Applied Soft Computing 13 (2013) 3816–3830

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Aggregation operators for fuzzy ontologies☆

Fernando Bobillo[a,*], Umberto Straccia[b]

[a] Department of Computer Science and Systems Engineering, University of Zaragoza, Spain
[b] Istituto di Scienza e Tecnologie dell'Informazione (ISTI), Consiglio Nazionale delle Ricerche (CNR), Pisa, Italy

ARTICLE INFO

Article history:
Received 31 July 2012
Received in revised form 26 February 2013
Accepted 2 May 2013
Available online 31 May 2013

Keywords:
Fuzzy ontologies
Fuzzy Description Logics
Aggregation operators
Fuzzy integrals
Logic for the Semantic Web

ABSTRACT

Fuzzy ontologies extend classical ontologies to allow the representation of imprecise and vague knowledge. Although a relatively important amount of work has been carried out in the field during the last years and they have been successfully used in several applications, several notions from fuzzy logic have not been considered yet in fuzzy ontologies. Among them are aggregation operators, mathematical functions used to fuse different pieces of information, which is a very common need. Some examples of aggregation operators are weighted sums, Ordered Weighting Averaging operators and fuzzy integrals.

In this work, we integrate fuzzy ontologies and aggregation operators. As a theoretical formalism, we provide the syntax and semantics of a fuzzy Description Logic with fuzzy aggregation operators. We provide a reasoning algorithm for the family of operators that are representable using a Mixed Integer Linear Programming optimization problem. We also show how to encode some examples of aggregation operators using the language Fuzzy OWL 2.

1. Introduction

In the last decade, *ontologies* have become the state-of-the-art knowledge representation formalism. An ontology is an explicit and formal specification of the concepts, individuals and relationships that exist in some area of interest, created by defining axioms that describe the properties of these entities [54].

Description Logics (DLs) [2] are a family of logics for representing structured knowledge that play a key role in the design of ontologies. Notably, DLs are essential in the theoretical counterpart of the *Web Ontology Language OWL 2* [20], the standard language to represent ontologies.

Nowadays, it is widely agreed that "classical" ontology languages are not appropriate to deal with *imprecise* and *vague* knowledge, which is inherent to several real world domains. Fuzzy set theory and fuzzy logic [69] have proved to be suitable formalisms to handle these types of knowledge. Therefore *fuzzy ontologies* emerge as useful in several applications, such as information retrieval [1,12,53,62], image interpretation [21,22,34], the Semantic Web and the Internet [19,47,51], decision making [13], recommendation [39], summarization [38], ontology merging [17], and many others [23,24,42,46,48,59].

So far, several fuzzy extensions of DLs can be found in the literature (see the survey in [40]). Existing approaches make it possible to reason with several fuzzy operators (such as different conjunctions, disjunctions, negations and implications), but they do not usually allow using aggregation operators.

Aggregation Operators (AOs) are mathematical functions that are used to combine information [58]. AOs have been widely used in computational intelligence because of their ability to fuse linguistically expressed pieces of information. The arithmetic mean, the weighted minimum and maximum, the weighted sum, the median and the Ordered Weighted Averaging (OWA) operators [64] are well-known examples.

Also worth mentioning are *fuzzy integrals*, which extend the concept of integral in the fuzzy case. Fuzzy integrals have been widely used in the setting of multi-criteria decision making because several AOs (including all the examples in the previous paragraph) can be expressed as particular cases of fuzzy integrals. Fuzzy integrals can also be used for the evaluation of type I quantified sentences [70]. Typical examples of fuzzy integrals are the Choquet [18], Sugeno [57], and the quasi-Sugeno [63] integrals.

Although AOs and fuzzy integrals have been extensively studied in the literature, they are still being researched. For a recent state of the art AOs we refer the reader to [28,29]; some examples of recent work on fuzzy integrals are [30,31,36].

Since existing fuzzy ontologies do not include AOs, the problem that we want to address in this paper is the integration of both fuzzy ontologies and AOs. Such integration is of great interest, in order to

☆ Part of this work has been previously published in the papers "Aggregation Operators and Fuzzy OWL 2", in the Proceedings of the 20th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011), and "Fuzzy Ontologies and Fuzzy Integrals", in the Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011). The present paper is a revised and considerably extended version.

* Corresponding author. Tel.: +34 876 55 55 46.
E-mail addresses: fbobillo@unizar.es (F. Bobillo), straccia@isti.cnr.it (U. Straccia).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.05.008

Table 1
Fuzzy connectives of some outstanding fuzzy logics.

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

combine the advantages of both formalisms. Let us firstly discuss how to improve fuzzy ontology applications with AOs. Very usually, users are not completely satisfied with the results provided with a search engine or a recommender system, because the information does not fit well to his/her needs. It is well known that AOs are very useful to represent *user preferences*, as they offer a very flexible way for the user to combine his/her individual preferences into one single value. Being able to express user preferences would improve the expressive power of fuzzy ontology querying. Furthermore, AOs also play a important role in *decision making* processes, specifying how to combine the different criteria that are relevant when taking a decision [27]. Up to now, fuzzy ontology applications can use different functions to combine information both in a conjunctive or a disjunctive way. However, sometimes it is necessary to consider functions that fill the gap between the intersection and the union. AOs play an important role there [52].

Conversely, extending AO applications with fuzzy ontologies makes it possible to take benefit of the advantages of using ontologies, such as making it easier the definition of a common and agreed vocabulary, improving the interoperability, knowledge reuse, information integration, and maintenance.

A lot of applications can benefit from the possibility of expressing user preferences with respect to a fuzzy ontology, but our research has a specially interesting application in *recommendation systems*. Throughout our paper, we will consider as a possible application of our research an intelligent system for accommodation recommendation, but our approach could be applied to any complex scenario affected by imprecision such as wines [13] or diets [39].

The objective of this paper is thus to integrate both fuzzy ontologies and AOs. More precisely, we provide syntax and semantics of a fuzzy DL extended with fuzzy AOs that make it possible to aggregate several fuzzy concepts. Then, we discuss how to represent them. To this end, we profit from a representation of fuzzy ontologies using OWL 2 annotation properties that has been recently proposed (we will call it *Fuzzy OWL 2*) [10]. The use of annotation properties makes it possible to use current OWL 2 editors for fuzzy ontology representation. This approach has reached a sufficient level of maturity: 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.

The rest of the paper is organized as follows. Section 2 recalls some background knowledge needed to read our paper. Then, Section 3 presents the syntax and semantics of a fuzzy DL with AOs and Section 4 shows how to support them within Fuzzy OWL 2. Next, Section 5 revises some related work and Section 6 sets out some conclusions and ideas for future work. A complete calculus for fuzzy DL reasoning for a family of AOs is provided in the appendix.

2. Preliminaries

2.1. Mathematical fuzzy logic

Fuzzy set theory and fuzzy logic were proposed by Zadeh [69] to manage imprecise and vague knowledge. In fuzzy logics, the convention prescribing that a statement is either true or false is changed. Now, a statement is not necessarily true or false, but its truth is a matter of degree measured on an ordered scale, usually the interval [0, 1]. Fuzzy logic can be interpreted in a wide sense (roughly speaking, as an engineering tool) or in a narrow sense, as a logical system which is a generalization of many-valued logic. In this latter case, it is called *Mathematical Fuzzy Logic* [33].

The main idea behind fuzzy logic is that of fuzzy set. Let X be a set of elements called the reference set. A *fuzzy subset* A of X is defined by a membership function $\mu_A(x)$, or simply $A(x)$, which assigns any $x \in X$ to a value in the interval of real numbers between 0 and 1. As in the classical case, 0 means no-membership and 1 full membership, but now a value between 0 and 1 represents the extent to which x can be considered as an element of X. Some popular membership functions, commonly used to define fuzzy sets, are the trapezoidal, triangular, left-shoulder, right-shoulder, and piecewise linear functions [56].

Fuzzy logics provide compositional calculi of degrees of truth. All crisp set operations are extended to fuzzy sets. The intersection, union, complement and implication set operations are performed in the fuzzy case by a t-norm function $\otimes$, a t-conorm function $\oplus$, a negation function $\ominus$ and an implication function $\Rightarrow$, respectively (see [33] for a formal definition of these functions and their properties). Several t-norms, t-conorms, implications, and negations have been given in the literature; some outstanding examples are shown in Table 1.

A quadruple composed by a t-norm, a t-conorm, an implication function and a negation function determines a *fuzzy logic* (usually called a family of fuzzy operators). The most important fuzzy logics are Łukasiewicz, Gödel, and Product logic, due to the fact that any continuous t-norm can be obtained as a combination of Łukasiewicz, Gödel, and Product t-norm [43]. The standard fuzzy logic (sometimes called "Zadeh logic" in the fuzzy DL literature) is subsumed by Łukasiewicz fuzzy logic, since every fuzzy operator of standard fuzzy logic can be simulated with the fuzzy operators of Łukasiewicz logic. The connectives of these four main fuzzy logics are shown in Table 1.

In our setting, *fuzzy statements* have the form $\phi \geqslant \alpha$ or $\phi \leqslant \alpha$, where $\alpha \in [0, 1]$ and ϕ is a statement, which encode that the degree of truth of ϕ is *at most* α (resp. *at least* α).

2.2. Aggregation operators

Aggregation operators (AOs) are mathematical functions that are used to combine different pieces of 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 [58].

There is no standard definition of AO. Usually, given a domain D (such as the reals), an AO of dimension n is a mapping $@ : \mathbb{D}^n \to \mathbb{D}$. For us, $D = [0, 1]$. Thus, an AO aggregates n values of n different criteria. In our scenario, such criteria will be represented by using fuzzy concepts from a fuzzy ontology.

Along this paper, we will use $N = \{1, \ldots, n\}$ to denote the set of indices of an AO with arity n. We will also use σ to denote a

Table 2
Some examples of aggregation operators.

Operator	Definition
Maximum	$x_{\sigma(1)}$
Minimum	$x_{\sigma(n)}$
kth order statistic	$x_{\sigma(k)}$
Arithmetic mean	$\frac{1}{n}\sum_{i=1}^{n} x_i$
Weighted sum	$\sum_{i=1}^{n} w_i x_i$
Weighted maximum	$\max_{i=1}^{n} \min\{v_i, x_i\}$
Weighted minimum	$\min_{i=1}^{n} \max\{\ominus v_i, x_i\}$
Median	$\begin{cases} \frac{1}{2}(x_{\sigma(n/2)} + x_{\sigma(n/2+1)}) & \text{if } n \text{ is even} \\ x_{\sigma((n+1)/2)} & \text{if } n \text{ is odd} \end{cases}$

permutation such that $x_{\sigma(1)} \geq x_{\sigma(2)} \geq \cdots \geq x_{\sigma(n)}$, i.e., $x_{\sigma(i)}$ is the ith largest of the values $x_1, \ldots, x_n$.

Often, an AO @ is parameterized with a vector of n weights $W = [w_1, \ldots, w_n]$ such that $w_i \in [0, 1]$ and $\sum_{i=1}^{n} w_i = 1$. In that case the AO is denoted as $@_W$. In other cases, AOs are parameterized with a vector of n weights $V = [v_1, \ldots, v_n]$ such that $v_i \in [0, 1]$ and $\max_{i=1}^{n} v_i = 1$.

Some examples of AOs are *maximum*, *minimum*, *order statistic*, *arithmetic mean*, *weighted sum* (or weighted mean) $@_W^{\text{ws}}$, *weighted maximum* $@_W^{\text{wmax}}$, *weighted minimum* $@_W^{\text{wmin}}$ and *median* (see the definitions in Table 2, with $\ominus$ being a negation function).

2.2.1. OWA

Another class of AOs are the *Ordered Weighted Averaging* (OWA) operators [64,68]. OWA operators provide a parameterized class of mean type AOs. Formally, an OWA operator of dimension n is an AO such that:

$$@_W^{\text{owa}}(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_i x_{\sigma(i)} \tag{1}$$

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. As a result, the OWA operator is non-linear.

By choosing different weights, OWA operators can implement different AOs, such as arithmetic mean, kth maximum, kth minimum, median or order statistic, among others. For example, the maximum and the minimum can be obtained by using $W = [1, 0, \ldots, 0]$ and $W = [0, \ldots, 0, 1]$, respectively. It is worth to note that weighted sum, maximum and minimum cannot be represented as OWA operators.

OWA operators verify internality, i.e., $\min(x_1, \ldots, x_n) \leq @(x_1, \ldots, x_n) \leq \max(x_1, \ldots, x_n)$. Since the two extreme cases of OWA operators correspond to the largest t-norm (the minimum) and the smallest t-conorm (maximum), the class of OWA operators nicely "fills the gap" between the intersection and the union [52].

2.2.2. Approximated OWA

In the literature, some approximations of OWA operators that are less demanding from a computational point of view have been proposed. We will consider the approximation in [41], which solves some problems in [26].

$$@_W^{\text{owa}}(x_1, \ldots, x_n) \approx \left(\frac{1}{n} - \frac{w_n - w_1}{2}\right) \sum_{i=1}^{n} x_i + \frac{w_n - w_1}{n-1} \sum_{\{i,j\} \subseteq N} \min\{x_i, x_j\} \tag{2}$$

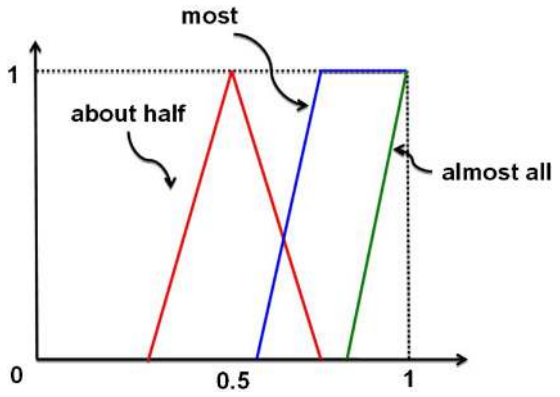

Fig. 1. Some examples of fuzzy quantifiers.

2.2.3. Quantifier-guided OWA

The problem of obtaining the weights of the OWA operators has been largely studied. The interested reader may look at the survey in [25]. In this section we will focus on one of the possibilities, proposed in the setting of *quantifier-guided aggregation* [67].

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 or proportional [37]. We will only 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. Fig. 1 shows some examples of fuzzy quantifiers.

An important class of quantifiers is the *Regular Increasing Monotone* (RIM) quantifiers [65] satisfying the boundary conditions $Q(0) = 0$, $Q(1) = 1$ and being monotone increasing, i.e., $Q(x_1) \leq Q(x_2)$ when $x_1 \leq x_2$. Essentially, these quantifiers are characterized by the idea that as the proportion increases, the degree of satisfaction does not decrease. Turning back to the example in Fig. 1, *most* and *almost all* are RIM quantifiers, but *about half* is not.

A RIM Q can be used to define an OWA weighting vector W_Q of dimension n. The weights are obtained as

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right) \tag{3}$$

where indeed $w_i \in [0, 1]$ and $\sum_i w_i = 1$, $i \in N$.

The resulting OWA operator can be used to evaluate the degree of truth of quantifier propositions (see [67] for details).

2.2.4. Fuzzy integral

In the previous sections, we have studied AOs where the criteria are mutually independent. In this section, we will consider fuzzy integrals, a more general class of AOs that make it possible to measure the importance of sets of criteria. In fuzzy integrals, fuzzy measures are taken into account, allowing redundancy, complementariness and other interactions among the different criteria to be expressed.

Let $X = \{x_1, \ldots, x_n\}$ be a *set of criteria*. A *fuzzy measure* μ is a function $\mu : 2^X \to [0, 1]$ verifying the boundary conditions $\mu(\emptyset) = 0$, $\mu(X) = 1$ and monotonicity: $A \subseteq B \Rightarrow \mu(A) \leq \mu(B)$.

Let $f : X \to \mathbb{R}^+$ be a function such that $f(x_i)$ measures the global satisfaction of an user according to some criterion x_i. The problem now is how to aggregate the evaluations of the individual criteria (i.e., the values $f(x_i)$), into an overall evaluation where the criteria are weighted according to a fuzzy measure μ. This can be done by considering the fuzzy integral of the function f with respect to a fuzzy measure μ.

As in the previous section, we will consider a permutation σ such that $f(x_\sigma(i))$ is the ith largest of the values $f(x_1), \ldots, f(x_n)$. We use a permutation such that the values are ordered in descending order because this is what we did for OWA operators, but please note that it is usual to find in the literature equivalent formulations based on a permutation such that the values are ordered in ascending order.

We will also define the set $A_{\sigma(i)} = \{x_{\sigma(1)}, \ldots, x_{\sigma(i)}\}$ for $i \in \{1, \ldots, n\}$ and $A_{\sigma(0)} = 0$.

In the following we will describe three types of fuzzy integrals: Choquet, Sugeno, and quasi-Sugeno integrals. The Choquet integral is suitable for cardinal aggregation (when the distance between the elements has a meaning), whereas the Sugeno integral is more suitable for ordinal aggregation (where only the order of elements is important) [35].

The *Choquet integral* [18] of a function $f: X \to [0, 1]$ with respect to μ is defined by

$$@^{\mathrm{ci}}_{\mu}(f) = \sum_{i=1}^{n} f(x_{\sigma(i)})[\mu(A_{\sigma(i)}) - \mu(A_{\sigma(i-1)})], \tag{4}$$

where $f(x_{\sigma(0)}) = 0$.

The Choquet integral is more general than other AOs, such as weighted sum and OWA. In particular, if the fuzzy measure is additive, then $\forall B \subseteq X$

$$\mu(B) = \sum_{x_i \in B} w_i \tag{5}$$

and hence the Choquet integral is equivalent to a weighted sum $@^{\mathrm{ws}}_{[w_1,\ldots,w_n]}$.

Observe that Eq. (4) can be equivalently rewritten as:

$$@^{\mathrm{ci}}_{\mu}(f) = \sum_{i=1}^{n} [f(x_{(i)}) - f(x_{(i-1)})]\mu(A_{(i)}) \tag{6}$$

where $\cdot_{(i)}$ indicates that the indices have been permuted so that $0 = f(x_{(0)}) \leq f(x_{(1)}) \leq \cdot \leq f(x_{(n)}) \leq 1$ and $A_{(i)} = \{x_{(i)}, x_{(i+1)}, \ldots, x_{(n)}\}$.

The *Sugeno integral* [57] of a function $f: X \to [0, 1]$ with respect to μ is defined by

$$@^{\mathrm{si}}_{\mu}(f) = \max_{i=1}^{n} \{\min\{f(x_{\sigma(i)}), \mu(A_{\sigma(i)})\}\} \tag{7}$$

This integral also generalizes some AOs, such as weighted minimum and weighted maximum.

Sugeno integral can be further generalized by considering any t-norm function $\otimes$. However, the only meaningful t-conorm is the maximum. This way, we obtained the *quasi-Sugeno integral* [63] of a function $f: X \to [0, 1]$ with respect to μ, which is defined by:

$$@^{\mathrm{qsi}}_{\mu}(f) = \max_{i=1}^{n} \{f(x_{\sigma(i)}) \otimes \mu(A_{\sigma(i)})\} \tag{8}$$

2.2.5. Some families of fuzzy measures

One of the problems of fuzzy integrals is that they need too many parameters. This complicates their practical utility, as they become more difficult for an user. More precisely, the definition of the fuzzy measure μ requires $2^n - 2$ parameters to aggregate a set X of size n, i.e., the power set of X excluding the values $\mu(\emptyset) = 0$, $\mu(X) = 1$ which are fixed by the boundary conditions.

Several approaches have been proposed to reduce the number of parameters. Among them, we can find $\oplus$-decomposable measures[1] for a given t-conorm $\oplus$ and additive measures.

An *additive measure* verifies, for each $i, j \in N$:

$$\mu\{x_i, x_j\} = \mu\{x_i\} + \mu\{x_j\} \tag{9}$$

A *$\oplus$-decomposable measure* [63] verifies, for each $i, j \in N$ with $\mu\{x_i\} \cap \mu\{x_j\} = \emptyset$, that:

$$\mu\{x_i, x_j\} = \mu\{x_i\} \oplus \mu\{x_j\} \tag{10}$$

Note that additive measures are a $\oplus$-decomposable measure for the Łukasiewicz t-conorm.

For these two families of fuzzy measures we just need to specify n parameters, namely the values of the singletons $\mu\{x_i\}$. Hence, we can parameterize the integrals using a vector W where $w_i = \mu\{x_i\}$.

As usual, additive measures require that $\sum_{i=1}^{n} w_i = 1$, but $\oplus$-decomposable measures require that $\oplus_{i=1}^{n} w_i = 1$.

3. A fuzzy Description Logic with aggregation operators

The aim of this section is to show how current fuzzy DLs can be extended to support AOs (in a general sense that includes fuzzy integrals).

In our setting, the values $x_1, \ldots, x_n$ that an AO has to aggregate are represented by using fuzzy concepts $C_1, \ldots, C_n$ from a fuzzy ontology. We will assume that the AOs are parameterized using a vector of weights $W = [w_1, \ldots, w_n]$, $w_i \in \mathbb{Q}$. Consequently, AOs can be denoted as $@_W(C_1, \ldots, C_n)$. We will not care here about the interpretation of the weights, as this will only be relevant in the reasoning algorithm.

As discussed in the previous section, in order to avoid fuzzy integrals having too many parameters to be practical, we will restrict to additive or $\oplus$-decomposable fuzzy measures. This way, w_i represents the relative importance of the fuzzy concept C_i (formally, $\mu\{C_i\}$). If $\sum_{i=1}^{n} w_i = 1$, we can assume an additive fuzzy measure. If $\oplus_{i=1}^{n} w_i = 1$, we can assume a $\oplus$-decomposable fuzzy measure. If none of these conditions is satisfied, the weighting vector is not correct.

For illustrative purposes, we will just consider a minimal fuzzy DL, namely fuzzy $\mathcal{ALCF}(\mathbf{D})$ [56], which is the basic fuzzy DL $\mathcal{ALC}$ extended with functional roles (letter $\mathcal{F}$) and fuzzy concrete domains (letter $\mathbf{D}$, defined below). For a more expressive fuzzy DL the reader is referred to e.g. [6].

A *fuzzy concrete 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 $\mathbf{d}^{\mathbf{D}} : \Delta_{\mathbf{D}}^n \to [0, 1]$, which is a n-ary fuzzy relation over $\Delta_{\mathbf{D}}$ [56]. In our specific fuzzy DL, we assume that predicates are unary and $\Delta_{\mathbf{D}}$ are non-negative real numbers.

In the following we formally describe the syntax, semantics and main reasoning tasks of this fuzzy DL.

3.1. Syntax

In fuzzy DLs there are three pairwise disjoint alphabets of individuals, fuzzy concepts and fuzzy roles. Fuzzy concepts are fuzzy sets of individuals, and fuzzy roles are fuzzy binary relations between individuals.

Individuals can be abstract or concrete, depending on whether they are instances of the fuzzy concepts or members of the concrete interpretation domain $\Delta_{\mathbf{D}}$, respectively. Similarly, roles can be abstract or concrete, depending on whether they relate an individual with another individual or with a fuzzy concrete value, respectively. Within the alphabet of abstract and concrete roles, we have to distinguish subsets of *abstract functional roles* (denoted f) and *concrete functional roles* (denoted t), respectively. Functional roles are also called *features*.

[1] In the literature, they are often named $\perp$-decomposable measures, but in this paper, as usual in DLs, we will use $\perp$ with a different meaning.

Example 3.1. Let us consider the problem of designing an intelligent system for accommodations recommendation. Fuzzy concepts in this domain are Hotel, HotelChain, Cheap and Comfortable. Any particular example of hotel, such as the OverlookHotel is an individual. The binary relation hasHotel between a hotel chain and a hotel is a fuzzy abstract role, as it relates two individuals. However, the binary relation price is a fuzzy concrete role, as it relates an individual with a numerical value or a fuzzy membership function, and it is functional (if we assume that all the rooms have the same price).

Concepts (denoted C) of the language can be built inductively from atomic concepts (A), top concept $\top$, bottom concept $\bot$, abstract roles (R), concrete roles (T) and predicates $\mathbf{d}$ as follows. The syntax of fuzzy concepts in $\mathcal{ALCF}(\mathbf{D})$ is as follows [6]:

$$C_1, C_2 \rightarrow \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \forall R.C \mid \exists R.C \mid \forall T.\mathbf{d} \mid \exists T.\mathbf{d}$$
$$\mathbf{d} \rightarrow ls(a,b) \mid rs(a,b) \mid tri(a,b,c) \mid trz(a,b,c,d)$$

where ls, rs, tri, trz stand for left-shoulder, right-shoulder, triangular and trapezoidal membership functions [56], which are our concrete domain predicates, and some roles R or T are *features*.

Example 3.2. In Example 3.1, the complex fuzzy concept $\mathsf{HotelChain} \sqcap \forall \mathsf{hasHo-tel}.\neg \mathsf{Cheap}$ denotes hotel chains only having hotels that are not cheap. Finally, given a quantity in euros denoting the price of a hotel, the cheapness can be represented using a left-shoulder membership function $ls(60, 120)$, indicating the prices below 60 are always considered cheap and prices in [60, 120] are considered cheap in some degree.

In order to support AOs in fuzzy DLs, we will extend the syntax of concepts as follows. Let $@_W$ be an AO parameterized with a vector of weights $W = [w_1, \ldots, w_n]$, $n \in \mathbb{N}$. Then, the following are concept expressions:

$$C_1, \ldots, C_n \rightarrow C \mid @_W(C_1, \ldots, C_n)$$

Example 3.3. Consider again Example 3.1. The following concept expressions denote the fuzzy set of hotels that satisfy three criteria (being cheap, comfortable, and close to the venue), where the degree of satisfaction of the criteria is computed using 4 different AOS, namely the weighted sum, an OWA operator, the Choquet integral and the Sugeno integral:

$\mathsf{Hotel} \sqcap @_W^{\mathsf{ws}}(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue})$

$\mathsf{Hotel} \sqcap @_W^{\mathsf{owa}}(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue})$

$\mathsf{Hotel} \sqcap @_W^{\mathsf{ci}}(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue})$

$\mathsf{Hotel} \sqcap @_W^{\mathsf{si}}(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue})$

A *Fuzzy Knowledge Base* (or *fuzzy Ontology*) $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ comprises a fuzzy ABox $\mathcal{A}$ and a fuzzy TBox $\mathcal{T}$.

A *fuzzy ABox* contains a finite set of *fuzzy assertions* of the following types:

- Fuzzy *concept assertions* of the form $\langle \mathrm{a} : \mathrm{C}, \alpha \rangle$, with $\alpha \in [0, 1]$ and stating that individual a is an instance of concept C with degree at least α.
- Fuzzy *role assertions* of the form $\langle (\mathrm{a}, \mathrm{b}) : \mathrm{R}, \alpha \rangle$, $\alpha \in [0, 1]$, meaning that the pair of individuals (a, b) is an instance of role R with degree at least α.

A *fuzzy TBox* consists of a finite set of *fuzzy General Concept Inclusions (fuzzy GCIs)*, which are expressions of the form $\langle C \sqsubseteq D, \alpha \rangle$, $\alpha \in [0, 1]$, meaning that the degree of subsumption between concept C and D is not less than α. $C \equiv D$ is a shorthand for both $\langle C \sqsubseteq D, 1 \rangle$ and $\langle D \sqsubseteq C, 1 \rangle$.

Example 3.4. Consider the following table with some data about hotels:

Hotel	Price	Distance	Stars
h_1	150	300	4
h_2	100	500	2

We may encode the information using a fuzzy KB $\mathcal{K}$ with the following ABox:

$\mathsf{Hotel}(\mathsf{h}_1)$,

$\mathsf{Hotel}(\mathsf{h}_2)$,

$\mathsf{price}(\mathsf{h}_1, 150)$,

$\mathsf{price}(\mathsf{h}_2, 100)$,

$\mathsf{distance}(\mathsf{h}_1, 300)$,

$\mathsf{distance}(\mathsf{h}_2, 500)$,

$\mathsf{stars}(\mathsf{h}_1, 4)$,

$\mathsf{stars}(\mathsf{h}_2, 2)$

where price, distance and stars are concrete features.

The TBox holds some definitions for cheap, comfortable and close (to some particular venue) hotels and contains the following axioms:

$\mathsf{Cheap} = \exists \mathsf{price}.ls(60, 120)$
$\mathsf{Comfortable} = \exists \mathsf{star}.rs(3, 5)$
$\mathsf{CloseToVenue} = \exists \mathsf{distance}.ls(200, 400)$

3.2. Semantics

The semantics of the logic is defined using a fuzzy interpretation. 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:

- To each abstract individual a an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.
- To each *concrete individual* v an element $v_{\mathbf{D}} \in \Delta_{\mathbf{D}}$.
- To each concept C a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow [0, 1]$.
- To each abstract role R a function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0, 1]$.
- To each abstract feature f a partial function $f^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow \{0, 1\}$[2] 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.
- To each concrete role T a function $T^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \rightarrow [0, 1]$.
- To each concrete feature t a partial function $t^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \rightarrow \{0, 1\}$ 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 $\Rightarrow$ [33], the fuzzy interpretation function is extended to *complex concepts* and *fuzzy axioms* as in Fig. 2, where $@_W$ is an AO.[3]

[2] In fuzzy DLs, abstract and concrete features are usually crisp [7].

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

Concept		Semantics
$(\top)^{\mathcal{I}}(x)$	=	1
$(\bot)^{\mathcal{I}}(x)$	=	0
$(A)^{\mathcal{I}}(x)$	=	$A^{\mathcal{I}}(x)$
$(C_1 \sqcap C_2)^{\mathcal{I}}(x)$	=	$C_1^{\mathcal{I}}(x) \otimes D^{\mathcal{I}}(x)$
$(C_1 \sqcup C_2)^{\mathcal{I}}(x)$	=	$C_1^{\mathcal{I}}(x) \oplus D^{\mathcal{I}}(x)$
$(\neg C)^{\mathcal{I}}(x)$	=	$\ominus C^{\mathcal{I}}(x)$
$(\forall R.C)^{\mathcal{I}}(x)$	=	$\inf_{y\in\Delta^{\mathcal{I}}}\{R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)\}$
$(\exists R.C)^{\mathcal{I}}(x)$	=	$\sup_{y\in\Delta^{\mathcal{I}}}\{R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)\}$
$(\forall T.\mathbf{d})^{\mathcal{I}}(x)$	=	$\inf_{v\in\Delta_{\mathbf{D}}}\{T^{\mathcal{I}}(x,v) \Rightarrow \mathbf{d}^{\mathbf{D}}(v)\}$
$(\exists T.\mathbf{d})^{\mathcal{I}}(x)$	=	$\sup_{v\in\Delta_{\mathbf{D}}}\{T^{\mathcal{I}}(x,v) \otimes \mathbf{d}^{\mathbf{D}}(v)\}$
$(@_W(C_1,\ldots,C_n))^{\mathcal{I}}(x)$	=	$@_W(C_1{}^{\mathcal{I}}(x),\ldots,C_n{}^{\mathcal{I}}(x))$
Axiom		**Semantics**
$(a{:}C)^{\mathcal{I}}$	=	$C^{\mathcal{I}}(a^{\mathcal{I}})$
$((a,b){:}R)^{\mathcal{I}}$	=	$R^{\mathcal{I}}(a^{\mathcal{I}},b^{\mathcal{I}})$
$(C \sqsubseteq D)^{\mathcal{I}}$	=	$\inf_{x\in\Delta^{\mathcal{I}}}\{C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)\}$

Fig. 2. Semantics of fuzzy concepts and axioms.

3.3. Reasoning tasks

Let $\phi \in \{a:C, (a,b):R, C \sqsubseteq D\}$ and $\tau = \langle t, \alpha\rangle$. A fuzzy interpretation $\mathcal{I}$ *satisfies* (is a *model* of) a fuzzy axiom τ iff $\phi^{\mathcal{I}} \geq \alpha$.

There several reasoning tasks in fuzzyDLs. In this paper, we will focus on computing the *Best Entailment Degree* (BED) of an axiom α w.r.t. a fuzzy KB $\mathcal{K}$, defined as $bed(\mathcal{K},\phi) = \sup\{\alpha \mid \mathcal{K} \models \langle\phi,\alpha\rangle\}$. Other reasoning tasks (such as KB consistency, entailment, subsumption, fuzzy concept satisfiability or best satisfiability degree of a concept) can be reduced to it [7].

Example 3.5. Consider again Example 3.4, where an user is looking for a cheap, close to some venue and comfortable hotel. Using some AO $@_W$, we may compute the degree to which hotel h_i satisfies our request by suitably aggregating the degree of cheapness, closeness and comfortableness of hotel h_i as follows:

$$bed(\mathcal{K}, \mathsf{h}_i : @_W(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue}))$$

The following table shows the membership degrees of hotels h_1, h_2 to the fuzzy concepts Cheap (CH), Comfortable (CO) and CloseToVenue (CV), as well as the global value obtained using different AOs. In weighted sum, OWA, approximated OWA (AOWA) and Choquet integral we assume $W = [0.7, 0.1, 0.2]$, whereas in weighted minimum, weighted maximum, Sugeno integral and quasi-Sugeno integral we assume $W = [1, 0.2, 0.3]$. In the two latter integrals we consider a minimum-decomposable fuzzy measure, and in quasi-Sugeno integral we assume Łukasiewicz t-norm.

	CH	CO	CV	$@^{\mathrm{ws}}_W$	$@^{\mathrm{owa}}_W$	$@^{\mathrm{aowa}}_W$	$@^{\mathrm{wmin}}_W$	$@^{\mathrm{wmax}}_W$	$@^{\mathrm{ci}}_W$	$@^{\mathrm{si}}_W$	$@^{\mathrm{qsi}}_W$
h_1	0	0.5	0.5	0.15	0.4	0.46	0	0.3	0.15	0.2	0
h_2	0.33	0	0	0.23	0.23	0.19	0.33	0.33	0.23	0.33	0.03

An important contribution of this paper is a reasoning algorithm to solve the BED problem in fuzzy DLs with AOs (recall that other reasoning tasks can be reduced to this problem). Our algorithm is similar to others in the literature with the only difference being the management of AOs in Section A.1. The algorithm is based on a combination of tableaux rules and a Mixed Integer Linear Programming (MILP) optimization problem. To this end, we require that the restrictions involving AOs can be represented as part of a MILP problem. Since the algorithm is rather technical, the interesting reader is referred to the Appendix for details. Here, we will just summarize in Table 3 the complexity of each studied AO in terms of the number of variables required to represent the aggregation of n concepts as part of a MILP problem. Note that the number of variables is not minimal and more efficient encodings could be found.

Table 3
Number of new variables introduced in the MILP encoding.

Operator	♯ variables
Weighted sum	0
Weighted maximum	$3n$
Weighted minimum (general case)	$\geq 3n$
Weighted minimum (Łukasiewicz negation)	$2n$
OWA	$n(n+1)$
Approximated OWA	$n(n+1)$
Quasi-Sugeno integral	$\geq n(n+5)$
Sugeno integral	$n(n+6)$

4. Representation in Fuzzy OWL 2

In this section, we will discuss the representation of fuzzy ontologies with some kinds of aggregation operators (namely weighted sum, weighted maximum, weighted minimum, OWA and quantifier-guided AO) and fuzzy integrals (namely the Choquet, the Sugeno and the quasi-Sugeno integrals) using a fuzzy extension of OWL 2.

In the previous section we have considered fuzzy ontologies using a DL syntax. This syntax is suitable for the formal definition of the syntax and semantics of the logic, their associated reasoning tasks and the algorithms for solving them. However, in practice, fuzzy ontology developers use more user-friendly syntaxes, based on fuzzy extensions of the ontology languages. In this section, we will consider the representation of fuzzy ontologies using a fuzzy extension of OWL 2. This representation can be used in a transparent way with a Protégé plug-in.[4]

The key idea of this representation is to use an OWL 2 ontology and 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 [10].

In order to store and to exchange OWL 2 ontologies, concrete syntaxes are needed. For this purpose, OWL 2 provides several different syntaxes [20]: OWL/XML, RDF/XML, functional style, Manchester or Turtle syntaxes. In the examples in this section, we will use OWL/XML syntax [44].

Example 4.1. Let us represent the fuzzy datatype $\mathsf{CheapPrice} = \mathsf{ls}(60, 120)$ denoting the age of a young person. This fuzzy datatype is represented using a datatype definition of base type `xsd:double` with range in [0, 1000], with an annotation indicating the type of the function (left-shoulder), and its parameters (a, b). More precisely, the representation is the following:

```
<AnnotationAssertion>
  <AnnotationProperty IRI='#fuzzyLabel'/>
  <IRI> #CheapPrice</IRI>
  <Literal datatypeIRI='&rdf;PlainLiteral'>
  <fuzzyOwl2 fuzzyType="datatype">
    <Datatype type="leftshoulder" a="60" b="120" />
  </fuzzyOwl2>
  </Literal>
</AnnotationAssertion>
```

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

Now we are ready to show how to represent some cases of AOs and fuzzy integrals. 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. 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 being 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.

4.1. Weighted sum, maximum and minimum

These three AOs can be dealt with in a similar way (even if the interpretation of the weights change). The value of the attribute `type` is one of `weightedSum`, `weightedMaximum` or `weightedMinimum`. 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, maximum and minimum concepts is the following:

```
<fuzzyOwl2 fuzzyType="concept">
  <Concept type=<WTYPE> >
    (<WEIGHTED>)+
  </Concept>
</fuzzyOwl2>
<WTYPE>:= "weightedSum" | "weightedMaximum" |
'weightedMinimum"
<WEIGHTED>:= <Concept type="weighted" value="<DOUBLE>
'base="<STRING> ' />
```

Example 4.2. Let us see how to represent the weighted sum concept $@^{ws}_{W}(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue})$, where $W = [0.2, 0.3, 0.5]$. So, we create a new atomic concept A and annotate it as follows:

```
<AnnotationAssertion>
  <AnnotationProperty IRI='#fuzzyLabel'/>
  <IRI> #A</IRI>
  <Literal datatypeIRI='&rdf;PlainLiteral'>
    <fuzzyOwl2 fuzzyType="concept">
      <Concept type="weightedSum">
        <Concept type="weighted" value="0.2" base="Cheap" />
        <Concept type="weighted" value="0.3" base="Comfortable" />
        <Concept type="weighted" value="0.5" base="CloseToVenue" />
      </Concept>
    </fuzzyOwl2>
  </Literal>
</AnnotationAssertion>
```

Fig. 3 shows how the plug-in can be used to represent the weighted sum concept defined in Example 4.2. Interestingly, users can specify the weights and the aggregated concepts without worrying about the underlying syntax.

4.2. OWA 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:

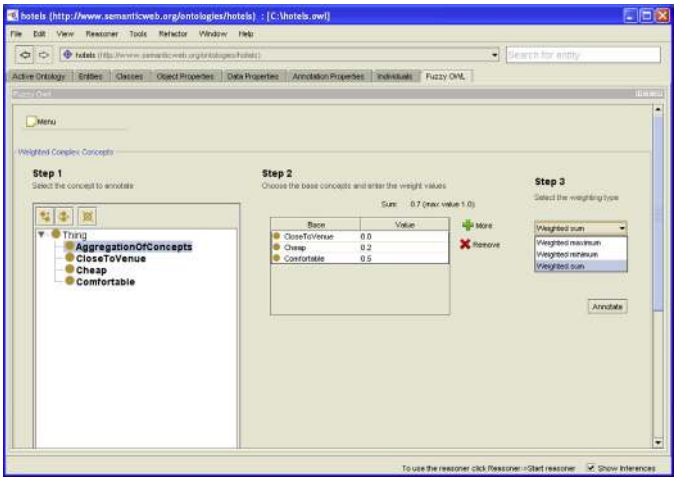

Fig. 3. Representation of weighted sum concepts using the Protégé plug-in.

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

Example 4.3. Assume that we want to represent the OWA aggregation concept $@^{owa}_{W}(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue})$, where $W = [0.2, 0.3, 0.5]$. So, we create a new atomic concept A and annotate it as follows:

```
<AnnotationAssertion>
  <AnnotationProperty IRI='#fuzzyLabel'/>
  <IRI> #A</IRI>
  <Literal datatypeIRI='&rdf;PlainLiteral'>
    <fuzzyOwl2 fuzzyType="concept">
      <Concept type="owa">
        <Weights>
          <Weight> 0.2</Weight>
          <Weight> 0.3</Weight>
          <Weight> 0.5</Weight>
        </Weights>
        <Names>
          <Name> Cheap</Name>
          <Name> Comfortable</Name>
          <Name> CloseToVenue</Name>
        </Names>
      </Concept>
    </fuzzyOwl2>
  </Literal>
</AnnotationAssertion>
```

Fig. 4 shows how to represent the OWA concept defined in Example 4.3 using the plug-in.

4.3. Quantifier-guided OWA 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 linear functions (Fig. 5(a))[5] and right-shoulder functions (Fig. 5(b)) restricted to the interval [0, 1]. These functions produce RIMs and have already being considered in Fuzzy OWL 2 [10].

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

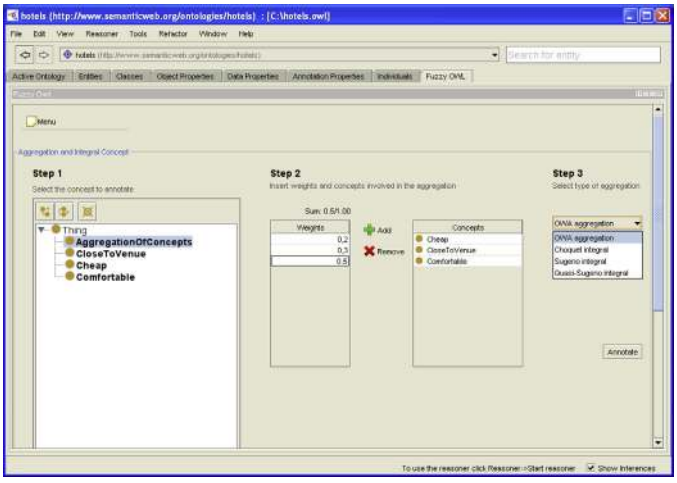

Fig. 4. Representation of OWA concepts using the Protégé plug-in.

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="rightshoulder" 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 4.4. Let $Q = \mathit{linear}(0.6, 0.4)$. We want to represent $@^{\mathrm{owa}}_{W_Q}(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue})$. 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'>
    <fuzzyOwl2 fuzzyType="concept">
      <Concept type="qowa" quantifier="Q" >
        <Names>
          <Name> Cheap</Name>
          <Name> CloseToVenue</Name>
            <Name> Comfortable</Name>
        </Names>
      </Concept>
    </fuzzyOwl2>
  </Literal>
</AnnotationAssertion>
```

Fig. 6 shows how the plug-in can be used to represent the quantifier-guided OWA concept defined in Example 4.4, assuming that the linear quantifier Q has already been defined using the plug-in.

4.4. Fuzzy integrals

We will show how to represent Choquet, the Sugeno and the quasi-Sugeno integrals. Once again, due to the reordering step it is not necessary to associate every weight to a particular concept, but we need two independent lists. The formal syntax is the following:

```
<fuzzyOwl2 fuzzyType="concept">  <Concept
type=<INTEGRAL> >    <Weights>      (<Weight>
<DOUBLE> </Weight>)+    </Weights>    <Names>
(<Name> <STRING> </Name>)+    </Names>
</Concept> </fuzzyOwl2> <INTEGRAL>:= "choquet"
| "sugeno" | "quasiSugeno"
```

Example 4.5. Let us show how to represent the concept $\mathcal{CI}_W(\mathsf{Cheap}, \mathsf{Comfortable}, \mathsf{CloseToVenue})$, where $W = [0.2, 0.3, 0.5]$. So, we create a new atomic concept A and annotate it as follows:

```
<AnnotationAssertion>
  <AnnotationProperty IRI='#fuzzyLabel'/>
  <IRI> #A</IRI>
  <Literal datatypeIRI='&rdf;PlainLiteral'>
    <fuzzyOwl2 fuzzyType="concept">
    <Concept type="choquet">
      <Weights>
        <Weight> 0.2</Weight>
        <Weight> 0.3</Weight>
        <Weight> 0.5</Weight>
      </Weights>
      <Names>
        <Name> Cheap</Name>
        <Name> CloseToVenue</Name>
        <Name> Comfortable</Name>
      </Names>
    </Concept>
  </fuzzyOwl2>
  </Literal>
</AnnotationAssertion>
```

The integral in Example 4.5 can be defined using the plug-in as in Fig. 4.

5. Related work

To the best of our knowledge, this is the first effort in the combination of fuzzy ontologies and fuzzy integrals. However, there is some previous research combining AOs and fuzzy DLs. [60,61] present a fuzzy extension of the DL $\mathcal{EL}$ with general AOs. Our work is more general as we consider a more expressive fuzzy DL and specifically show how to represent and reason with several cases of AOs. However, the complexity of reasoning in fuzzy $\mathcal{EL}$ is smaller.

Another close work is [50], which considers fuzzy ontologies with general fuzzy quantifiers that could be used for some type of quantifier-guided aggregation. The approach differs to ours

[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.

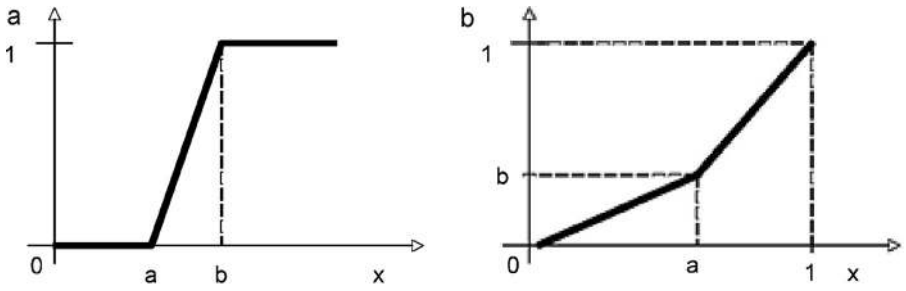

Fig. 5. (a) Left-shoulder function; (b) linear function.

because, roughly speaking, they quantify the degree of membership of fuzzy relations whereas we quantify the degree of membership of fuzzy concepts. Thus, both approaches can be seen as complementary.

There are some applications using fuzzy ontologies in recommendation systems. Carlsson et al. developed a wine recommender system composed of a fuzzy ontology encoding real data about wines which can be used to select a wine given a specific context and reasoning using OWA operators [13]. Lee et al. apply fuzzy ontologies to personal diabetic-diet recommendations, with several experts taking part of the process and having to aggregate their different opinions [39]. These are two very nice applications of fuzzy ontologies solving a real-world problem. Our approach is more general, as we support more general AOs that could fit better to other applications, and consists of one integrated system supporting both fuzzy ontologies and AOs, not requiring to perform separately the two steps as in [13,39]. Hence, it could be probably help to extend these applications or to develop new ones from scratch.

There is a big amount of work in fuzzy DLs without AOs. For a good overview the reader is referred to [40]. The literature provides ideas to complement our work, as other fuzzy constructors could be considered to make our fuzzy DL more expressive. Conversely, existing algorithms to reason in fuzzy DLs based on a reduction to a MILP optimization problem, could be extended by considering our MILP representation of AOs proposed in the appendix.

Before concluding this section, we would like to mention some references where we have published part of this work. Bobillo and Straccia [8] began the study of fuzzy ontologies with some AOs and Bobillo and Straccia [9] considered some fuzzy integrals. The present paper is a revised and considerably extended version, including an integrated formulation of both papers, the study of more operators (weighted minimum, weighted maximum, approximated OWA), a new interpretation of fuzzy integral weights, an analysis of their complexities in the reasoning algorithm, a revised representation in Fuzzy OWL 2, a description of a Protégé plug-in implementing such representation, a discussion of the applications of our approach and a comparison with the related work.

Leaving aside the contributions of the present paper with respect to our previous works, there are no significant technical differences with [8] except for the new integrated formulation of the fuzzy DLs.

However, it is convenient to remark the new interpretation of fuzzy integral weights with respect to [9]. There, we assumed that the weights of fuzzy integrals verified $w_i = \mu(A_{(i)})$. Thus, every weight w_i measures the degree of satisfaction when the criteria $C_{\sigma(1)}, \ldots, C_{\sigma(i)}$ are satisfied simultaneously, where the permutation σ ensures that $C_{\sigma(i)}$ is the ith largest of the degrees of satisfaction of the fuzzy concepts $C_1, \ldots, C_n$. Note that in order to really have a fuzzy measure we should require that $w_1 \leq w_2 \leq \cdots \leq w_n = 1$. Hence, a Choquet integral concept $@^{\mathrm{ci}}_{[w_1,\ldots,w_n]}(C_1, \ldots, C_n)$ is equivalent to an OWA concept $@^{\mathrm{owa}}_{[w_1, w_2-w_1, \ldots, w_n-w_{n-1}]}(C_1, \ldots, C_n)$.

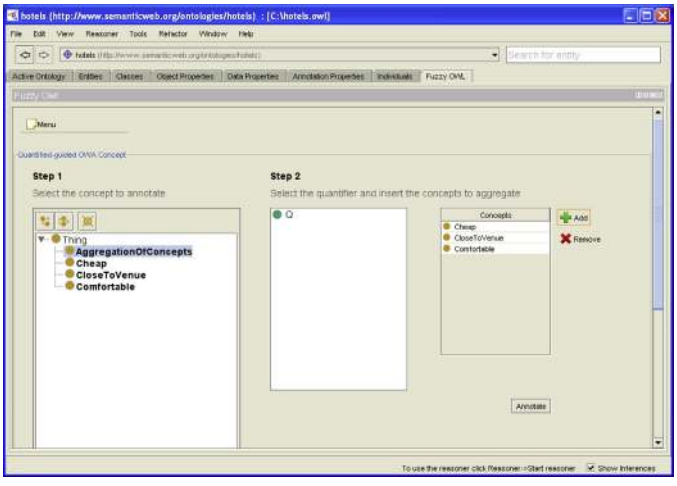

Fig. 6. Representation of quantifier-guided OWA concepts using the Protégé plug-in.

6. Conclusions and future work

In this work we have shown how to include aggregation operators and fuzzy integrals within fuzzy ontologies. This integration has several applications, such as user preferences representation, information fusion, decision making applications and recommendation systems.

In particular, we have defined the syntax and semantics for a fuzzy DL supporting such operators. We have focused in a minimal fuzzy DL, but the idea can be easily applied to more expressive logics.

We have provided a reasoning algorithm for a class of aggregations operators and fuzzy integrals that are MILP-representable. This makes it possible to integrate these operators in any fuzzy DL for which there is a reasoning procedure based on a reduction to a MILP optimization problem [7]. This algorithm is implemented in the fuzzyDL reasoner [6].

We have shown how to represent using MILP some common AOs, namely the weighted sum, weighted minimum, weighted maximum, OWA, quantifier-based OWA, approximated OWA, Sugeno integral and quasi-Sugeno integral. Choquet integral has been represented using MIQCP or, under some restrictions, MILP. We have also studied the complexity of these operators in terms of the size of the equivalent MILP problem.

We have also shown how to support these fuzzy integrals in Fuzzy OWL 2, a methodology to encode fuzzy version using the standard ontology language OWL 2. Our approach is implemented in a Protégé plug-in for fuzzy ontology development [10].

In our ongoing work, we are studying the linear approximation using Taylor's Theorem of the MIQCP restriction that we obtained

for the Choquet integral, in order to end up with a MILP problem in the more general case of a $\oplus$-decomposable measure.

In order to avoid the problems of the fuzzy integrals requiring too many parameters, we have restricted to $\oplus$-decomposable or additive fuzzy measures. In the future, we would like to study alternative restrictions already proposed in the literature [58].

Acknowledgements

We would like to thank to the anonymous referees for their valuable comments on an earlier version of this paper.

Appendix A. Reasoning in fuzzy ontologies with AOs

A.1. Optimization problems

In this section we recall Mixed Integer Linear Programming (MILP) optimization problems [45,49]. As we will see, our algorithm to reason with fuzzy ontologies will need to solve a MILP optimization problem.

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. The variables in y are called *control variables*. Let A, B be integer matrices, h an integer vector and $f(x, y)$ be an $k+m$-ary linear function. The general MILP problem is to find $\bar{x} \in \mathbb{Q}^k, \bar{y} \in \mathbb{Z}^m$ such that $f(\bar{x}, \bar{y}) = \min\{f(x, y) | Ax + By \geq h\}$.

The general case can be restricted to what concerns the paper as we can deal with *bounded* MILP. That is, the rational variables usually range over [0, 1], while the integer variables range over $\{0, 1\}$. It is well known that the bounded MILP problem is NP-COMPLETE (for the belonging to NP, guess the y and solve in polynomial time the linear system, NP-hardness follows from NP-hardness of 0-1 Integer Programming).

We say that a fuzzy operator $g : [0, 1]^n \otimes [0, 1]$ is *MILP representable* if the inequation $g(x_1, \ldots, x_n) \bowtie \alpha$ can be encoded using a set of MILP constraints, where $\bowtie \in \{<, \leq, >, \geq, =\}$ and $\alpha \in [0, 1]$. For instance, in Łukasiewicz fuzzy logic, $x_1 \otimes x_2 \geq \alpha$ can be encoded using the set of MILP constraints $\{\alpha \leq y,\ x_1 + x_2 - 1 \leq \alpha,\ x_1 + x_2 - y \geq \alpha, y \in \{0, 1\}\}$. Therefore, $\otimes$ is MILP-representable.

A.2. A reasoning algorithm for the BED problem

We next provide a reasoning algorithm to solve the BED problem in fuzzy DLs with AOs. Our algorithm is similar to others in the literature with the only difference being the management of AOs in Section A.3, but it is described here in detail to make the paper self-contained.

To simplify our exposition, we will make the following assumptions. We restrict the fuzzy DL to the case of Łukasiewicz fuzzy logic (and, thus, support standard fuzzy logic as well). Other fuzzy logics could be dealt with similarly as in [7], which considers t-norm based fuzzy DLs. Furthermore, we will restrict KBs to have an *unfoldable* TBox [4]. In short, unfoldable TBoxes have no cycles, only have atomic concepts in the left side of the GCIs, and no atomic concept appears more than once in the left side of a GCI. The reason of this restriction is that the combination of GCIs and fuzzy DLs is problematic (see Section 5 for details), but such problems do not appear here due to our, by purpose, simplified assumptions.

The basic idea of the algorithm follows. Let $\mathcal{K} = \langle \mathcal{A}, \mathcal{T}\rangle$, with $\mathcal{T}$ 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) [32]. 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 *minimize* the original variable x such that all constraints are satisfied.

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 [45]. In this paper, however, due to our restrictions above and those we will impose to AOs, we will end up with a *bounded Mixed Integer Linear Program* (MILP) optimization problem [45,49]. 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.

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, standard fuzzy logic, and Łukasiewicz fuzzy logic 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. [7,56]). Furthermore, as we will see in Section A.3, some typical AOs are also MILP-representable. Notice that the restriction of being MILP representable also affects the negation of weighted minimum, the t-norm of quasi-Sugeno integrals, and the t-conorm in fuzzy integrals with a $\oplus$-decomposable fuzzy measure.

Note that under Łukasiewicz fuzzy logic a fuzzy KB can be transformed into an equivalent *Negation Normal Form* (NNF) where the negation only appears before an atomic concept, a predicate and an aggregation concept. Given a fuzzy concept $\neg C$, we define the function nnf as follows: $nnf(\neg\top) = \bot$, $nnf(\neg\bot) = \top$, $nnf(\neg A) = \neg A$, $nnf(\neg\neg C) = C$, $nnf(\neg(C \sqcap D)) = C \sqcup D$, $nnf(\neg(C \sqcup D)) = C \sqcap D$, $nnf(\neg \exists R.C) = \forall R.\neg C$, $nnf(\neg \forall R.C) = \exists R.\neg C$, $nnf(\neg \exists R.\mathbf{d}) = \forall R.\neg \mathbf{d}$, $nnf(\neg \forall R.\ \mathbf{d}) = \exists R.\neg \mathbf{d}$, $nnf(\neg \mathbf{d}) = \neg \mathbf{d}$, $nnf(\neg @_W(C_1, \ldots, C_n)) = \neg @_W(C_1, \ldots, C_n)$.

Like most of the tableaux algorithms, our algorithm works on *completion-forests* since an ABox might contain several individuals with arbitrary roles connecting them. A completion-forest $\mathcal{F}$ for $\mathcal{K}$ is a collection of trees whose distinguished roots are arbitrarily connected by edges. The nodes can be abstract or concrete, depending if they represent abstract or concrete individuals.

- Each abstract node v is labeled with a set $\mathcal{L}(v)$ of concepts C. If $C \in \mathcal{L}(v)$ then we consider a variable $x_{v:C}$. The intuition is that v is an instance of C to degree equal or greater than the value of the variable $x_{v:C}$.
- Each concrete node c is labeled with a set $\mathcal{L}(c)$ of fuzzy predicates. For each concrete node c, we consider a variable x_c representing the value of the concrete individual c.
- Each edge $\langle v, w\rangle$ is labeled with a set $\mathcal{L}(\langle v, w\rangle)$ of roles R and if $R \in \mathcal{L}(\langle v, w\rangle)$ then we consider a variable $x_{\langle v,w\rangle:R}$ representing the degree of being $\langle v, w\rangle$ and instance of R.

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 linear expressions using the variables occurring in the forest.

Let us discuss now how to deal with *feature roles*. Let $\mathcal{F}$ be a forest, f a feature such that we have two edges $\langle v, w_1\rangle$ and $\langle v, w_1\rangle$ such that f occur in $\mathcal{L}(\langle v, w_1\rangle)$ and $\mathcal{L}(\langle v, w_2\rangle)$. Such a pair is called a *fork*. As f is a partial function, it 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

Table 4
Completion rules of the tableaux algorithm.

(var_1)	If $x \in \{x_{v:C}, x_{(v,w):R}, x_{(v,c):T}\}$ occurs in $\mathcal{C}_\mathcal{F}$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x \in [0,1]\}$
(var_2)	If $x \in \{x_{(v,w):f}, x_{(v,c):t}\}$ occurs in $\mathcal{C}_\mathcal{F}$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x \in \{0,1\}\}$
$(\bot)$	f $\bot \in \mathcal{L}(v)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:\bot} = 0\}$
$(\top)$	If $\top \in \mathcal{L}(v)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:\top} = 1\}$
$(\neg)$	If $\neg C \in \mathcal{L}(v)$ and this rule has not yet been applied to node v then *(i)* append $\mathsf{nnf}(\neg C)$ to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:\neg C} = x_{v:\mathsf{nnf}(\neg C)}\} \cup \{x_{v:\mathsf{nnf}(\neg C)} = 1 - x_{v:C}\}$
$(\sqcap)$	If $C \sqcap D \in \mathcal{L}(v)$ then *(i)* append C, D to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:C} \otimes x_{v:D} \geq x_{v:C \sqcap D}\}$
$(\sqcup)$	If $C \sqcup D \in \mathcal{L}(v)$ then *(i)* append C, D to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:C} \oplus x_{v:D} \geq x_{v:C \sqcup D}\}$
$(\exists)$	If $\exists R.C \in \mathcal{L}(v)$ then *(i)* create a new node w, and *(ii)* append R to $\mathcal{L}(\langle v, w\rangle)$, and *(iii)* append C to $\mathcal{L}(w)$, and *(iv)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{(v,w):R} \otimes x_{w:C} \geq x_{v:\exists R.C}\}$
$(\forall)$	If $\forall R.C \in \mathcal{L}(v)$ and $R \in \mathcal{L}(\langle v, w\rangle$ then *(i)* append C to $\mathcal{L}(w)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{w:C} \geq x_{v:\forall R.C} \otimes x_{(v,w):R}\}$
$(\exists_\mathbf{D})$	If $\exists T.\mathbf{d} \in \mathcal{L}(v)$, the case is similar as in rule $(\exists)$ but creating a concrete node c
$(\forall_\mathbf{D})$	If $\forall T.\mathbf{d} \in \mathcal{L}(v)$, the case is similar as in rule $(\forall)$ but considering a concrete node c
$(\mathbf{d})$	If $\mathbf{d} \in \mathcal{L}(c)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \gamma(\mathbf{d}(x_c) \geq x_{c:d})$, where the set $\gamma(\mathbf{d}(x_c) \geq x_{c:d})$ is obtained from the MILP-representation of $\mathbf{d}$
$(\neg\mathbf{d})$	If $\neg\mathbf{d} \in \mathcal{L}(v)$ then $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \gamma(\neg\mathbf{d}(x_c) \geq x_{c:d})$, where the set $\gamma(\neg\mathbf{d}(x_c) \geq x_{c:d})$ is obtained from the MILP-representation of $\neg\mathbf{d}$
$(@)$	If $@_W(C_1, \ldots, C_n)\mathcal{L}(v)$ then *(i)* append all C_i to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{@_W(x_{v:C_1}, \ldots, x_{v:C_n}) \geq x_{@_W(C_1,\ldots,C_n)}\}$
$(\neg @)$	If $\neg@_W(C_1, \ldots, C_n)\mathcal{L}(v)$ then *(i)* append all $\neg C_i$ to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{@_W(x_{v:C_1}, \ldots, x_{v:C_n}) \geq 1 - x_{@_W(C_1,\ldots,C_n)}\}$
$(\sqsubseteq)$	If $\langle A \sqsubseteq C, \alpha\rangle \in \mathcal{T}, A \in \mathcal{L}(v)$ and this rule has not yet been applied to node v then *(i)* append $\neg A \sqcup C$ to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:\neg A \sqcup C} \geq \alpha\}$
$(\equiv)$	If $\langle A \equiv C, \in\rangle\mathcal{T}, A \in \mathcal{L}(v)$ and this rule has not yet been applied to node v then *(i)* append C to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:A} = x_{v:C}\}$
$(\neg\equiv)$	If $\langle A \equiv C, \in\rangle\mathcal{T}, \neg A \in \mathcal{L}(v)$ and this rule has not yet been applied to node v then *(i)* append $\neg C$ to $\mathcal{L}(v)$, and *(ii)* $\mathcal{C}_\mathcal{F} = \mathcal{C}_\mathcal{F} \cup \{x_{v:\neg A} = x_{v:\neg C}\}$

appear (as part of a rule application) with the proviso that newly generated nodes are replaced by older ones and not vice-versa.

The algorithm initializes a forest $\mathcal{F}$ to contain:

- A root node v^i, for each individual a_i occurring in $\mathcal{A}$.
- An edge $\langle v^i, v^j\rangle$, for each fuzzy assertion $\langle(a_i, a_j) : R_i, \alpha\rangle \in \mathcal{A}$.
- For each $\langle a_i : C, \alpha\rangle \in \mathcal{A}$, we add both C to $\mathcal{L}(v^i)$ and $x_{v^i:C} \geq \alpha$ to $\mathcal{C}_\mathcal{F}$.
- For each $\langle(a_i, a_j) : R, \alpha\rangle \in \mathcal{A}$, we add both R to $\mathcal{L}(\langle v^i, v^j\rangle)$ and $x_{(v^i,v^j):R} \geq \alpha$ to $\mathcal{C}_\mathcal{F}$.

$\mathcal{F}$ is then expanded by repeatedly applying the completion rules in Table 4, where $\mathbf{d} \in \{ls, rs, tri, trz\}$ and $@_W$ is a MILP-representable AO. The completion-forest is *complete* when none of the completion rules are applicable. We assume that each rule is only applied once for the same concept and the same node. Then, the MILP problem on the set of constraints $\mathcal{C}_\mathcal{F}$ is solved. If the MILP problem does not have a solution, the KB is inconsistent and it has no models.

Example A.1. Consider the fuzzy KB $\{c : \forall \mathsf{hasHotel}.\mathsf{Cheap} \geq 0.7, (c, h) : \mathsf{hasHotel} \geq 0.6, h : \neg\mathsf{Cheap} \geq 0.8\}$, where c and h are individuals representing a hotel chain and a hotel, respectively. Fig. 7 shows the completion graph after the initialization and the application of the rules $(\neg)$ and $(\forall)$. It can be seen that the KB has no solution because we both have $x_{h:\mathsf{Cheap}} \geq x_{c:\forall\mathsf{hasHotel}.\mathsf{Cheap}} \otimes x_{(c,h):\mathsf{hasHotel}} \geq 0.7 \otimes 0.6 = 0.3$ and $x_{h:\mathsf{Cheap}} = 1 - x_{h:\neg\mathsf{Cheap}} \leq 1 - 0.8 = 0.2$. Hence, the KB is inconsistent.

The following proposition can be shown.

Proposition A.1 (Termination, Soundness, Completeness). *Let $\mathcal{K} = \langle\mathcal{A}, \mathcal{T}\rangle$ with $\mathcal{T}$ unfoldable. Then, the following properties are satisfied:*

- *The tableau algorithm terminates.*
- *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.*
- *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.*

The proof can be obtained as follows. On the one hand, this result has been shown for fuzzy $\mathcal{ALCF}(\mathbf{D})$ [7,16]. On the other hand, it can be extended to support AOs if we provide a correct MILP encoding of the restrictions of the form $@_W(x_1, \ldots, x_n) \geq x$. This will be the objective of the next subsection.

A.3. MILP encoding of aggregation operators

In the rest of this section we will show how to encode using MILP equations the restrictions that involve some AOs. Since the complexity of a MILP optimization problem is proportional to its size, we will also discuss how many new variables are introduced by each operator. As we will see, it is more efficient to consider weighted sums and OWA as special cases of a Choquet integral.

A.3.1. MILP encoding of weighted sums

The case of weighted sums is very easy. In order to show that $@^{\mathsf{ws}}_W(x_1, \ldots, x_n) \geq x$ is MILP representable, we just have to introduce the following equation:

$$\sum_{i=1}^{n} w_i x_i \geq x \tag{11}$$

Note that Eq. (11) does not introduce any new variable.

Proposition A.2 (Weighted sum). *Eq. (11) is a MILP encoding of $@^{\mathsf{ws}}_W(x_1, \ldots, x_n) \geq x$.*

A.3.2. MILP encoding of weighted maxima

In this case we firstly define n new [0,1] variables $\min_i$ and, for each $i \in N$, we state that $\min_i = \min\{w_i, x_i\}$ using the following constraints:

$$\begin{aligned}
&\min_i \leq w_i, \forall i \in N\\
&\min_i \leq x_i, \forall i \in N\\
&w_i \leq \min_i + y_{1i}, \forall i \in N\\
&x_i \leq \min_i + (1 - y_{1i}), \forall i \in N\\
&\min_i \in [0,1], \forall i \in N\\
&y_{1i} \in \{0,1\}, \forall i \in N
\end{aligned} \tag{12}$$

where y_{1i} are new binary variables.

$$\begin{array}{l} \mathsf{c} \quad \mathcal{L}(\mathsf{c}) = \{\forall \mathsf{hasHotel.Cheap}\} \\ \quad \downarrow \quad \mathcal{L}(\langle \mathsf{c}, \mathsf{h}\rangle) = \{\mathsf{hasHotel}\} \\ \mathsf{h} \quad \mathcal{L}(\mathsf{h}) = \{\neg \mathsf{Cheap}, \mathsf{Cheap}\} \end{array}$$

$$\mathcal{C}_{\mathcal{F}} = \{x_{\mathsf{c}:\forall \mathsf{hasHotel.Cheap}} \in [0,1], x_{(\mathsf{c},\mathsf{h}):\mathsf{hasHotel}} \in [0,1], x_{\mathsf{h}:\neg \mathsf{Cheap}} \in [0,1], x_{\mathsf{h}:\mathsf{Cheap}} \in [0,1],$$
$$x_{\mathsf{c}:\forall \mathsf{hasHotel.Cheap}} \geq 0.7, x_{(\mathsf{c},\mathsf{h}):\mathsf{hasHotel}} \geq 0.6, x_{\mathsf{h}:\neg \mathsf{Cheap}} \geq 0.8,$$
$$x_{\mathsf{h}:\neg \mathsf{Cheap}} = 1 - x_{\mathsf{h}:\mathsf{Cheap}}, x_{\mathsf{h}:\mathsf{Cheap}} \geq x_{\mathsf{c}:\forall \mathsf{hasHotel.Cheap}} \otimes x_{(\mathsf{c},\mathsf{h}):\mathsf{hasHotel}} \}$$

Fig. 7. Completion graph for the fuzzy KB in Example A.1.

Next, we encode that the maximum of the variables $\min_i$ is greater or equal than x. This can be done as follows:

$$\begin{array}{l} \min_i + y_{2i} \geq x, \forall i \in N \\ \sum_{i=1}^{n} y_{2i} = n - 1 \\ y_{2i} \in \{0,1\}, \forall i \in N \end{array} \quad (13)$$

where y_{2i} are new binary variables.

Proposition A.3 (Weighted maximum). *Eqs. (12) and (13) are a MILP encoding of* $@_W^{\mathrm{wmax}}(x_1, \ldots, x_n) \geq x$.

The weighted maximum creates $3n$ new variables: $2n$ binary variables y_{1i}, y_{2i} and n variables $\min_i$.

A.3.3. MILP encoding of weighted minima

Now, we start by defining n new [0, 1] variables neg_i to store the negation of the weight w_i. Assuming that the negation is MILP representable, this can be done as follows:

$$\begin{array}{l} neg_i = \ominus w_i, \forall i \in N \\ neg_i \in [0,1], \forall i \in N \end{array} \quad (14)$$

Next, we define n new [0,1] variables $\max_i$ and, for each $i \in N$, we state that $\max_i = \max\{neg_i, x_i\}$ as follows:

$$\begin{array}{l} \max_i \geq neg_i, \forall i \in N \\ \max_i \geq x_i, \forall i \in N \\ neg_i + y_{1i} \geq \max_i, \forall i \in N \\ x_i + (1 - y_{1i}) \geq \max_i, \forall i \in N \\ \max_i \in [0,1], \forall i \in N \\ y_{1i} \in \{0,1\}, \forall i \in N \end{array} \quad (15)$$

where y_{1i} are new binary variables.

In the particular case of Łukasiewicz negation, we do not need the variables neg_i, and we can replace Eqs. (14) and (15) with this one:

$$\begin{array}{l} \max_i \geq 1 - w_i, \forall i \in N \\ \max_i \geq x_i, \forall i \in N \\ 1 - w_i + y_{1i} \geq \max_i, \forall i \in N \\ x_i + (1 - y_{1i}) \geq \max_i, \forall i \in N \\ \max_i \in [0,1], \forall i \in N \\ y_{1i} \in \{0,1\}, \forall i \in N \end{array} \quad (16)$$

where y_{1i} are new binary variables.

Finally, we force the minimum of the $\max_i$ to be greater or equal than x, which is simply done as:

$$\max_i \geq x, \forall i \in N \quad (17)$$

Proposition A.4 (Weighted minimum).

- *Eqs. (14), (15) and (17) are a MILP encoding of* $@_W^{\mathrm{wmin}}(x_1, \ldots, x_n) \geq x$.
- *In the case of Łukasiewicz negation, Eqs. (16) and (17) are a MILP encoding of* $@_W^{\mathrm{wmin}}(x_1, \ldots, x_n) \geq x$.

In the general case, the weighted minimum creates at least $3n$ new variables: $n\ neg_i$, $n \max_i$ and $n\ y_i$. We may also need some more variables to compute the MILP-representation of the negation function. In the particular case of Łukasiewicz negation, the weighted minimum creates $2n$ new variables: $n \max_i$ and $n\ y_i$.

A.3.4. MILP encoding of OWA operators

R. Yager showed that the *maximization* of $@_W^{\mathrm{owa}}(x_1, \ldots, x_n)$ can indeed be encoded as a MILP problem [66]. However, this encoding does not work in our setting as we are not maximizing $@_W^{\mathrm{owa}}(x_1, \ldots, x_n)$, but rather need to show that $@_W^{\mathrm{owa}}(x_1, \ldots, x_n) \geq x$ is MILP-representable.

Similarly to [66], we introduce new [0, 1]-valued variables y_i ($i \in N$) and impose:

$$\begin{array}{l} y_1 \geq y_2 \geq \cdots \geq y_n \\ y_i \in [0,1], \forall i \in N \end{array} \quad (18)$$

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

$$\sum_{i=1}^{n} w_i y_i \geq x \quad (19)$$

The remaining of the encoding consists of establishing a bijection between the sets of variables $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$. This can be done by introducing the following restrictions:

$$\begin{array}{l} y_i \leq x_j + 2z_{ij}, \forall i, j \in N \\ x_j \leq y_i + 2z_{ij}, \forall i, j \in N \\ \sum_{j=1}^{n} z_{ij} = n - 1, \forall i \in N \\ \sum_{i=1}^{n} z_{ij} = n - 1, \forall j \in N \\ z_{ij} \in \{0,1\}, \forall i, j \in N \end{array} \quad (20)$$

where z_{ij} are new binary variables.

It can be verified that the following conditions are indeed verified:

- If $z_{ij}=0$ then $y_i=x_j$ is imposed.
- For any y_i, there is only one x_j imposed to be equal to y_i.
- For any x_j, there is only one y_i imposed to be equal to x_j.

Proposition A.5 (OWA). *Eqs. (18)–(20) are a MILP encoding of* $@_W^{\mathrm{owa}}(x_1,\ldots,x_n)\geq x$.

Notice that our encoding of an OWA operators introduce n^2 new variables z_{ij} and n variables y_i, which makes a total of $n(n+1)$ new variables.

Note also that this encoding can be used in quantifier-guided OWA AOs. To this end, we just have to compute firstly the weights by using Eq. (3).

A.3.5. MILP encoding of approximated OWA operators

Now, we will show how to compute $@_W^{\mathrm{aowa}}(x_1,\ldots,x_n)\geq x\}$, which is the approximation of an OWA operator described in Eq. (2).

Firstly, note that the set $\{\{i,j\}\subseteq N\}$, used in Eq. (2), requires to consider the pairs of elements such that $1\leq i<j\leq n$, because it only makes sense to consider the case $i\neq j$ and, for symmetry, the pair $\{i,j\}$ excludes considering $\{j,i\}$. This means $\binom{n}{2}=n(n-1)/2$ pairs.

For $1\leq i<j\leq n$, we can introduce a new variable $\min_{ij}$ that will take the value of the minimum between x_i and x_j. This can be done by introducing some new [0,1] variables $\min_{ij}$ and proceedings similarly as in Eq. (12):

$$\begin{aligned}
&\min_{ij}\leq x_i,\ \forall 1\leq i<j\leq n\\
&\min_{ij}\leq x_j,\ \forall 1\leq i<j\leq n\\
&x_i\leq \min_{ij}+y_{ij},\ \forall 1\leq i<j\leq n\\
&x_j\leq \min_{ij}+(1-y_{ij}),\ \forall 1\leq i<j\leq n\\
&y_{ij}\in\{0,1\},\ \forall 1\leq i<j\leq n
\end{aligned}\tag{21}$$

where y_{ij} are new binary variables.

Next, we can add the following MILP restriction:

$$\left(\frac{1}{n}-\frac{w_n-w_1}{2}\right)\sum_{i=1}^{n}x_i+\frac{w_n-w_1}{n-1}\sum_{1\leq i<j\leq n}\min_{ij}\geq x\tag{22}$$

Proposition A.6 (Approximated OWA). *Eqs. (21) and (22) are a MILP encoding of* $@_W^{\mathrm{owa}}(x_1,\ldots,x_n)\geq x$ *if OWA is approximated as in Definition hyperlinkeq:approxOWArefeq:approxOWA.*

Our encoding introduces $n(n-1)$ new variables: $n(n-1)/2$ $\min_{ij}$ and $n(n-1)/2$ y_{ij}. Hence, approximated OWA operators require the same number of variables than OWA operators.

A.3.6. MILP encoding of quasi-Sugeno and Sugeno integrals

Now we will consider the case of the quasi-Sugeno integral which extends the case of the Sugeno integral.

As in OWA operators, we introduce new variables y_i to order the aggregated values in descending order. This is done by using Eq. (18).

Next, we establish the relation between the x_i variables and the y_j ($i,j\in N$) using Eq. (20).

Now, let us focus on the calculus of μ from the parameters $w_1,\ldots,w_n$ is the fuzzy integral. We can provide the inductive definition

$$\begin{aligned}
&\mu(A_{\sigma(1)})=\mu(x_{\sigma(1)})=w_{\sigma(1)}\\
&\mu(A_{\sigma(i)})=\mu(A_{\sigma(i-1)})\oplus\mu(x_{\sigma(i)})=\mu(A_{\sigma(i-1)})\oplus w_{\sigma(i)}
\end{aligned}\tag{23}$$

since the fuzzy measure is $\oplus$-decomposable.

This implies that we must order the weights. To this end, we will create n new [0,1]-variables ow_i (ow stands for ordered weight) and establish a bijection with the parameters w_i such that $ow_j=w_{\sigma(j)}$. This can be done by introducing the following constraints:

$$\begin{aligned}
&ow_j\geq(1-z_{ij})w_i,\ \forall i,j\in N\\
&ow_j\leq z_{ij}+w_i,\ \forall i,j\in N\\
&ow_j\in[0,1],\ \forall j\in N
\end{aligned}\tag{24}$$

where z_{ij} are the variables introduced in Eq. (20). As already discussed, $z_{ij}=0$ represents that x_j is the ith largest variable. Now, $z_{ij}=0$ makes $ow_j=p_i$, so ow_j contains the weight of the ith largest x_i.

The next step is to define n new variables a_i to store the values $\mu(A_{\sigma(i)})$, which is done as follows:

$$\begin{aligned}
&a_1=ow_1\\
&a_i=ow_i\oplus a_{i-1},\ \forall i\in\{2,\ldots,n\}\\
&a_i\in[0,1],\ \forall i\in N
\end{aligned}\tag{25}$$

which is possible since the t-conorm $\oplus$ is required to be MILP-representable.

Next, we introduce n new [0,1]-valued variables c_i that will take the value of the expression $y_i\otimes a_i$. This is done as follows:

$$\begin{aligned}
&c_i=y_i\otimes a_i,\ \forall i\in N\\
&c_i\in[0,1],\ \forall i\in N
\end{aligned}\tag{26}$$

Note that this is possible because the t-norm is required to be MILP-representable. In the case of the Sugeno integral, the t-norm $\otimes$ will be fixed to the minimum, which can be encoded similarly as in Eq. (12).

Finally, we have to ensure that the maximum of the c_i is greater or equal than x. To this end, we introduce n new binary variables b_i($i\in N$) and add the following constraints:

$$\begin{aligned}
&c_i+b_i\geq x,\ \forall i\in N\\
&\sum_{i=1}^{n}b_i=n-1\\
&b_i\in\{0,1\},\ \forall i\in N
\end{aligned}\tag{27}$$

Proposition A.7 (Quasi-Sugeno integral). *Eqs. (18), (20), (26) and (27) are a MILP encoding of* $@_W^{\mathrm{qsi}}(x_1,\ldots,x_n)\geq x$.

The number of new variables is at least $n(n+5)$: n^2 z_{ij}, n y_i, n ow_i, n a_i,n c_i and n b_i. Furthermore, we may also need some variables to encode Eq. (26), which depend on the particular t-norm. In the case of the Sugeno integral, we need $n(n+6)$ variables, as the minimum can be managed as in Eq. (12).

A.3.7. Encoding of Choquet integrals

Let us discuss now the case of the more complicated case of Choquet integral. In this case, we are able to provide an encoding using a Mixed Integer Quadratically Constrained Programming (MIQCP) problem [45]. Firstly, Eqs. (18), (20), (24) and (25) are needed as well.

The remaining of the encoding concerns to the definition of the Choquet integral itself:

$$y_1\cdot a_1+\sum_{i=2}^{n}(a_i-a_{i-1})y_i\geq x\tag{28}$$

Note that in fact Eq. (28) cannot be represented in a MILP problem because it involves the product of two variables a_i and y_i.

Proposition A.8 (Choquet integral). *Eqs. (18), (20), (24), (25) and (28) are a MIQCP encoding of* $@_W^{\mathrm{ci}}(x_1,\ldots,x_n)\geq x$.

The number of new variables is $n(n+3)$: n^2 z_{ij}, n y_i, n ow_i and n a_i.

Solving MIQCP problems is more difficult than solving MILP problems for theoretical and practical reasons, as the availability of efficient algorithms and tools is smaller. In order to have a MILP representation we can impose several restrictions.

- If the fuzzy measure is additive, then we have an weighted sum operator and hence can use Proposition A.2.
- If the weights of the Choquet integral are interpreted as $w_i = \mu(A_{\sigma(i)})$, then we can have an OWA operator by considering Eqs. (18), (20) and

$$y_1 \cdot w_1 + \sum_{i=2}^{n} (y_i - y_{i-1}) w_i \geq x \tag{29}$$

A.4. Final remarks

For example, our reasoning algorithm could be extended to work with other fuzzy logics by taking profit of the algorithms to reason with arbitrary continuous t-norms. Among them, [7] provides a reasoning algorithm for the fuzzy DL $\mathcal{ALC}$ under arbitrary continuous t-norms extended with Łukasiewicz negation. Our reasoning algorithm in Section A.2 assumes Łukasiewicz fuzzy logic, but other fuzzy operators could be considered by reusing the ideas behind the algorithm for arbitrary t-norms.

Our reasoning algorithm also assumes that the TBox is unfoldable. GCIs are known to work properly for standard fuzzy logic [55] and finite fuzzy DLs [5,14,15] but in Łukasiewicz and Product fuzzy DLs they produce undecidability [4,11,16,3]. In our approach, it is yet unknown if we can assume a less restrictive condition than unfoldable TBoxes, but arbitrary GCIs cannot be supported for sure. This is an advantage of [5,14,15,55], although they cannot support AOs.

Finally, note also that [9] uses a different but equivalent definition of fuzzy integrals with the aggregated values in ascending order. In the current paper, we order them in decreasing order for coherence with the case of aggregation operators. It is worth to note that this is the reason why the MILP encodings in [9] replace Eq. (18) with:

$$\begin{aligned} &y_1 \leq y_2 \leq \ldots \leq y_n \\ &y_i \in [0,1], \forall i \in N \end{aligned} \tag{30}$$

References

[1] T. Andreasen, H. Bulskov, Conceptual querying through ontologies, Fuzzy Sets and Systems 160 (2009) 2159–2172.
[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P.F. Patel-Schneider, The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, New York, USA, 2003.
[3] F. Baader, R. Peñaloza, On the undecidability of fuzzy description logics with GCIs and product t-norm, in: Proceedings of the 8th International Symposium on Frontiers of Combining Systems (FroCoS 2011), Springer, 2011, pp. 55–70.
[4] F. Bobillo, F. Bou, U. Straccia, On the failure of the finite model property in some fuzzy description logics, Fuzzy Sets and Systems 172 (2011) 1–12.
[5] F. Bobillo, M. Delgado, J. Gómez-Romero, U. Straccia, Joining Gödel and Zadeh fuzzy logics in fuzzy description logics, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 20 (2012) 475–508.
[6] F. Bobillo, U. Straccia, fuzzyDL: an expressive fuzzy description logic reasoner, in: Proceedings of the 17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008), IEEE Computer Society, 2008, pp. 923–930.
[7] F. Bobillo, U. Straccia, Fuzzy description logics with general t-norms and datatypes, Fuzzy Sets and Systems 160 (2009) 3382–3402.
[8] F. Bobillo, U. Straccia, Aggregation operators and Fuzzy OWL 2, in: Proceedings of the 20th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011), IEEE Press, 2011, pp. 1727–1734.
[9] F. Bobillo, U. Straccia, Fuzzy ontologies and fuzzy integrals, in: Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), 2011, pp. 1311–1316.
[10] F. Bobillo, U. Straccia, Fuzzy ontology representation using OWL 2, International Journal of Approximate Reasoning 52 (2011) 1073–1094.
[11] S. Borgwardt, R. Peñaloza, Undecidability of fuzzy description logics, in: Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012), AAAI Press, 2012, pp. 232–242.
[12] S. Calegari, E. Sanchez, Object-fuzzy concept network: an enrichment of ontologies in semantic information retrieval, Journal of the American Society for Information Science and Technology 59 (2008) 2171–2185.
[13] C. Carlsson, M. Brunelli, J. Mezei, Decision making with a fuzzy ontology, Soft Computing 16 (2012) 1143–1152.
[14] M. Cerami, F. Esteva, À. García-Cerdaña, On finitely valued fuzzy description logics: the Łukasiewicz case, in: Proceedings of the 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2012), Springer, 2012, pp. 235–244.
[15] M. Cerami, À. García-Cerdaña, F. Esteva, From classical description logic to n-graded fuzzy description logic, in: Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010), IEEE, 2010, pp. 1506–1513.
[16] M. Cerami, U. Straccia, On the undecidability of fuzzy description logics with GCIs with Łukasiewicz t-norm, Information Sciences 227 (2013) 1–21.
[17] R.C. Chen, C.T. Bau, C.J. Yeh, Merging domain ontologies based on the WordNet system and fuzzy formal concept analysis techniques, Applied Soft Computing 11 (2011) 1908–1923.
[18] G. Choquet, Theory of capacities, Annales de l'Institut Fourier 5 (1953) 131–295.
[19] P.C.G. Costa, K.B. Laskey, T. Lukasiewicz, Uncertainty representation and reasoning in the semantic web, in: J. Cardoso, M.D. Lytras (Eds.), Semantic Web Engineering in the Knowledge Society, IGI Global, 2008, pp. 315–340.
[20] B. Cuenca-Grau, I. Horrocks, B. Motik, B. Parsia, P.F. Patel-Schneider, U. Sattler, OWL 2: the next step for OWL, Journal of Web Semantics 6 (2008) 309–322.
[21] S. Dasiopoulou, I. Kompatsiaris, M.G. Strintzis, Applying fuzzy DLs in the extraction of image semantics, Journal of Data Semantics XIV (2009) 105–132.
[22] S. Dasiopoulou, I. Kompatsiaris, M.G. Strintzis, Investigating fuzzy DLs-based reasoning in semantic image analysis, Multimedia Tools and Applications 46 (2010) 331–370.
[23] C. De Maio, G. Fenza, M. Gaeta, V. Loia, F. Orciuoli, S. Senatore, RSS-based e-learning recommendations exploiting fuzzy FCA for knowledge modeling, Applied Soft Computing 12 (2012) 113–124.
[24] C. De Maio, G. Fenza, V. Loia, S. Senatore, Hierarchical web resources retrieval by exploiting fuzzy formal concept analysis, Information Processing and Management 48 (2012) 399–418.
[25] R. Fullér, On obtaining OWA operator weights: a sort survey of recent developments, in: Proceedings of the 5th IEEE International Conference on Computational Cybernetics (ICCC 2007), 2007, pp. 241–244.
[26] M. Grabisch, Alternative representations of OWA operators, in: The Ordered Weighted Averaging Operators: Theory and Applications, Kluwer, Dordrecht, Netherlands, 1997, pp. 73–85.
[27] M. Grabisch, C. Labreuche, Fuzzy measures and integrals in MCDA, in: J. Figueira, S. Greco, M. Ehrgott (Eds.), Multiple Criteria Decision Analysis: State of the Art Surveys, Springer Verlag, Heidelberg, Germany, 2005, pp. 563–604.
[28] M. Grabisch, J.L. Marichal, R. Mesiar, E. Pap, Aggregation functions: construction methods, conjunctive, disjunctive and mixed classes, Information Sciences 181 (2011) 23–43.
[29] M. Grabisch, J.L. Marichal, R. Mesiar, E. Pap, Aggregation functions: means, Information Sciences 181 (2011) 1–2.
[30] S. Greco, F. Rindone, Bipolar fuzzy integrals, Fuzzy Sets and Systems (2013), http://dx.doi.org/10.1016/j.fss.2012.11.021.
[31] S. Greco, F. Rindone, Robust integrals, Fuzzy Sets and Systems (2013), http://dx.doi.org/10.1016/j.fss.2013.01.008.
[32] R. Hähnle, Advanced many-valued logics, in: D.M. Gabbay, F. Guenthner (Eds.), in: Handbook of Philosophical Logic, vol. 2, 2nd ed., Kluwer, Dordrecht, Netherlands, 2001, pp. 297–395.
[33] P. Hájek, Metamathematics of Fuzzy Logic, Kluwer, Dordrecht, Netherlands, 1998.
[34] C. Hudelot, J. Atif, I. Bloch, Fuzzy spatial relation ontology for image interpretation, Fuzzy Sets and Systems 159 (2008) 1929–1951.
[35] Jeroen Janssen, D.V. Steven Schockaert, M.D. Cock, Aggregated fuzzy answer set programming, Annals of Mathematics and Artificial Intelligence 63 (2012) 103–147.
[36] E.P. Klement, R. Mesiar, E. Pap, A universal integral as common frame for Choquet and Sugeno integral, IEEE Transactions on Fuzzy Systems 18 (2010) 178–187.
[37] G.J. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice-Hall, Upper Saddle River, USA, 1995.
[38] C.S. Lee, Z.W. Jian, L.K. Huang, A fuzzy ontology and its application to news summarization, IEEE Transactions on Systems, Man and Cybernetics, Part B 35 (2005) 859–880.
[39] C.S. Lee, M.H. Wang, H. Hagras, A type-2 fuzzy ontology and its application to personal diabetic-diet recommendation, IEEE Transactions on Fuzzy Systems 18 (2010) 374–395.
[40] T. Lukasiewicz, U. Straccia, Managing uncertainty and vagueness in description logics for the semantic web, Journal of Web Semantics 6 (2008) 291–308.
[41] J.L. Marichal, Aggregation Operators for Multicriteria Decision Aids, Universite de Liège, Belgium, 1999 (Ph.D. thesis).
[42] C. Martínez-Cruz, A. van der Heide, D. Sánchez, G. Triviño, An approximation to the computational theory of perceptions using ontologies, Expert Systems with Applications 39 (2012) 9494–9503.

[43] P.S. Mostert, A.L. Shields, On the structure of semigroups on a compact manifold with boundary, Annals of Mathematics 65 (1957) 117–143.
[44] B. Motik, B. Parsia, P.F. Patel-Schneider (Eds.), OWL 2 Web Ontology Language XML Serialization, 2009, Available at: http://www.w3.org/TR/owl2-xml-serialization (Online).
[45] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley-Interscience, New York, USA, 1999.
[46] T.T. Quan, S.C. Hui, A.C.M. Fong, Automatic fuzzy ontology generation for semantic help-desk support, IEEE Transactions on Industrial Informatics 2 (2006) 155–164.
[47] T.T. Quan, S.C. Hui, A.C.M. Fong, T.H. Cao, Automatic fuzzy ontology generation for semantic web, IEEE Transactions on Knowledge and Data Engineering 18 (2006) 842–856.
[48] J.A. Rodger, A fuzzy linguistic ontology payoff method for aerospace real options valuation, Expert Systems with Applications 40 (2013) 2828–2840.
[49] H.M. Salkin, K. Mathur, Foundations of Integer Programming, North-Holland, Amsterdam, Netherlands, 1989.
[50] D. Sánchez, A.G.B. Tettamanzi, Fuzzy quantification in fuzzy description logics, in: E. Sanchez (Ed.), Fuzzy Logic and the Semantic Web, Capturing Intelligence, vol. 1, Elsevier Science, Heidelberg, Germany, 2006, pp. 135–159.
[51] E. Sanchez (Ed.), Fuzzy Logic and the Semantic Web, Capturing Intelligence, vol. 1, Elsevier Science, Heidelberg, Germany, 2006.
[52] R. Senge, E. Hüllermeier, Top-down induction of fuzzy pattern trees, IEEE Transactions on Fuzzy Systems 19 (2011) 241–252.
[53] N. Simou, T. Athanasiadis, G. Stoilos, S. Kollias, Image indexing and retrieval using expressive fuzzy description logics, Signal, Image and Video Processing 2 (2008) 321–335.
[54] S. Staab, R. Studer (Eds.), Handbook on Ontologies, Springer-Verlag, Heidelberg, Germany, 2009.
[55] G. Stoilos, U. Straccia, G. Stamou, J.Z. Pan, General concept inclusions in fuzzy description logics, in: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006), 2006, pp. 457–461.
[56] U. Straccia, Description logics with fuzzy concrete domains, in: F. Bachus, T. Jaakkola (Eds.), Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005), AUAI Press, 2005, pp. 559–567.
[57] M. Sugeno, Theory of fuzzy integrals and its applications, Tokyo Institute of Technology, Japan, 1974 (Ph.D. thesis).
[58] V. Torra, Y. Narukawa, Modeling Decisions – Information Fusion and Aggregation Operators, Springer, Heidelberg, Germany, 2007.
[59] P. Tzouveli, N. Simou, G. Stamou, S. Kollias, Semantic classification of Byzantine icons, IEEE Intelligent Systems 24 (2009) 35–43.
[60] V. Vaneková, P. Vojtás, A description logic with concept instance ordering and top-k restriction, in: Proceedings of the 18th European–Japanese Conference on Information Modelling and Knowledge Bases (EJC 2008), IOS Press, 2008, pp. 139–153.
[61] 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.
[62] M. Wallace, Ontologies and soft computing in flexible querying, Control and Cybernetics 38 (2009) 481–507.
[63] S. Weber, $\perp$-decomposable measures and integral for Archimedean t-conorms $\perp$, Journal of Mathematical Analysis and Applications 101 (1984) 114–138.
[64] R.R. Yager, On ordered weighted averaging aggregation operators in multicriteria decision making, EEE Transactions on Systems, Man and Cybernetics 18 (1988) 183–190.
[65] R.R. Yager, Connectives and quantifiers in fuzzy sets, Fuzzy Sets and Systems 40 (1991) 39–75.
[66] R.R. Yager, Constrained OWA aggregation, Fuzzy Sets and Systems 81 (1996) 10–89.
[67] R.R. Yager, Quantifier guided aggregation using OWA operators, International Journal of Intelligent Systems 11 (1996) 49–73.
[68] R.R. Yager, J. Kacprzyk (Eds.), The Ordered Weighted Averaging Operators: Theory and Applications, Kluwer, Dordrecht, Netherlands, 1997.
[69] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.
[70] L.A. Zadeh, A computational approach to fuzzy quantifiers in natural languages, Computers and Mathematics with Applications 9 (1983) 149–184.