ALC}$ DL concepts (DL-Learner) with some of the most up-to-date fuzzy ontology tools whereas [Konstantopoulos and Charalambidis (2010)] is based on an ad-hoc translation of fuzzy Łukasiewicz $\mathcal{ALC}$ DL constructs into logic programs.

Bibliography

F. Baader, D. Calvanese, D. McGuinness, D. Nardi and P. F. Patel-Schneider (eds). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, Cambridge. 2003.

D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini and R. Rosati. Data complexity of query answering in description logics. In P. Doherty, J. Mylopoulos and C. A. Welty (eds). *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*. AAAI Press, Boston, Massachusetts, pp. 260–270. 2006.

M. Drobics, U. Bodenhofer and E.-P. Klement. FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions. *Int. J. Approximate Reasoning*, **32(2–3)**, 131–152. 2003.

P. Hájek. *Metamathematics of Fuzzy Logic*. Springer, Berlin. 1998.

T. Horváth and P. Vojtás. Induction of fuzzy and annotated logic programs. In S. H. Muggleton, R. P. Otero, and A. Tamaddoni-Nezhad (eds). *Revised Selected Papers from the 16th International Conference on Inductive Logic Programming*, LNCS, vol. 4455. Springer, Berlin, pp. 260–274. 2007.

J. Iglesias and J. Lehmann. Towards Integrating Fuzzy Logic Capabilities into an Ontology-based Inductive Logic Programming Framework. In *Proceedings of the 11th International Conference on Intelligent Systems Design and Applications*. IEEE Press, Piscataway, New Jersey, pp. 1323–1328. 2011.

S. Konstantopoulos and A. Charalambidis. Formulating description logic learning as an inductive logic programming task. In *Proceedings of the IEEE International Conference on Fuzzy Systems*. IEEE Press, Piscataway, New Jersey, pp. 1–7. 2010.

F. A. Lisi and U. Straccia. An inductive logic programming approach to learning inclusion axioms in fuzzy description logics. In F. Fioravanti (ed.). *Proceedings of the 26th Italian Conference on Computational Logic*. CEUR-WS.org, pp. 57–71. 2011.

T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *J. Web Semant.*, **6**, 291–308. 2008.

J. R. Quinlan. Learning logical definitions from relations. *Mach. Learn.*, **5**, 239–266. 1990.

M. Serrurier and H. Prade. Improving expressivity of inductive logic programming by learning different kinds of fuzzy rules. *Soft Comput.*, **11(5)**, 459–466. 2007.

D. Shibata, N. Inuzuka, S. Kato, T. Matsui and H. Itoh. An induction algorithm based on fuzzy logic programming. In N. Zhong and L. Zhou (eds). *Methodologies for Knowledge Discovery and Data Mining*, LNCS, vol. 1574. Springer, Berlin, pp. 268–273. 1999.

U. Straccia. SoftFacts: A top-k retrieval engine for ontology mediated access to relational databases. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 4115–4122. IEEE Press, Piscataway, New Jeresey. 2010.