Fuzzy Logic, Annotation Domains and Semantic Web Languages

Umberto Straccia

ISTI - CNR, Pisa, Italy
straccia@isti.cnr.it
http://www.umberto-straccia.name

Abstract. This talk presents a detailed, self-contained and comprehensive account of the state of the art in representing and reasoning with fuzzy knowledge in Semantic Web Languages such a RDF/RDFS, OWL 2 and RIF and discuss some implementation related issues. We further show to which extend we may generalise them to so-called annotation domains, that cover also e.g. temporal, provenance and trust extensions.

Keywords: Fuzzy Logic, Semantic Web Languages, RDFS, OWL, RIF.

1 Introduction

Managing uncertainty and fuzzyness is starting to play an important role in Semantic Web research, and has been recognised by a large number of research efforts in this direction (see, e.g., [68] for a concise overview).

We recall 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 [28]. We recall here the salient concepts for the inexpert reader. Under *uncertainty theory* fall all those approaches in which statements rather than being either true or false, are true or false to some *probability* or *possibility* (for example, "it will rain tomorrow"). That is, a statement is true or false in any world/interpretation, but we are "uncertain" about which world to consider as the right one, and thus we speak about e.g. a probability distribution or a possibility distribution over the worlds. For example, we cannot exactly establish whether it will rain tomorrow or not, due to our *incomplete* knowledge about our world, but we can estimate to which degree this is probable, possible, and necessary. On the other hand, under *fuzzy theory* fall all those approaches in which statements (for example, "the tomato is ripe") 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 "ripe", which do not have an *precise* definition (we cannot always say whether a tomato is ripe or not). Note that all fuzzy statements are truth-functional, that is, the degree of truth of every statement can be calculated from the degrees of truth of its constituents, while uncertain statements cannot always be a function of the uncertainties of their

S. Benferhat and J. Grant (Eds.): SUM 2011, LNAI 6929, pp. 2–21, 2011.

constituents [27]. More concretely, in probability theory, only negation is truth-functional, while in possibility theory, only disjunction (resp. conjunction) is truth-functional in possibilities (resp. necessities) of events. Furthermore, mathematical fuzzy logics are based on truly many-valued logical operators, while uncertainty logics are defined on top of standard binary logical operators.

We present here some salient aspects in representing and reasoning with fuzzy knowledge in Semantic Web Languages (SWLs) such as *triple languages* RDF & RDFS [19] (see, e.g. [69,70]), *conceptual languages* or *frame-based languages* of the OWL 2 family [50] (see, e.g. [45,58,62]) and *rule languages*, such as RIF [53] (see, e.g. [65,66,68]).

In the following, we overview briefly SWLs and relate them to their logical counterpart. Then, we briefly sketch the basic notions of Mathematical Fuzzy Logic, which we require in the subsequent sections in which we illustrate the fuzzy variants of SWLs.

2 Semantic Web Languages: Overview

The Semantic Web is a 'web of data' whose goal is to enable machines to understand the semantics, or meaning, of information on the World Wide Web. In rough terms, it should extend the network of hyperlinked human-readable web pages by inserting machine-readable *metadata*[1] about pages and how they are related to each other, enabling automated agents to access the Web more intelligently and perform tasks on behalf of users.

Semantic Web Languages (SWL) are the languages used to provide a formal description of concepts, terms, and relationships within a given knowledge domain to be used to write the metadata. There are essentially three family of languages: namely, *triple languages* RDF & RDFS [19] (*Resource Description Framework*), *conceptual languages* of the OWL 2 family (*Ontology Web Language*) [50] and *rule languages* of the RIF family (*Rule Interchange Format*) [53]. While their syntactic specification is based on XML [74], their semantics is based on logical formalisms, which will be the focus here (see Fig. 1): briefly,

- RDFS is a logic having intensional semantics and the logical counterpart is ρdf [47];
- OWL 2 is a family of languages that relate to *Description Logics* (DLs) [6];
- RIF relates to the *Logic Programming* (LP) paradigm [43];
- both OWL 2 and RIF have an extensional semantics.

RDF & RDFS. The basic ingredients of *RDF* are *triples* of the form (s, p, o), such as $(umberto, likes, tomato)$, stating that *subject* s has *property* p with *value* o. In *RDF Schema* (RDFS), which is an extension of RDF, additionally some special keywords may be used as properties to further improve the expressivity of the language. For instance we may also express that the class of 'tomatoes are a subclass of the class of vegetables', $(tomato, \mathsf{sc}, vegetables)$, while Zurich is an instance of the class of cities, $(zurich, \mathsf{type}, city)$.

[1] Obtained manually, semi-automatically, or automatically.

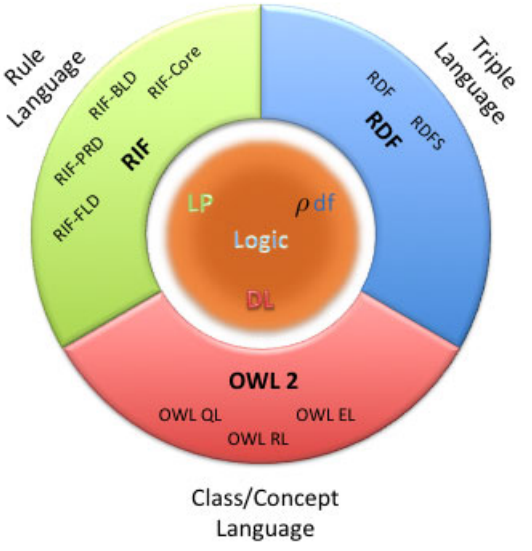


Fig. 1. Semantic Web Languages from a Logical Perspective

Form a computational point of view, one computes the so-called *closure* (denoted $cl(\mathcal{K})$) of a set of triples $\mathcal{K}$. That is, one infers all possible triples using inference rules [46,47,52], such as

$$\frac{(A, \mathsf{sc}, B), (X, \mathsf{type}, A)}{(X, \mathsf{type}, B)}$$

"if A subclass of B and X instance of A then infer that X is instance of B",

and then store all inferred triples into a relational database to be used then for querying. We recall also that there also several ways to store the closure $cl(\mathcal{K})$ in a database (see [1,37]). Essentially, either we may store all the triples in table with three columns *subject, predicate, object*, or we use a table for each predicate, where each table has two columns *subject, object*. The latter approach seems to be better for query answering purposes. Note that making all implicit knowledge explicit is viable due to the low complexity of the closure computation, which is $\mathcal{O}(|\mathcal{K}|^2)$ in the worst case.

OWL Family. The Web Ontologoy Language *OWL* [49] and its successor *OWL 2* [23,50] are "object oriented" languages for defining and instantiating Web ontologies. Ontology (see, e.g. [31]) is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related. An OWL ontology may include descriptions of classes, properties and their instances, such as

```
class Person partial Human
      restriction (hasName someValuesFrom String)
      restriction (hasBirthPlace someValuesFrom Geoplace)
```

> "The class Person is a subclass of class Human and has two attributes: hasName having a string as value, and hasBirthPlace whose value is an instance of the class Geoplace".

Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences. For example, if an individual Peter is an instance of the class Student, and Student is a subclass of Person, then one can derive that Peter is also an instance of Person in a similar way as it happens for RDFS. However, OWL is much more expressive than RDFS, as the decision problems for OWL are in higher complexity classes [51] than for RDFS. In Fig. 2 we report the various OWL languages, their computational complexity and as subscript the DL their relate to [6,26].

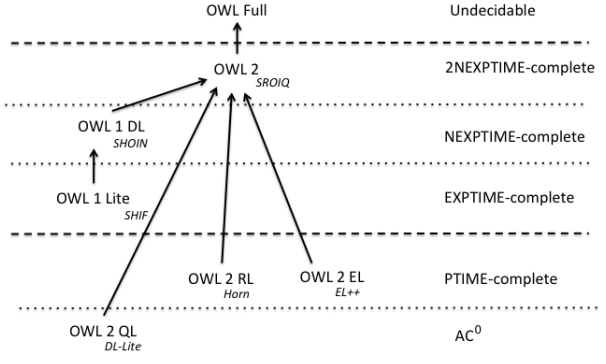


Fig. 2. OWL family and complexity

OWL 2 [23,50] is an update of OWL 1 adding several new features, including an increased expressive power. OWL 2 also defines several *OWL 2 profiles*, i.e. OWL 2 language subsets that may better meet certain computational complexity requirements or may be easier to implement. The choice of which profile to use in practice will depend on the structure of the ontologies and the reasoning tasks at hand. The OWL 2 profiles are:

OWL 2 EL is particularly useful in applications employing ontologies that contain very large numbers of properties and/or classes (basic reasoning problems can be performed in time that is polynomial with respect to the size of the ontology [5]). The EL acronym reflects the profile's basis in the $\mathcal{EL}$ family of description logics [5].

OWL 2 QL is aimed at applications that use very large volumes of instance data, and where query answering is the most important reasoning task. In OWL 2 QL, conjunctive query answering can be implemented using conventional relational database systems. Using a suitable reasoning technique,

sound and complete conjunctive query answering can be performed in LOGSPACE with respect to the size of the data (assertions) [4,21]. The QL acronym reflects the fact that query answering in this profile can be implemented by rewriting queries into a standard relational Query Language such as SQL [72].

OWL 2 RL is aimed at applications that require scalable reasoning without sacrificing too much expressive power. OWL 2 RL reasoning systems can be implemented using rule-based reasoning engines as a mapping to *Logic Programming* [43], specifically *Datalog* [72], exists. The RL acronym reflects the fact that reasoning in this profile can be implemented using a standard rule language [30]. The computational complexity is the same as for Datalog [25] (polynomial in the size of the data, EXPTIME w.r.t. the size of the knowledge base).

RIF Family. The *Rule Interchange Format* (RIF) aims at becoming a standard for exchanging rules, such as

```
Forall ?Buyer ?Item ?Seller
   buy(?Buyer ?Item ?Seller) :- sell(?Seller ?Item ?Buyer)
```

"Someone buys an item from a seller if the seller sells that item to the buyer"

among rule systems, in particular among Web rule engines. RIF is in fact a family of languages, called *dialects*, among which the most significant are:

RIF-BLD The *Basic Logic Dialect* is the main logic-based dialect. Technically, this dialect corresponds to Horn logic with various syntactic and semantic extensions. The main syntactic extensions include the frame syntax and predicates with named arguments. The main semantic extensions include datatypes and externally defined predicates.

RIF-PRD The *Production Rule Dialect* aims at capturing the main aspects of various production rule systems. Production rules, as they are currently practiced in main-stream systems like Jess[2] or JRules[3], are defined using ad hoc computational mechanisms, which are not based on a logic. For this reason, RIF-PRD is not part of the suite of logical RIF dialects and stands apart from them. However, significant effort has been extended to ensure as much sharing with the other dialects as possible. This sharing was the main reason for the development of the RIF Core dialect;

RIF-Core The *Core Dialect* is a subset of both RIF-BLD and RIF-PRD, thus enabling limited rule exchange between logic rule dialects and production rules. RIF-Core corresponds to Horn logic without function symbols (i.e., Datalog) with a number of extensions to support features such as objects and frames as in F-logic [38].

RIF-FLD The *Framework for Logic Dialects* is not a dialect in its own right, but rather a general logical extensibility framework. It was introduced in order to

[2] `http://www.jessrules.com/`
[3] `http://www.ilog.com/products/jrules/`

drastically lower the amount of effort needed to define and verify new logic dialects that extend the capabilities of RIF-BLD.

3 Mathematical Fuzzy Logic Basics

Given that SWLs are grounded on Mathematical Logic, it is quite natural to look at *Mathematical Fuzzy Logic* [36] to get inspiration for a fuzzy logic extensions of SWLs. So, we recap here briefly that in Mathematical Fuzzy Logic, 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 $[0,1]$. This degree is called *degree of truth* (or *score*) of the logical statement ϕ in the interpretation $\mathcal{I}$. In this section, *fuzzy statements* have the form $\phi\colon r$, where $r \in [0,1]$ (see, e.g. [35,36]) and ϕ is a statement, which encode that the degree of truth of ϕ is *greater or equal* r. A *fuzzy interpretation* $\mathcal{I}$ maps each basic statement p_i into $[0,1]$ and is then extended inductively to all statements:

$$\begin{aligned} \mathcal{I}(\phi \wedge \psi) &= \mathcal{I}(\phi) \otimes \mathcal{I}(\psi) \quad , \quad \mathcal{I}(\phi \vee \psi) = \mathcal{I}(\phi) \oplus \mathcal{I}(\psi) \\ \mathcal{I}(\phi \rightarrow \psi) &= \mathcal{I}(\phi) \Rightarrow \mathcal{I}(\psi) \quad , \quad \mathcal{I}(\neg\phi) = \ominus \mathcal{I}(\phi) \\ \mathcal{I}(\exists x.\phi(x)) &= \sup_{a \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(a)) \quad , \quad \mathcal{I}(\forall x.\phi(x)) = \inf_{a \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(a)) \ , \end{aligned}$$

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 [40]. Usually, the implication function $\Rightarrow$ is defined as *r-implication*, that is, $a \Rightarrow b = \sup\{c \mid a \otimes c \leq b\}$. The notions of satisfiability and logical consequence are defined in the standard way. A fuzzy interpretation $\mathcal{I}$ *satisfies* a fuzzy statement $\phi\colon r$ or $\mathcal{I}$ is a *model* of $\phi\colon r$, denoted $\mathcal{I} \models \phi\colon r$ iff $\mathcal{I}(\phi) \geq r$.

One usually distinguishes three different logics, namely *Łukasiewicz, Gödel*, and *Product logics* [36], whose combination functions are reported in Table 1. Zadeh logic, namely $a \otimes b = \min(a,b)$, $a \oplus b = \max(a,b)$, $\ominus a = 1 - a$ and $a \Rightarrow b = \max(1-a,b)$, is entailed by Łukasiewicz logic, as $\min(a,b) = a \otimes (a \Rightarrow b)$ and $\max(a,b) = 1 - \min(1-a, 1-b)$. Table 2 and 3 report axioms these functions have to satisfy. Table 4 recalls some salient properties of the various fuzzy logics. Worth noting is that a fuzzy logic satisfying all the listed properties has

Table 1. Combination functions of various fuzzy logics

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

Table 2. Properties for t-norms and s-norms

Axiom Name	T-norm	S-norm
Tautology / Contradiction	$a \otimes 0 = 0$	$a \oplus 1 = 1$
Identity	$a \otimes 1 = a$	$a \oplus 0 = a$
Commutativity	$a \otimes b = b \otimes a$	$a \oplus b = b \oplus a$
Associativity	$(a \otimes b) \otimes c = a \otimes (b \otimes c)$	$(a \oplus b) \oplus c = a \oplus (b \oplus c)$
Monotonicity	if $b \leq c$, then $a \otimes b \leq a \otimes c$	if $b \leq c$, then $a \oplus b \leq a \oplus c$

Table 3. Properties for implication and negation functions

Axiom Name	Implication Function	Negation Function
Tautology / Contradiction	$0 \Rightarrow b = 1,\ \ a \Rightarrow 1 = 1,\ \ 1 \Rightarrow 0 = 0$	$\ominus 0 = 1,\ \ \ominus 1 = 0$
Antitonicity	if $a \leq b$, then $a \Rightarrow c \geq b \Rightarrow c$	if $a \leq b$, then $\ominus a \geq \ominus b$
Monotonicity	if $b \leq c$, then $a \Rightarrow b \leq a \Rightarrow c$	

necessarily to collapse to the Boolean, two-valued, case. As a note, [29] claimed that fuzzy logic collapses to boolean logic, but didn't recognise that to prove it, all the properties of Table 4 have been used. Additionally, we have the following inferences: let $a \geq n$ and $a \Rightarrow b \geq m$. Then, under Kleene-Dienes implication, we infer that "if $n > 1 - m$ then $b \geq m$". More importantly, to what concerns our paper, is that under an r-implication relative to a t-norm $\otimes$, we have that

$$\text{from } a \geq n \text{ and } a \Rightarrow b \geq m, \text{ we infer } b \geq n \otimes m\ . \tag{1}$$

To see this, as $a \geq n$ and $a \Rightarrow b = \sup\{c \mid a \otimes c \leq b\} = \bar{c} \geq m$ it follows that $b \geq a \otimes \bar{c} \geq n \otimes m$. In a similar way, under an r-implication relative to a t-norm $\otimes$, we have that

$$\text{from } a \Rightarrow b \geq n \text{ and } b \Rightarrow c \geq m, \text{ we infer that } a \Rightarrow c \geq n \otimes m\ . \tag{2}$$

We say $\phi\colon n$ is a *tight logical consequence* of a set of fuzzy statements $\mathcal{K}$ iff n is the infimum of $\mathcal{I}(\phi)$ subject to all models $\mathcal{I}$ of $\mathcal{K}$. Notice that the latter is

Table 4. Some additional properties of combination functions of various fuzzy logics

Property	Łukasiewicz Logic	Gödel Logic	Product Logic	Zadeh Logic
$x \otimes \ominus x = 0$	+	+	+	−
$x \oplus \ominus x = 1$	+	−	−	−
$x \otimes x = x$	−	+	−	+
$x \oplus x = x$	−	+	−	+
$\ominus \ominus x = x$	+	−	−	+
$x \Rightarrow y = \ominus x \oplus y$	+	−	−	+
$\ominus (x \Rightarrow y) = x \otimes \ominus y$	+	−	−	+
$\ominus (x \otimes y) = \ominus x \oplus \ominus y$	+	+	+	+
$\ominus (x \oplus y) = \ominus x \otimes \ominus y$	+	+	+	+

equivalent to $n = \sup\{r \mid \mathcal{K} \models \phi\colon r\}$. n is called the *best entailment degree* of ϕ w.r.t. $\mathcal{K}$ (denoted $bed(\mathcal{K}, \phi)$), i.e.

$$bed(\mathcal{K}, \phi) = \sup\{r \mid \mathcal{K} \models \phi\colon r\} .$$

On the other hand, the *best satisfiability degree* of ϕ w.r.t. $\mathcal{K}$ (denoted $bsd(\mathcal{K}, \phi)$) is

$$bsd(\mathcal{K}, \phi) = \sup_{\mathcal{I}}\{\mathcal{I}(\phi) \mid \mathcal{I} \models \mathcal{K}\} .$$

We refer the reader to [34,35,36] for reasoning algorithms for fuzzy propositional and First-Order Logics. For illustrative purpose, we recap here a simple method to determine $bed(\mathcal{K}, \phi)$ and $bsd(\mathcal{K}, \phi)$ via Mixed Integer Linear Programming (MILP) for the case of propositional Łukasiewicz logic. To this end, it can be shown that

$$\begin{aligned} bed(\mathcal{K}, \phi) &= \min x. \text{ such that } \mathcal{K} \cup \{\neg\phi\colon 1 - x\} \text{ satisfiable} \\ bsd(\mathcal{K}, \phi) &= \max x. \text{ such that } \mathcal{K} \cup \{\phi\colon x\} \text{ satisfiable} . \end{aligned}$$

Now, for a formula ϕ consider a variable x_ϕ (with intended meaning: the degree of truth of ϕ is greater or equal to x_ϕ). Now we apply the following transformation σ that generates a set of MILP in-equations:

$$\begin{aligned} bed(\mathcal{K}, \phi) = \min x. \text{ such that } & x \in [0,1], x_{\neg\phi} \geq 1 - x, \sigma(\neg\phi), \\ & \text{for all } \phi' \geq n \in \mathcal{K}, x_{\phi'} \geq n, \sigma(\phi'), \end{aligned}$$

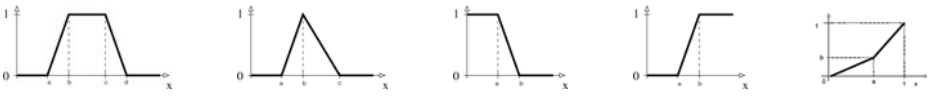
$$\sigma(\phi) = \begin{cases} x_p \in [0,1] & \text{if } \phi = p \\ x_\phi = 1 - x_{\phi'}, x_\phi \in [0,1] & \text{if } \phi = \neg\phi' \\ x_{\phi_1} \otimes x_{\phi_2} \geq x_\phi, \\ \sigma(\phi_1), \sigma(\phi_2), x_\phi \in [0,1] & \text{if } \phi = \phi_1 \wedge \phi_2 \\ x_{\phi_1} \oplus x_{\phi_2} \geq x_\phi & \text{if } \phi = \phi_1 \vee \phi_2 \\ \sigma(\neg\phi_1 \vee \phi_2) & \text{if } \phi = \phi_1 \Rightarrow \phi_2 . \end{cases}$$

In the definition above, $z \leq x_1 \oplus x_2$ and $z \leq x_1 \otimes x_2$, with $0 \leq x_i, z \leq 1$, can be encoded as the sets of constraints:

$$\begin{aligned} & z \leq x_1 \oplus x_2 \mapsto \{z \leq x_1 + x_2\}, \\ & z \leq x_1 \otimes x_2 \mapsto \{y \leq 1 - z, x_1 + x_2 - 1 \geq z - y, y \in \{0,1\}\} . \end{aligned}$$

As the set of constraints is linearly bounded by $\mathcal{K}$ and as MILP satisfiability is NP-complete, we get the well-known result that determining the best entailment/satisfiability degree is NP-complete for propositional Łukasiewicz logic.

We conclude with the notion of *fuzzy set* [76]. A *fuzzy set* R over a countable crisp set X is a function $R\colon X \to [0,1]$. The *degree of subsumption* between two fuzzy sets A and B, denoted $A \sqsubseteq B$, is defined as $\inf_{x \in X} A(x) \Rightarrow B(x)$, where $\Rightarrow$ is an implication function. Note that if $A(x) \leq B(x)$, for all $x \in [0,1]$, then $A \sqsubseteq B$ evaluates to 1. Of course, $A \sqsubseteq B$ may evaluate to a value $v \in (0,1)$ as well.



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

Fig. 3. (a) Trapezoidal function $trz(a, b, c, d)$, (b) triangular function $tri(a, b, c)$, (c) left shoulder function $ls(a, b)$, (d) right shoulder function $rs(a, b)$ and (e) linear modifier $lm(a, b)$

A (binary) *fuzzy relation* R over two countable crisp sets X and Y is a function $R\colon X \times Y \to [0, 1]$. The *inverse* of R is the function $R^{-1}\colon Y \times X \to [0, 1]$ with membership function $R^{-1}(y, x) = R(x, y)$, for every $x \in X$ and $y \in Y$. The *composition* of two fuzzy relations $R_1\colon X \times Y \to [0, 1]$ and $R_2\colon Y \times Z \to [0, 1]$ is defined as $(R_1 \circ R_2)(x, z) = \sup_{y \in Y} R_1(x, y) \otimes R_2(y, z)$. A fuzzy relation R is *transitive* iff $R(x, z) \geqslant (R \circ R)(x, z)$.

Eventually, the trapezoidal (Fig. 3 (a)), the triangular (Fig. 3 (b)), the L-function (left-shoulder function, Fig. 3 (c)), and the R-function (right-shoulder function, Fig. 3 (d)) are frequently used to specify membership degrees. For instance, the left-shoulder function is defined as

$$ls(x; a, b) = \begin{cases} 1 & \text{if } x \leq a \\ 0 & \text{if } x \geq b \\ (b - x)/(b - a) & \text{if } x \in [a, b] \end{cases} \tag{3}$$

4 Fuzzy Logic and Semantic Web Languages

We have seen in the previous section how to "fuzzyfy" a classical language such as propositional logic and FOL, namely fuzzy staements are of the form $\phi\colon n$, where ϕ is a statement and $n \in [0, 1]$.

The natural extension to SWLs consists then in replacing ϕ with appropriate expressions belonging to the logical counterparts of SWLs, namely ρdf, DLs and LPs, as we will illustrate next.

4.1 Fuzzy RDFS

In *Fuzzy RDFS* (see [69] and references therein), triples are annotated with a degree of truth in $[0, 1]$. For instance, "Rome is a big city to degree 0.8" can be represented with $(Rome, \mathsf{type}, BigCity)\colon 0.8$. More formally, *fuzzy triples* are expressions of the form $\tau\colon n$, where τ is a RDFS triple (the truth value n may be omitted and, in that case, the value $n = 1$ is assumed).

The interesting point is that from a computational point of view the inference rules parallel those for "crisp" RDFS: indeed, the rules are of the form

$$\frac{\tau_1\colon n_1,\ \ldots,\ \tau_k\colon n_k, \{\tau_1, \ldots, \tau_k\} \vdash_{\mathsf{RDFS}} \tau}{\tau\colon \bigotimes_i n_i} \tag{4}$$

Essentially, this rule says that if a classical RDFS triple τ can be inferred by applying a classical RDFS inference rule to triples $\tau_1, \ldots, \tau_k$ (denoted $\{\tau_1, \ldots, \tau_k\} \vdash_{\mathsf{RDFS}} \tau$), then the truth degree of τ will be $\bigotimes_i n_i$.

As a consequence, the rule system is quite easy to implement for current inference systems. Specifically, as for the crisp case, one may compute the closure $cl(\mathcal{K})$ of a set of fuzzy triples $\mathcal{K}$, store them in a relational database and thereafter query the database.

Concerning the query language, *SPARQL* [55] is the current standard, but a new version (*SPARQL 1.1*) is close to be finalised [56]. From a logical point of view, a SPARQL query may be seen as a *Conjunctive Query* (CQ), or an union of them, a well-known notion in database theory [2]. Specifically, an *RDF query* is of the rule-like form

$$q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y}) \ , \tag{5}$$

where $q(\boldsymbol{x})$ is the *head* and $\exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y})$ is the *body* of the query, which is a conjunction (we use the symbol ",″ to denote conjunction in the rule body) of triples τ_i $(1 \leq i \leq n)$. $\boldsymbol{x}$ is a vector of variables occurring in the body, called the *distinguished variables*, $\boldsymbol{y}$ are so-called *non-distinguished variables* and are distinct from the variables in $\boldsymbol{x}$, each variable occurring in τ_i is either a distinguished or a non-distinguished variable. If clear from the context, the existential quantification $\exists \boldsymbol{y}$ may be omitted. In a query, built-in triples of the form (s, p, o) are allowed, where p is a built-in predicate taken from a reserved vocabulary and having a *fixed interpretation*. Built-in predicates are generalised to any n-ary predicate p. For convenience, "functional predicates"[4] are written as *assignments* of the form $x{:=}f(\boldsymbol{z})$ and it is assumed that the function $f(\boldsymbol{z})$ is safe (also non functional built-in predicate $p(\boldsymbol{z})$ should be safe as well). A query example is:

$$q(x, y) \leftarrow (y, created, x), (y, \mathsf{type}, Italian), (x, exhibitedAt, Uffizi) \tag{6}$$

having intended meaning to retrieve all the artefacts x created by Italian artists y, being exhibited at Uffizi Gallery.

Roughly, the *answer set* of a query q w.r.t. a set of tuples $\mathcal{K}$ (denoted $ans(\mathcal{K}, q)$) is the set of tuples $\boldsymbol{t}$ such that there exists $\boldsymbol{t'}$ such that the instantiation $\varphi(\boldsymbol{t}, \boldsymbol{t'})$ of the query body is true in the closure of $\mathcal{K}$, i.e., all triples in $\varphi(\boldsymbol{t}, \boldsymbol{t'})$ are in $cl(\mathcal{K})$.

Once we switch to the fuzzy setting, queries are similar as for the crisp case, except that fuzzy triples are used in the query body in place of crisp triples. A special attention is required to the fact that now all answers are graded and, thus, an order is induced on the answer set. Specifically, a *fuzzy query* is of the form

$$q(\boldsymbol{x}) \colon s \leftarrow \exists \boldsymbol{y}.\tau_1 \colon s_1, \ldots, \tau_n \colon s_n, s{:=}f(\boldsymbol{s}, \boldsymbol{x}, \boldsymbol{y}) \ , \tag{7}$$

where now additionally s_i is the score of triple τ_i and the final score s of triple $\boldsymbol{x}$ is computed according to a user function f applied to variables occurring in the query body. For instance, the query

$$q(x) \colon s \leftarrow (x, \mathsf{type}, SportsCar) \colon s_1, (x, hasPrice, y), s = s_1 \cdot cheap(y) \tag{8}$$

[4] A predicate $p(\boldsymbol{x}, y)$ is functional if for any $\boldsymbol{t}$ there is *unique* t' for which $p(\boldsymbol{t}, t')$ is true.

where e.g. $cheap(y) = ls(20000, 30000)(y)$, has intended meaning to retrieve all cheap sports car. Then, any answer is scored according to the product of being cheap and a sports car.

It is not difficult to see that indeed fuzzy CQs can easily be mapped into SQL as well. For further details see [69].

Annotation Domains and RDFS. We have seen that fuzzy RDFS extends triples with an *annotation* $n \in [0, 1]$. Interestingly, we may further generalise fuzzy RDFS, by allowing a triple being annotated with a value λ taken from a so-called *annotation domain* [3,20,48,70][5], which allow to deal with several domains (such as, fuzzy, temporal, provenace) and their combination, in a uniform way. Formally, let us consider a non-empty set L. Elements in L are our annotation values. For example, in a fuzzy setting, $L = [0, 1]$, while in a typical temporal setting, L may be time points or time intervals. In the annotation framework, an interpretation will map statements to elements of the annotation domain. Now, an *annotation domain* for RDFS is an idempotent, commutative semi-ring

$$D = \langle L, \oplus, \otimes, \bot, \top \rangle \ ,$$

where $\oplus$ is $\top$-annihilating [20]. That is, for $\lambda, \lambda_i \in L$

1. $\oplus$ is idempotent, commutative, associative;
2. $\otimes$ is commutative and associative;
3. $\bot \oplus \lambda = \lambda$, $\top \otimes \lambda = \lambda$, $\bot \otimes \lambda = \bot$, and $\top \oplus \lambda = \top$;
4. $\otimes$ is distributive over $\oplus$, i.e.$\lambda_1 \otimes (\lambda_2 \oplus \lambda_3) = (\lambda_1 \otimes \lambda_2) \oplus (\lambda_1 \otimes \lambda_3)$;

It is well-known that there is a natural partial order on any idempotent semiring: an annotation domain $D = \langle L, \oplus, \otimes, \bot, \top \rangle$ induces a partial order $\preceq$ over L defined as:

$$\lambda_1 \preceq \lambda_2 \ \text{ if and only if } \lambda_1 \oplus \lambda_2 = \lambda_2 \ .$$

The order $\preceq$ is used to express redundant/entailed/subsumed information. For instance, for temporal intervals, an annotated triple $(s, p, o) : [2000, 2006]$ entails $(s, p, o) : [2003, 2004]$, as $[2003, 2004] \subseteq [2000, 2006]$ (here, $\subseteq$ plays the role of $\preceq$).

Remark 1. $\oplus$ is used to combine information about the same statement. For instance, in temporal logic, from $\tau : [2000, 2006]$ and $\tau : [2003, 2008]$, we infer $\tau : [2000, 2008]$, as $[2000, 2008] = [2000,\ 2006] \cup [2003, 2008]$; here, $\cup$ plays the role of $\oplus$. In the fuzzy context, from $\tau : 0.7$ and $\tau : 0.6$, we infer $\tau : 0.7$, as $0.7 = \max(0.7, 0.6)$ (here, max plays the role of $\oplus$).

Remark 2. $\otimes$ is used to model the "conjunction" of information. In fact, a $\otimes$ is a generalisation of boolean conjunction to the many-valued case. In fact, $\otimes$ satisfies also that

[5] The readers familiar with the annotated logic programming framework [39], will notice the similarity of the approaches.

1. $\otimes$ is bounded: i.e.$\lambda_1 \otimes \lambda_2 \preceq \lambda_1$.
2. $\otimes$ is $\preceq$-monotone, i.e. for $\lambda_1 \preceq \lambda_2$, $\lambda \otimes \lambda_1 \preceq \lambda \otimes \lambda_2$

For instance, on interval-valued temporal logic, from $(a, \mathsf{sc}, b)\colon [2000, 2006]$ and $(b, \mathsf{sc}, c)\colon [2003, 2008]$, we will infer $(a, \mathsf{sc}, c)\colon [2003, 2006]$, as $[2003, 2006] = [2000, 2006] \cap [2003, 2008]$; here, $\cap$ plays the role of $\otimes$.[6] In the fuzzy context, one may chose any t-norm [36,40], e.g.product, and, thus, from $(a, \mathsf{sc}, b)\colon 0.7$ and $(b, \mathsf{sc}, c)\colon 0.6$, we will infer $(a, \mathsf{sc}, c)\colon 0.42$, as $0.42 = 0.7 \cdot 0.6$) (here, $\cdot$ plays the role of $\otimes$).

Remark 3. Observe that the distributivity condition is used to guarantee that e.g. we obtain the same annotation $\lambda \otimes (\lambda_2 \oplus \lambda_3) = (\lambda_1 \otimes \lambda_2) \oplus (\lambda_1 \otimes \lambda_3)$ of the triple (a, sc, c) that can be inferred from triples $(a, \mathsf{sc}, b)\colon \lambda_1$, $(b, \mathsf{sc}, c)\colon \lambda_2$ and $(b, \mathsf{sc}, c)\colon \lambda_3$.

The use of annotation domains appears to be quite appealing as

1. it applies to several domains, such as the fuzzy domain, the temporal domain, provenance, trust and any combination of them [3];
2. from an inference point of view, the rules are conceptually the same as for the fuzzy case: indeed, just replace in Rule 4, the values n_i with λ_i, i.e.

$$\frac{\tau_1\colon \lambda_1,\ \ldots,\ \tau_k\colon \lambda_k, \{\tau_1, \ldots, \tau_k\} \vdash_{\mathsf{RDFS}} \tau}{\tau\colon \bigotimes_i \lambda_i} \tag{9}$$

3. annotated conjunctive queries are as fuzzy queries, except that now variables s and s_i range over L in place of $[0, 1]$;
4. a query answering procedure is similar as for the fuzzy case: compute the closure, store it on a relation database and transform an annotated CQ into a SQL query.

From a computational complexity point of view, it is the same as for crisp RDFS plus the cost of $\otimes$, $\oplus$ and the scoring function f in the body of a query. A prototype implementation is available from `http://anql.deri.org/`.

4.2 Fuzzy OWL

Description Logics. (DLs) [6] are the logical counterpart of the family of OWL languages. So, to illustrate the basic concepts of fuzzy OWL, it suffices to show the fuzzy DL case (see [45], for a survey). Briefly, one starts from a classical DL, and attaches to the basic statements a degree $n \in [0, 1]$, similarly as we did for fuzzy RDFS. As a matter of example, consider the DL $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplement), a major DL representative used to introduce new extensions to DLs: the table below shows its syntax, semantics and provides examples.

[6] As we will see, $\oplus$ and $\otimes$ may be more involved.

Syntax		*Semantics*	*Example*
$C, D \rightarrow$	$\top$	$\top(x)$	
	$\bot$	$\bot(x)$	
	A	$A(x)$	$Human$
	$C \sqcap D$	$C(x) \wedge D(x)$	$Human \sqcap Male$
	$C \sqcup D$	$C(x) \vee D(x)$	$Nice \sqcup Rich$
	$\neg C$	$\neg C(x)$	$\neg Meat$
	$\exists R.C$	$\exists y.R(x,y) \wedge C(y)$	$\exists has\_child.Blond$
	$\forall R.C$	$\forall y.R(x,y) \Rightarrow C(y)$	$\forall has\_child.Human$
$C \sqsubseteq D$		$\forall x.C(x) \Rightarrow D(x)$	$Happy\_Father \sqsubseteq Man \sqcap \exists has\_child.Female$
a:C		$C(a)$	$John$:$Happy\_Father$
(a,b):R		$R(a,b)$	$(John, Mary)$:$Loves$

The upper pane describes how *concepts/classes* can be formed, while the lower pane shows the form of *statements/formulae* a knowledge base may be build of. Statements of the form $C \sqsubseteq D$, called, *General Inclusion Axioms* (GCIs), dictated that the class C is a subclass of the class D, a:C dictates that individual a is an instance of class C, while (a,b):R states that $\langle a, b\rangle$ is an instance of the binary relation R. The definition $A = C$, is used in place of having both $A \sqsubseteq C$ and $C \sqsubseteq A$, stating that class A is defined to be equivalent to C.

Fuzzy DLs [58,64,45] are then obtained by interpreting the statements as fuzzy FOL formulae and attaching a weight n to DL statements, yielding *fuzzy DL statements*, such as

$$C \sqsubseteq D\colon n \ , \ a\!:\!C\colon n \ \text{ and } \ (a,b)\!:\!R\colon n \ .$$

A notable difference to fuzzy RDFS is that one may use additionally some special constructs to enhance the expressivity of fuzzy DLs [12,15,16,60], these include

- fuzzy modifiers applied to concepts, such as

  $$NiceVeryExpensiveItem = Nice \sqcap very(ExpensiveItem)$$

  defining the class of nice and very expensive items, where $Nice$ and $ExpensiveItem$ are classes/concepts and $very$ is a linear modifier, such as $ln(x, 0.7, 0.3)$;
- the possibility of defining *fuzzy concrete concepts* [60], i.e. concepts having a specific fuzzy membership function, e.g., allowing a definition for $ExpensiveItem$

  $$ExpensiveItem = Item \sqcap \exists hasPrice.HighPrice$$
  $$HighPrice = rs(100, 200)$$

- various forms of *concept aggregations* [15] using so-called *Aggregation Operators* (AOs). These are mathematical functions that are used to combine information [71]. The arithmetic mean, the weighted sum, the median and, more generally Ordered Weighted Averaging (OWA) [75] are the most well-known AOs. For instance,

  $$Hotel \sqcap (0.3 \cdot Cheap + 0.5 \cdot CloseToVenue + 0.2 \cdot Comfortable) \sqsubseteq GoodHotel \quad (10)$$

may be used to define a sufficient condition for a good hotel as a weighted sum of being cheap, close to the venue and comfortable ($Cheap, CloseToVenue$ and $Comfortable$ are classes here).

From a decision procedure point of view, one may proceed similarly as for the best entailment degree problem for fuzzy propositional logic. That is, the decision procedure consists of a set of inference rules that generate a set of in-equations (that depend on the t-norm and fuzzy concept constructors) that have to be solved by an operational research solver (see, e.g. [14,60]). An informal rule example is as follows:

> "If individual a is instance of the class intersection $C_1 \sqcap C_2$ to degree greater or equal to $x_{a:C_1 \sqcap C_2}$[7], then a is instance of C_i $(i = 1, 2)$ to degree greater or equal to $x_{a:C_i}$, where additionally the following in-equation holds:
>
> $$x_{a:C_1 \sqcap C_2} \leq x_{a:C_1} \otimes x_{a:C_2} \ .$$"

Note that for Zadeh Logic and Łukasiewicz Logic a MILP solver is enough to determine whether the set of in-equations has a solution or not.

However, recently there have been some unexpected surprises [7,8,9,22]. [9] shows that $\mathcal{ALC}$ with GCIs *(i)* does not have the finite model property under Łukasiewicz Logic or Product Logic, contrary to the classical case; *(ii)* illustrates that some algorithms are neither complete not correct; and *(iii)* shows some interesting conditions under which decidability is still guaranteed. [7,8] show that knowledge base satisfiability is an undecidable problem for Product Logic. The same holds for Łukasiewicz Logic as well [22]. In case the truth-space is finite and defined a priori, decidability is guaranteed (see, e.g. [13,11,59]).

Some fuzzy DLs solvers are: *fuzzyDL* [12], *Fire* [57], *GURDL* [32], *DeLorean* [10], *GERDS* [33], and *YADLR* [41]. There is also a proposal to use OWL 2 itself to represent fuzzy ontologies [16]. More precisely, [16] identifies the syntactic differences that a fuzzy ontology language has to cope with, and shows how to encode them using OWL 2 annotation properties. The use of annotation properties makes possible *(i)* to use current OWL 2 editors for fuzzy ontology representation, *(ii)* that OWL 2 reasoners discard the fuzzy part of a fuzzy ontology, producing almost the same results as if it would not exist; and *(ii)* an implementation is provided as a Protégé plug-in.

Eventually, as for RDFS, the notion of conjunctive query straightforwardly extends to DLs and to fuzzy DLs as well: in the classical DL case, a query is of the form (compare to Eq. (5))

$$q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y}) \ , \tag{11}$$

where now $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of unary and binary predicates. For instance, the DL analogue of the RDFS query (6) is

$$q(x, y) \leftarrow Created(y, x), Italian(y), ExhibitedAt(x, uffizi) \ . \tag{12}$$

[7] As for the fuzzy propositional case, for a fuzzy DL formula ϕ we consider a variable x_ϕ with intended meaning: the degree of truth of ϕ is greater or equal to x_ϕ.

Similarly, a *fuzzy DL query* is of the form (compare to Eq. (7))

$$q(\boldsymbol{x}) \colon s \leftarrow \exists \boldsymbol{y}.A_1 \colon s_1, \ldots, A_n \colon s_n, s \coloneqq f(\boldsymbol{s}, \boldsymbol{x}, \boldsymbol{y}) \ , \tag{13}$$

where now A_i is either an unary or binary predicate. For instance, the fuzzy DL analogue of the RDFS query (8) is

$$q(x) \colon s \leftarrow SportsCar(x) \colon s_1, HasPrice(x, y), s \coloneqq s_1 \cdot cheap(y) \ . \tag{14}$$

Annotation Domains and OWL. The generalisation of fuzzy OWL to the case in which an annotation $n \in [0, 1]$ is replaced with an annotation value λ taken from an annotation domain proceeds as for RDFS, except that now the annotation domain has the form of a complete lattice [63].

From a computational complexity point of view, similar results hold as for the $[0, 1]$ case [17,18,63]. While [63] provides a decidability result in case the lattice is finite, [17] further improves the decidability result by characterising the computational complexity of KB satisfiability problem for $\mathcal{ALC}$ with GCIs over finite lattices being EXPTIME-complete, as for the crisp variant, while [18] shows that the KB satisfiability problem for $\mathcal{ALC}$ with GCIs over non finite lattices is undecidable.

4.3 Fuzzy RIF

The foundation of the core part of RIF is *Datalog* [72], i.e. a Logic Programming Language (LP) [43]. In LP, the management of imperfect information has attracted the attention of many researchers and numerous frameworks have been proposed. Addressing all of them is almost impossible, due to both the large number of works published in this field (early works date back to early 80-ties [54]) and the different approaches proposed.

Basically [43], a Datalog program $\mathcal{P}$ is made out by a set of rules and a set of facts. *Facts* are ground *atoms* of the form $P(\boldsymbol{c})$. On the other hand rules are similar as conjunctive DL queries and are of the form

$$A(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}.\varphi(\boldsymbol{x}, \boldsymbol{y}) \ ,$$

where now $\varphi(\boldsymbol{x}, \boldsymbol{y})$ is a conjunction of n-ary predicates. In Datalog it is further assumed that no fact predicate may occur in a rule head (facts are the so-called extensional database, while rules are the intentional database). A *query* is a rule and the *answer set* of a query q w.r.t. a set $\mathcal{K}$ of facts and rules (denoted $ans(\mathcal{K}, q)$) is the set of tuples $\boldsymbol{t}$ such that there exists $\boldsymbol{t'}$ such that the instantiation $\varphi(\boldsymbol{t}, \boldsymbol{t'})$ of the query body is true in *minimal model* of $\mathcal{K}$, which is guaranteed to exists.

As pointed out, there are several proposals for fuzzy Datalog (see [68] for an extensive list). However, a sufficiently general form is obtained in case facts are graded with $n \in [0, 1]$, i.e. facts are of the form $P(\boldsymbol{c}) \colon n$ and rules generalise fuzzy DL queries (compare to Eq. (13)): i.e., a *fuzzy rule* is of the form

$$A(\boldsymbol{x}) \colon s \leftarrow \exists \boldsymbol{y}.A_1 \colon s_1, \ldots, A_n \colon s_n, s \coloneqq f(\boldsymbol{s}, \boldsymbol{x}, \boldsymbol{y}) \ , \tag{15}$$

where now A_i is an n-ary predicate. For instance, the fuzzy GCI in Eq. (10), can be expressed easily as the fuzzy rule

$$GoodHotel(x)\colon s \leftarrow Hotel(x), Cheap(x)\colon s_1, CloseToVenue(x)\colon s_2, \\ Comfortable(x)\colon s_3, s := 0.3 \cdot s_1 + 0.5 \cdot s_2 + 0.2 \cdot s_3 \quad (16)$$

A *fuzzy query* is a fuzzy rule and, informally, the *fuzzy answer set* is the ordered set of weighted tuples $\langle \boldsymbol{t}, s \rangle$ such that all the fuzzy atoms in the rule body are true in the minimal model and s is the result of the scoring function f applied to its arguments. The existence of a minimal is guaranteed if the scoring functions in the query and in the rule bodies are *monotone* [68].

We conclude by saying that most works deal with logic programs without negation and some may provide some technique to answer queries in a top-down manner, as e.g. [24,39,42,73,61]. Deciding whether a wighted tuple $\langle \boldsymbol{t}, s \rangle$ is the answer set is undecidable in general, though is decidable if the truth space is finite and fixed a priory, as then the minimal model is finite.

Another rising problem is the problem to compute the top-k ranked answers to a query, without computing the score of all answers. This allows to answer queries such as "find the top-k closest hotels to the conference location". Solutions to this problem can be found in [44,66,67].

Annotation Domains and RIF. The generalisation of fuzzy RIF to the case in which an annotation $n \in [0, 1]$ is replaced with an annotation value λ taken from an annotation domain is straightforward and proceeds as for RDFS. From a computational complexity point of view, similarly to the fuzzy case, deciding whether a wighted tuple $\langle \boldsymbol{t}, \lambda \rangle$ is the answer set is undecidable in general, though is decidable if the annotation domain is finite.

5 Conclusions

We have provided a "crash course" through the realm of Semantic Web Languages, their fuzzy variants and their generalisation to annotation domains, by illustrating the basics of these languages, some issues, and related them to the logical formalisms on which they are based.

References

1. Abadi, D.J., Marcus, A., Madden, S., Hollenbach, K.: Sw-store: a vertically partitioned dbms for semantic web data management. VLDB J. 18(2), 385–406 (2009)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co., Reading (1995)
3. Zimmermann, A.P.A., Lopes, N., Straccia, U.: A general framework for representing, reasoning and querying with annotated semantic web data. Technical report, Computing Research Repository (2011), Available as CoRR technical report, at `http://arxiv.org/abs/1103.1255`

4. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. Journal of Artificial Intelligence Research 36, 1–69 (2009)
5. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005), pp. 364–369. Morgan-Kaufmann Publishers, Edinburgh (2005)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
7. Baader, F., Peñaloza, R.: Are fuzzy description logics with general concept inclusion axioms decidable? In: Proceedings of 2011 IEEE International Conference on Fuzzy Systems (Fuzz-IEEE 2011). IEEE Press, Los Alamitos (to appear, 2011)
8. Baader, F., Peñaloza, R.: Gcis make reasoning in fuzzy dl with the product t-norm undecidable. In: Proceedings of the 24th International Workshop on Description Logics (DL 2011), CEUR Electronic Workshop Proceedings (to appear, 2011)
9. Bobillo, F., Bou, F., Straccia, U.: On the failure of the finite model property in some fuzzy description logics. Fuzzy Sets and Systems 172(1), 1–12 (2011)
10. Bobillo, F., Delgado, M., Gómez-Romero, J.: Delorean: A reasoner for fuzzy OWL 1.1. In: Proceedings of the 4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2008), CEUR Workshop Proceedings, vol. 423 (October 2008)
11. Bobillo, F., Delgado, M., Gómez-Romero, J., Straccia, U.: Fuzzy description logics under gödel semantics. International Journal of Approximate Reasoning 50(3), 494–514 (2009)
12. Bobillo, F., Straccia, U.: fuzzyDL: An expressive fuzzy description logic reasoner. In: 2008 International Conference on Fuzzy Systems (FUZZ 2008), pp. 923–930. IEEE Computer Society, Los Alamitos (2008)
13. Bobillo, F., Straccia, U.: Towards a crisp representation of fuzzy description logics under Łukasiewicz semantics. In: An, A., Matwin, S., Raś, Z.W., Ślęzak, D. (eds.) Foundations of Intelligent Systems. LNCS (LNAI), vol. 4994, pp. 309–318. Springer, Heidelberg (2008)
14. Bobillo, F., Straccia, U.: Fuzzy description logics with general t-norms and datatypes. Fuzzy Sets and Systems 160(23), 3382–3402 (2009)
15. Bobillo, F., Straccia, U.: Aggregations operators and fuzzy owl 2. In: 2011 International Conference on Fuzzy Systems (FUZZ 2011). IEEE Computer Society, Los Alamitos (2011)
16. Bobillo, F., Straccia, U.: Fuzzy ontology representation using owl 2. International Journal of Approximate Reasoning (2011)
17. Borgwardt, S., Peñaloza, R.: Description logics over lattices with multi-valued ontologies. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI 2011 (to appear, 2011)
18. Borgwardt, S., Peñaloza, R.: Fuzzy ontologies over lattices with t-norms. In: Proceedings of the 24th International Workshop on Description Logics (DL 2011), CEUR Electronic Workshop Proceedings (to appear, 2011)
19. Brickley, D., Guha, R.V.: RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, W3C (2004), `http://www.w3.org/TR/rdf-schema/`
20. Buneman, P., Kostylev, E.: Annotation algebras for rdfs. In: The Second International Workshop on the role of Semantic Web in Provenance Management (SWPM 2010), CEUR Workshop Proceedings (2010)

21. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The dl-lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
22. Cerami, M., Straccia, U.: Undecidability of KB satisfiability for Ł-$\mathcal{ALC}$ with GCIs (July 2011) (Unpublished Manuscript)
23. Cuenca-Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. Journal of Web Semantics 6(4), 309–322 (2008)
24. Damásio, C.V., Medina, J., Ojeda Aciego, M.: A tabulation proof procedure for residuated logic programming. In: Proceedings of the 6th European Conference on Artificial Intelligence, ECAI 2004 (2004)
25. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Surveys 33(3), 374–425 (2001)
26. Description Logics Web Site, `http://dl.kr.org`
27. Dubois, D., Prade, H.: Can we enforce full compositionality in uncertainty calculi? In: Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI 1994), Seattle, Washington, pp. 149–154 (1994)
28. 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)
29. Elkan, C.: The paradoxical success of fuzzy logic. In: Proc. of the 11th Nat. Conf. on Artificial Intelligence (AAAI 1993), pp. 698–703 (1993)
30. Grosof, B.N., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proceedings of the Twelfth International Conference on World Wide Web, pp. 48–57. ACM Press, New York (2003)
31. Guarino, N., Poli, R.: Formal ontology in conceptual analysis and knowledge representation. International Journal of Human and Computer Studies 43(5/6), 625–640 (1995)
32. Haarslev, V., Pai, H.-I., Shiri, N.: Optimizing tableau reasoning in alc extended with uncertainty. In: Proceedings of the 2007 International Workshop on Description Logics, DL 2007 (2007)
33. Habiballa, H.: Resolution strategies for fuzzy description logic. In: Proceedings of the 5th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2007), vol. 2, pp. 27–36 (2007)
34. Hähnle, R.: Many-valued logics and mixed integer programming. Annals of Mathematics and Artificial Intelligence 3,4(12), 231–264 (1994)
35. Hähnle, R.: Advanced many-valued logics. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn., vol. 2. Kluwer, Dordrecht (2001)
36. Hájek, P.: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht (1998)
37. Ianni, G., Krennwallner, T., Martello, A., Polleres, A.: A rule system for querying persistent rdfs data. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 857–862. Springer, Heidelberg (2009)
38. Kifer, M., Lausen, G., Wu, J.: Logical foundations of Object-Oriented and frame-based languages. Journal of the ACM 42(4), 741–843 (1995)
39. Kifer, M., Subrahmanian, V.S.: Theory of generalized annotated logic programming and its applications. Journal of Logic Programming 12, 335–367 (1992)
40. Klement, E.P., Mesiar, R., Pap, E.: Triangular Norms. Trends in Logic - Studia Logica Library. Kluwer Academic Publishers, Dordrecht (2000)

41. Konstantopoulos, S., Apostolikas, G.: Fuzzy-dl reasoning over unknown fuzzy degrees. In: Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems, OTM 2007, vol. Part II, pp. 1312–1318. Springer, Heidelberg (2007)
42. Lakshmanan, L.V.S., Shiri, N.: A parametric approach to deductive databases with uncertainty. IEEE Transactions on Knowledge and Data Engineering 13(4), 554–570 (2001)
43. Lloyd, J.W.: Foundations of Logic Programming. Springer, Heidelberg (1987)
44. Lukasiewicz, T., Straccia, U.: Top-k retrieval in description logic programs under vagueness for the semantic web. In: Prade, H., Subrahmanian, V.S. (eds.) SUM 2007. LNCS (LNAI), vol. 4772, pp. 16–30. Springer, Heidelberg (2007)
45. Lukasiewicz, T., Straccia, U.: Managing uncertainty and vagueness in description logics for the semantic web. Journal of Web Semantics 6, 291–308 (2008)
46. Marin, D.: A formalization of rdf. Technical Report TR/DCC-2006-8, Deptartment of Computer Science, Universidad de Chile (2004), http://www.dcc.uchile.cl/cgutierr/ftp/draltan.pdf
47. Muñoz, S., Pérez, J., Gutierrez, C.: Minimal deductive systems for rdf. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC 2007. LNCS, vol. 4519, pp. 53–67. Springer, Heidelberg (2007)
48. Straccia, U., Lopes, N., Polleres, A., Zimmermann, A.: Anql: Sparqling up annotated rdf. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 518–533. Springer, Heidelberg (2010)
49. OWL Web Ontology Language overview, W3C (2004), http://www.w3.org/TR/owl-features/
50. OWL 2 Web Ontology Language Document Overview, W3C (2009), http://www.w3.org/TR/2009/REC-owl2-overview-20091027/
51. Papadimitriou, C.H.: Computational Complexity. Addison Wesley Publ. Co., Reading (1994)
52. RDF Semantics, W3C (2004), http://www.w3.org/TR/rdf-mt/
53. Rule Interchange Format (RIF), W3C (2011), http://www.w3.org/2001/sw/wiki/RIF
54. Shapiro, E.Y.: Logic programs with uncertainties: A tool for implementing rule-based systems. In: Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI 1983), pp. 529–532 (1983)
55. SPARQL, http://www.w3.org/TR/rdf-sparql-query/
56. SPARQL, http://www.w3.org/TR/sparql11-query/
57. Stoilos, G., Simou, N., Stamou, G., Kollias, S.: Uncertainty and the semantic web. IEEE Intelligent Systems 21(5), 84–87 (2006)
58. Straccia, U.: Reasoning within fuzzy description logics. Journal of Artificial Intelligence Research 14, 137–166 (2001)
59. Straccia, U.: Transforming fuzzy description logics into classical description logics. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 385–399. Springer, Heidelberg (2004)
60. Straccia, U.: Description logics with fuzzy concrete domains. In: Bachus, F., Jaakkola, T. (eds.) 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005), pp. 559–567. AUAI Press, Edinburgh (2005)
61. Straccia, U.: Uncertainty management in logic programming: Simple and effective top-down query answering. In: Khosla, R., Howlett, R.J., Jain, L.C. (eds.) KES 2005. LNCS (LNAI), vol. 3682, pp. 753–760. Springer, Heidelberg (2005)

62. Straccia, U.: Answering vague queries in fuzzy DL-Lite. In: Proceedings of the 11th International Conference on Information Processing and Managment of Uncertainty in Knowledge-Based Systems (IPMU 2006), pp. 2238–2245. E.D.K, Paris (2006)
63. Straccia, U.: Description logics over lattices. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 14(1), 1–16 (2006)
64. Straccia, U.: A fuzzy description logic for the semantic web. In: Sanchez, E. (ed.) Fuzzy Logic and the Semantic Web, Capturing Intelligence. ch.4, pp. 73–90. Elsevier, Amsterdam (2006)
65. Straccia, U.: Fuzzy description logic programs. In: Proceedings of the 11th International Conference on Information Processing and Managment of Uncertainty in Knowledge-Based Systems (IPMU 2006), pp. 1818–1825. E.D.K, Paris (2006)
66. Straccia, U.: Towards top-k query answering in deductive databases. In: Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics (SMC 2006), pp. 4873–4879. IEEE, Los Alamitos (2006)
67. Straccia, U.: Towards vague query answering in logic programming for logic-based information retrieval. In: Melin, P., Castillo, O., Aguilar, L.T., Kacprzyk, J., Pedrycz, W. (eds.) IFSA 2007. LNCS (LNAI), vol. 4529, pp. 125–134. Springer, Heidelberg (2007)
68. Straccia, U.: Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In: Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. LNCS, vol. 5224, pp. 54–103. Springer, Heidelberg (2008)
69. Straccia, U.: A minimal deductive system for general fuzzy RDF. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 166–181. Springer, Heidelberg (2009)
70. Straccia, U., Lopes, N., Lukacsy, G., Polleres, A.: A general framework for representing and reasoning with annotated semantic web data. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010), pp. 1437–1442. AAAI Press, Menlo Park (2010)
71. Torra, V., Narukawa, Y.: Information Fusion and Aggregation Operators. In: Cognitive Technologies. Springer, Heidelberg (2007)
72. Ullman, J.D.: Principles of Database and Knowledge Base Systems, vol. 1,2. Computer Science Press, Potomac (1989)
73. Vojtás, P.: Fuzzy logic programming. Fuzzy Sets and Systems 124, 361–370 (2001)
74. XML, W3C, `http://www.w3.org/XML/`
75. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Trans. Syst. Man Cybern. 18, 183–190 (1988)
76. Zadeh, L.A.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)