2011 IEEE International Conference on Fuzzy Systems
June 27-30, 2011, Taipei, Taiwan

Aggregation Operators and Fuzzy OWL 2

Fernando Bobillo
Dpt. of Computer Science and Systems Engineering
University of Zaragoza, Spain
fbobillo@unizar.es

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

***Abstract*—Fuzzy Description Logics (Fuzzy DLs) are logics that allow to deal with structured knowledge affected by fuzziness. Fuzzy DLs are at the heart of Fuzzy OWL 2, a fuzzy version of the standard ontology language OWL 2. Although a relatively important amount of work has been carried out in the last years, fuzzy DLs are open to be extended with several features worked out in other fields. In particular, the integration of aggregation operators (AOs) in fuzzy DLs has received little attention so far.**

In this work, we show how to support aggregation operators in fuzzy DLs. We provide syntax and semantics of a fuzzy DL extended with AOs, and provide a calculus for a family of AOs (weighted sum, OWA and quantifier-guided OWA). We also show how to encode them into our proposal for Fuzzy OWL 2.

***Keywords*—Fuzzy ontologies, Fuzzy Description Logics, Fuzzy OWL 2, Aggregation operators, OWA operators**

I. Introduction

Description Logics (DLs) [1] play a key role in the design of *Ontologies*. An Ontology consists of a hierarchical description of important concepts in a particular domain, along with the description of the properties (of the instances) of each concept. In this context, DLs are important as they are essentially the theoretical counterpart of the *Web Ontology Language OWL 2* [2], the standard language to represent ontologies.

It is well-known that "classical" ontology languages are not appropriate to deal with *vague knowledge*, which is inherent to several real world domains [3], [4]. *Fuzzy ontologies* emerge as useful in several applications, such as (multimedia) information retrieval, image interpretation, ontology mapping, matchmaking, multi-criteria decision making [5] and the Semantic Web [6]. So far, several fuzzy extensions of DLs can be found in the literature (see the survey in [6]).

A representation of fuzzy ontologies using OWL 2 annotation properties has been recently proposed (in this paper, we will call it *Fuzzy OWL 2*) [7]. The use of annotation properties makes it possible to use current OWL 2 editors for fuzzy ontology representation, such as Protégé. This approach has reached a sufficient level of maturity and there exists a Protégé plug-in to edit fuzzy ontologies, as well as some parsers that translate fuzzy ontologies represented using this approach into the languages supported by some fuzzy DL reasoners.

Aggregation Operators (AOs) are mathematical functions that are used to combine information [8]. The arithmetic mean, the weighted sum, the median and, more generally Ordered Weighted Averaging (OWA) [9] are the most well-known AOs. AOs have been widely used in computational intelligence because of their ability to fuse linguistically expressed pieces of information.

The integration of AOs and fuzzy ontologies is of great interest, in order to combine the advantages of both formalisms. In this work, we show *how to support AOs in fuzzy ontologies*. In particular, we provide syntax and semantics of a fuzzy DL extended with AOs, provide a calculus for a family of aggregation operators, such as weighted sum, OWA and quantifier-guided OWA, and show how to represent them using our previous proposal for Fuzzy OWL 2 [7].

There is little related work combining AOs and fuzzy DLs. [10] presents a fuzzy DL with AOs, but the expressivity is very limited. Also, weighted sum concepts have been proposed in the literature (see e.g., [11]). Our contributions consists of a more expressive fuzzy DL, independent of the fuzzy operators considered, together with a reasoning algorithm and a representation using OWL 2, for a family of AOs.

In the following, we proceed as follows. Section II recalls basic notions on mathematical fuzzy logic and aggregation operators. Then, Section III introduces fuzzy DLs that support aggregation operators. We provide the syntax, semantics, and reasoning algorithms for a family of AOs. Then, Section IV shows how to support them within fuzzy OWL 2 [7]. Finally, Section V sets out some conclusions and discusses future work.

II. Preliminaries

A. Mathematical Fuzzy Logic

In *Mathematical Fuzzy Logic* [12], the convention prescribing that a statement is either true or false is changed and is a matter of degree measured on an ordered scale that is no longer $\{0,1\}$, but the $[0,1]$. This degree is called *degree of truth* of the statement ϕ in the interpretation $\mathcal{I}$. In this section, *fuzzy statements* have the form $\phi \geqslant r$ or $\phi \leqslant r$, where $r \in [0,1]$ (see, e.g. [12], [13]) and ϕ is a statement, which encode that the degree of truth of ϕ is *at least* r (resp. *at most* r).

A *fuzzy interpretation* $\mathcal{I}$ maps each basic statement p_i into $[0,1]$ and is then extended inductively to all statements: $\mathcal{I}(\phi \wedge \psi) = \mathcal{I}(\phi) \otimes \mathcal{I}(\psi)$, $\mathcal{I}(\phi \vee \psi) = \mathcal{I}(\phi) \oplus \mathcal{I}(\psi)$, $\mathcal{I}(\phi \rightarrow \psi) = \mathcal{I}(\phi) \Rightarrow \mathcal{I}(\psi)$, $\mathcal{I}(\neg\phi) = \ominus\mathcal{I}(\phi)$, $\mathcal{I}(\exists x.\phi(x)) = \sup_{a \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(a))$, $\mathcal{I}(\forall x.\phi(x)) = \inf_{a \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(a))$, where $\Delta^{\mathcal{I}}$ is the domain of $\mathcal{I}$, and $\otimes$, $\oplus$, $\Rightarrow$, and $\ominus$ are *t-norms*, *t-conorms*, *implication functions*, and *negation functions*, respectively, which extend the Boolean conjunction, disjunction, implication, and negation, respectively, to the fuzzy case.

One usually distinguishes three different logics, namely Łukasiewicz, Gödel, and Product logics [12]. Zadeh logic,

978-1-4244-7316-8/11/$26.00 ©2011 IEEE

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

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.

A *fuzzy set* R over a countable crisp set X is a function $R\colon X \to [0,1]$. The trapezoidal (Fig. 1 (a)), the triangular (Fig. 1 (b)), the L-function (left-shoulder function, Fig. 1 (c)), and the R-function (right-shoulder function, Fig. 1 (d)) are frequently used to specify membership degrees.

The notions of satisfiability and logical consequence are defined in the standard way. A fuzzy interpretation $\mathcal{I}$ *satisfies* a fuzzy statement $\phi \geqslant r$ (resp., $\phi \leqslant r$) or $\mathcal{I}$ is a *model* of $\phi \geqslant r$ (resp., $\phi \leqslant r$), denoted $\mathcal{I} \models \phi \geqslant r$ (resp., $\mathcal{I} \models \phi \leqslant r$), iff $\mathcal{I}(\phi) \geqslant r$ (resp., $\mathcal{I}(\phi) \leqslant r$).

B. Aggregation Operators

Aggregation Operators (AOs) (see, e.g. [8]) are mathematical functions that are used to combine information. There exist a large number of different AOs that differ on the assumptions on the data (data types) and about the type of information that we can incorporate in the model. To what concerns us, an AO of dimension n is a mapping $@ : \mathbb{R}^n \to \mathbb{R}$ that satisfies:

1) $@(a) = a$ (idempotent if unary)
2) $@(0,\ldots,0) = 0$ and $@(1,\ldots,1) = 1$ (boundary conditions)
3) $@(a_1,\ldots,a_n) \leq @(b_1,\ldots,b_n)$ if $\forall i, a_i \leq b_i$ (monotone)

Note that we always have that

$$\min(a_1,\ldots,a_n) \leq @(a_1,\ldots,a_n) \leq \max(a_1,\ldots,a_n) \ .$$

Often, an AO $@$ is parameterised with a vector of n weights $W = [w_1,\ldots,w_n]$ such that $w_i \in [0,1]$ and $\sum_i w_i = 1$. In that case we will denote the AO as $@_W$.

Examples of AOs are the *arithmetic mean*,

$$@^{\text{avg}}(a_1,\ldots,a_n) = \frac{1}{n}\sum_i a_i \tag{1}$$

the *weighted sum*

$$@^{\text{ws}}_W(a_1,\ldots,a_n) = \sum_i w_i a_i \ , \tag{2}$$

and the *Ordered Weighted Averaging* (OWA) operators and the quantifier-guided OWAs that we shall present next.

The OWA operators [9], [14], [15] provide a parameterised class of mean type AOs. Formally, an OWA operator of dimension n is an AO such that

$$@^{\text{owa}}_W(a_1,\ldots,a_n) = \sum_j w_j b_j \ , \tag{3}$$

where b_j is the j-th largest of the a_i.

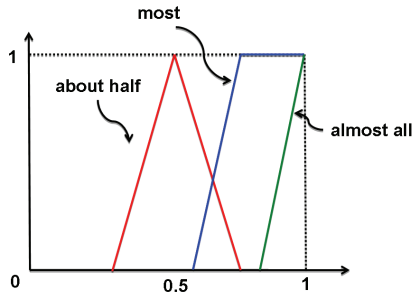

Fig. 2. Some examples of fuzzy quantifiers.

A fundamental aspect of these operators is the reordering step. In particular, a weight w_i is not associated with a specific argument but with an ordered position of the aggregate. By choosing different W we can implement different AOs. The OWA operator is a non-linear operator as a result of the process of determining the b_j. An OWA operator $@$ is a mean operator and is also symmetric and idempotent:

$$\begin{aligned} &\mathit{Symmetric} : @(a_1,\ldots,a_n) = @(a_{\pi(1)},\ldots,a_{\pi(n)}) \\ &\mathit{Idempotent} : @(a,\ldots,a) = a \end{aligned}$$

where π is a permutation. Notable OWA operators are:

$$\begin{aligned} f(a_1,\ldots,a_n) &= \max(a_1,\ldots,a_n) \text{ for } W = [1,0,\ldots,0] \\ f(a_1,\ldots,a_n) &= \min(a_1,\ldots,a_n) \text{ for } W = [0.\ldots,0,1] \\ f(a_1,\ldots,a_n) &= \text{avg}(a_1,\ldots,a_n) \text{ for } W = [1/n,1/n,\ldots,1/n] \\ f(a_1,\ldots,a_n) &= \text{med}(a_1,\ldots,a_n) \text{ for } w_i = 0, \\ &\qquad n \text{ odd and } w_{(n+1)/2} = 1, \text{ or} \\ &\qquad n \text{ even and } w_{n/2} = 0.5 = w_{n/2+1} \end{aligned}$$

In the equations above, avg and med are the average and median value of the a_i, respectively.

As next we recap *quantifier aggregations* [14], [16], [17], [18]. Classical logic has two quantifiers, the universal $\forall$ and the existential $\exists$ quantifier. These are extremal ones among several other linguistic quantifiers such as *most, few, about half, some, many*, etc. Quantifiers can be seen as absolute of proportional (see, e.g. [18]). To what concerns us, we consider the proportional ones. In this case, a proportional type quantifier, such as *most*, can be represented as a fuzzy subset $Q : [0,1] \to [0,1]$ such that for each $r \in [0,1]$, the membership grade $Q(r)$ indicates the degree to which the proportion r satisfies the linguistic quantifier that Q represents. See Figure 2 for examples of fuzzy quantifiers.

An important class of quantifiers are the *monotone quantifiers* that satisfy the following conditions:

1) $Q(0) = 0$,
2) $Q(1) = 1$,
3) $Q(r_1) \leq Q(r_2)$ if $r_1 \leq r_2$.

Essentially, these quantifiers are characterised by the idea that as the proportion increases, the degree of satisfaction does not decrease. For two quantifiers Q_1, Q_2, we write $Q_1 \leq Q_2$ if for all r, $Q_1(r) \leq Q_2(r)$. We recall that the two classical quantifiers $\exists, \forall$ may be defined as the monotone quantifiers $Q_\exists(r) = 0$ if $r \neq 1$ and $Q_\forall(r) = 1$ if $r \neq 0$, respectively. Note that for any monotone quantifier Q, we have that

$$Q_\forall \leq Q \leq Q_\exists \tag{4}$$

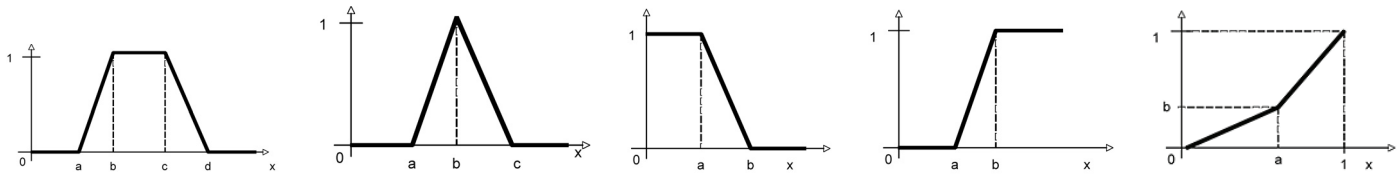

Fig. 1. (a) Trapezoidal function; (b) Triangular function; (c) L-function; (d) R-function; (e) Linear function .

dictating that $\forall$ and $\exists$ are indeed the lower and upper bounds of monotone quantifiers.

To introduce quantifier-guided aggregation, we consider first the evaluation of quantified propositions. So, let Y be a (crisp) set of objects and B some fuzzy subset of Y. A quantified proposition is a statement of the form

$$QY\text{'s are } B \quad .$$

An example of such a statement is "most students are young". Yager [16] suggested an approach to evaluate the truth of such quantified propositions using OWA. In the following we will assume that Q is a monotone quantifier. The first step consists in associating to Q an OWA weighting vector W_Q of dimension $n = |Y|$, where $|S|$ is the cardinality of a denumerable set S. The weights are obtained as

$$w_j = Q(\frac{j}{n}) - Q(\frac{j-1}{n}) \ . \tag{5}$$

Note that for Q monotone, $w_j \in [0,1]$ and $\sum_j w_j = 1$. Now, we can obtain the truth t of the statement "QY's are B" as

$$t = @^{\text{owa}}_{W_Q}(a_1, \ldots, a_n) \tag{6}$$

where $a_i = B(y_i)$, $y_i \in Y$ and $@^{\text{owa}}_{W_Q}$ is the OWA operator obtained using the weights W_Q found from Q as above.

We turn now to the issue of quantifier-guided aggregation, that have been shown to be useful for evaluating multi-criteria decision problems [19]. So, assume X to be a set of alternatives and we have a collection of n criteria A_i that are of concern in a given problem. For $x \in X$, with $A_i(x)$ we indicate the degree to which the i-th criterion is satisfied by alternative x. It is easily verified that the degree to which *all* criteria are satisfied by x can be determined by $D(x) = \min(A_1(x), \ldots, A_n(x))$ while the degree to which *some* criteria is satisfied by x can be determined by $D(x) = \max(A_1(x), \ldots, A_n(x))$. In general, for a monotone quantifier Q, the degree to which

$$Q \text{ of the criteria are satisfied by } x$$

can be formalised as

$$D(x) = @^{\text{owa}}_{W_Q}(A_1(x), \ldots, A_n(x)) \ , \tag{7}$$

where $@^{\text{owa}}_{W_Q}$ is the OWA operator obtained using the weights W_Q found from Q.

Thus, by selecting an appropriate weighting vector W, we are essentially implementing a kind of quantifier-guided aggregation.

III. Aggregation Operators and Fuzzy DLs

The aim of this section is to show how current fuzzy DLs can be extended to support AOs. For illustrative purposes and for reasons of space, we will just consider a minimal fuzzy DL (see, e.g. [11] for a more expressive fuzzy DL). For the sake of our purpose, we will consider $\mathcal{ALCF}(\mathbf{D})$ [20], which is the basic DL $\mathcal{ALC}$ extended with functional roles (letter $\mathcal{F}$) and fuzzy concrete domains (letter $\mathbf{D}$).

A. The Fuzzy DL $\mathcal{ALCF}(\mathbf{D})$ with Aggregation Operators

In general, a *fuzzy concrete domain* (or simply *fuzzy domain*) is a pair $\langle \Delta_{\mathbf{D}}, \Phi_{\mathbf{D}} \rangle$, where $\Delta_{\mathbf{D}}$ is an interpretation domain and $\Phi_{\mathbf{D}}$ is the set of *fuzzy domain predicates* $\mathbf{d}$ with a predefined arity n and an interpretation $\mathbf{d}^{\mathbf{D}} \colon \Delta^n_{\mathbf{D}} \to [0,1]$, which is a n-ary fuzzy relation over $\Delta_{\mathbf{D}}$ [20]. In our specific fuzzy DL, we assume that predicates are unary and $\Delta_{\mathbf{D}}$ are non-negative real numbers.

Now, consider pair wise disjoint alphabets of *concepts names* (denoted A), *abstract roles names* (denoted R) and *concrete roles names* (denoted T). Within the alphabet of abstract and concrete roles, we have distinguished subsets of *abstract functional roles names* (denoted f) and *concrete functional roles names* (denoted t), respectively. Functional roles are also called *features*.

Concepts (denoted C or D) of the language can be built inductively from atomic concepts (A), top concept $\top$, bottom concept $\bot$, abstract roles (R), concrete roles (T) as follows. The syntax of fuzzy concepts is as follows [11]:

$$\begin{array}{rcl} C, D & \to & \top \mid \ \bot \ \mid A \mid C \sqcap D \mid C \sqcup D \mid \neg C \mid \\ & & \forall R.C \mid \exists R.C \mid \forall T.\mathbf{d} \mid \exists T.\mathbf{d} \\ \mathbf{d} & \to & ls(a,b) \mid rs(a,b) \mid tri(a,b,c) \mid trz(a,b,c,d) \end{array}$$

where ls, rs, tri, trz stand for left-shoulder, right-shoulder, triangular and trapezoidal membership functions. These are our concrete domain predicates. So, for instance the expression $Human \sqcap \exists hasChild.Lovely$ is a concept (unary predicate), which will denote the fuzzy set of humans that have a lovely child, while $Human \sqcap \forall hasChild.Lovely$ will denote the fuzzy set of humans that have only lovely children. Also, the expression $Human \sqcap \exists hasAge.ls(10,30)$ will denote the set of young humans (their age is $ls(10,30)$).

We next extend fuzzy DLs to support AOs. Consider a family of AOs $(@_n)$, $n \in \mathbb{N}$, each of arity $a(n)$. Then, similarly to [10], the following are concept expressions

$$C \quad \to \quad C \mid @_n(C_1, \ldots, C_{a(n)}) \ .$$

Example 1: The following concept expressions denote the fuzzy set of hotels that are cheap, close to the venue and comfortable, in which the degree is computed respectively as the weighted sum, the OWA aggregation and the quantifier-guided aggregation.

$$Hotel \sqcap @^{\text{ws}}_{W}(Cheap, CloseToVenue, Comfortable)$$
$$Hotel \sqcap @^{\text{owa}}_{W}(Cheap, CloseToVenue, Comfortable)$$
$$Hotel \sqcap @^{\text{owa}}_{W_Q}(Cheap, CloseToVenue, Comfortable)$$

It is interesting to note that a weighted sum concept $@^{\text{ws}}_{W}(C_1, \ldots, C_n)$ cannot be represented using an OWA operator $@^{\text{owa}}_{W}(C_1, \ldots, C_n)$, because in a weighted sum concept every weight is assigned to a specific concept, whereas in OWA operators every weight is associated to the concept that corresponds to a specific position after a reordering step.

A *Fuzzy Knowledge Base* (or *fuzzy Ontology*) comprises a fuzzy ABox $\mathcal{A}$ and a fuzzy TBox $\mathcal{T}$. A *fuzzy ABox* consists of a finite set of *fuzzy assertions* of one of the following types: a fuzzy *concept assertion* of the form $\langle a{:}C, r\rangle$ (with informal meaning, individual a is an instance of concept C with degree at least r), or a fuzzy *role assertion* of the form $\langle (a,b){:}R, r\rangle$ (the pair of individuals (a,b) is an instance of role R with degree at least r), with $r \in [0,1]$. In FOL, $\langle a{:}C, r\rangle$ may be seen as a fuzzy statement of the form $C(a) \geqslant r$, while $\langle (a,b){:}R, r\rangle$ may be seen as a fuzzy statement of the form $R(a,b) \geqslant r$.

A *fuzzy TBox* consists of a finite set of *fuzzy General Concept Inclusions (fuzzy GCIs)*, which are expressions of the form $\langle C \sqsubseteq D, r\rangle$ (with informal meaning, the degree of subsumption between concept C and D is not less than r). $C \equiv D$ is a shorthand for both $\langle C \sqsubseteq D, 1\rangle$ and $\langle D \sqsubseteq C, 1\rangle$.

From a semantics point of view, a *fuzzy interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ relative to the fuzzy concrete domain $\langle \Delta_{\mathbf{D}}, \Phi_{\mathbf{D}}\rangle$, consists of a nonempty set $\Delta^{\mathcal{I}}$ (the *domain*), disjoint from $\Delta_{\mathbf{D}}$, and of a *fuzzy interpretation function* $\cdot^{\mathcal{I}}$ that coincides with $\cdot_{\mathbf{D}}$ on every fuzzy concrete predicate, and it assigns: (i) to each abstract concept C a function $C^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \to [0,1]$; (ii) to each abstract role R a function $R^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1]$; (iii) to each abstract feature f a partial function $f^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to \{0,1\}$ such that for all $u \in \Delta^{\mathcal{I}}$ there is an unique $w \in \Delta^{\mathcal{I}}$ on which $f^{\mathcal{I}}(u,w)$ is defined; (iv) to each concrete role T a function $T^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \to [0,1]$; (v) to each concrete feature t a partial function $t^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \to \{0,1\}$ such that for all $u \in \Delta^{\mathcal{I}}$ there is an unique $v \in \Delta_{\mathbf{D}}$ on which $t^{\mathcal{I}}(u,v)$ is defined.

Given arbitrary t-norm $\otimes$, t-conorm $\oplus$, negation function $\ominus$ and implication function $\Rightarrow$, the fuzzy interpretation function is extended to *complex concepts* and *fuzzy axioms* as in Figure 3, where @ is an AO[1].

Now, we will define some common reasoning tasks [3]. Let $\alpha \in \{a{:}C, (a,b){:}R, C \sqsubseteq D\}$ and $\tau = \langle \alpha, r\rangle$. A fuzzy interpretation $\mathcal{I}$ *satisfies* (is a *model* of) a fuzzy axiom τ iff $\alpha^{\mathcal{I}} \geq r$. A fuzzy KB $\mathcal{K}$ is *satisfiable* iff there is a model $\mathcal{I}$ which satisfies each of the fuzzy axioms in $\mathcal{K}$. A fuzzy axiom τ is a *logical consequence* of a fuzzy KB $\mathcal{K}$ iff, for every model $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{I} \models \tau$. The *Best Entailment Degree* (BED) of an axiom α w.r.t. a fuzzy KB $\mathcal{K}$ is defined as $bed(\mathcal{K}, \alpha) = \sup\{r \mid \mathcal{K} \models \langle \alpha, r\rangle\}$. The *Best Satisfiability Degree* (BSD) of a concept C w.r.t. a fuzzy KB $\mathcal{K}$ as $bsd(\mathcal{K}, C) = \sup_{\mathcal{I} \models \mathcal{K}} \sup_{x \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(x)$ [11].

Example 2: Suppose we have some data about hotels, as shown in the following table:

$hotel$	$price$	$distance$	$star$
h_1	150	300	4
h_2	100	500	2

Assume that we are looking for a cheap, close to the venue and comfortable hotel. To this end, we may encode the problem using a fuzzy KB $\mathcal{K}$ with the following ABox $\mathcal{A}$:

$$Hotel(h_1), Hotel(h_2), price(h_1, 150), price(h_2, 100),$$
$$distance(h_1, 300), distance(h_2, 500), star(h_1, 4), star(h_2, 2)$$

where $price, distance$ and $star$ are concrete features.

The TBox $\mathcal{T}$ holds some definitions for cheap, close and comfortable hotels and contains the following axioms:

$$\begin{aligned} Cheap &= Hotel \sqcap \exists price.ls(60, 120) \\ CloseToVenue &= Hotel \sqcap \exists distance.ls(200, 400) \\ Comfortable &= Hotel \sqcap \exists star.ls(3, 5)\ . \end{aligned}$$

Note that, e.g., we have that $\mathcal{K} \models \langle h_2{:}Cheap, 0.33\rangle$, i.e., h_2 is a cheap hotel to degree no less than 0.33 (it is easily verified that this degree is also tight). Now, for some suitable AO @, we may compute the degree n_i to which hotel h_i satisfies our request as

$$n_i = bed(\mathcal{K}, h_i{:}@(Cheap, CloseToVenue, Comfortable))$$

B. A Reasoning Algorithm

We next provide a reasoning procedure that supports a family of AOs for fuzzy DLs. To simplify our exposition, we will make the following assumptions. We restrict the fuzzy DL to the case of Lukasiewicz fuzzy logic (and, thus, support "Zadeh logic" as well). Other fuzzy logics could be dealt with similarly as in [20], [21]. Furthermore, we will restrict KBs to be *unfoldable* [22], which is defined as follows.

A fuzzy TBox $\mathcal{T}$ is *unfoldable* iff it verifies the following three constraints: (i) every axiom in $\mathcal{T}$ is either of the form $\langle A \sqsubseteq C, 1\rangle$ or $A \equiv C$, where A is an atomic concept [2]; (ii) there is no concept A such that it appears more than once on the left hand side of some axiom in $\mathcal{T}$; and (iii) no cyclic definitions are present in $\mathcal{T}$. We will say that A *directly uses* primitive concept B in $\mathcal{T}$, if there is some axiom $\tau \in \mathcal{T}$ such that A is on the left hand side of τ and B occurs in the right hand side of τ. Let *uses* be the transitive closure of the relation *directly uses* in $\mathcal{T}$. $\mathcal{T}$ is *cyclic* iff there is A such that A uses A in $\mathcal{T}$. Analogously, a fuzzy KB is said to be *unfoldable* when its TBox is unfoldable.

Unfoldable TBoxes can be eliminated through an *expansion process* [23].Essentially, each axiom $A \sqsubseteq C \in \mathcal{T}$ can be replaced with $A = C \sqcap A^*$, where A^* is a new concept name.

[1]For ease of presentation, we identify the interpretation of ls, rs, tri, trz and @ with the functions themselves.

[2]In particular this forbids concept $\top$ on the left hand side of axioms.

Concept		Semantics
$(\top)^{\mathcal{I}}(x)$	=	1
$(\bot)^{\mathcal{I}}(x)$	=	0
$(A)^{\mathcal{I}}(x)$	=	$A^{\mathcal{I}}(x)$
$(C \sqcap D)^{\mathcal{I}}(x)$	=	$C^{\mathcal{I}}(x) \otimes D^{\mathcal{I}}(x)$
$(C \sqcup D)^{\mathcal{I}}(x)$	=	$C^{\mathcal{I}}(x) \oplus D^{\mathcal{I}}(x)$
$(\neg C)^{\mathcal{I}}(x)$	=	$\ominus C^{\mathcal{I}}(x)$
$(@(C_1, \ldots, C_n))^{\mathcal{I}}(x)$	=	$@({C_1}^{\mathcal{I}}(x), \ldots, {C_n}^{\mathcal{I}}(x))$

Concept		Semantics
$(\forall R.C)^{\mathcal{I}}(x)$	=	$\inf_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)\}$
$(\exists R.C)^{\mathcal{I}}(x)$	=	$\sup_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)\}$
$(\forall T.\mathbf{d})^{\mathcal{I}}(x)$	=	$\inf_{v \in \Delta_{\mathbf{D}}} \{T^{\mathcal{I}}(x,v) \Rightarrow \mathbf{d}^{\mathbf{D}}(v)\}$
$(\exists T.\mathbf{d})^{\mathcal{I}}(x)$	=	$\sup_{v \in \Delta_{\mathbf{D}}} \{T^{\mathcal{I}}(x,v) \otimes \mathbf{d}^{\mathbf{D}}(v)\}$
$(a{:}C)^{\mathcal{I}}$	=	$C^{\mathcal{I}}(a^{\mathcal{I}})$
$((a,b){:}R)^{\mathcal{I}}$	=	$R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}})$
$(C \sqsubseteq D)^{\mathcal{I}}$	=	$\inf_{x \in \Delta^{\mathcal{I}}} \{C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)\}$

Fig. 3. Semantics of fuzzy concepts and axioms.

Let $\mathcal{K}'$ the obtained knowledge base. $\mathcal{K}'$ can be *expanded* by substituting every concept name A occurring in $\mathcal{K}$, which is defined in $\mathcal{T}$, with its defining term in $\mathcal{T}$.

Note that reasoning with fuzzy GCIs requires so-called blocking conditions [1] and are known to work properly for Zadeh logic only [24]. See [22] for problems risen by GCIs within Lukasiewicz logic. These problems do not appear here due to our, by purpose, simplified assumptions.

The basic idea of the reasoning algorithm is as follows and is common to most fuzzy DLs algorithms (see, e.g. [21]). Consider $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is unfoldable. In order to solve the BED problem, we combine appropriate DL tableaux rules with methods developed in the context of *many-valued logics* (MVLs) [13]. In order to determine e.g., $bed(\mathcal{K}, a{:}C)$, we consider an expression of the form $\langle a{:}\neg C, 1 - x \rangle$ (informally, $\langle a{:}C \leq x \rangle$), where x is a $[0,1]$-valued variable. Then we construct a tableaux for $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \cup \{\langle a{:}\neg C, 1 - x \rangle\} \rangle$ in which the application of satisfiability preserving rules generates new fuzzy assertion axioms together with *inequations* over $[0,1]$-valued variables. These inequations have to hold in order to respect the semantics of the DL constructors. Hence, we *minimise* the original variable x such that all constraints are satisfied[3].

Similarly, for $C \sqsubseteq D$, we can compute $bed(\mathcal{K}, C \sqsubseteq D)$ as the minimal value of x such that $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \cup \{\langle a : C, x_1 \rangle\} \cup \{\langle a : D, x_2 \rangle\} \rangle$ is satisfiable under the constraints expressing that $x_1 \Rightarrow x_2 \leq x, x_1 \in [0,1]$ and $x_2 \in [0,1]$, where a is new abstract individual.

Finally, $bed(\mathcal{K}, (a,b){:}R)$ is equivalent to $bed(\mathcal{K} \cup \{\langle b : B, 1 \rangle\}, a{:}\exists R.B)$, where B is a new concept (which does not appear in $\mathcal{K}$).

Therefore, the BED problem can be reduced to minimal satisfiability problem of a KB. Finally, concerning the best satisfiability bound problem, $bsd(\mathcal{K}, C)$ is determined by the maximal value of x such that $\langle \mathcal{T}, \mathcal{A} \cup \{\langle a{:}C, x \rangle\} \rangle$ is satisfiable.

In general, depending on the semantics of the DL constructors and fuzzy domain predicates we may end up with a general, bounded *Mixed Integer Non Linear Programming* (MINLP) optimization problem. In this paper, however, due to our restrictions above and those we will impose to AOs, we will end up with a *Mixed Integer Linear Program* (MILP) optimization problem [25]. Interestingly, as for the MVL case, the tableaux we are generating contains *one* branch only and, thus, just *one* MILP problem has to be solved.

We recall that a general MILP problem consists in minimizing a linear function with respect to a set of constraints that are linear inequations in which rational and integer variables can occur. More precisely, let $x = \langle x_1, \ldots, x_k \rangle$ and $y = \langle y_1, \ldots, y_m \rangle$ be variables over $\mathbb{Q}$ and $\mathbb{Z}$ respectively, over the integers and let A, B be integer matrices and h an integer vector. The variables in y are called *control variables*. Let $f(x,y)$ be an $k+m$-ary linear function. Then the general MILP problem is to find $\bar{x} \in \mathbb{Q}^k, \bar{y} \in \mathbb{Z}^m$ such that $f(\bar{x}, \bar{y}) = \min\{f(x,y) \mid Ax + By \geq h\}$. Furthermore, we say that $M \subseteq [0,1]^k$ is MILP-representable iff there is a MILP (A, B, h) with k real and m $0-1$ variables such that $M = \{x : \exists y \in \{0,1\}^m$ such that $Ax + By \geq h\}$. In general, we require that a constructor f is MILP representable. In particular, the sets $g(f) = \{\langle x_1, \ldots, x_k, x \rangle \colon f(x_1, \ldots, x_k) \geq x\}$ and $\bar{g}(f) = \{\langle x_1, \ldots, x_k, x \rangle \colon f(x_1, \ldots, x_k) \leq x\}$ should be MILP-representable.

For instance, under Łukasiewicz t-norm, $x_1 \otimes x_2 \geq l$ can be written as $\{l \leq y, x_1 + x_2 - 1 \leq l, x_1 + x_2 - y \geq l, y \in \{0,1\}\}$. Therefore, $g(\otimes)$ is MILP representable.

To guarantee that our reasoning process ends up to solve a MILP problem, we require that every constructor is MILP representable. For instance, classical logic, Zadehs fuzzy logic, and Lukasiewicz connectives, are MILP-representable. It is not difficult to see that indeed the functions ls, rs, tri and trz are MILP representable as well (see, e.g. [20], [21]).

To guarantee that also a concept expression of the form $@(C_1, \ldots, C_n)$, involving an AO $@$, can be handled appropriately by our reasoning method, we have further to impose that $@$ is indeed MILP representable. For instance, it is easy to see that this is the case of the weighted sum, $@^{\text{ws}}_W$, while more difficult is the notably case of the OWAs, $@^{\text{owa}}_W$, for which we provide the encoding in the Appendix. Therefore, this covers also the case of quantifier based OWAs. Note that in this latter case, no particular restriction is due on the quantifier except of being monotone and computable in finite time. Furthermore, we may support also so-called OWA approximations such as those proposed in [26], [27], that are less demanding from a computational point of view.

Like most of the tableaux algorithms, our algorithm works on *completion-forests*. Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a fuzzy KB. A completion-forest $\mathcal{F}$ for $\mathcal{K}$ is a collection of trees whose

[3]Informally, suppose the minimal value is $\bar{n}$. We will know then that for any interpretation $\mathcal{I}$ satisfying the knowledge base such that $(a{:}C)^{\mathcal{I}} < \bar{n}$, the starting set is unsatisfiable and, thus, $(a{:}C)^{\mathcal{I}} \geq \bar{n}$ has to hold. Which means that $bed(\mathcal{K}, a{:}C) = \bar{n}$

distinguished roots are arbitrarily connected by edges. Each node v is labelled with a set $\mathcal{L}(v)$ of expressions of the form $\langle C, l\rangle$, where $C \in sub(\mathcal{K})$ ($sub(\mathcal{K})$ is the set of named concepts appearing in $\mathcal{K}$), and l is either a rational, a variable x, or a negated variable, i.e., of the form $1 - x$, where x is a variable. The intuition is that v is an instance of C to degree equal or greater than the evaluation of l.

Each edge $\langle v, w\rangle$ is labelled with a set $\mathcal{L}(\langle v, w\rangle)$ of expressions of the form $\langle R, l\rangle$, where $R \in \mathcal{R}_\mathcal{K}$ ($\mathcal{R}_\mathcal{K}$ is the set of roles in $\mathcal{K}$). The intuition here is that $\langle v, w\rangle$ is an instance of R to degree greater or equal than the evaluation of l.

The forest has associated a set $\mathcal{C}_\mathcal{F}$ of constraints of the form $l \leq l'$, $l = l'$, $x_i \in [0, 1], y_i \in \{0, 1\}$, where l, l' are arithmetic expressions, on the variables occurring in the node labels and edge labels.

The algorithm initializes a forest $\mathcal{F}$ to contain *(i)* a root node v_0^i, for each individual a_i occurring in $\mathcal{A}$, labelled with $\mathcal{L}(v_0^i)$ such that $\mathcal{L}(v_0^i)$ contains $\langle C_i, r\rangle$ for each fuzzy assertion $\langle a_i{:}C_i, r\rangle \in \mathcal{A}$, and *(ii)* an edge $\langle v_0^i, v_0^j\rangle$, for each fuzzy assertion $\langle (a_i, a_j){:}R_i, r\rangle \in \mathcal{A}$, labelled with $\mathcal{L}(\langle v_0^i, v_0^j\rangle)$ such that $\mathcal{L}(\langle v_0^i, v_0^j\rangle)$ contains $\langle R_i, r\rangle$. $\mathcal{F}$ is then expanded by repeatedly applying the completion rules described below. The completion-forest is complete when none of the completion rules are applicable. Then, the MILP problem on the set of constraints $\mathcal{C}_\mathcal{F}$ is solved.

With x_α we denote the variable associated to the *assertion* α of the form $a{:}C$ or $(a, b){:}R$. x_α will take the truth value associated to α. With x_c we will denote the variable associated to the concrete individual c.

Let us discuss now how to deal with feature roles. Let $\mathcal{F}$ be a forest, p an abstract or concrete feature such that we have two edges $\langle v, w_1\rangle$ and $\langle v, w_1\rangle$ such that $\langle p, l_1\rangle$ and $\langle p, l_2\rangle$ occur in $\mathcal{L}(\langle v, w_1\rangle)$ and $\mathcal{L}(\langle v, w_2\rangle)$, respectively (informally, $\mathcal{F}$ contains $\langle (v, w_1){:}p, l_1\rangle$ and $\langle (v, w_2){:}p, l_2\rangle$). Then we call such a pair a *fork*. As p is a partial function, such a fork means that w_1 and w_2 have to be interpreted as the same individual. Such a fork can be deleted by adding both $\mathcal{L}(\langle v, w_2\rangle)$ to $\mathcal{L}(\langle v, w_1\rangle)$ and $\mathcal{L}(w_2)$ to $\mathcal{L}(w_1)$, and then deleting node w_2. At the beginning, we remove the forks from the initial forest. We assume that forks are eliminated as soon as they appear (as part of a rule application) with the proviso that newly generated nodes are replaced by older ones and not vice-versa.

Now we are ready to present the inference rules. Recall that $\mathbf{d} \in \{ls, rs, tri, trz\}$, while $@ \in \{@^{\text{avg}}, @_W^{\text{ws}}, @_W^{\text{owa}}\}$:

($\bot$). If $\langle \bot, l\rangle \in \mathcal{L}(v)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{l = 0\}$.

(R). If $\langle R, l\rangle \in \mathcal{L}(\langle v, w\rangle)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{(v,w):R} \geq l\} \cup \{x_{(v,w):R} \in [0, 1]\}$. The case for concrete roles T is similar.

(f). If $\langle f, l\rangle \in \mathcal{L}(\langle v, w\rangle)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{(v,w):f} \geq l\} \cup \{x_{(v,w):f} \in \{0, 1\}\}$. The case for concrete features t is similar.

(C). If $\langle C, l\rangle \in \mathcal{L}(v)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:C} \geq l\} \cup \{x_{v:C} \in [0, 1]\}$.

($\sqcap$). If $\langle C \sqcap D, l\rangle \in \mathcal{L}(v)$ then *(i)* append $\langle C, x_{v:C}\rangle$, and $\langle D, x_{v:D}\rangle$ to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:C} \otimes x_{v:D} \geq l\}$.

($\sqcup$). If $\langle C \sqcup D, l\rangle \in \mathcal{L}(v)$ then *(i)* append $\langle C, x_{v:C}\rangle$ and $\langle D, x_{v:D}\rangle$ to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:C} \oplus x_{v:D} \geq l\}$.

($\neg$). If $\langle \neg C, l\rangle \in \mathcal{L}(v)$ then *(i)* append $\langle C, x_{v:C}\rangle$ to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:\neg C} = 1 - x_{v:C}\}$.

($\forall$). If *(i)* $\langle \forall R.C, l_1\rangle \in \mathcal{L}(v)$, $\langle R, l_2\rangle \in \mathcal{L}(\langle v, w\rangle$, and *(ii)* the rule has not been already applied to this pair then *(i)* append $\langle C, x_{w:C}\rangle$ to $\mathcal{L}(w)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{w:C} \geq l_1 \otimes l_2\}$.

($\exists$). If $\langle \exists R.C, l\rangle \in \mathcal{L}(v)$ then *(i)* create a new node w, and *(ii)* append $\langle R, x_{(v,w):R}\rangle$ to $\mathcal{L}(\langle v, w\rangle)$, and *(iii)* append $\langle C, x_{w:C}\rangle$ to $\mathcal{L}(w)$, and *(iv)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{w:C} \otimes x_{(v,w):R} \geq l\}$.

($\forall_\mathbf{D}$). If $\langle \forall T.\mathbf{d}, l\rangle \in \mathcal{L}(v)$, the case is similar as in rule ($\forall$).

($\exists_\mathbf{D}$). If $\langle \exists T.\mathbf{d}, l\rangle \in \mathcal{L}(v)$, the case is similar as in rule ($\exists$).

($\mathbf{d}$). If $\langle d, l\rangle \in \mathcal{L}(v)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{\gamma(v{:}\mathbf{d}, l)\}$, where the set $\gamma(v{:}\mathbf{d}, l)$ is obtained from the MILP representation of $g(\mathbf{d})$ by replacing all occurrences of x with l and x_1 with x_v.

($\bar{\mathbf{d}}$). The case $\langle \neg\mathbf{d}, l\rangle \in \mathcal{L}(v)$ is similar as in rule ($\mathbf{d}$), where we use the MILP representation of $\bar{g}(\mathbf{d})$ in place of $g(\mathbf{d})$.

(@). If $\langle @(C_1, \ldots, C_n), l\rangle \in \mathcal{L}(v)$ then *(i)* append all $\langle C_i, x_{v:C_i}\rangle$ to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{\gamma(x_{v:C_1}, \ldots, x_{v:C_n}, l)\}$, where the set $\gamma(x_{v:C_1}, \ldots, x_{v:C_n}, l)$ is obtained from the MILP representation of $g(@)$ by replacing all occurrences of x with l and x_i with $x_{v:C_i}$.

Now, the following proposition can be shown.

Proposition 1 (Termination, Soundness, Completeness): For a given a KB $\mathcal{K}$ (i) the tableau algorithm terminates; (ii) if the expansion rules can be applied to $\mathcal{K}$ such that they yield a complete completion-forest $\mathcal{F}$ such that $\mathcal{C}_\mathcal{F}$ has a solution, then $\mathcal{K}$ has a model; and (iii) if $\mathcal{K}$ has a model, then the expansion rules can be applied in such a way that the tableaux algorithm yields a complete completion-forest for $\mathcal{K}$ such that $\mathcal{C}_\mathcal{F}$ has a solution. □

IV. Representation of some AOs in Fuzzy OWL 2

In this section, we will consider the representation of fuzzy ontologies using a fuzzy extension of OWL 2[4]. In particular, we will show how to represent weighted concepts, OWA operators, and quantifier-guided AOs.

The idea of this methodology for fuzzy ontology representation is to use an OWL 2 ontology and to extend their elements with annotations representing the features of the fuzzy ontology that OWL 2 cannot directly encode. In order to separate the annotations including fuzzy information from other annotations, we use a new annotation property called `fuzzyLabel`, and every annotation is identified by the tag `fuzzyOwl2`, with an attribute `fuzzyType` that specifies the type of element that is being annotated. For full details, we refer the reader to [7].

Now we are ready to show how to represent some cases of AOs, namely weighted concepts, OWA operators, and quantifier-guided AOs. For each of these concepts, we create a new concept and add an annotation property to it (describing the type of the constructor and the value of their parameters).

[4]http://www.straccia.info/software/FuzzyOWL/

Hence, the domain of the annotation is an OWL 2 concept declaration, and the value of the attribute `fuzzyType` is `concept`.

Then, we use a tag `Concept`, with an attribute `type`, describing the type of fuzzy concept that is represented, and other attributes that depend on the concept constructor. In the case of the concept constructors that we consider in this section, such attributes will be used to represent the weights, the concepts to be aggregated, and (possibly) the quantifier. Let us consider each case in detail.

A. Weighted sum concepts

The value of the attribute `type` is `weightedSum`. There are also several additional tags, each of them represented by a pair formed by a weight and a concept. Such pairs are usually called weighted concepts in the fuzzy DL literature and can be considered as fuzzy concepts themselves. For coherence, the syntax of a weighted concept use another tag `Concept`, with the `type` being `weighted`, and attributes `value` and `base` for the weight and the name of the base concept, respectively. Formally, the syntax for the annotation of weighted sum concepts is the following:

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="weightedSum">
    (<WEIGHTED>)+
  </Concept>
</fuzzyOwl2>

<WEIGHTED> := <Concept type="weighted" value="<DOUBLE>"
              base="<STRING>" />
```

B. OWA aggregation concepts

Now, the value of the attribute `type` is `owa`. Due to the reordering step, it is not necessary to associate every weight to a particular concept, so we use two separate list of weights and concept names. The formal syntax is the following:

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="owa">
    <Weights>
        ( <Weight><DOUBLE></Weight> )+
    </Weights>
    <Names>
        ( <Name><STRING></Name> )+
    </Names>
  </Concept>
</fuzzyOwl2>
```

Example 3: Assume that we want to represent the concept $@^{\text{owa}}_{W}(Cheap, CloseToVenue, Comfortable)$, where $W = (0.1, 0.3, 0.6)$. So, we create a new atomic concept A and annotate it as follows: (the example uses OWL/XML syntax [28]):

```
<AnnotationAssertion>
  <AnnotationProperty IRI='#fuzzyLabel'/>
  <IRI>#A</IRI>
  <Literal datatypeIRI='&rdf;PlainLiteral'>
    <fuzzyOwl2 fuzzyType="concept">
      <Concept type="owa">
        <Weights>
            <Weight>0.1</Weight>
            <Weight>0.3</Weight>
            <Weight>0.6</Weight>
        </Weights>
        <Names>
            <Name>Cheap</Name>
            <Name>CloseToVenue</Name>
            <Name>Comfortable</Name>
        </Names>
      </Concept>
    </fuzzyOwl2>
  </Literal>
</AnnotationAssertion>
```

C. Quantified-guided OWA aggregation concepts

This case is similar to the previous one, with the difference that we must also represent the quantifier. As possible quantifiers, we propose to use left-shoulder functions (Fig. 1 (b)) and linear functions (Fig. 1 (e))[5] restricted to the interval $[0, 1]$. These functions produce monotone quantifiers and have already being considered in Fuzzy OWL 2 [7].

So, the first step is to create a new datatype representing the quantifier, and add an annotation to it, specifying the `fuzzyType` as `datatype`, and including a tag `Datatype` with some attributes encoding the type of the function (`rightshoulder` or `linear`), and the parameters `a` and `b`. Formally, the syntax is the following[6]:

```
<fuzzyOwl2 fuzzyType="datatype">
  <DATATYPE>
</fuzzyOwl2>

<DATATYPE> :=
  <Datatype type="leftshoulder" a="<DOUBLE>" b="<DOUBLE>" /> |
  <Datatype type="linear" a="<DOUBLE>" b="<DOUBLE>" />
```

Now, the differences with the previous case is that the value of the attribute `type` is `qowa`, that there is an additional attribute `quantifier` relating the concept with the previously defined quantifier, and that the quantifiers do not need to be represented, as they are implicitly determined by the quantifier. Formally, the syntax is:

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type="qowa" quantifier="<STRING>" >
    <Names>
        ( <Name><STRING></Name> )+
    </Names>
  </Concept>
</fuzzyOwl2>
```

Example 4: Let $Q = linear(0.6, 0.4)$. We want to represent $@^{\text{owa}}_{W_Q}(Cheap, CloseToVenue, Comfortable)$. Note that $w_1 = Q(1/3) - Q(0) = 0.22$, $w_2 = Q(2/3) - Q(1/3) = 0.49 - 0.22 = 0.27$, and $w_3 = Q(1) - Q(2/3) = 1 - 0.49 = 0.51$. Now, we create a new datatype called Q and annotate it as follows:

```
<AnnotationAssertion>
   <AnnotationProperty IRI='#fuzzyLabel'/>
   <IRI>#Q</IRI>
   <Literal datatypeIRI='&rdf;PlainLiteral'>
   <fuzzyOwl2 fuzzyType="datatype">
     <Datatype type="linear" a="0.6" b="0.4" />
  </fuzzyOwl2>
  </Literal>
</AnnotationAssertion>
```

Then, we create a new atomic concept A and annotate it as:

```
<AnnotationAssertion>
  <AnnotationProperty IRI='#fuzzyLabel'/>
  <IRI>#A</IRI>
  <Literal datatypeIRI='&rdf;PlainLiteral'>
```

[5]Linear functions are determined by a parameter c that makes it possible to define $a = c/(c+1), b = 1/(c+1)$.

[6]Note that other functions (e.g. ls, tri, trz) can be used as fuzzy datatypes, but they are not covered here because they cannot be used as quantifiers.

```
    <fuzzyOwl2 fuzzyType="concept">
      <Concept type="qowa" quantifier="Q" >
        <Names>
            <Name>Cheap</Name>
            <Name>CloseToVenue</Name>
            <Name>Comfortable</Name>
        </Names>
      </Concept>
    </fuzzyOwl2>
  </Literal>
</AnnotationAssertion>
```

V. Conclusion

In this work we have shown how to include aggregation operators within fuzzy DLs. Our contribution consist of: (i) definition of syntax and semantics for fuzzy DLs supporting AOs; (ii) a reasoning algorithm for a wide class of AOs that are MILP representable, that includes average, weighted sum and notably OWA and quantifier-based OWA operators; and (iii) a mechanism to support these AOs in Fuzzy OWL 2, a fuzzy version of the standard ontology language OWL 2. Our approach is implemented in the plug-in Fuzzy OWL 2 and in the fuzzyDL reasoner [11].

Concerning future work, we note that the realm of AOs is quite large (see, e.g. [8]). It remains to see which of them and especially *how* they can be encoded within our reasoning algorithm framework based on (especially) MILP or, more generally, on MINLP optimisation tools.

Appendix

We succinctly show that OWA operators are MILP representable. Yager [17] has shown that the *maximisation* of $@_W^{\text{owa}}(x_1, \ldots, x_n)$ can indeed be encoded as a MILP problem. However, this encoding does not work in our setting as we are not maximising $@_W^{\text{owa}}(x_1, \ldots, x_n)$, but rather need to show that the set $g(@_W^{\text{owa}}) = \{\langle x_1, \ldots, x_k, x\rangle \colon @_W^{\text{owa}}(x_1, \ldots, x_k) \geq x\}$ is MILP representable. To this end, we need a different encoding. So, let $N = \{1, \ldots, n\}$. Similarly to [17], we introduce new $[0, 1]$-valued variables y_i ($i \in N$) and impose

$$y_1 \geq y_2 \geq \ldots \geq y_n \ . \tag{8}$$

The intuition is that y_i will take the value of the i-th largest of the x_j. Hence, by definition of OWA, y_i has weight w_i and therefore, we add also the constraint

$$\textstyle\sum_i w_i y_i \geq x \ . \tag{9}$$

The remaining of the encoding concerns now to establish which one among $x_1, \ldots, x_n$ are the y_i. Consider the following set of equations: for $i, j \in N$, $k = 2$, new binary variables z_{ij}

$$\begin{array}{lll} y_i \leq x_j + k z_{ij} & & x_j \leq y_i + k z_{ij} \\ \textstyle\sum_j z_{ij} = n - 1 & \textstyle\sum_i z_{ij} = n - 1 & z_{ij} \in \{0, 1\} \ . \end{array} \tag{10}$$

It can be verified that (i) if $z_{ij} = 0$ then $y_i = x_j$ is imposed; (ii) for any y_i, there is only one x_j imposed to be equal to y_i; and (iii) for any x_j, there is only one y_i imposed to be equal to x_j. That is, there is a bijection among the y_j and the x_i, which together with Eq. (8) guarantees that the equations Eq. (8) - (10) correctly encode $@_W^{\text{owa}}(x_1, \ldots, x_k) \geq x$.

References

[1] 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, 2003.

[2] B. Cuenca-Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, and U. Sattler, "OWL 2: The next step for OWL", Journal of Web Semantics, vol. 6:4, pp. 309–322, 2008.

[3] U. Straccia, "Reasoning within fuzzy Description Logics", Journal of Artificial Intelligence Research, vol. 14, pp. 137–166, 2001.

[4] U. Straccia, "A fuzzy Description Logic for the semantic web", in Fuzzy Logic and the Semantic Web, E. Sanchez, Ed. Elsevier, 2006, pp. 73–90.

[5] U. Straccia, "Multi-criteria decision making in fuzzy Description Logics: a first step", in Proceedings of the 13th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES 2009), ser. LNAI, no. 5711. Springer, 2009, pp. 79–87.

[6] T. Lukasiewicz and U. Straccia, "Managing uncertainty and vagueness in Description Logics for the semantic web", Journal of Web Semantics, vol. 6, pp. 291–308, 2008.

[7] F. Bobillo and U. Straccia. "Representing fuzzy ontologies in OWL 2", in Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010). IEEE Press, 2010, pp. 2695–2700.

[8] V. Torra, Y. Narukawa, Modeling Decisions - Information Fusion and Aggregation Operators. Springer Verlag, 2007.

[9] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decision making", IEEE Transactions on Systems, Man and Cybernetics, vol. 18, pp. 183–190, 1988.

[10] P. Vojtáš, "A fuzzy $\mathcal{EL}$ Description Logic with crisp roles and fuzzy aggregation for web consulting", in Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2006), 2006, pp. 1834–1841.

[11] F. Bobillo and U. Straccia, "fuzzyDL: an expressive fuzzy Description Logic reasoner", in 2008 International Conference on Fuzzy Systems (FUZZ-IEEE 2008). IEEE Computer Society, 2008, pp. 923–930.

[12] P. Hájek, Metamathematics of fuzzy logic. Kluwer, 1998.

[13] R. Hähnle, "Advanced many-valued logics", in Handbook of Philosophical Logic, 2nd Edition, D. M. Gabbay and F. Guenthner, Eds. Kluwer, 2001.

[14] R. R. Yager, "Families of OWA operators", Fuzzy Sets and Systems, vol. 59, pp. 125–148, 1993.

[15] R. R. Yager, J. Kacprzyk, Eds., The ordered weighted averaging operators: theory and applications. Kluwer Academic Publishers, 1997.

[16] R. R. Yager, "Connectives and quantifiers in fuzzy sets", Fuzzy Sets and Systems, vol. 40, pp. 39–75, 1991.

[17] R. R. Yager, "Constrained OWA aggregation", Fuzzy Sets and Systems, vol. 81, pp. 89–101, 1996.

[18] G. J. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications." Prentice-Hall, Inc., 1995.

[19] R. Bellmann and L. Zadeh, "Decision-making in a fuzzy environment", Management Sciences, vol. 17, pp. 141–164, 1970.

[20] U. Straccia, "Description logics with fuzzy concrete domains", in Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005). AUAI Press, 2005, pp. 559–567.

[21] F. Bobillo and U. Straccia, "Fuzzy Description Logics with general t-norms and datatypes", Fuzzy Sets Systems, vol. 160(23), pp. 3382—3402, 2009.

[22] F. Bobillo, F. Bou, and U. Straccia, "On the failure of the finite model property in some fuzzy Description Logics", Fuzzy Sets and Systems, vol. 172(1), pp. 1-12, 2011.

[23] B. Nebel, "Terminological reasoning is inherently intractable, Artificial Intelligence", vol. 43, pp. 235–249, 1990.

[24] G. Stoilos, U. Straccia, G. Stamou, and J. Pan, "General concept inclusions in fuzzy Description Logics", in Proceedings of the 17th Eureopean Conference on Artificial Intelligence (ECAI 2006). IOS Press, 2006, pp. 457–461.

[25] H. Salkin and M. Kamlesh, Foundations of Integer Programming. North-Holland, 1988.

[26] M. Grabisch, "Alternative representations of OWA operators", in The ordered weighted averaging operators. Kluwer Academic Publishers, 1997, pp. 73–85.

[27] J. L. Marichal, Aggregation Operators for Multicriteria Decision Aid. Ph.D. dissertation, Universite de Liege, Belgium, 1999.

[28] B. Motik, B. Parsia, and P. F. Patel-Schneider, Eds., OWL 2 Web Ontology Language XML serialization, http://www.w3.org/TR/owl2-xml-serialization, 2009.