The Fuzzy Ontology Reasoner *fuzzyDL*

Fernando Bobillo
Department of Computer Science and Systems Engineering
University of Zaragoza, Spain

Umberto Straccia
ISTI-CNR
Pisa, Italy

Abstract

Classical, two-valued, ontologies have been successfully applied to represent the knowledge in many domains. However, it has been pointed out that they are not suitable in domains where vague or imprecise pieces of information play an important role. To overcome this limitation, several extensions to classical ontologies based on fuzzy logic have been proposed. We believe, however, that the success of fuzzy ontologies strongly depends on the availability of effective reasoners able to deal with fuzzy ontologies.

In this paper we describe *fuzzyDL*, an expressive fuzzy ontology reasoner with some unique features. We discuss its possibilities for fuzzy ontology representation, the supported reasoning services, the different interfaces to interact with it, some implementation details, a comparison with other fuzzy ontology reasoners, and an overview of the main applications that have used it so far.

1 Introduction

In the last decade, *OWL 2 ontologies* [61], or simply *ontologies*, have become standard for knowledge representation. 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 [74].

The theoretical underpinnings of ontologies are strongly based on *Description Logics* (DLs) [5]. DLs 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 design of OWL 2 (Web Ontology Language) [31, 61], the current standard language to represent ontologies. As a matter of fact, OWL 2 is almost equivalent to the DL $\mathcal{SROIQ}(\mathbf{D})$.

Ontologies have been successfully used as part of expert and multiagent systems, as well as a core element in the Semantic Web, which proposes to extend

the current web to give information a well-defined meaning [7]. Despite of their many advantages, ontologies also have some limitations. One of them is that their classical two-valued semantics cannot directly manage *imprecise* or vague pieces of knowledge, which are inherent to several real-world problems [71].

Fuzzy set theory and fuzzy logic [99] 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 [3, 23, 55, 93], recommendation systems [25, 51, 64, 97], image interpretation [32, 33], the Semantic Web and the Internet [30, 66, 71], ambient intelligence [35, 54], ontology merging [27, 88], matchmaking [68], decision making [79], summarisation [50], robotics [36], and many others [6, 37, 45, 49, 52, 58, 59, 65, 69, 73]. The interested reader is referred to Section 5 for details about some of these applications.

The main formalism behind fuzzy ontologies are fuzzy DLs, proposed as an extension to classical DLs with the aim of dealing with *fuzzy/vague/imprecise information*. Since the first work of J. Yen in 1991 [98], an important number of works can be found in the literature. For a good survey on *fuzzy DLs* we refer the reader to [56, 81].

Fuzzy ontologies require the development of new languages and reasoning algorithms, together with the implementation of tools. Although there is not a standard for fuzzy ontology representation yet, some languages based on fuzzy DLs have been proposed [14], such as *Fuzzy OWL 2*. Due to the capability of ontologies to enable the automatic deduction of implicit knowledge, the success of fuzzy ontologies will strongly depend on the availability of effective reasoners.

fuzzyDL is an ontology reasoner supporting fuzzy logic reasoning. The objective of this paper is to present the latest version of the reasoner *fuzzyDL*, which offers various novel features with respect to previous versions. From a historical point of view, *fuzzyDL* can be considered as the first fuzzy DL reasoner. It is a popular and well-known tool that has been applied in several applications. In this paper we will present its possibilities for fuzzy ontology representation, the supported reasoning services, the different interfaces to interact with the system, some useful implementation details for ontology developers, a comparison of the main differences with other fuzzy ontology reasoners, and an overview of some applications that have successfully used *fuzzyDL*.

The rest of this paper is organised as follows. Section 2 describes an overview of the system, including its architecture and the different interfaces to interact with the tool. Section 3 enumerates the features and capabilities of the reasoner for fuzzy ontology representation and reasoning. Section 4 discusses some implementation details and summarises some optimisations. Next, Section 5 illustrates the usefulness of the system by revisiting some applications using it. Then, Section 6 compares *fuzzyDL* with other existing fuzzy DL reasoners, while Section 7 concludes and addresses some ideas for future work. Finally, we include two appendixes with some technical background (Appendix A) and a user manual with the syntax, semantics, and deeper details about the interaction with the tool (Appendix B).

2 System Overview

fuzzyDL is a fuzzy ontology reasoner. It supports a very expressive language: a fuzzy extension of a fragment of the language OWL 2 extended with some unique features and capabilities of fuzzy logic. It supports fuzzy operators from several families of fuzzy logics and implements a unique algorithm based on a combination of tableau rules and an optimisation problem.

fuzzyDL is publicly available in its webpage.[1] The installation of *fuzzyDL* is easy, since the user can download the file, uncompress it and run the main jar file without any further installation. However, the user must also install Gurobi (a *Mixed Integer Linear Programming* solver, MILP)[2] and get a valid license. Currently, Gurobi offers free licenses for academic purposes and limited trials otherwise.

This section provides a quick overview of the system. We will firstly discuss the architecture of the system in Section 2.1, specifying the inputs and outputs. Then, Section 2.2 details the different interfaces to interact with *fuzzyDL*. A detailed description of what types of fuzzy ontologies can be represented and what reasoning types are supported can be found in Section 3.

2.1 Architecture

The architecture of the reasoner is depicted in Figure 1 and describes its main components. There are two main parts: the upper part of the figure is dedicated to manage the user inputs, while the lower part orchestrates the reasoning procedures. These parts will be analysed in more detail in Sections 2.2 and 4, respectively.

The input of the reasoner is a fuzzy ontology and a set of user queries. As we will discuss in the next section, there are three different interfaces to provide this information: a Protégé plug-in to build fuzzy OWL 2 ontologies, its own syntax, and a Java API. These interfaces are handled by a Fuzzy OWL 2 parser, a *fuzzyDL* parser, and a Java API component, respectively. These components are responsible for representing the fuzzy KB knowledge and the queries using appropriate data structures. *fuzzyDL* also uses a configuration file that specifies the value of some parameters. The default values would be enough for most of the users; advanced users are referred to Appendix B.

After obtaining the fuzzy KB, the reasoning component answers user queries. We will explain now how the reasoning components are orchestrated and give more details in Section 4. Firstly, the reasoner performs some basic preprocessing to speed up the subsequent reasoning. Apart from some normalisations and optimisations that will be discussed later, *fuzzyDL* expands the axioms in the fuzzy KB once and reuse the results of this expansion for the different queries. For example, the TBox expansion module applies the terminological axioms to the known individuals, the RBox expansion computes the transitive closure of

[1] `http://straccia.info/software/fuzzyDL/fuzzyDL.html`
[2] `http://www.gurobi.com`

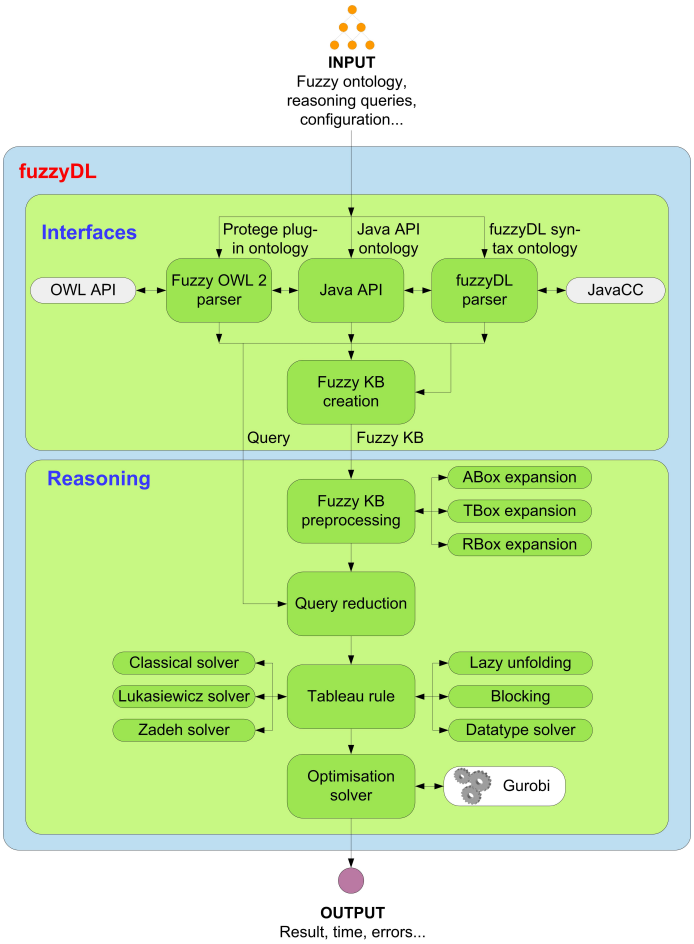


Figure 1: Architecture of *fuzzyDL* reasoner.

the role inclusion relation, and the ABox expansion generates new role assertions according to that relation. Next, the query reduction module reduces each of the user queries to a single reasoning task, and then *fuzzyDL* tries to solve it.

As anticipated, solving reasoning tasks combines some tableau rules with an optimisation problem. The tableau rule component is very complex because it includes an important number of complicated tableau rules (more than 80), implements several optimisation techniques (such as lazy unfolding or blocking), and interacts with different solvers: a datatype solver and one fuzzy logic solver for each of the supported semantics (currently, classical, Łukasiewicz, and Zadeh). The tableau rule component generates an optimisation problem, which is solved by an Optimisation solver component that interacts with an external tool.

As a result, *fuzzyDL* provides the following output information:

- Solution of the query, such as a numerical value, or a message warning that the fuzzy ontology is inconsistent.
- Time in seconds necessary to compute the solution.
- Main ingredients of a model of the fuzzy ontology (if it exists) that satisfies a query.
- Statistics of the fuzzy ontology: expressivity and number of elements of each type.
- Statistics of the reasoning, such as size of the generated model, number of applications of each reasoning rule, or size of the optimisation problem.

2.2 fuzzyDL Interfaces

There are three different interfaces to interact with *fuzzyDL*: command line interpreter, a Protégé plug-in, and a Java API. In the rest of this section we will briefly overview them; the interested reader can find more details in Appendix B.

Command line interpreter The direct way to use the reasoner is to run it from a command line shell. In this case, a graphical interface is not used: the user just inputs a text file with a fuzzy ontology and a set of queries, both of them written in the *fuzzyDL* syntax described in Appendix B. Any text editor can be used to write the input files in *fuzzyDL* syntax but, in order to make this task a little bit easier, we have adapted the *Sublime* text editor[3] to provide *fuzzyDL* syntax highlighting, as illustrated in Figure 2.

Protégé plug-in A more convenient way is to use a graphical interface: the FuzzyOWL Protégé plug-in.[4] This plug-in makes it possible to develop fuzzy ontologies using the independent language Fuzzy OWL 2 and to query *fuzzyDL*.

Once the plug-in is installed, a new tab in Protégé, named `Fuzzy OWL`, enables to use it. The plug-in is based on a methodology for fuzzy ontology representation using OWL 2 [14]. The key idea of this representation is to use an OWL 2 ontology and extend its elements with annotations representing the features of the fuzzy ontology that OWL 2 cannot directly encode. Note that the approach is backwards compatible so OWL 2 ontologies are valid fuzzy ontologies as well.

The non-fuzzy part of the ontology can be created by using Protégé as usual. After that, the user can define the fuzzy elements of the ontology by using the plug-in; namely, fuzzy axioms, fuzzy datatypes, fuzzy modifiers, and fuzzy concepts. Figure 3 (a) shows the available options, whereas Figure 3 (b) illustrates the plug-in's use by showing how to create a new fuzzy datatype.

[3] `http://www.sublimetext.com`
[4] `http://straccia.info/software/FuzzyOWL`

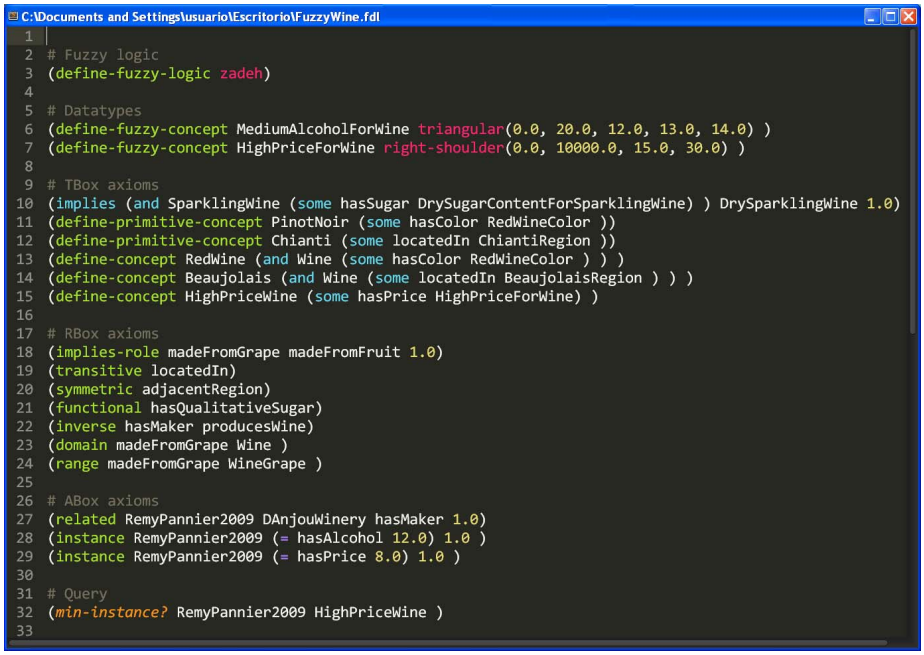


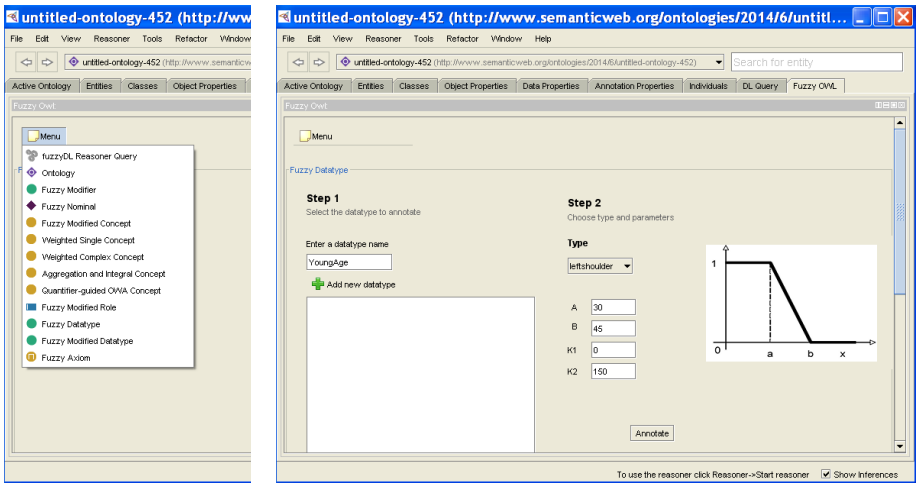
Figure 2: *fuzzyDL* syntax highlighting.

Using the option fuzzyDL *reasoner query* it is possible to submit queries to *fuzzyDL*. Such queries must be written in *fuzzyDL* syntax in the *Query* field and the results are displayed in the *Solution* field. Figure 4 shows how to check, given a fuzzy ontology about wines, whether the individual `RemyPannier2009` is an instance of the concept `HighPriceWine`.

Before submitting queries to *fuzzyDL*, the fuzzy ontology is translated into *fuzzyDL* syntax. Fuzzy OWL 2 is a more expressive language than the one supported by *fuzzyDL*, so some Fuzzy OWL 2 statements that are currently not supported by *fuzzyDL* are dropped during the translation and a warning message is shown to the user (see Appendix B.1.2 for a list of the not supported elements). The dropped statements are also saved into a log file for further inspection.

Java API A third option is to create fuzzy ontologies and answer queries over fuzzy ontologies using the *fuzzyDL* API for Java applications. This is especially useful if the user does not only want to visualise the results but wants to post-process them.

The *fuzzyDL* API is a Java library that allows both fuzzy ontology management and reasoning. The Javadoc documentation of the API, including all the classes and methods available for these tasks, can be found along with the



(a) (b)

Figure 3: Fuzzy OWL 2 plug-in: (a) Menu options; (b) Creation of a fuzzy datatype.

distribution.[5] Some examples of use of the API can be found in Appendix B.

With the *fuzzyDL* API it is possible to load existing fuzzy ontologies in *fuzzyDL* or fuzzy OWL 2 syntax. Another option is to create a new empty ontology and populate it using the different methods available to build complex concepts or add axioms to the fuzzy ontology. There are also methods to save fuzzy ontologies in *fuzzyDL* syntax and Fuzzy OWL 2, so it is possible to translate *fuzzyDL* syntax into Fuzzy OWL 2, and vice versa. Finally, there are several classes to represent the different reasoning tasks and methods to retrieve the same output information as from command line.

Example 2.1 *This example shows how to represent a very simple fuzzy ontology using the three different interfaces. There is one fuzzy concept (`Tall`), one fuzzy relation (`isFriendOf`), two individuals (`fernando` and `umberto`), and two axioms stating that `umberto` is tall, and that `fernando` and `umberto` are friends. The novelty of this fuzzy ontology is that the axioms are graded: `umberto` belongs to the fuzzy concept `tall` with degree greater or equal than 0.9, and `fernando` is related to `umberto` via the fuzzy relation `isFriendOf` with degree greater or equal than 0.8. In* fuzzyDL *syntax, the fuzzy ontology can be represented as*

```
(define-primitive-concept Tall *top* )
(instance fernando *top* 1.0)
(instance umberto Tall 0.9)
(related fernando umberto isFriendOf 0.8)
```

[5] `http://straccia.info/software/fuzzyDL/download/old/documents/javadoc`

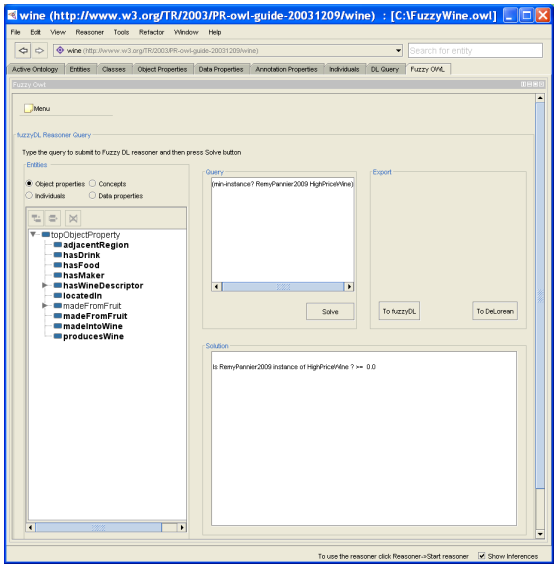


Figure 4: Using *fuzzyDL* reasoner from the Fuzzy OWL 2 plug-in.

where ****top**** *denotes the universal concept, similar to the OWL 2 class* ***Thing*** *(see Appendix B.1.1 for details). Now, let us consider the fuzzy OWL 2 representation obtained using the plug-in. In short, the following ontology in Manchester syntax has two annotations to represent the lower bounds of the two axioms:*

```
AnnotationProperty: fuzzyLabel Class: owl:Thing Class: Tall
ObjectProperty: isFriendOf Individual: umberto
    Types:
        Annotations: fuzzyLabel
            "<fuzzyOwl2 fuzzyType=\"axiom\">
                <Degree value=\"0.9\" />
            </fuzzyOwl2>"
        Tall
Individual: fernando
    Types:
        owl:Thing
    Facts:
        Annotations: fuzzyLabel
            "<fuzzyOwl2 fuzzyType=\"axiom\">
                <Degree value=\"0.8\" />
            </fuzzyOwl2>"
        isFriendOf  umberto
```

Finally, let us include a fragment of a program using the Java API to create an empty fuzzy ontology and add the two axioms to it:

```
// Creates a new fuzzy ontology
KnowledgeBase kb = new KnowledgeBase();

// Adds a fuzzy concept assertion
```

```
Individual i1 = kb.getIndividual("umberto");
Concept c = new Concept("Tall");
Degree d = Degree.getDegree(0.9);
kb.addAssertion(i1, c, d);

// Adds a fuzzy role assertion
Individual i2 = kb.getIndividual("fernando");
String r = "isFriendOf";
d = Degree.getDegree(0.8);
kb.addRelation(i2, r, i1, d);
```

3 Features and Capabilities

In this section we detail the main features and capabilities of *fuzzyDL*. Firstly, Section 3.1 details the possibilities for fuzzy ontology representation. Secondly, Section 3.2 enumerates the options for fuzzy ontology reasoning.

3.1 Fuzzy Ontology Elements

This section gives an overview of the elements of the fuzzy ontologies supported by *fuzzyDL*. The interested reader can find the specific syntax and semantics in Appendix A. We will assume that the reader is familiar with fuzzy ontologies or fuzzy DLs; some basic references for a quick refreshment are [56, 81].

The main ingredients of fuzzy DLs are *fuzzy concepts* (or classes), which denote unary predicates, *fuzzy roles* (or properties), which denote binary predicates, *individuals* (or instances), and *fuzzy datatypes* (or fuzzy concrete domains). Apart from these elements which have a counterpart in the classical case, there are other elements specific to the fuzzy case, such as *fuzzy modifiers* or *fuzzy quantifiers*. We will also have *variables* in our language which, depending on the case, can take the value of a data value or of a degree of truth. *Axioms* are formal statements involving these elements, like a recipe that defines how to combine the previous ingredients to represent the knowledge of some particular domain. Such axioms may hold to some *degree of truth*. Finally, the choice of the fuzzy logic determines the semantics of the fuzzy ontology. In the remaining of this section, we will analyse these elements in detail.

3.1.1 Fuzzy Logics

fuzzyDL currently supports 3 main semantics for the fuzzy ontology: Zadeh, Łukasiewicz, and, for backwards compatibility with crisp ontologies, classical. However, it is also possible to combine fuzzy operators of different fuzzy logics (including some operators of Gödel fuzzy logic) within the same fuzzy ontology.[6] Table 1 summarises the supported fuzzy operators.

[6] Furthermore, in Zadeh fuzzy ontologies it is usual to consider another implication in the semantics of the axioms, see Appendix B for details).

Table 1: Fuzzy operators supported by *fuzzyDL*.

Operator	Łukasiewicz logic	Gödel logic	Zadeh logic
Conjunction $\alpha \wedge \beta$	$\max(\alpha + \beta - 1, 0)$	$\min(\alpha, \beta)$	$\min(\alpha, \beta)$
Disjunction $\alpha \vee \beta$	$\min(\alpha + \beta, 1)$	$\max(\alpha, \beta)$	$\max(\alpha, \beta)$
Negation $\neg\alpha$	$1 - \alpha$	$\begin{cases} 1 & \text{if } \alpha = 0 \\ 0 & \text{otherwise} \end{cases}$	$1 - \alpha$
Implication $\alpha \rightarrow \beta$	$\min(1 - \alpha + \beta, 1)$	$\begin{cases} 1 & \text{if } \alpha \leq \beta \\ \beta & \text{otherwise} \end{cases}$	$\max(1 - \alpha, \beta)$

Instead of dealing with the infinite set of truth degrees in $[0, 1]$ as usual in fuzzy DLs, *fuzzyDL* actually restricts to a discretised set $\mathcal{D} = \{0, \frac{1}{n}, \ldots, \frac{n-1}{n}, 1\}$ for an appropriate big natural n satisfying that there is a machine representable $0 < \epsilon < \frac{1}{n}$. Assuming a finite set of truth degrees is unavoidable as *fuzzyDL* relies on numerical calculus to solve MILP problems using fixed precision. Specifically, in *fuzzyDL*, $\mathcal{D}$ is the set of double numbers that can be represented in Java and are supported by the Gurobi solver.

3.1.2 Fuzzy Concepts

In ontologies, concepts are inductively defined from simpler concepts, according to the concept constructors available in the language. Currently, *fuzzyDL* supports 46 different types of fuzzy concepts.

- *fuzzyDL* supports a fuzzy extension of a fragment of the language OWL 2. In particular, it supports the fuzzy DL $\mathcal{SHIF}(\mathbf{D})$ extended with object-data value restrictions and local reflexivity concepts. Recall that $\mathcal{SHIF}(\mathbf{D})$ concepts include atomic concepts, top, bottom, conjunction, disjunction, negation, and object-data existential and universal restrictions.

- In the classical case, an implication `C1` $\rightarrow$ `C2` can be represented as the disjunction between `C2` and the negation of `C1`. In fuzzy ontologies, this is not true in general, so *fuzzyDL* supports implication concepts explicitly.

- In fuzzy DLs, the semantics of conjunction, disjunction, negation and implication concepts depend on the choice of the fuzzy operator (t-norm, t-conorm, negation or implication function, respectively). Hence, in *fuzzyDL* there are several types of conjunctions, disjunctions, and implications, corresponding to different fuzzy logics: Zadeh, Łukasiewicz, and Gödel.

- *fuzzyDL* supports threshold concepts [15] that specify an upper or a lower bound for the degree of membership of every individual to a fuzzy concept.

- *fuzzyDL* data value restrictions can replace a data value with a real-valued variable or a fuzzy number [76].

- *fuzzyDL* supports the application of fuzzy modifiers [76] to change the membership function of the fuzzy concepts.
- *fuzzyDL* makes it possible to use aggregation concepts [11] (including fuzzy integrals) combining several fuzzy concepts as a generalisation of the conjunction and the disjunction.
- Finally, *fuzzyDL* supports fuzzy rough concepts [16], combining two different formalisms (fuzzy logic and rough set theory) to handle vagueness. This way, it is possible to represent different types of upper and lower approximations of the vague concepts.

Example 3.1 *The fuzzy concept* `(g-and Human (some hasAge very(Young))` *denotes the fuzzy set of very young humans, using Gödel conjunction and the fuzzy modifier* `very` *to change the interpretation of the fuzzy concept* `Young`. *The fuzzy rough concept* `(ua s Buyer)` *represents a potential buyer as an upper approximation of a unknown rough set of buyers using a fuzzy similarity relation* `s`. *The fuzzy concept* `(and Human (owa (0.6 0.3 0.1) (Young Tall Blond)))` *denotes the fuzzy set of humans that satisfy three criteria (being young, tall, and blond), where the degrees of satisfaction of the criteria are computed using an OWA operator (see Appendix A.3) with respect to a vector of weights* $W = [0.6, 0.3, 0.1]$.

3.1.3 Individuals

As usual in classical ontologies, individuals are split into two categories: *abstract individuals*, that are instances of the concepts, and *concrete individuals* (or data values), that belong to *concrete domains* (or datatypes), which are already structured and whose structure is already known to the machine, such as the integers, reals, or strings.

3.1.4 Fuzzy Roles

Similarly, roles are split into two categories: *abstract roles* (or object properties), that link two abstract individuals, and *concrete roles*, which relate an abstract individual and a data value. For example, `hasFriend` relates two abstract individuals.

3.1.5 Datatypes and Data Values

As usual in classical DLs, data values can be used as the range of data properties. Currently, *fuzzyDL* supports the following types of classical data values: integers, reals, strings, Booleans, and dates.

Furthermore, *fuzzyDL* supports fuzzy datatypes defined over a subinterval of the rational numbers. In particular, it supports trapezoidal (Figure 5 (a)), triangular (Figure 5 (b)), left-shoulder (Figure 5 (c)), and right-shoulder (Figure 5 (f)) membership functions. A crisp function (Figure 5 (d)) is included for backwards compatibility as it allows expressing exact integer or real values. It

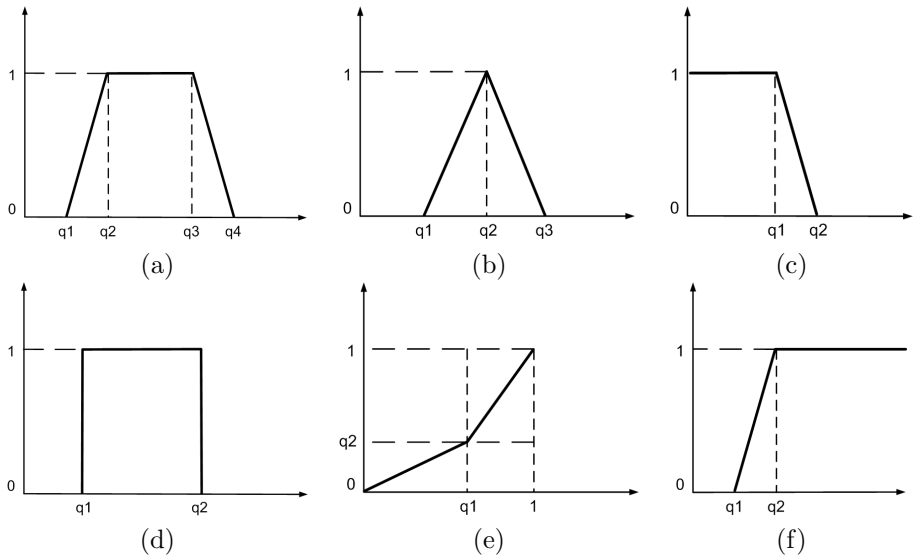


Figure 5: (a) Trapezoidal function; (b) Triangular function; (c) Left-shoulder function; (d) Crisp interval; (e) Linear function; (f) Right shoulder function.

is also possible to apply a fuzzy modifier to change the shape of these functions. Note also that in practice we assume a finite number of degrees of truth (see Section 4.1) so these functions are discretised as usual in fuzzy logic [46].

A common difficulty when building fuzzy ontologies is the definition of the fuzzy membership functions. A simple method to define such functions which usually behaves well in practice consists in uniformly partitioning the domain into an odd number of parts, usually 5. Then, the smaller value is generalised using a left-shoulder function, the higher value is generalised using a right-shoulder function, and the medium values are generalised using triangular functions, as illustrated in Figure 6. The functions are usually built to intersect with other functions in a point where the value of both functions is 0.5. For more details the interested reader is referred to [46].

Triangular membership functions can be seen as fuzzy numbers and, hence, it is possible to define arithmetical combinations of them. *fuzzyDL* supports addition, subtraction, multiplication, and division of two fuzzy numbers or of a fuzzy number and a numerical value.

3.1.6 Fuzzy Modifiers and Fuzzy Quantifiers

Fuzzy modifiers can change the interpretation of fuzzy concepts and fuzzy datatypes. *fuzzyDL* supports fuzzy modifiers defined using triangular (Figure 5 (b)) and linear functions (Figure 5 (e)).

Fuzzy quantifiers can be used in some aggregation concepts (in particular, in

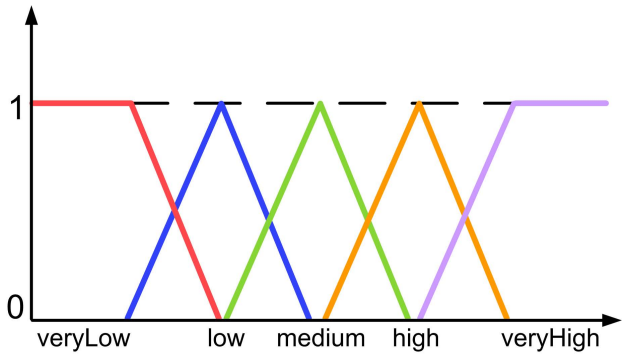


Figure 6: Partitioning a domain using fuzzy membership functions.

quantifier-guided OWA concepts). *fuzzyDL* supports fuzzy quantifiers defined by means of linear (Figure 5 (e)) and right-shoulder (Figure 5 (f)) functions.

3.1.7 Knowledge Bases

A *fuzzy ontology* or *fuzzy knowledge base* (KB) is a finite set of axioms. A fuzzy KB includes a fuzzy *ABox* (Assertional Box) with axioms about individuals, a fuzzy *TBox* (Terminological Box) with axioms about concepts, and a fuzzy *RBox* (Role Box) with axioms about roles. *fuzzyDL* supports the axioms in the fuzzy DL $\mathcal{SHIF}(\mathbf{D})$ plus reflexive object property axioms. We recall that fuzzy $\mathcal{SHIF}(\mathbf{D})$ supports concept-role assertions, General Concept Inclusions (GCIs), and role axioms (role inclusion axioms, transitivity, functionality, inverse, inverse functionality, symmetry, and data property range axioms). The difference with classical DLs is that now it is possible to specify a degree of truth in some fuzzy axioms, denoting a lower bound for the degree of satisfaction of the axiom.

fuzzyDL provides a syntactic sugar for some common cases of GCIs, such as concept equivalences, concept disjointness, disjoint union of concepts, domain, and range axioms. Furthermore, it supports different types of GCIs involving different fuzzy implications in the semantics.

Example 3.2 *The fuzzy concept assertion* `(instance bob Tall 0.4)` *states that Bob is a tall person with degree greater or equal than* 0.4. *The fuzzy role assertion* `(related alice bob hasFriend)` *ensures that Alice and Bob are friends to degree* 1. *The fuzzy GCI* `(l-implies Lion Dangerous 0.8)` *states that lions can be considered as dangerous animals with degree greater or equal than* 0.8*, where the inclusion is measured using Łukasiewicz implication.*

3.1.8 Degrees of Truth

As already mentioned, threshold concepts and some fuzzy axioms can include a degree of truth. In *fuzzyDL*, the possible degrees include not only $\mathcal{D} =$

$\{0, \frac{1}{n}, \ldots, \frac{n-1}{n}, 1\}$, as usual in the literature of finite fuzzy DLs, but also linguistic labels and $\mathcal{D}$-valued variables that may also appear in arithmetic inequations (see, e.g., Example B.7).

3.2 Reasoning Tasks

fuzzyDL allows typical tasks in fuzzy DLs like consistency or subsumption check but also supports the optimisation of variables or the computation of a defuzzification. *fuzzyDL* currently offers the following reasoning services:

- *KB consistency.* A fuzzy KB $\mathcal{K}$ is *consistent* if there is a model of $\mathcal{K}$ that satisfies each axiom in $\mathcal{K}$.
- *Concept satisfiability.* A fuzzy concept `C` is `D`-satisfiable w.r.t. a fuzzy KB $\mathcal{K}$ if there exists a model of $\mathcal{K}$ where `C` can have some instance with degree greater or equal than `D`, where `D` is a degree of truth. In *fuzzyDL*, this task can also consider some particular individual `o` instead of an arbitrary one.
- *Best satisfiability degree* (BSD) of a fuzzy concept `C` w.r.t. a fuzzy KB $\mathcal{K}$ is the maximal degree `D` such that `C` is `D`-satisfiable w.r.t. $\mathcal{K}$.
- *Minimal satisfiability degree* (MSD) of a fuzzy concept `C` is similar to the BSD but considering the minimal degree rather than the maximal one.
- *Concept subsumption.* `C2` `D`-subsumes `C1` w.r.t. a fuzzy KB $\mathcal{K}$ if in every model of $\mathcal{K}$, `C1` is included in `C2` with degree greater or equal than `D`. The degree of inclusion is computed using a fuzzy implication.
- *Entailment.* A fuzzy KB $\mathcal{K}$ entails an axiom if every model of $\mathcal{K}$ satisfies it. *fuzzyDL* computes entailments of assertions and GCIs.
- *Best Entailment Degree* (BED) of a non-graded axiom with respect to a fuzzy KB $\mathcal{K}$ is the maximal degree `D` such that the axiom is satisfied in every model of $\mathcal{K}$ with degree greater or equal than `D`.
- *Maximal Entailment Degree* (MED) of a non-graded axiom is similar to the BED but considering some model rather than any model.
- *Instance retrieval.* Given a concept `C` and a fuzzy KB $\mathcal{K}$, the instance retrieval problem computes the individuals that belong to `C` with a non-zero degree together with the minimal degree of membership in every model of $\mathcal{K}$.
- *Variable maximisation.* Given a fuzzy KB $\mathcal{K}$ and a variable `x`, maximise `x` such that $\mathcal{K}$ is consistent.
- *Variable minimisation.* Given a fuzzy KB $\mathcal{K}$ and a variable `x`, minimise `x` such that $\mathcal{K}$ is consistent.

- *Defuzzification.* Given a fuzzy KB $\mathcal{K}$, a concrete role `t`, a concept `C`, and an individual `o`, compute the BSD of `C` for the individual `o` and then defuzzify the value of `t` for the individual `o` using some defuzzification method: largest of maxima (LOM), smallest of maxima (SOM), or the middle of maxima (MOM).

- *Best Non-Fuzzy Performance (BNP).* Given a triangular fuzzy number `F` = `(triangular q1 q2 q3)`, $BNP(\texttt{F}) = (\texttt{q1} + \texttt{q2} + \texttt{q3})/3$. This task is particularly useful when fuzzy numbers are arithmetically combined.

Section 5 illustrates the use of some of these reasoning tasks in practical applications. Note that the term "Best" is used with a different meaning in entailment and satisfiability for historical reasons [56, 83]: the entailment degree of an axiom is in the interval $[BED, MED]$, while the satisfiability degree of a concept is in the interval $[MSD, BSD]$.

4 Implementation and Optimisation

This section provides some implementation details of *fuzzyDL*, paying special attention to the optimisation techniques. Section 4.1 starts with an overview of *fuzzyDL* reasoning algorithm. Next, Section 4.2 discusses some implementation details. Finally, Section 4.3 enumerates some implemented optimisations that improve the performance of the reasoner.

4.1 Overview of the Reasoning Algorithm

The *fuzzyDL* reasoning algorithm is an extension of some algorithms presented in the literature [11, 12, 13, 16, 17, 18, 26, 76, 81, 82, 85].

As we will see in Appendix B.4.2, each reasoning task is reduced to a variable minimisation problem. Indeed, the algorithm combines tableaux rules with an optimisation problem. Firstly, the reasoner applies some preprocessing steps to speed up the reasoning: normalisation of the fuzzy ontology, application of optimisation techniques (see Section 4.3), etc. Secondly, it reduces the user query to a reasoning task: variable minimisation with respect to a fuzzy KB. Then, *fuzzyDL* applies several tableau rules that decompose complex concept expressions into simpler ones, as usual in tableau algorithms, but also generate a system of linear inequation constraints. Such inequations need to hold in order to preserve the semantics of the fuzzy DL constructors. After all rules have been applied, an optimisation problem must be solved before obtaining the final result.

It is worth to note that the tableau rules are deterministic and only one optimisation problem is obtained. With the fuzzy operators supported by *fuzzyDL*, we will end up with a bounded MILP [70] problem, whose size, however, may be exponential w.r.t. the size of the initial knowledge base. We also recall that solving a MILP problem is an NP-complete problem (one can guess the assignment

to the binary variables and solve the system of linear inequations in polynomial time).

Appendix B shows that indeed all the reasoning tasks can be reduced to variable minimisation, that is, to minimise a $[0, 1]$-variable `x` given a fuzzy KB $\mathcal{K}$. For example, the BED of a concept assertion involving an individual `o` and a fuzzy concept `C` is equivalent to minimise the variable x given a fuzzy KB $\mathcal{K}'$ obtained after adding to $\mathcal{K}$ the fuzzy assertion `(instance o (not C) 1 - x)`. The interested reader can find more details about the tableau rules in [15, 81] and a running example in the appendix.

Remark 4.1 (Computational issues) *At the time of writing, two computational issues may arise within* fuzzyDL*.*

1. *A first issue may arise by the numerical solution identified by the underlying MILP solver. In some cases, the numerical precision adopted by the underlying MILP solver may not be enough to guarantee that the identified solution is indeed correct.*

2. *Another, orthogonal, issue is due to the fact that in some cases the reasoning problems addressed by* fuzzyDL*, and, by fuzzy DLs in general, have been show to be undecidable in the presence of GCIs (see, e.g., [4, 20, 26]) and in such cases the correct behaviour of fuzzyDL is not guaranteed.*[7] *However, this is not the case, e.g.,, in the case of Zadeh DLs or in Łukasiewicz DLs with an acyclic TBox [26, 76].*[8]

The reader is referred to the fuzzyDL *documentation for up to date information on computational issues related to* fuzzyDL*.*

4.2 Implementation Details

fuzzyDL has been developed in the Java language, reusing some existing Java libraries:

- *JavaCC*[9] parser generator is used to import *fuzzyDL* syntax input files.
- The de-facto standard *OWL API* [44][10] is used to import and export OWL 2 ontologies.
- *Gurobi* optimiser is used to solve MILP problems.

[7] Although some less expressive finitely-valued Łukasiewicz fuzzy DLs have shown to be decidable [21], *fuzzyDL* has not been proven to behave correctly as the reasoning algorithm is completely different than the one presented in [21].

[8] It is worth to stress that non-acyclic TBoxes can usually be (totally or partially) transformed into equivalent acyclic TBoxes using an absorption algorithm [18].

[9] `https://javacc.dev.java.net`

[10] `http://owlapi.sourceforge.net`

Older versions of *fuzzyDL* [15] used the MILP solver Cbc.[11] Cbc is implemented in C++, so it is necessary to access it using Java Native Interface (JNI). In contrast to that, Gurobi is accessible via a Java API, making the communication with the MILP solver easier and improving the performance of the reasoner.

As it happens in any (fuzzy) DL reasoner, implementing the reasoning algorithms efficiently is a difficult task. A careful choice of efficient data structures is needed. For example, it is convenient to use hash sets and hash tables to allow for easy retrieval of the searched elements. Furthermore, since the tableau rules are deterministic, no backtracking is currently necessary. Avoiding recursive implementations of the subalgorithms to compute blocking or ABox expansion significantly reduces the overhead. Finally, it is important to have a modular design of the application to enable an easy update with new fuzzy ontology elements.

fuzzyDL computes some tasks only once, such as the load of the fuzzy ontology, a normalisation of the original axioms, the transitive closure of the role inclusions axioms, the partitioning of the TBox, the expansion of the ABox according to the terminological knowledge, etc. Every user query is resolved against a cloned copy of the expanded knowledge base, thus reusing those inferences that can be deduced from the original fuzzy ontology but not those inferences derived from previous user queries.

Direct implementations of the reasoning algorithm do not behave well in practice. Since the complexity of the reasoning is high in the worst case, it is necessary to implement several optimisation techniques improving the performance in the most common scenarios in practice. The next section describes some of the most important optimisation techniques that are currently implemented and that could be reused in other fuzzy ontology reasoners.

4.3 Optimisations

fuzzyDL implements several optimisation techniques that improve the performance of the reasoning (similar techniques for other fuzzy ontology reasoners have been proposed [40, 72]). We will describe here the most important ones.

- *Rules introducing less variables.* In older versions of the reasoning algorithm, every assertion always generated some new variables [76]. More recent versions of the algorithm make it possible to reuse the variables that involve the same individuals and concepts/roles, thus reducing the total number of variables [13]. Obviously, the smaller number of variables, the better.

- *Rules with special cases.* Tableaux rules can be optimised in several special cases that are common in practice. For example, top and bottom concepts are very often part of TBox axioms, and *fuzzyDL* treats them as special cases improving the efficiency of the reasoning [18].

[11] `http://www.coin-or.org/projects/Cbc.xml`

- *N-ary conjunction and disjunction.* This is a notable example of special case. *fuzzyDL* considers conjunction and disjunction as n-ary operators instead of binary operators. This makes it possible to reduce the number of variables in the optimisation problem [18].

- *Different blocking strategies.* In order to guarantee the termination in some expressive DLs, some blocking mechanism is needed to avoid infinite applications of some rules. Often, more expressive logics require more complex blocking strategies [5]. Depending on the expressivity of the fuzzy ontology, *fuzzyDL* is able to avoid any blocking whenever possible or to choose the most appropriate strategy. This avoids using a more complicated strategy than necessary, as illustrated in Figure 7. The supported strategies are simple subset (of labels), simple equality (of labels), simple pairwise, anywhere subset, anywhere equality, and anywhere pairwise [39]. Furthermore, *fuzzyDL* can identify if the blocking status must be static or can be dynamic.

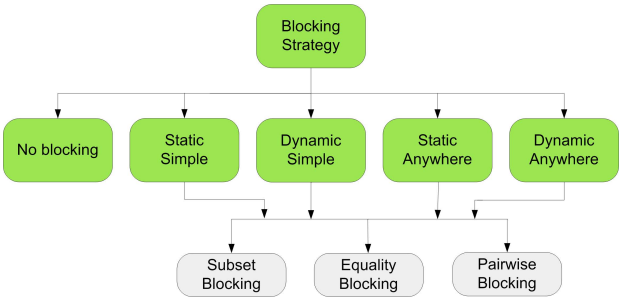


Figure 7: Different blocking strategies implemented in *fuzzyDL*.

- *Lazy unfolding.* Primitive concept inclusions can be handled more efficiently than GCIs. In classical DLs, the basic idea is that a GCI is applied (unfolded) to every individual considered in the reasoning, stating that either it is not an instance of the subclass or it is an instance of the superclass. However, a primitive concept inclusion can be lazily unfolded, i.e., only for those individuals which are known to be an instance of the atomic concept that appears in the axioms [5]. This technique can be generalised to the fuzzy case, reducing the application space of the primitive concept inclusion axioms [18]. A similar idea is implemented for concept equivalences, domain, range, and disjoint axioms. These axioms are formally defined in Appendix B.

- *Absorption.* This optimisation consists in transforming GCIs into primitive concept inclusions or other axioms to which lazy unfolding can be applied. The absorption algorithm computes a TBox partitioning that splits the original TBox into two parts: one with the axioms to which lazy unfolding can be applied, and another one with the axioms that do not

allow this technique [18]. Note that some absorption strategies used in the classical case are not valid in the fuzzy case.

- *Concept simplifications.* This technique replaces concept expressions with simpler ones. Usually, concept expressions are found to be equivalent to the top or the bottom concept. In other cases, concept expressions can be simplified. For example, the expression `(g-and C1 C1 C2)` can be simplified to `(g-and C1 C2)`. Note that some concept simplifications from classical DLs are not valid in the fuzzy case [40].

- *GCI simplification.* This technique makes it possible removing superfluous axioms or transforming axioms in equivalent but more efficient forms. For instance, the axiom `(implies C *top*)`, which essentially defines `C` as part of the ontology signature, can be omitted during the reasoning [18]. Again, some GCI simplifications from classical DLs do not hold in the fuzzy case.

- *Lexical normalisation.* This technique transforms concepts into a canonical form, by reordering the subconcepts [40]. This makes the application of other optimisations (such as concept simplification or absorption) easier. It is worth to note that in other reasoners lexical normalisation can detect obvious inconsistencies sooner, but this is not always the case in *fuzzyDL* since they cannot usually be found until the end of the reasoning, once the optimisation problem has been solved.

- *Degrees normalisation.* If a fuzzy ontology contains several axioms that only differ in the degree of truth, *fuzzyDL* just considers one axiom. If the degrees are numerical, *fuzzyDL* takes the axiom with the most restrictive one, i.e., the one with the highest degree [72]. If the degrees are variables, *fuzzyDL* adds some inequalities to make sure that the axiom is satisfied with a degree greater or equal than all the involved variables.

- *Optimisation partitioning.* This technique consists in splitting the inequation constraints that are derived during the reasoning algorithm into independent sets, with the aim of solving easier optimisation problems [17].

- *Caching.* In order to save the reasoner from repeating the same work, *fuzzyDL* uses caching techniques to store some intermediate results [40]. This happens in the absorbed TBoxes, expanded ABoxes, intermediate queries (for instance, the value `z` in a MOM defuzzification can be computed just once), etc.

- *Encoding strings as numbers.* The names of the elements in the tableau (concept, properties, individuals, datatypes, variables, etc.) are encoded as unique integer numbers. For instance, it is computationally cheaper to check the equality of two individuals by comparing two integers than by comparing two strings.

5 Use Cases

In this section we overview some practical applications that have successfully used *fuzzyDL*. This also allows the reader to better understand the benefits one may have by using fuzzy ontologies compared to crisp ontologies only.

Matchmaking *fuzzyDL* has been applied to solve matchmaking problems in e-commerce applications [68]. In this setting, two parts (a buyer and a seller) express their constraints and preferences using fuzzy concepts, so they would match with a certain degree. Typically, their requests are expressed as two concepts `Buyer` and `Seller`, respectively, formed as a combinations of their constraints and of their preferences, using conjunctions or aggregation operators expressing their relative importance. For example, assume that the buyer is looking for a dark-color item and prefers not paying more than 200 euro, although he could go up to 300 to a lower degree of satisfaction. Assuming that the price is more important than the color, this can be expressed as

```
(define-concept Buyer  (w-sum (0.3 DarkColor)
                              (0.7 (some hasPrice (left 0 0 200 300))))).
```

Of course, there can also be a background theory about the domain, defining for instance the meaning of dark-color:

```
(define-concept DarkColor  (some hasColor (or Black Grey))).
```

Next, the best agreement between the buyer and the seller can be computed as the maximal satisfiability degree of the conjunction of the preferences, i.e.,

```
(max-sat?  (l-and Buyer Seller)).
```

Using Łukasiewicz conjunction, the solution (if non-zero) can be proved to be Pareto optimal.

Fuzzy control and rule-based systems *fuzzyDL* is able to represent and reason with Mamdani fuzzy IF-THEN rules, commonly applied in fuzzy control problems, in combination with background knowledge. In particular, it has been applied in a recommender system of touristic attractions [97]. Given a touristic attraction, its attributes (relevance, timing, etc.) are defined using fuzzy role assertions, such as

```
(instance stadium1 (some relevance 4) 1.0).
```

Then, fuzzy rules are used to infer the user satisfaction from the touristic attraction attributes, such as

```
(define-concept Rule1  (g-and (some relevance highRelevance)
                              (some timing lowTiming)
                              (some satisfaction highSatisfaction)))
```

This encodes that if the relevance is high and the timing is low, then the satisfaction is high. These rules use fuzzy membership functions that are defined using fuzzy datatypes, such as

```
(define-fuzzy-concept highRelevance (right 0 10 5 9)).
```

Finally, one computes the satisfaction of some particular tourist attraction given a set of rules via a defuzzification method as follows:

```
(defuzzify-som?  (or Rule1 ...  Rule6) stadium1 satisfaction).
```

Another possibility is using *fuzzyDL* in combination with rule management systems (such as Drools) in order to provide a unified framework. This has also been explored in the domain of tourism [22].

Multicriteria decision making (MCDM) In a MCDM problem, a group of experts decides among a set of alternatives the one with the highest degree of desirability according to a set of criteria. To this end, the experts define the degree to which every alternative satisfies a criterion (the performance of the alternative) and the relative importance of the criterion. The application of *fuzzyDL* to MCDM has been discussed in [14, 79]. The performance of alternative i and criterion j according to expert k is defined with the help of a triangular fuzzy number, such as

```
(define-concept performance-ijk
                (some hasScore (triangular 0.6 0.7 0.9))).
```

Evaluations of an alternative i for an expert k are defined as weighted sums of the performances of the criteria, where the relative importance of the criteria are defined using numerical weights. For example, if there are 3 criteria,

```
(define-concept Evaluation-ik  (w-sum (0.33 performance-i1k)
                               (0.33 performance-i2k)
                               (0.33 performance-i3k))).
```

The evaluation of an alternative i is computed as a weighted sum of the evaluations for every expert, being possible to use the weights to give more importance to some experts than other ones. For example, if there are 2 experts and one of them is more reliable,

```
(define-concept Evaluation-i   (w-sum (0.6 performance-i1)
                              (0.4 performance-i2))).
```

Finally, the alternative i with a highest maximal satisfiability degree of the concept `Evaluation-i` is selected.

Recommendation systems Recommendation systems can benefit from *fuzzyDL* features. In fact, *fuzzyDL* has been used as part of a system recommending wines [25, 64, 95]. The use of a fuzzy ontology makes it possible to represent wine attributes such as price, alcohol level, sugar, or acidity in a more convenient way with the help of membership functions. Then, a Java application uses *fuzzyDL* API to solve instance retrieval queries where the output is a list of wines that satisfy some features combined using an OWA operator. For instance, a possible query is

```
(all-instances?  (owa (0.33 0.33 0.33)
                 (RedWine HighAlcohol HighAcidity))).
```

Robotic perception *fuzzyDL* has been used in the context of an ambitious research project in robotics, oriented to robot planning and plan execution [36]. One of the robot tasks is the localisation and transportation of sample containers from a lunar surface. The robot has a perception module responsible of differentiating between different types of sample containers (with different heights and sizes), and other objects on the lunar surface. Reasoning is based on a fuzzy ontology where the TBox contains spatial descriptions and the ABox contains extracted features of the objects. The authors differentiate between damaged, that have fallen down, and usable containers, that are standing upright and can be transported. The heights and sizes that are used to differentiate the containers are interpreted in a fuzzy way. For example, `DamagedContainer` is defined combining (using conjunction or weighted sum) fuzzy concepts, one of them being

```
(some hasDistanceToGround (triangular 0 0.2 0.4)).
```

The queries

```
(min-instance?  object-i DamagedContainer)
```

and

```
(min-instance?  object-i UsableContainer)
```

detect the container type of a given object.

Ambient intelligence Ambient intelligence systems need a precise representation of the environment and a recognition of the human activity in order to answer with appropriate actions. *fuzzyDL* has been used for modelling (using fuzzy ontologies) and recognition (using reasoning services) of complex human activities in real-life (office and public buildings) scenarios [35, 54]. The attributes that characterise an activity (such as location, lights, or temperature) are represented using fuzzy assertions. Possible activities (such as having lunch or doing exercise) are represented as fuzzy classes. The system includes rules to deduce activities from the current situation, such as

```
(define-concept (l-implies  (g-and User
                                   (some performsAction (w-sum
                                     (0.1 UseBottle)
                                     (0.4 UsePlate)
                                     (0.2 UseFork)
                                     (0.2 UseSpoon)
                                     (0.1 UseKnife))))
                            (some performsActivity HaveLunch))).
```

Finally, the query

```
(min-instance?  ana (some performsActivity HaveLunch))
```

computes the minimal degree to which Ana is performing the activity of having lunch.

Text mining of Twitter data A method for detecting influenza rates in a geographical region using the knowledge extracted from Twitter messages has been proposed in [6]. Extracted data are represented using a fuzzy ontology. A fuzzy property `isManifested` links flu symptoms with linguistic variables (`Weak`, `Moderate`, and `Strong`). There are 9 rules detecting flu from its symptoms, such as

```
(define-concept R1 (g-and Fever (some isManifested Strong))).
```

Next, the fuzzy concept `Influenza`, that represents positive flu tweets, is defined as a weighted maximum of the 9 rules. *fuzzyDL* computes the degree of a tweet `o` being a relevant message as

```
(min-sat?  Influenza o).
```

Argumentation Fuzzy ontologies can improve some aspects of argumentation theory such as the weakest link principle (Gödel fuzzy logic) and accrual of arguments (Łukasiewicz fuzzy logic) by tolerating inconsistency [52]. The approach is applied to the domain of law (argumentation in an assault accusation) using *fuzzyDL*. In this setting, the domain can be modelled using fuzzy concepts such as `WeakPerson` or `StrongPerson`. Then, the argument that a small and weak person with an attack role towards a large and strong person leads to an implausible situation can be modelled using an axiom

```
(l-implies  (and SmallPerson WeakPerson
                 (some attack (and LargePerson StrongPerson)))
            ImplausibleAttack)
```

Hence, it is possible to query whether an individual is a victim of an implausible attack as

```
(min-instance?  david ImplausibleAttack).
```

Ontology learning In [45], *fuzzyDL* is used to extend an ontology-based Inductive Logic Programming (ILP) system (DL-Learner) with fuzzy logic capabilities. Instead of only considering positive and negative training examples, they are represented using fuzzy concept assertions including a minimal degree of truth. The learning algorithm is based on generating candidate concepts that may solve the ILP problem and calculating their quality for some set of examples. This is achieved with BED queries computing the minimal degree for an individual being an instance of a candidate concept.

Image interpretation and retrieval An image interpretation framework using *fuzzyDL* for the extraction of the semantics of the images is described in [32, 33]. Their framework is applied to the domain of outdoor images, where they use fuzzy assertions like

```
(instance image (some contains Vegetation) 0.72)
```

to describe images. Then, different fuzzy reasoning techniques are applied to improve the extracted descriptions: determining the most plausible description, resolving inconsistencies, enriching the descriptions using background knowledge, etc. These tasks require *fuzzyDL* to solve several concept satisfiability and BED tests.

In the framework of geospatial semantic retrieval, a fuzzy geospatial ontology including fuzzy spatial relations has been developed [55]. For example, `distance` is considered as a fuzzy property. During the development of this fuzzy ontology, *fuzzyDL* has been used to check the consistency and to discover subsumptions and role assertions. The authors show that their system increase the precision and recall with respect to crisp models.

Cognitive vision *fuzzyDL* has been used for recognizing and describing meaningful events in video sequences from different domains [37]. The reasoner has been used to define a fuzzy ontology of spatio-temporal occurrences and to reason with it. For example, a `meet` event between two individuals is represented using a fuzzy concept. The authors define fuzzy rules to compute the confidence on two people meeting using their present and past distances, described using fuzzy membership functions. Finally, the resulting value is obtained after a LOM defuzzification.

Workflow management systems A workflow management system provides a procedural automation of a business process by managing the sequence of its work activities. A methodology for integrating workflow management systems and fuzzy ontologies has been recently proposed [73]. This way, it is possible to represent fuzzy workflows where limits are not strict but vague. There is a prototypical implementation using *fuzzyDL* to represent the fuzzy ontology and to reason with it. The main objective of the reasoning is to update the fuzzy ontology according to the feedback obtained during the workflow execution. This is achieved by computing BEDs of concept assertions and by solving concept subsumption queries.

Cultural object descriptions In [49] *fuzzyDL* is used as part of an authoring environment providing representations of cultural heritage object descriptions with linguistic information. The system is able to perform some reasoning, inferring missing values by computing a membership degree. Some notions such as being an `Interesting` object are interpreted as fuzzy concepts.

6 Related Work

In this section we will overview other existing fuzzy DL reasoners. FIRE, FPLGERDS, YADLR, DELOREAN, GURDL, FRESG, LIFR and SMT-BASED SOLVER support standard tasks for DL reasoners. Other reasoners are more oriented to the integration with databases, namely DLMEDIA, SOFTFACTS and ONTOSEARCH2. Finally, FUZZYRDF is oriented to fuzzy RDF graphs, and FUZZYDL-LEARNER is a tool focused on learning GCIs from OWL 2 facts. It is also worth to note a fuzzy extension of the classical reasoner KAON2.

fuzzyDL supports a very expressive language, with elements that no other fuzzy ontology reasoner supports. However, as we will see, most of the other reasoners also provide features that *fuzzyDL* is not able to support, so the choice of the fuzzy ontology reasoner may be application dependent.

Fire *Fuzzy Reasoning Engine* implements a tableau algorithm for fuzzy $\mathcal{SHIN}$ under Zadeh logic [75]. Developed in Java, it is publicly available.[12] It supports several reasoning tasks (consistency, entailment, BED, concept satisfiability, subsumption problems, and classification). An interesting feature is its graphical interface, although users need to deal directly with the syntax of the language for the fuzzy KB representation. A difference with *fuzzyDL* is the ability to serialise fuzzy $\mathcal{SHIF}$ ontologies into RDF triples and the integration with classical RDF storing systems, which provide persistent storing and querying over large-scale fuzzy information.

FPLGERDS *Fuzzy Predicate Logic Generalized Resolution Deductive System* is an implementation of a resolution algorithm for fuzzy predicate logic. Being able to map fuzzy DLs to fuzzy predicate logics, it can be used to solve the BED problem in fuzzy $\mathcal{ALC}$ extended with some role constructors (role negation, top role and bottom role) under Łukasiewicz fuzzy logic [41]. It is a fuzzy extension of the reasoner GERDS (GEneralised Resolution Deductive System), which is often erroneously credited as a fuzzy ontology reasoner. It is publicly available.[13]

YADLR It is a Prolog implementation of a combination of a resolution-based algorithm with linear programming for reasoning with a multi-valued DL [47, 48]. It supports fuzzy ABox reasoning in fuzzy $\mathcal{ALCOQ}$ extended with local reflexivity concepts under Łukasiewicz and Zadeh fuzzy logics. An interesting

[12] `http://www.image.ece.ntua.gr/~nsimou/FiRE`
[13] `http://irafm.osu.cz/en/c104_fplgerds`

feature is that it allows variables as degrees of truth. Currently supported reasoning tasks are concept assertion entailment, instance retrieval, concept realisation, BED of a concept assertion, and fuzzy concept description learning. It is publicly available.[14]

DeLorean *DEscription LOgic REasoner with vAgueNess* supports fuzzy rough $\mathcal{SROIQ}(\mathbf{D})$ ontologies under Zadeh and Gödel fuzzy logics [9]. Nowadays, DeLorean is the only one that currently supports fuzzy OWL 2, although it does not support several elements of *fuzzyDL*. Its reasoning algorithm consists in a reduction to reasoning in crisp $\mathcal{SROIQ}(\mathbf{D})$ [8, 10, 14, 19]. As a consequence, it allows to reuse classical languages and resources (editors, tools, reasoners ...). There are some optimisations of the reduction to crisp $\mathcal{SROIQ}(\mathbf{D})$ but the only existing optimisations for reasoning in crisp $\mathcal{SROIQ}(\mathbf{D})$ are those implemented by the reused classical reasoner. It allows linguistic labels as degrees of truth and is publicly available.[15]

GURDL It supports an extension of $\mathcal{ALC}$ with an abstract and more general notion of uncertainty [40]. This makes more difficult to extend the reasoner to more expressive fuzzy DLs. Degrees of truth are taken from a certainty lattice. The reasoning algorithm is based on a mixture of tableau rules and resolution of a set of inequations. Moreover, it implements some interesting optimisation techniques (lexical normalisation, concept simplification, partitions based on individual connectivity, caching) and studies the applicability of some techniques used in the crisp case. It is not publicly available.

FRESG *Fuzzy Reasoning Engine Supporting Fuzzy Data Type Group* is a prototype reasoner that implements a tableau algorithm for fuzzy $\mathcal{ALC}(\mathbf{D})$ under Zadeh logic, with customised fuzzy data types and customised fuzzy data type predicates but with an empty TBox [94]. It supports several reasoning tasks (ABox consistency, entailment, fuzzy concept satisfiability, instance retrieval, and concept realisation). It has a simple graphical interface, but it is not publicly available.

LiFR *Lighweight Fuzzy semantic Reasoner* is a fuzzy extension of the Pocket KRHyper mobile reasoner [90]. In older versions, the reasoner was called f-PocketKRHyper [91]. The reasoner is oriented to mobile devices and the supported language is the tractable DLP fragment. There are fuzzy concept assertions and weighted concepts, but there are no GCIs and role assertions are forced to be crisp. The reasoner implements a hyper-tableaux calculus. The supported reasoning tasks are concept satisfiability, subsumption, concept assertion entailment, and BED. It is not publicly available currently, but it is announced to be soon.[16]

[14] `http://sourceforge.net/projects/yadlr`
[15] `http://webdiis.unizar.es/~fbobillo/delorean`
[16] `http://mklab.iti.gr/project/lifr`

SMT-based solver There is an implementation of a fuzzy DL concept satisfiability checker for infinite-valued Product fuzzy $\mathcal{ALE}$ using a Sat Modulo Theory (SMT) solver but, as far as we know, the reasoner does not have an official name [2]. The only supported reasoning task is concept satisfiability over quasi-witnessed models. The reasoner is based on a translation to non-linear real arithmetic formulae. There is an online version of the tool.[17]

DLMedia & SoftFacts *DLMedia* is a fuzzy ontology-based multimedia information retrieval system combining logic-based retrieval with multimedia feature-based similarity retrieval [80]. The supported language is *DLR–Lite* [24] with fuzzy concrete domains expressing similarity relations between keywords. Retrievable data may be stored into a multimedia database. The query language are fuzzy conjunctive queries and top-k query answering (CQA) can be computed. It is publicly available.[18] *SoftFacts* is similar to DLMedia and supports *DLR–Lite*, but rather accessing to a multimedia retrieval engine to retrieve relevant images, SoftFacts is tailored to connect to a relational DBMS for large-scale retrieval of facts [84]. Results may be scored and fuzzy conjunctive queries and top-k query answering (CQA) is supported. It is publicly available.[19]

ONTOSEARCH2 It is the first scalable query engine for fuzzy ontologies [62]. It can answer fuzzy conjunctive queries over fuzzy *DL-Lite*$_{\mathcal{R}}$ [24] ontologies, allowing queries to be defined using a fuzzy extension of SPARQL. The reasoner is implemented as an extension of TrOWL [87] and is accessible from a HTML web page or using a web service. Currently, it is not publicly available.

fuzzyRDF This reasoner supports a fuzzy version of RDFS, where RDF triples can have a degree of truth associated [78]. Given a fuzzy conjunctive query, it computes a top-k retrieval of data stored into a relational database. It is publicly available.[20]

FuzzyDL-Learner It is a tool to learn fuzzy GCIs for OWL 2 ontologies [53]. Given an OWL 2 ontology, FuzzyDL-Learner allows to learn fuzzy concept expressions in fuzzy $\mathcal{EL}(\mathbf{D})$++ and the degree of subsumption. The concept expressions can contain fuzzy concrete domains. It is publicly available.[21]

KAON2 It is a DL reasoner, which has been extended with the reduction of fuzzy DLs to classical DLs, as first mentioned in [1]. The reduction to classical DLs is similar to that implemented in DeLorean, although KAON2 implements a less expressive language (fuzzy $\mathcal{ALCH}$ in Zadeh fuzzy logic). An empirical study of the scalability of reasoning with fuzzy ontologies using this

[17] `http://arinf.udl.cat/fuzzydlsolver`
[18] `http://straccia.info/software/DL-Media/DL-Media.html`
[19] `http://nmis.isti.cnr.it/~straccia/software/SoftFacts/SoftFacts.html`
[20] `http://straccia.info/software/fuzzyRDF/fuzzyRDF.html`
[21] `http://straccia.info/software/FuzzyDL-Learner`

reasoner has been performed in [29]. The reasoner is publicly available,[22] but the fuzzy extension is not. We will not consider this reasoner in the discussion below as we could not find the necessary information.

Discussion Table 2 compares the languages supported by the reasoners and the existence of a GUI, while Table 3 compares their reasoning services and optimisations (the columns "CON", "ENT", "CSAT", "SUB", "IR", "BED", and "OPT" represent consistency, entailment, concept satisfiability, subsumption, instance retrieval, BED, and optimisations, respectively).

Table 2: Languages supported by the fuzzy ontology reasoners.

Reasoner	Fuzzy DL	Logic	Degrees	Other constructors	GUI
fuzzyDL	$\mathcal{SHIF}(\mathbf{D})$	Z, Ł	General	Modifiers, rough, aggregation	Yes
Fire	$\mathcal{SHIN}$	Z	Numbers		Yes
FPLGERDS	$\mathcal{ALC}$	Ł	Numbers	Role negation/top/bottom	No
YADLR	$\mathcal{ALCOQ}$	Z, Ł	General	Local reflexivity	No
DeLorean	$\mathcal{SROIQ}(\mathbf{D})$	Z, G	General	Modifiers, rough DL	Yes
GURDL	$\mathcal{ALC}$	General	Numbers		No
FRESG	$\mathcal{ALC}(\mathbf{D})$	Z	Numbers	Fuzzy datatype expressions	Yes
LiFR	DLP fragment	Z	Numbers	Weighted concepts	No
SMT-based solver	$\mathcal{ALE}$	Π	No	No	No
DLMedia	*DLR–Lite*	Z, G	Numbers	Fuzzy datataypes	Yes
ONTOSEARCH2	$DLLite_R$	General	Numbers		Yes
fuzzyRDF	RDFS	General	Numbers		Yes
FuzzyDL-Learner	$\mathcal{EL}$++(**D**)	G	Numbers		Yes

Let us note that *fuzzyDL* supports an expressive logic, with elements that no other fuzzy ontology reasoner is able to support. In particular, aggregation of fuzzy concepts, some explicit fuzzy set membership functions and fuzzy modifiers, or defuzzification are unique features of *fuzzyDL* or it allows a more general use than the reasoners that support them. However, many of the other reasoners also provide features that *fuzzyDL* is not able to support:

- Fire implements a (basic) classification algorithm, persistent storing support and number restrictions;
- FPLGERDS supports additional role constructors;
- YADLR and FuzzyDL-Learner support fuzzy DL learning;
- DeLorean supports (unrestricted) nominals, qualified number restrictions and some additional role axioms;
- GURDL supports a more general representation of uncertainty;
- FRESG supports customised datatypes;

[22] `http://kaon2.semanticweb.org`

Table 3: Reasoning services offered by the fuzzy ontology reasoners.

Reasoner	CON	ENT	CSAT	SUB	IR	BDB	Other tasks	OPT
fuzzyDL	Yes	Yes	Yes	Yes	Yes	Yes	Defuzzification	Yes
Fire	Yes	Yes	Yes	Yes	No	Yes	Classification	Yes
FPLGERDS	No	Yes	No	No	No	No		No
YADLR	No	Partial	No	No	Yes	Partial	Realisation	No
DeLorean	Yes	Yes	Yes	Yes	No	Yes		Yes
GURDL	Yes	Yes	No	Yes	No	No		Yes
FRESG	Yes	Yes	Yes	No	Yes	No	Realisation	No
LiFR	No	Partial	Yes	Yes	No	Yes		No
SMT-based solver	No	No	Yes	No	No	No		No
DLMedia	No	No	No	No	No	No	Top-k CQA	No
ONTOSEARCH2	No	No	No	No	No	No	Retrieval	No
fuzzyRDF	No	No	No	No	No	No	Top-k CQA	No
FuzzyDL-Learner	No	No	No	No	No	No	GCI learning	No

- SMT-based solver supports Product fuzzy DLs;
- LiFR, DLMedia, SoftFacts and ONTOSEARCH2 support tractable fuzzy ontology languages;
- DLMedia, SoftFacts and fuzzyRDF support integration with relational databases;

It would be interesting to perform a worst case complexity of the different algorithms. Unfortunately, the complexities of the other fuzzy ontology reasoners have not been published (with the exception of DeLorean which computes a reduction to classical ontologies).

An empirical comparison of all these fuzzy ontology reasoners is out of the scope of this paper. The comparison is challenging because each of the reasoners have some unique features that a common comparison should discard: the common fragment might also not be representative of the usefulness or the performance of any of the reasoners. Furthermore, the reasoners support different syntaxes and many of them are not even publicly available, so this evaluation seems more appropriate for a collaborative work with other reasoner developers.

7 Conclusions and Future Work

This paper has presented the main features of the fuzzy ontology reasoner *fuzzyDL*. We have enumerated the supported fuzzy ontology features (a fuzzy extension of a subset of OWL 2 with some unique fuzzy ontology features), the supported reasoning services, and the different ways to interact with the tool: as a stand-alone application, through a Java API or using a Protégé plug-in with the aim to offer ontology developers a tool to manage fuzzy ontologies.

We have also given some key ideas of the underlying reasoning algorithms, based on a combination of tableau rules and an optimisation problem, and discussed some recent optimisation techniques that have improved its performance. We believe that these implementation details will be interesting for the developers of other fuzzy reasoning engines as well.

fuzzyDL is arguably the most popular fuzzy ontology reasoner and several applications have taken advantage of its reasoning capabilities, including matchmaking applications or fuzzy control systems. We believe that the presented use cases could encourage new applications to be developed.

fuzzyDL provides several unique features with respect to the other similar fuzzy ontology reasoners. However, most of the other fuzzy ontology reasoners also include some unique features, so the most appropriate will depend on the particular application. This paper contains a detailed comparison that will hopefully be of help in the choice of the appropriate reasoning engine.

fuzzyDL reasoner is in continuous update. A first line of future research is related to the extension of the expressivity of the logic. In this regard, we are especially interested in a more general support for nominals and in the missing constructors of fuzzy OWL 2. Furthermore, we are very interested in the implementation of the algorithms to reason with different families of fuzzy operators [10, 13]. We would like to dedicate some effort to the design and implementation of more advanced optimisation techniques to reduce the running time, especially related to the classification and conjunctive query answering problems. In this line, classification is more involved in the fuzzy case because, given a fuzzy ontology $\mathcal{O}$ and contrary to the crisp case, one may have e.g., cyclic subsumptions among concepts without them being equivalent. That is, e.g., one may have that $\mathtt{A}$ is subsumed by $\mathtt{B}$ to degree $\mathtt{D}_1$, while $\mathtt{B}$ is subsumed by $\mathtt{A}$ to degree $\mathtt{D}_2$, with $\mathtt{D}_1 \neq \mathtt{D}_2$. Last but not least, we would like to perform a more detailed evaluation of *fuzzyDL* after implementing these optimisations, completing the preliminary evaluation described at [18], and possibly comparing it with other fuzzy ontology reasoners.

Acknowledgments

F. Bobillo has been partially supported by the CICYT projects TIN2012-30939, TIN2013-46238-C4-4-R and DGA-FSE. We would like to thank the anonymous reviewers for their valuable comments on an earlier version of this paper.

References

[1] S. Agarwal and P. Hitzler. sMart - A semantic matchmaking portal for electronic markets. In *Proceedings of the 7th International IEEE Conference on E-Commerce Technology (CEC 2005)*, 2005.

[2] T. Alsinet, D. Barroso, R. Béjar, F. Bou, M. Cerami and F. Esteva. On the implementation of a fuzzy DL solver over infinite-valued Product logic

with SMT solvers. In *Proceedings of the 7th International Conference on Scalable Uncertainty Management (SUM 2013)*, volume 8078 of *Lecture Notes in Computer Science*, pages 325–330. Springer Verlag, 2013.

[3] T. Andreasen and H. Bulskov. Conceptual querying through ontologies. *Fuzzy Sets and Systems*, 160(15):2159–2172, 2009.

[4] F. Baader, S. Borgwardt, and R. Peñaloza. On the decidability status of fuzzy $\mathcal{ALC}$ with general concept inclusions. *Journal of Philosophical Logic*, 44(2):117–146, 2015.

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

[6] R. Balaj and A. Groza. Detecting influenza epidemics based on real-time semantic analysis of Twitter streams. In *Proceedings of the 3rd International Conference on Modelling and Development of Intelligent Systems (MDIS 2013)*, pages 30–39, 2013.

[7] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *The Scientific American*, 284(5):34–43, 2001.

[8] F. Bobillo, M. Delgado, and J. Gómez-Romero. Crisp representations and reasoning for fuzzy ontologies. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 17(4):501–530, 2009.

[9] F. Bobillo, M. Delgado, and J. Gómez-Romero. DeLorean: A reasoner for fuzzy OWL 2. *Expert Systems with Applications*, 39(1):258–272, 2012.

[10] F. Bobillo, M. Delgado, J. Gómez-Romero, and U. Straccia. Joining Gödel and Zadeh fuzzy logics in fuzzy description logics. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 20(4):475–508, 2012.

[11] F. Bobillo and U. Straccia. Aggregation operators for fuzzy ontologies. *Applied Soft Computing*, 13(9):3816–3830, 2013.

[12] F. Bobillo and U. Straccia. Extending datatype restrictions in fuzzy description logics. In *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications (ISDA 2009)*, pages 785–790, 2009.

[13] F. Bobillo and U. Straccia. Fuzzy description logics with general t-norms and datatypes. *Fuzzy Sets and Systems*, 160(23):3382–3402, 2009.

[14] F. Bobillo and U. Straccia. Fuzzy ontology representation using OWL 2. *International Journal of Approximate Reasoning*, 52(7):1073–1094, 2011.

[15] F. Bobillo and U. Straccia. fuzzyDL: An expressive fuzzy description logic reasoner. In *Proceedings of the 17th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*, pages 923–930, 2008.

[16] F. Bobillo and U. Straccia. Generalized fuzzy rough description logics. *Information Sciences*, 189(1):43–62, 2012.

[17] F. Bobillo and U. Straccia. On partitioning-based optimisations in expressive fuzzy description logics. In *Proceedings of the 24th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015)*, 2015.

[18] F. Bobillo and U. Straccia. Optimising fuzzy description logic reasoners with general concept inclusions absorption. *Fuzzy Sets and Systems*, In press.

[19] F. Bobillo and U. Straccia. Reasoning with the finitely many-valued Łukasiewicz fuzzy description logic $\mathcal{SROIQ}$. *Information Sciences*, 181(4):758–778, 2011.

[20] S. Borgwardt, F. Distel, and R. Peñaloza. The limits of decidability in fuzzy description logics with general concept inclusions. *Artificial Intelligence*, 218:23–55, 2015.

[21] S. Borgwardt and R. Peñaloza. The complexity of lattice based fuzzy description logics. *Journal on Data Semantics* 2(1):1–19, 2013.

[22] S. Bragaglia, F. Chesani, A. Ciampolini, P. Mello, M. Montali, and D. Sottara. An hybrid architecture integrating forward rules with fuzzy ontological reasoning. In *Proceedings of the 5th International Conference on Hybrid Artificial Intelligence Systems (HAIS 2010), Part I*, pages 438–445, 2010.

[23] S. Calegari and 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(13):2171–2185, 2008.

[24] D. Calvanese, G. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[25] C. Carlsson, M. Brunelli, and J. Mezei. Decision making with a fuzzy ontology. *Soft Computing*, 16(7):1143–1152, 2012.

[26] M. Cerami and U. Straccia. On the (un)decidability of fuzzy description logics under Łukasiewicz t-norm. *Information Sciences*, 227:1–21, 2013.

[27] R.-C. Chen, C. T. Bau, and C.-J. Yeh. Merging domain ontologies based on the WordNet system and fuzzy formal concept analysis techniques. *Applied Soft Computing*, 11(2):1908–1923, 2011.

[28] G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5:131–295, 1953.

[29] P. Cimiano, P. Haase, Q. Ji, T. Mailis, G. B. Stamou, G. Stoilos, T. Tran, and V. Tzouvaras. Reasoning with large A-Boxes in fuzzy description logics using DL reasoners: An experimental valuation. In *Proceedings of the 1st Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (ARea 2008)*, volume 350. CEUR Workshop Proceedings, 2008.

[30] P. C. G. Costa, K. B. Laskey, and T. Lukasiewicz. Uncertainty representation and reasoning in the semantic web. In *Semantic Web Engineering in the Knowledge Society*, pages 315–340. IGI Global, 2008.

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

[32] S. Dasiopoulou, I. Kompatsiaris, and M. G. Strintzis. Applying fuzzy DLs in the extraction of image semantics. *Journal of Data Semantics*, XIV:105–132, 2009.

[33] S. Dasiopoulou, I. Kompatsiaris, and M. G. Strintzis. Investigating fuzzy DLs-based reasoning in semantic image analysis. *Multimedia Tools and Applications*, 46(2):331–370, 2010.

[34] M. De Cock, C. Cornelis, and E. E. Kerre. Fuzzy rough sets: The forgotten step. *IEEE Transactions on Fuzzy Systems*, 15(1):121–130, 2007.

[35] N. Díaz-Rodríguez, M. Pegalajar-Cuéllar, J. Lilius, and M. Delgado. A fuzzy ontology for semantic modelling and recognition of human behaviour. *Knowledge-Based Systems*, 66:46–60, 2014.

[36] M. Eich, R. Hartanto, S. Kasperski, S. Natarajan, and J. Wollenberg. Towards coordinated multirobot missions for lunar sample collection in an unknown environment. *Journal of Field Robotics*, 31(1):35–74, 2014.

[37] C. Fernández. *Understanding image sequences: the role of ontologies in cognitive vision systems*. PhD thesis, Universitat Autònoma de Barcelona, Spain, 2010.

[38] C. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[39] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: An OWL 2 reasoner. *Journal of Automated Reasoning* 53 3:245–269, 2014.

[40] V. Haarslev, H.-I. Pai, and N. Shiri. Optimizing tableau reasoning in $\mathcal{ALC}$ extended with uncertainty. In *Proceedings of the 20th International Workshop on Description Logics (DL 2007)*, volume 250, pages 307–314. CEUR Workshop Proceedings, 2007.

[41] H. Habiballa. Resolution strategies for fuzzy description logic. In *Proceedings of the 5th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2007)*, volume 2, pages 27–36, 2007.

[42] P. Hájek. *Metamathematics of Fuzzy Logic*, volume 4 of *Trends in Logic*. Kluwer, 1998.

[43] P. Hájek. Making fuzzy description logic more general. *Fuzzy Sets and Systems*, 154(1):1–15, 2005.

[44] M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web Journal*, 2(1):11–21, 2011.

[45] J. Iglesias and J. Lehmann. Towards integrating fuzzy logic capabilities into an ontology-based inductive logic programming framework. In *Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011)*, pages 1323–1328, 2011.

[46] G. J. Klir and B. Yuan. *Fuzzy sets and fuzzy logic: theory and applications*. Prentice-Hall, Inc., 1995.

[47] S. Konstantopoulos and G. Apostolikas. Fuzzy-DL reasoning over unknown fuzzy degrees. In *Proceedings of the 3rd International Workshop on Semantic Web and Web Semantics (SWWS 07), Part II*, volume 4806 of *Lecture Notes in Computer Science*, pages 1312–1318. Springer-Verlag, 2007.

[48] S. Konstantopoulos and A. Charalambidis. Formulating description logic learning as an inductive logic programming task. In *Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010)*, pages 1–7. IEEE Press, 2010.

[49] S. Konstantopoulos, V. Karkaletsis, and D. Bilidas. An intelligent authoring environment for abstract semantic representations of cultural object descriptions. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education (LaTeCHSHELT&R 2009)*, pages 10–17, 2009.

[50] C.-S. Lee, Z.-W. Jian, and L.-K. Huang. A fuzzy ontology and its application to news summarization. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 35(5):859–880, 2005.

[51] C.-S. Lee, M. H. Wang, and H. Hagras. A type-2 fuzzy ontology and its application to personal diabetic-diet recommendation. *IEEE Transactions on Fuzzy Systems*, 18(2):374–395, 2010.

[52] I. A. Letia and A. Groza. Modelling imprecise arguments in description logic. *Advances in Electrical and Computer Engineering*, 9(3):94–99, 2009.

[53] F. A. Lisi and U. Straccia. A logic-based computational method for the automated induction of fuzzy ontology axioms. *Fundamenta Informaticae*, 124(4):503–519, 2013.

[54] C. Liu, D. Liu, and S. Wang. Situation modeling and identifying under uncertainty. In *Proceedings of the 2nd Pacific-Asia Conference on Circuits, Communications and System (PACCS 2010)*, pages 296–299, 2010.

[55] C. Liu, D. Liu, and S. Wang. Fuzzy geospatial information modeling in geospatial semantic retrieval. *Advances in Mathematical and Computational Methods*, 2(4):47–53, 2012.

[56] T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6(4):291–308, 2008.

[57] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7:1–13, 1975.

[58] C. Martínez-Cruz, A. van der Heide, D. Sánchez, and G. Triviño. An approximation to the computational theory of perceptions using ontologies. *Expert Systems with Applications*, 39(10):9494-9503, 2012.

[59] C. Martínez-Cruz, C. Porcel, J. Bernabé-Moreno, and E. Herrera-Viedma. A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling. *Information Sciences*, 311:102-118, 2015.

[60] P. S. Mostert and A. L. Shields. On the structure of semigroups on a compact manifold with boundary. *Annals of Mathematics*, 65(1):117–143, 1957.

[61] OWL 2 Web Ontology Language Document Overview. *http://www.w3.org/TR/2009/REC-owl2-overview-20091027/*. W3C, 2009.

[62] J. Z. Pan, E. Thomas, Y. Ren, and S. Taylor. Exploiting tractable fuzzy and crisp reasoning in ontology applications. *IEEE Computational Intelligence Magazine*, 7(2):45–53, 2012.

[63] Z. Pawlak. Rough sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

[64] I. J. Pérez, R. Wikström, J. Mezei, C. Carlsson, and E. Herrera-Viedma. A new consensus model for group decision making using fuzzy ontology. *Soft Computing*, 17(9):1617–1627, 2013.

[65] T. T. Quan, S. C. Hui, and A. C. M. Fong. Automatic fuzzy ontology generation for semantic help-desk support. *IEEE Transactions on Industrial Informatics*, 2(3):155–164, 2006.

[66] T. T. Quan, S. C. Hui, A. C. M. Fong, and T. H. Cao. Automatic fuzzy ontology generation for semantic web. *IEEE Transactions on Knowledge and Data Engineering*, 18(6):842–856, 2006.

[67] A. M. Radzikowska and E. E. Kerre. A comparative study of fuzzy rough sets. *Fuzzy Sets and Systems*, 126(2):137–155, 2002.

[68] A. Ragone, U. Straccia, F. Bobillo, T. D. Noia, and E. D. Sciascio. Fuzzy bilateral matchmaking in e-marketplaces. In *Proceedings of the 12th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2008), Part III*, volume 5179 of *Lecture Notes in Computer Science*, pages 293–301. Springer-Verlag, 2008.

[69] J. A. Rodger. A fuzzy linguistic ontology payoff method for aerospace real options valuation. *Expert Systems with Applications*, 40(8), 2013.

[70] H. M. Salkin and K. Mathur. *Foundations of Integer Programming*. North-Holland, 1989.

[71] E. Sanchez, editor. *Fuzzy Logic and the Semantic Web*, volume 1 of *Capturing Intelligence*. Elsevier Science, 2006.

[72] N. Simou, T. P. Mailis, G. Stoilos, G. B. Stamou. Optimization techniques for fuzzy description logics. In *Proceedings of the 23rd International Workshop on Description Logics (DL 2010)*, volume 573. CEUR Workshop Proceedings, 2010.

[73] V. Slavícek. An ontology-driven fuzzy workflow system. In *Proceedings of the 39th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2013)*, volume 7741 of *Lecture Notes in Computer Science*, pages 515–527. Springer, 2013.

[74] S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.

[75] G. Stoilos, N. Simou, G. Stamou, and S. Kollias. Uncertainty and the semantic web. *IEEE Intelligent Systems*, 21(5):84–87, 2006.

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

[77] U. Straccia. A fuzzy description logic for the semantic web. In *Fuzzy Logic and the Semantic Web*, Capturing Intelligence, chapter 4, pages 73–90. Elsevier, 2006.

[78] U. Straccia. A minimal deductive system for general fuzzy RDF. In *Proceedings of the 3rd International Conference on Web Reasoning and Rule Systems (RR 2009)*, number 5837 in Lecture Notes in Computer Science, pages 166–181. Springer-Verlag, 2009.

[79] U. Straccia. Multi-criteria decision making in fuzzy description logics: A first step. In *Proceedings of the 13th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems (KES 2009)*, volume 5711 of *Lecture Notes in Artificial Intelligence*, pages 79–87. Springer-Verlag, 2009.

[80] U. Straccia. An ontology mediated multimedia information retrieval system. In *Proceedings of the the 40th International Symposium on Multiple-Valued Logic (ISMVL 2010)*, pages 319–324, 2010.

[81] U. Straccia. *Foundations of Fuzzy Logic and Semantic Web Languages*. CRC Studies in Informatics Series. Chapman & Hall, 2013.

[82] U. Straccia. Reasoning in Ł-$\mathcal{SHIF}$: an expressive fuzzy description logic under Łukasiewicz semantics. Technical report TR-2007-10-18, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy, 2007.

[83] U. Straccia. Reasoning within fuzzy Description Logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.

[84] U. Straccia. Softfacts: A top-k retrieval engine for ontology mediated access to relational databases. In *Proceedings of the 2010 IEEE International Conference on Systems, Man and Cybernetics (SMC 2010)*, pages 4115–4122. IEEE Press, 2010.

[85] U. Straccia and F. Bobillo. Mixed integer programming, general concept inclusions and fuzzy description logics. *Mathware & Soft Computing*, 14(3):247–259, 2007.

[86] M. Sugeno. *Theory of fuzzy integrals and its applications*. PhD thesis, Tokyo Institute of Technology, Japan, 1974.

[87] E. Thomas, J. Z. Pan, and Y. Ren. TrOWL: Tractable OWL 2 reasoning infrastructure. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 2, pages 431–435, 2010.

[88] K. Todorov, C. Hudelot, A. Popescu, and P. Geibel. Fuzzy ontology alignment using background knowledge. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 22(1):75–112, 2014.

[89] V. Torra and Y. Narukawa. *Modeling decisions - Information fusion and aggregation operators*. Springer, 2007.

[90] D. Tsatsou, S. Dasiopoulou, I. Kompatsiaris, and V. Mezaris. LiFR: A lightweight fuzzy DL reasoner. In *Proceedings of the 11th Extended Semantic Web Conference (ESWC 2014), Posters and Demo sessions*, 2014.

[91] D. Tsatsou, F. Menemenis, I. Kompatsiaris, and P. C. Davis. A semantic framework for personalized ad recommendation based on advanced textual analysis. In *Proceedings of the 3rd ACM conference on Recommender systems (RecSys 2009)*, pages 217–220, 2009.

[92] W. Van Leekwijck and E. E. Kerre. Defuzzification: Criteria and classification. *Fuzzy Sets and Systems*, 108(2):159–178, 1999.

[93] M. Wallace. Ontologies and soft computing in flexible querying. *Control and Cybernetics*, 38(2):481–507, 2009.

[94] H. Wang, Z. M. Ma, and J. Yin. Fresg: A kind of fuzzy description logic reasoner. In *Proceedings of the 20th International Conference on Database and Expert Systems Applications (DEXA 2009)*, volume 5690 of *Lecture Notes in Computer Science*, pages 443–450. Springer-Verlag, 2009.

[95] R. Wikström. *Fuzzy Ontology for Knowledge Mobilisation and Decision Support*. PhD thesis, Åbo Akademi, Turku, Finland, 2014.

[96] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.

[97] C. A. Yaguinuma, M. T. P. Santos, H. A. Camargo, and M. Reformat. A FML-based hybrid reasoner combining fuzzy ontology and Mamdani inference. In *Proceedings of the 22nd IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2013)*, pages 1–8, 2013.

[98] J. Yen. Generalizing term subsumption languages to fuzzy logic. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991)*, pages 472–477. Morgan Kaufmann, 1991.

[99] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[100] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, 3(1):28–44, 1973.

A Appendix: Theoretical Background

In this appendix we will recall some basic background knowledge on fuzzy set theory and fuzzy logic (Section A.1), approximate reasoning (Section A.2), aggregation operators (Section A.3), and fuzzy rough set theory (Section A.4).

A.1 Fuzzy Set Theory and Fuzzy Logic

Fuzzy set theory and fuzzy logic were proposed by Zadeh [99] to manage imprecise and vague knowledge. While in classical set theory elements either belong to a set or not, in fuzzy set theory elements can belong to some degree. More formally, 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 to every $x \in X$ a degree of truth, measured as a value in a truth space L. The truth space is usually $L = [0, 1]$, but another popular choice is the finite truth space $\mathcal{D} = \{0, \frac{1}{n-1}, \ldots, \frac{n-2}{n-1}, 1\}$ for some natural number $n > 1$. Of course, L_2 is the classical two-valued case. 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 A.

Fuzzy logics provide compositional calculi of degrees of truth. The conjunction, disjunction, complement and implication 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. For a formal definition of these functions we refer the reader to [42, 46].

A quadruple composed by a t-norm, a t-conorm, an implication function and a negation function determines a *fuzzy logic*. One usually distinguishes three fuzzy logics, namely Łukasiewicz, Gödel, and Product [42], due to the fact that any continuous t-norm can be obtained as a combination of Łukasiewicz, Gödel, and Product t-norm [60]. It is also usual to consider Zadeh logic, including the conjunction, disjunction, and negation originally proposed by Zadeh [99] together with Kleene-Dienes implication defined as $\alpha \Rightarrow_{KD} \beta = \max(1 - \alpha, \beta)$. The name of Zadeh fuzzy logic is used following the tradition in the setting of fuzzy DLs, even if it might lead to confusion because the logic does not include the sometimes called *Zadeh implication* (because it corresponds to Zadeh's fuzzy set inclusion) or Rescher implication, denoted $\Rightarrow_Z$ and defined as:

$$\alpha \Rightarrow_Z \beta = \begin{cases} 1, & \text{if } \alpha \leq \beta \\ 0 & \text{if } \alpha > \beta \ . \end{cases}$$

Table 1 summarises the fuzzy operators in the families supported by *fuzzyDL*, namely Zadeh, Łukasiewicz, and Gödel. We will often use the subscripts $Ł, G$ and Z to indicate that an operator belongs to Łukasiewicz, Gödel, and Zadeh fuzzy logics, respectively.

Fuzzy numbers are a generalisation of real numbers. A fuzzy number is a convex and normalised fuzzy set of the real line. Fuzzy numbers can be defined using some of the functions in Figure 5. However, the triangular function is used most of the times.

Fuzzy modifiers apply to fuzzy sets to change their membership function. Examples are very, moreOrLess, and slightly. Formally, a *modifier* is a function $f_m \colon [0, 1] \to [0, 1]$. We will allow modifiers defined in terms of triangular functions (Figure 5 (b)) and linear hedges (Figure 5 (e)). For instance, the modifier very can be defined as a linear function with parameters $q1 = 0.7$ and $q2 = 0.3$.

Relations can also be extended by considering fuzzy subsets of the Cartesian product over some reference sets. A (binary) *fuzzy relation* R over two countable sets X and Y is a function $R \colon X \times Y \to [0, 1]$. Fuzzy relations can be trivially extended to the n-ary case. Several properties of the relations (such as reflexive, irreflexive, symmetric, asymmetric, transitive, or disjoint with another relation) and operations (inverse, composition) can easily be extended to the fuzzy case.

A.2 Approximate Reasoning

One of the most important features of fuzzy logic is its ability to perform *approximate reasoning* [100], which involves inference rules with fuzzy propositions. A

very popular formalism, due to their practical success, are *fuzzy rule-based systems*. A *fuzzy IF-THEN system* consists in a rule base (a set of IF-THEN rules) and a reasoning algorithm performing an inference mechanism. In general, the input of the system are the current values for the input variables and the output is a fuzzy set, which can be defuzzified into a single value. In a fuzzy IF-THEN rule, its antecedents, consequents or both are fuzzy. Fuzzy IF-THEN rules are fired to a degree which is a function of the degree of matching between their antecedent and the input. The deduction rule is *Generalised Modus Ponens*. Roughly speaking, given a rule "IF A THEN B", where A and B are fuzzy propositions, it is possible from a premise A' which matches A to some degree, to deduce B', which is similar to B.

One of the most popular IF-THEN systems is the *Mamdani* model [57]. In a Mamdani model, fuzzy rules have the form

$$\text{IF } x_1 \text{ IS } A_1 \text{ AND } \ldots \text{ AND } x_k \text{ IS } A_k \text{ THEN } y \text{ IS } B \ ,$$

where X_i and Y are reference sets, x_i and y are variables taking values in X_i and Y, respectively, and A_i and B are linguistic values defined by fuzzy sets over X_i and Y, respectively, for each $i \in \{1, \ldots, k\}$.

For every clause in the antecedent of the rule, the matching degree between the current value of the variable and the linguistic label in the rule is computed (typically, using the minimum or another t-norm). If there exist several clauses, they are aggregated into a firing degree, using a fuzzy logic operator (typically, the maximum or another t-conorm). Then, this firing degree is used for modifying the consequent of the rule using some function (typically the minimum). Rules are fired using some inference algorithm such as Rete [38]. The computed consequences related to the same variable are aggregated (typically, using the maximum).

Finally, the output variables can be defuzzified. *Defuzzification* is a function that given a fuzzy set B defined over a reference set Y returns a value in Y [92]. Some examples of defuzzification methods are the largest of maxima (LOM), smallest of maxima (SOM), and the middle of maxima (MOM), defined next:

- $x_{LOM} \in Y$ is the LOM iff $\mu_B(x_{LOM}) \geq \mu_B(y)$ and, if $\mu_B(x_{LOM}) = \mu_B(y)$ then $x_{LOM} > y$, for all $y \in Y$ such that $y \neq x_{LOM}$.
- $x_{SOM} \in Y$ is the SOM iff $\mu_B(x_{SOM}) \geq \mu_B(y)$ and, if $\mu_B(x_{SOM}) = \mu_B(y)$ then $x_{SOM} < y$, for all $y \in Y$ such that $y \neq x_{SOM}$.
- $x_{MOM} \in Y$ is the MOM iff $x_{MOM} = (x_{LOM} + x_{SOM})/2$.

LOM and SOM require that Y is an ordered set, whereas MOM requires that Y is the set of the rational numbers.

A.3 Aggregation Operators

Aggregation Operators (AOs) are mathematical functions that are used to combine different pieces of information [89]. Given a domain $\mathbb{D}$ (such as the reals),

an AO of dimension k is a mapping $@ : \mathbb{D}^k \to \mathbb{D}$. For us, $\mathbb{D} = [0, 1]$. Thus, an AO aggregates k values of k different criteria. In our scenario, such criteria will be represented by using fuzzy concepts from a fuzzy ontology. Some examples of AOs are *maximum*, *minimum*, *order statistic*, *arithmetic mean*, *weighted sum* (or weighted mean) $@_W^{\mathrm{ws}}$, strict weighted sum ($@_W^{\mathrm{wsz}}$), *weighted maximum* $@_W^{\mathrm{wmax}}$, *weighted minimum* $@_W^{\mathrm{wmin}}$, and *median*.

We will also use σ to denote a permutation such that $x_{\sigma(1)} \geq x_{\sigma(2)} \geq \cdots \geq x_{\sigma(k)}$, i.e., $x_{\sigma(i)}$ is the i-th largest of the values $x_1, \ldots, x_k$. Often, an AO $@$ is parameterised with a vector of k weights $W = [w_1, \ldots, w_k]$ such that $w_i \in [0, 1]$ and $\sum_{i=1}^{k} w_i = 1$. In that case the AO is denoted as $@_W$. In other cases, AOs are parameterised with a vector of k weights $R = [r_1, \ldots, r_k]$ such that $r_i \in [0, 1]$ and $\max_{i=1}^{k} r_i = 1$.

Ordered Weighted Averaging (OWA) operators [96] are a parameterised class of mean type AO. An OWA operator of dimension n is denoted as $@_W^{\mathrm{owa}}$. By choosing different weights, OWA operators can implement different AOs, such as arithmetic mean, n-th maximum, n-th minimum, median or order statistic. For example, the maximum and the minimum can be obtained by using $W = [1, 0, \ldots, 0]$ and $W = [0, \ldots, 0, 1]$, respectively. However, weighted sum, maximum, and minimum cannot be represented as OWA operators. Table 4 includes the definition of some AOs, where $\ominus$ is a negation function.

The problem of obtaining the weights of the OWA operators has been largely studied. A possible idea is using *quantifier-guided aggregation* by means of *Regular Increasing Monotone* (RIM) quantifiers Q that satisfy the boundary conditions $Q(0) = 0$, $Q(1) = 1$ and are monotone increasing, i.e., $Q(x_1) \leq Q(x_2)$ when $x_1 \leq x_2$. Essentially, these quantifiers are characterised by the idea that as the proportion increases, the degree of satisfaction does not decrease. A RIM Q can be used to define an OWA weighting vector W_Q of dimension k, called *quantifier-guided OWA*, as

$$w_i = Q(\tfrac{i}{k}) - Q(\tfrac{i-1}{k}) \quad .$$

We have considered so far AOs where the criteria are mutually independent. *Fuzzy integrals* are a more general class of AOs that make it possible to measure the importance of sets of criteria [89]. 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. In our setting, the values $x_1, \ldots, x_k$ that an AO has to aggregate are represented by using fuzzy concepts $C_1, \ldots, C_k$ from a fuzzy ontology. Aggregating the evaluations of the individual criteria into an overall evaluation can be done by considering the fuzzy integral of the function f with respect to a fuzzy measure μ. We can parameterise the integrals using a vector W where $w_i = \mu\{x_i\}$.

We will consider three types of fuzzy integrals of a function $f : X \to [0, 1]$: Choquet [28] (denoted $@_W^{\mathrm{ci}}(f)$), Sugeno [86] ($@_W^{\mathrm{si}}(f)$), and quasi-Sugeno ($@_W^{\mathrm{qsi}}(f)$) integrals. Choquet integral is more general than other AOs, such as weighted sum and OWA. Sugeno integral generalises some AOs, such as weighted minimum and weighted maximum. Quasi-Sugeno integral is more

general than the Sugeno integral. These integrals are defined in Table 4, where $A_{\sigma(i)} = \{x_{\sigma(1)}, \ldots, x_{\sigma(i)}\}$ for $i \in \{1, \ldots, k\}$, $A_{\sigma(0)} = 0$, and $f(x_{\sigma(0)}) = 0$.

Table 4: Some examples of aggregation operators.

Aggregation operator	Definition
Maximum	$x_{\sigma(1)}$
Minimum	$x_{\sigma(k)}$
n-th order statistic	$x_{\sigma(n)}$
Arithmetic mean	$\frac{1}{k}\sum_{i=1}^{k} x_i$
OWA	$\sum_{i=1}^{k} w_i x_{\sigma(i)}$
Weighted sum	$\sum_{i=1}^{k} w_i x_i$
Strict weighted sum	$\begin{cases} 0, & \text{if some } x_i = 0 \\ \sum_{i=1}^{k} w_i x_i, & \text{otherwise} \end{cases}$
Weighted maximum	$\max_{i=1}^{k} \min\{v_i, x_i\}$
Weighted minimum	$\min_{i=1}^{k} \max\{\ominus v_i, x_i\}$
Fuzzy integral	**Definition**
Choquet integral	$\sum_{i=1}^{k} f(x_{\sigma(i)})\Big[\mu(A_{\sigma(i)}) - \mu(A_{\sigma(i-1)})\Big]$
Sugeno integral	$\max_{i=1}^{k}\{\min\{f(x_{\sigma(i)}), \mu(A_{\sigma(i)})\}\}$
Quasi-Sugeno integral	$\max_{i=1}^{k}\{f(x_{\sigma(i)}) \otimes \mu(A_{\sigma(i)})\}$

A.4 Rough and Fuzzy Rough Sets

Fuzzy set theory offers a quantitative way to manage vagueness, where it is expressed using degrees of membership to fuzzy concepts. On the contrary, rough set theory offers a qualitative approach to model vagueness. Instead of providing a degree of membership, a vague concept is approximated by using a rough set. This approach is very useful when it is not possible to quantify the membership function of a vague concept.

The notion of rough set was introduced by Z. Pawlak in 1982 [63]. The key idea in rough set theory is the approximation of a vague concept when there is only incomplete information about the concept. More precisely, there are only some examples of elements that belong to the concept, and an indiscernibility equivalence (reflexive, symmetric, and transitive) or similarity (reflexive and symmetric) relation between elements of the domain. In this case, a vague concept is approximated by means of a pair of concepts: a sub-concept or lower approximation, and a super-concept or upper approximation. The *lower approximation* describes the sets of elements that definitely belong to the vague set, but it can be incomplete and not include some of the elements of the vague set. This set is formed by those elements such that all of their indistinguishable elements belong to the vague concept. The *upper approximation* describes the sets of elements that possibly belong to the vague set, but it can include elements

that might not actually belong to the vague set. This set is formed by those elements having some indistinguishable element belonging to the vague concept. A *rough set* is then defined as a pair of concepts: a lower approximation and an upper approximation of a vague concept.

Fuzzy logic and rough logic are complementary formalisms to manage vagueness and hence it is natural to combine them. A very natural extension to rough sets is to consider a fuzzy similarity relation instead of an indiscernibility relation, which gives raise to *fuzzy rough sets* [34, 67]. While in rough sets an element of the domain can only belong to one equivalence class of similar elements, in fuzzy rough sets one element can belong to several fuzzy similarity classes (with different degrees of truth). Thus, the notions of tight and loose approximation naturally appear: a *tight approximation* considers all the existing fuzzy similarity classes, whereas a *loose approximation* considers the best one among the similarity classes.

Given a fuzzy similarity relation s, a t-norm $\otimes$ and an implication function $\Rightarrow$, the lower approximation $s \downarrow A$, the upper approximation $s \uparrow A$, the tight lower approximation $s \downarrow\downarrow A$, the loose lower approximation $s \uparrow\downarrow A$, the tight upper approximation $s \downarrow\uparrow A$, and the loose upper approximation $s \uparrow\uparrow A$ of a fuzzy subset A of X are defined by the following membership functions:

$$
\begin{array}{lcl}
s \downarrow A(x) & = & \inf_{y \in X}\{s(x,y) \Rightarrow A(y)\} \\
s \uparrow A(x) & = & \sup_{y \in X}\{s(x,y) \otimes A(y)\} \\
s \downarrow\downarrow A(x) & = & \inf_{z \in X}\{s(x,z) \Rightarrow \inf_{y \in X}\{s(z,y) \Rightarrow A(y)\}\} \\
s \uparrow\downarrow A(x) & = & \sup_{z \in X}\{s(x,z) \otimes \inf_{y \in X}\{s(z,y) \Rightarrow A(y)\}\} \\
s \downarrow\uparrow A(x) & = & \inf_{z \in X}\{s(x,z) \Rightarrow \sup_{y \in X}\{s(z,y) \otimes A(y)\}\} \\
s \uparrow\uparrow A(x) & = & \sup_{z \in X}\{s(x,z) \otimes \sup_{y \in X}\{s(z,y) \otimes A(y)\}\}
\end{array}
$$

B Appendix: User Manual

This appendix provides details for the advanced users of *fuzzyDL*, as a user manual of the reasoner. The three different interfaces to interact with *fuzzyDL*, the command line interpreter, a Protégé plug-in, and a Java API, are covered in Sections B.1, B.2, and B.3, respectively. Eventually, Section B.4 explains some advanced issues of the reasoning algorithm.

B.1 fuzzyDL Syntax

We next present the fuzzy DL which is currently behind the *fuzzyDL* reasoner. It can be seen as a fuzzy extension of $\mathcal{SHIF}(D)$, with several elements that are not present in the classical case. Most of the elements of the logic have already been presented in [11, 16, 77, 81].

This section is organised as follows. We start by presenting the syntax of the supported language in Section B.1.1. Rather than using a DL-like syntax, we will use *fuzzyDL* syntax which is more easily readable for non-expert users. Next, Section B.1.2 details the elements of Fuzzy OWL 2 that *fuzzyDL* cannot

support yet. Then, we present the semantics in Section B.1.3 of the language. Finally, Section B.1.4 describes *fuzzyDL* syntax to represent the reasoning tasks.

B.1.1 Fuzzy Ontology Syntax

In the following, we will define the syntax of the fuzzy DL language supported by *fuzzyDL*. We will start with the syntax of fuzzy concepts. Then, we will focus on the syntax of elements that are required to define fuzzy concepts: degrees of truth, data values, datatypes, modifiers, and quantifiers. Finally, we will define fuzzy knowledge bases. Note that roles are always atomic.

Before defining the syntax, we need to introduce some notation for referring to the main ontology elements. We use `A` to denote fuzzy atomic concepts (or fuzzy *concept names*), `C` for any fuzzy concept, `R` for fuzzy object properties, `T` for fuzzy data properties, `r` for fuzzy object features, `t` for fuzzy data features, `o` for abstract individuals, `v` for concrete values, and `d` for fuzzy datatypes. Furthermore, the syntax of fuzzy ontologies requires some additional notation for the elements specific of the fuzzy case. We use `m` for fuzzy modifiers, `F` for fuzzy numbers, `Q` for fuzzy quantifiers, `D` for degrees of truth, and `x` for variables.

Concepts The syntax of fuzzy $\mathcal{SHIF}$ concepts is as follows:

(C1)	`A`	atomic concept
(C2)	`*top*`	top concept
(C3)	`*bottom*`	bottom concept
(C4)	`(and C1 C2 ... Ck)`	concept conjunction
(C5)	`(or C1 C2 ... Ck)`	concept disjunction
(C6)	`(not C)`	negation
(C7)	`(some R C)`	object existential restriction
(C8)	`(all R C)`	object universal restriction

Example B.1 *The concept* `(and Human (some hasChild Male))` *will denote the set of humans with a son, while* `(and Human (all hasChild Male))` *denotes the set of humans such that if they have children then they all must be male.*

Fuzzy $\mathcal{SHIF}(\mathbf{D})$ extends $\mathcal{SHIF}$ with concrete data types, i.e., it has the additional concept constructs:

(C9)	`(some T d)`	data existential restriction
(C10)	`(all T d)`	data universal restriction
(C11)	`(some t {v})`	data value restriction
(C12)	`(some t [>= v])`	data minimal value restriction
(C13)	`(some t [<= v])`	data maximal value restriction

Example B.2 *The concept* `(and Human (some hasAge [<= 18]))` *denotes the set of humans with an age less or equal than* 18*, while the concept expression* `(and Human (some hasAge (left 0 100 10 30) ))` *denotes the fuzzy set of*

young humans with an age defined using a left-shoulder function, to be defined below.

The syntax of concepts is extended to cover some of the ones in the more expressive DL $\mathcal{SROIQ}(\mathbf{D})$. *fuzzyDL* currently supports two of those concepts:

(C14)	`(some R {o})`	object value restriction
(C15)	`(self R)`	local reflexivity concept

Example B.3 `(some hasFriend {bob})` *denotes the fuzzy set of people having a friend called Bob, while* `(self shave)` *denotes the set of people that shave themselves.*

Table 5 shows how to express concepts (C1)–(C15) using OWL 2 and DL syntaxes.

Table 5: *fuzzyDL* concepts (C1)–(C15) in OWL 2 and DL syntaxes.

fuzzyDL concept	OWL 2 abstract syntax	DL syntax
(C1)	Class (A)	A
(C2)	Class (owl:Thing)	$\top$
(C3)	Class (owl:Nothing)	$\bot$
(C4)	ObjectIntersectionOf $(C_1, \ldots, C_k)$	$C_1 \sqcap \cdots \sqcap C_k$
(C5)	ObjectUnionOf $(C_1, \ldots, C_k)$	$C_1 \sqcup \cdots \sqcup C_k$
(C6)	ObjectComplementOf (C)	$\neg C$
(C7)	ObjectSomeValuesFrom (R, C)	$\exists R.C$
(C8)	ObjectAllValuesFrom (R, C)	$\forall R.C$
(C9)	DataSomeValuesFrom (T, d)	$\exists T.d$
(C10)	DataAllValuesFrom (T, d)	$\forall T.d$
(C11)	DataHasValue (t, v)	$\exists t.\{v\}$
(C12)	DataSomeValuesFrom $(t, [\geq v])$	$\exists t.[\geq v]$
(C13)	DataSomeValuesFrom $(t, [\leq v])$	$\exists t.[\leq v]$
(C14)	ObjectHasValue (R, o)	$\exists R.\{o\}$
(C15)	ObjectExistsSelf (R)	$\exists R.\texttt{Self}$

So far, every fuzzy concept is a direct extension of a similar concept in the classical case (the remaining fuzzy concepts will be specific from the fuzzy case):

(C16)	`(g-and C1 ... Ck)`	Gödel concept conjunction
(C17)	`(l-and C1 ... Ck)`	Łukasiewicz concept conjunction
(C18)	`(g-or C1 ... Ck)`	Gödel concept disjunction
(C19)	`(l-or C1 ... Ck)`	Łukasiewicz concept disjunction
(C20)	`(implies C1 C2)`	concept implication
(C21)	`(g-implies C1 C2)`	Gödel concept implication
(C22)	`(l-implies C1 C2)`	Łukasiewicz concept implication
(C23)	`(m C)`	modified fuzzy concept
(C24)	`(C [>= D])`	minimum threshold concept
(C25)	`(C [<= D])`	maximum threshold concept

Example 3.1 illustrates the use of the new conjunction concepts and modified fuzzy concepts. Example B.4 illustrates the definition of a threshold concept.

Example B.4 `(YoungHuman [>= 0.8])` *represents the set of people that belong to the fuzzy set of young humans with degree greater or equal than* 0.8.

Data value restrictions can be extended to the fuzzy case by replacing a single value with a real valued variable or a fuzzy number `F` (defined later on):

(C26)	`(some t {x})`	data value restriction
(C27)	`(some t {F})`	data value restriction
(C28)	`(some t [>= x])`	data minimal value restriction
(C29)	`(some t [>= F])`	data minimal value restriction
(C30)	`(some t [<= x])`	data maximal value restriction
(C31)	`(some t [<= F])`	data maximal value restriction

Next, *fuzzyDL* supports fuzzy rough concepts defined using upper and lower approximation values according to a fuzzy similarity relation (a reflexive and symmetric object property) `s`:

(C32)	`(la s C)`	lower approximation fuzzy rough concept
(C33)	`(tla s C)`	tight lower approximation fuzzy rough concept
(C34)	`(lla s C)`	loose lower approximation fuzzy rough concept
(C35)	`(ua s C)`	upper approximation fuzzy rough concept
(C36)	`(tua s C)`	tight upper approximation fuzzy rough concept
(C37)	`(lua s C)`	loose upper approximation fuzzy rough concept

To conclude the syntax of fuzzy concepts, *fuzzyDL* supports some aggregation concepts defined using a fuzzy quantifier Q or two types of weights such that $\texttt{wi} \in [0,1]$ with $\sum_{i=1}^{k} \texttt{wi} = 1$, and $\texttt{ri} \in [0,1]$ with $\max_{i=1}^{k} \texttt{ri} = 1$:

(C38)	`(w-sum (w1 C1) ... (wk Ck))`	weighted sum concept
(C39)	`(w-sum-zero (w1 C1) ... (wk Ck))`	strict weighted sum concept
(C40)	`(owa (w1 ... wk) (C1 ... Ck))`	OWA concept
(C41)	`(q-owa Q C1 ... Ck)`	quantifier-guided OWA concept

(C42)	`(choquet (w1 ... wk) (C1 ... Ck))`	Choquet integral concept
(C43)	`(sugeno (r1 ... rk) (C1 ... Ck))`	Sugeno integral concept
(C44)	`(q-sugeno (r1 ... rk) (C1 ... Ck))`	Quasi-Sugeno integral concept
(C45)	`(w-max (r1 C1) ... (rk Ck))`	weighted maximum concept
(C46)	`(w-min (r1 C1) ... (rk Ck))`	weighted minimum concept

Example 3.1 illustrates the use of fuzzy rough concepts and aggregation concepts.

After having defined the syntax of fuzzy concepts, it remains to define the syntax of some elements that have already appeared in the text, but have been left unspecified so far.

Degrees of truth In the literature of fuzzy DLs, some fuzzy concepts and fuzzy axioms can contain a numerical degree of truth expressing a lower bound for its satisfiability degree. In *fuzzyDL*, this is generalised and the degrees of truth `D` can be a rational number in $[0, 1]$, a variable `x` taking values in $[0, 1]$, or a linguistic label (as an alias for a defined degree). Whenever a degree of truth is omitted, the maximal degree 1 is assumed. At the time of writing, variables cannot occur in the TBox or in the RBox.

Datatypes and data values Data values are used in fuzzy concepts (C11)–(C13). Currently, *fuzzyDL* supports the following types of classical data values `v`: integers, reals, strings, booleans, and dates.

Furthermore, datatypes are used in fuzzy concepts (C9)–(C10). *fuzzyDL* supports fuzzy datatypes defined over a dense total ordered concrete domain in $[\mathtt{k1}, \mathtt{k2}]$ according to the following functions, where `m` is a fuzzy modifier, `d` is a fuzzy datatype, and the parameters $\mathtt{q1}, \mathtt{q2}, \mathtt{q3}, \mathtt{q4}$ are rational numbers as illustrated in Figure 5:

(D1)	`(crisp k1 k2 q1 q2)`	crisp membership function
(D2)	`(left k1 k2 q1 q2)`	left-shoulder membership function
(D3)	`(right k1 k2 q1 q2)`	right-shoulder membership function
(D4)	`(triangular k1 k2 q1 q2 q3)`	triangular membership function
(D5)	`(trapezoidal k1 k2 q1 q2 q3 q4)`	trapezoidal membership function
(D6)	`(modified m d)`	modified fuzzy datatype

Note that classical integer or real values can be represented using a crisp membership function.

Fuzzy modifiers Fuzzy modifiers are used to form fuzzy concepts (C23) and fuzzy modified datatypes (D6). As fuzzy modifiers, *fuzzyDL* supports linear and triangular functions. However, now $\mathtt{k1} = 0$ and $\mathtt{k2} = 1$ (so these parameters are not needed), and $\mathtt{q1}, \mathtt{q2}, \mathtt{q3} \in [0, 1]$. Formally, the syntax is:

(M1)	`(triangular q1 q2 q3)`	triangular fuzzy modifier
(M2)	`(linear q1 q2)`	linear fuzzy modifier

Fuzzy quantifiers Fuzzy quantifiers appear in quantifier-guided OWA concepts (C41). Currently, *fuzzyDL* supports fuzzy quantifiers defined by means of right-shoulder and linear functions. As for fuzzy modifiers, $\texttt{k1} = 0$ and $\texttt{k2} = 1$ (so these parameters are not needed), and $\texttt{q1}, \texttt{q2} \in [0, 1]$. The formal syntax is:

(Q1)	`(right q1 q2)`	right-shoulder quantifier
(Q2)	`(linear q1 q2)`	linear quantifier

Fuzzy numbers Fuzzy numbers are used in extended value restrictions (C27), (C29), and (C31). The fuzzy numbers supported by *fuzzyDL* can be numerical, triangular membership functions, or arithmetical combinations of fuzzy numbers. Let $\texttt{F1}, \texttt{F2}$ be fuzzy numbers and $\texttt{q}, \texttt{q1}, \texttt{q2}, \texttt{q3} \in \mathbb{Q}$. The syntax is:

(F1)	`q`	rational number
(F2)	`(triangular q1 q2 q3)`	triangular fuzzy numbers
(F3)	`(F1 + F2)`	addition
(F4)	`(F1 - F2)`	subtraction
(F5)	`(F1 * F2)`	multiplication
(F6)	`(F1 / F2)`	division
(F7)	$\texttt{q} * \texttt{F}$	product by a constant

Knowledge bases In *fuzzyDL*, an *ABox* (Assertional Box) contains axioms about individuals (A1)–(A2), a *TBox* (Terminological Box) contains axioms about concepts (A3)–(A14), and an *RBox* $\mathcal{R}$ (Role Box) contains axioms about roles (A15)–(A21). Some of the axioms include a degree of truth `D` which should be understood as optional (if `D` is not specified, the maximal degree 1 is assumed).

The supported ABox axioms in *fuzzyDL* are similar to their crisp counterparts but could include a degree `D`:

(A1)	`(instance o C D)`	concept assertion
(A2)	`(related o1 o2 R D)`	role assertion

Example 3.2 illustrates the use of fuzzy concept and role assertions.
fuzzyDL TBoxes can include the following axioms:

(A3)	`(implies C1 C2 D)`	General Concept Inclusion (GCI)
(A4)	`(define-primitive-concept A C D)`	primitive concept inclusion
(A5)	`(equivalent-concepts C1 ... Ck)`	concept equivalence
(A6)	`(define-concept A C)`	atomic concept definition
(A7)	`(domain R C)`	object property domain axiom
(A8)	`(range R C)`	object property range axiom
(A9)	`(disjoint-concepts C1 ... Ck)`	disjoint concepts axiom
(A10)	`(disjoint-union C1 ... Ck)`	disjoint union of concepts

Example B.5 *Example 3.2 illustrates the use of a simple fuzzy GCI. The meaning of the fuzzy concept* `YoungPerson` *can be set with the definition* `(define-concept YoungPerson (and Human (some hasAge (left 0 100 10 30))))`. *The two axioms* `(domain hasDaughter Human)` *and* `(range hasDaughter Woman)` *guarantee that the relation* `hasDaughter` *links instances of* `Human` *with instances of* `Woman`. `(disjoint Man Woman)` *forces men and women to be disjoint (i.e., being impossible for anybody to belong to both classes). Moreover, the axiom* `(disjoint-union Human Man Woman)` *defines humans as the disjoint union of man and woman, forcing every human to belong to exactly one of the classes of men and women.*

It is worth noting that axioms (A4)–(A10) can be represented using GCIs and that the syntax of the language is not ambiguous even if the terms `implies` is used both in concepts and axioms. Note also that in Zadeh fuzzy DLs, the degree `D` in axiom (A3) is actually discarded because Zadeh implication can only take values in $\{0, 1\}$.

fuzzyDL also allows to specify in the GCIs a particular implication function (currently, Kleene-Dienes, Łukasiewicz, Gödel, or Zadeh) to be used in the semantics of the axiom.

(A11)	`(g-implies C1 C2 D)`	Gödel GCI
(A12)	`(kd-implies C1 C2 D)`	Kleene-Dienes GCI
(A13)	`(l-implies C1 C2 D)`	Łukasiewicz GCI
(A14)	`(z-implies C1 C2)`	Zadeh GCI

fuzzyDL RBoxes can include the following axioms:

(A15)	`(implies-role R1 R2 D)`	object property inclusion axiom
(A16)	`(transitive R)`	transitive object property axiom
(A17)	`(inverse R1 R2)`	inverse object property axioms
(A18)	`(functional r)`	object feature declaration axiom
(A19)	`(inverse-functional R)`	inverse-functional object property axiom
(A20)	`(symmetric R)`	symmetric object property axiom
(A21)	`(reflexive R)`	reflexive object property axiom
(A22)	`(functional t)`	data feature declaration axiom
(A23)	`(range t *integer* k1 k2)`	range of data feature axiom
(A24)	`(range t *real* k1 k2)`	range of data feature axiom
(A25)	`(range t *string*)`	range of data feature axiom
(A26)	`(range t *boolean*)`	range of data feature axiom

Furthermore, concepts (C15) and axioms (A18)–(A19) require roles to be *simple*, which means that they cannot be transitive or have transitive sub object properties. Axioms (A15)–(A19) can be used in classical $\mathcal{SHIF}(\mathbf{D})$ with the only difference that *fuzzyDL* makes it possible to specify a degree of truth `D` in (A15). (A21) can be used in more expressive classical DLs. (A22) together with one of (A23)–(A26) are used in order to declare a data feature and its related concrete domain.

Example B.6 `(implies-role hasSon hasChild)` *states that* `hasSon` *is a sub-property of* `hasChild`. `(inverse hasChild hasParent)` *states that if one individual is related via* `hasChild` *with another individual, then the latter one is related via* `hasParent` *with the former one.* `(inverse-functional hasMother)` *specifies that the inverse of* `hasMother` *is functional (everybody has exactly one biological mother).*

Table 6 shows how to express axioms using OWL 2 and DL syntaxes. Axioms (A11)–(A14) are not included because they do not have a direct equivalence in the classical case.

Table 6: *fuzzyDL* axioms in OWL 2 and DL syntaxes.

fuzzyDL axiom	OWL 2 abstract syntax	DL syntax
(A1)	ClassAssertion (o, C)	$o\colon C$
(A2)	ObjectPropertyAssertion (R, o_1, o_2)	$(o_1, o_2)\colon R$
(A3),(A4)	SubClassOf (C_1, C_2)	$C_1 \sqsubseteq C_2$
(A5),(A6)	EquivalentClasses $(C_1, \ldots, C_k)$	$C_1 \equiv \cdots \equiv C_k$
(A7)	ObjectPropertyDomain (R, C)	$\exists R.\top \sqsubseteq C$
(A8)	ObjectPropertyRange (R, C)	$\top \sqsubseteq \forall R.C$
(A9)	DisjointClasses $(C_1, \ldots, C_k)$	$C_1 \sqcap \cdots \sqcap C_k \sqsubseteq \bot$,
(A10)	DisjointUnion $(C, C_1, \ldots, C_k)$	$C \equiv C_1 \sqcup \cdots \sqcup C_k$, $C_i \sqcap C_j \sqsubseteq \bot, 1 \leq i < j \leq k$
(A15)	SubObjectPropertyOf $(R_1 R_2)$	$R_1 \sqsubseteq R_2$
(A15)	SubDataPropertyOf (T_1, T_2)	$T_1 \sqsubseteq T_2$
(A16)	TransitiveObjectProperty (R)	$\mathtt{trans}(R)$
(A17)	InverseObjectProperties (R_1, R_2)	$R_1 \equiv R_2^-$
(A18)	FunctionalObjectProperty (R)	$\top \sqsubseteq (\leq 1\ R.\top)$
(A19)	InverseFunctionalObjectProperty (R)	$\top \sqsubseteq (\leq 1\ R^-.\top)$
(A20)	SymmetricObjectProperty (R)	$\mathtt{sym}(R)$
(A21)	ReflexiveObjectProperty (R)	$\mathtt{ref}(R)$
(A22)	FunctionalDataProperty (t)	$\top \sqsubseteq (\leq 1\ t.\top)$
(A23)–(A26)	DataPropertyRange (t, d)	$\top \sqsubseteq \forall t.d$

In addition to the ABox, the TBox, and the RBox, *fuzzyDL* fuzzy KBs can also include a *VBox* (variable Box) $\mathcal{V}$, that makes it possible to restrict the values of the variables occurring in *fuzzyDL* expressions. Our VBox contains constraints about variables (A27)–(A29), where $\mathtt{a0}, \ldots, \mathtt{ak}$ are constants and $\bowtie \in \{\geq, \leq, =\}$:

(A27)	`(a1 * x1 + ... + ak * xk` $\bowtie$ `a0)`	linear inequation
(A28)	`(binary x)`	binary variable
(A29)	`(free x)`	unrestricted variable

Example B.7 *One may encode that Bob is at least as tall as Alice by specifying* `(instance bob Tall x1)`, `(instance alice (not Tall) x2)`, *together with the constraint* `x1 + x2 = 1`. *As we will see, the second fuzzy assertion requires*

that `alice` *belongs to* `Tall` *with a degree less or equal than* `1 - x2`*, which according to the constraint is equivalent to* `x1`*. Furthermore, the range of a variable* `x` *can be set by means of a pair of linear inequations. For instance, the set* {`x >= 0.5, x <= 0.8`} *ensures that* `x` $\in [0.5, 0.8]$.

Finally, there are some axioms to give names to the fuzzy datatypes:

(A30) `(define-fuzzy-concept name d)` datatype definition

Fuzzy logic Before concluding the syntax of the language, the user is also able to specify the default semantics for the fuzzy operators. For example, if the ontology developer uses `g-and`, the semantics considers Gödel conjunction, but if s/he uses `and`, the semantics considers the default fuzzy logic. Currently, *fuzzyDL* supports 3 semantics: Łukasiewicz, Zadeh, and, for backwards compatibility with crisp ontologies, classical.

(FL1) `(define-fuzzy-logic lukasiewicz)`
(FL2) `(define-fuzzy-logic zadeh)`
(FL3) `(define-fuzzy-logic classical)`

B.1.2 Unsupported Fuzzy OWL 2 Elements

At the time of writing, *fuzzyDL* does not support fuzzy nominals and modified fuzzy roles [81], which are elements of the Fuzzy OWL 2 ontologies without a counterpart in classical OWL 2. Furthermore, *fuzzyDL* cannot support some elements of classical OWL 2 [61] which for backwards compatibility are also present in Fuzzy OWL 2:

- The following concepts are not supported: ObjectOneOf, ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, DataAllValuesFrom, DataHasValue, DataMinCardinality, DataMaxCardinality, and DataExactCardinality.
- The following properties are not supported: TopObjectProperty, BottomObjectProperty, TopDataProperty, and BottomDataProperty.
- The following axioms are not supported: NegativeObjectPropertyAssertion, NegativeDataPropertyAssertion, SameIndividual, DifferentIndividuals, ObjectPropertyChain, AsymmetricObjectProperty, IrreflexiveObjectProperty, DisjointObjectProperties, and DisjointDataProperties.
- Datatypes are restricted to strings and real and integer numbers (see the list of datatypes in [61]). Other datatypes such as booleans, dates, or URIs are considered as strings.
- Data ranges are restricted to atomic datatypes, datatype restrictions on numerical datatypes, and intersections of datatype restrictions.

The reader is referred to the *fuzzyDL* webpage to get updated information.

B.1.3 Fuzzy Ontology Semantics

The main idea is that concepts and roles are interpreted, respectively, as fuzzy sets and fuzzy relations over an interpretation domain. Thus, axioms, rather than being "classically" evaluated (either true or false), are "many-valued" evaluated in a finite truth space $\mathcal{D}$ which is a subset of $[0, 1]$.

To do so, we need beforehand to fix a fuzzy logic. By default, the fuzzy operators are interpreted according to the semantics chosen by the user using (FL1), (FL2), or (FL3), except if a particular fuzzy operator is indicated, as in some fuzzy conjunctions, disjunctions, implications, or axioms. If the chosen semantics is classical, every evaluation is restricted to the interval $\{0, 1\}$. Exceptionally, Zadeh fuzzy DLs use Zadeh implication in the semantics of GCIs and RIAs (again, unless a particular fuzzy implication is specified), since Kleene-Dienes implication is known to produce some counter-intuitive effects [8]. However, it is worth to note that the result of Zadeh implication is always in $\{0, 1\}$ (see Apprendix A.1) and Kleene-Dienes implication can also produce counter-intuitive effects in concepts (C8), (C10), (C20), (C33), and (C36) [43].

The semantics of the logic is given by a fuzzy interpretation $\mathcal{I}$ (for the abstract individuals) and a *fuzzy concrete domain* or *fuzzy data type theory* [76] $\mathbf{D}$ (for the concrete values). The fuzzy data type theory consists of a tuple $\mathbf{D} = \langle \Delta^{\mathbf{D}}, \cdot^{\mathbf{D}} \rangle$, with a data type domain $\Delta^{\mathbf{D}}$ and a function $\cdot^{\mathbf{D}}$ that assigns to each data type predicate $\mathbf{d}$ a function $\mathbf{d}^{\mathbf{D}} : \Delta^{\mathbf{D}} \to \mathcal{D}$ (recall that we are restricting to unary data types).

A *fuzzy interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ relative to a fuzzy data type theory $\mathbf{D} = \langle \Delta^{\mathbf{D}}, \cdot^{\mathbf{D}} \rangle$ 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 data value, data type, and fuzzy data type predicate, and it assigns:

1. to each atomic concept $\mathtt{A}$ a function $\mathtt{A}^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \to \mathcal{D}$;
2. to each object property $\mathtt{R}$ a function $\mathtt{R}^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to \mathcal{D}$;
3. to each data property $\mathtt{T}$ a function $\mathtt{T}^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}} \to \mathcal{D}$;
4. to each individual $\mathtt{o}$ an element $\mathtt{o}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ such that $\mathtt{o1}^{\mathcal{I}} \neq \mathtt{o2}^{\mathcal{I}}$ if $\mathtt{o1} \neq \mathtt{o2}$ (called *Unique Name Assumption*, UNA),
5. to each concrete value $\mathtt{v}$ an element $\mathtt{v}^{\mathcal{I}} \in \Delta^{\mathbf{D}}$,
6. to each variable $\mathtt{x}$ an element $\mathtt{x}^{\mathcal{I}} \in \Delta^{\mathbf{D}}$,
7. to each object feature $\mathtt{r}$ a partial function $\mathtt{r}^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to \mathcal{D}$ such that for all $x \in \Delta^{\mathcal{I}}$ there is a unique $y \in \Delta^{\mathcal{I}}$ on which $\mathtt{r}^{\mathcal{I}}(x, y)$ is defined;
8. to each data feature $\mathtt{t}$ a partial function $\mathtt{t}^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}} \to \{0, 1\}$ such that for all $x \in \Delta^{\mathcal{I}}$ there is a unique $w \in \Delta^{\mathbf{D}}$ on which $\mathtt{t}^{\mathcal{I}}(x, w)$ is defined;
9. to each modifier $\mathtt{m}$ a function $\mathtt{m}^{\mathcal{I}} \colon \mathcal{D} \to \mathcal{D}$;
10. to each aggregation operator @ a function $@^{\mathcal{I}} : \mathcal{D}^k \to \mathcal{D}$.

$\cdot^{\mathcal{I}}$ is extended to complex concepts as follows, where for ease of presentation, we identify the interpretation of fuzzy membership functions (appearing in datatypes, modifiers, quantifiers, and fuzzy numbers) and aggregation operators with the functions themselves:

(C2) $(\texttt{*top*})^{\mathcal{I}}(x) := 1$

(C3) $(\texttt{*bottom*})^{\mathcal{I}}(x) := 0$

(C4) $(\texttt{and C1 C2 ... Ck})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \otimes \cdots \otimes \texttt{Ck}^{\mathcal{I}}(x)$

(C5) $(\texttt{or C1 C2 ... Ck})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \oplus \cdots \oplus \texttt{Ck}^{\mathcal{I}}(x)$

(C6) $(\texttt{not C})^{\mathcal{I}}(x) := 1 - \texttt{C}^{\mathcal{I}}(x)$

(C7) $(\texttt{some R C})^{\mathcal{I}}(x) := \sup_{y \in \Delta^{\mathcal{I}}} \texttt{R}^{\mathcal{I}}(x,y) \otimes \texttt{C}^{\mathcal{I}}(y)$

(C8) $(\texttt{all R C})^{\mathcal{I}}(x) := \inf_{y \in \Delta^{\mathcal{I}}} \texttt{R}^{\mathcal{I}}(x,y) \Rightarrow \texttt{C}^{\mathcal{I}}(y)$

(C9) $(\texttt{some T d})^{\mathcal{I}}(x) := \sup_{v \in \Delta^{\mathcal{D}}} \texttt{T}^{\mathcal{I}}(x,v) \otimes \texttt{d}^{\mathcal{I}}(v)$

(C10) $(\texttt{all T d})^{\mathcal{I}}(x) := \inf_{v \in \Delta^{\mathcal{D}}} \texttt{T}^{\mathcal{I}}(x,v) \Rightarrow \texttt{d}^{\mathcal{I}}(v)$

(C11) $(\texttt{some t v })^{\mathcal{I}}(x) := \texttt{t}^{\mathcal{I}}(x, \texttt{v}^{\mathcal{I}})$

(C12) $(\texttt{some t [>= v]})^{\mathcal{I}}(x) := \sup_{w \in \Delta^{\mathcal{D}}} \texttt{t}^{\mathcal{I}}(x,w) \otimes (w \geq \texttt{v}^{\mathcal{I}})$

(C13) $(\texttt{some t [<= v]})^{\mathcal{I}}(x) := \sup_{w \in \Delta^{\mathcal{D}}} \texttt{t}^{\mathcal{I}}(x,w) \otimes (w \leq \texttt{v}^{\mathcal{I}})$

(C14) $(\texttt{some R o})^{\mathcal{I}}(x) := \texttt{R}^{\mathcal{I}}(x, \texttt{o}^{\mathcal{I}})$

(C15) $(\texttt{self R})^{\mathcal{I}}(x) := \texttt{R}^{\mathcal{I}}(x,x)$

(C16) $(\texttt{g-and C1 ... Ck})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \otimes_G \cdots \otimes_G \texttt{Ck}^{\mathcal{I}}(x)$

(C17) $(\texttt{l-and C1 ... Ck})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \otimes_{\text{L}} \cdots \otimes_{\text{L}} \texttt{Ck}^{\mathcal{I}}(x)$

(C18) $(\texttt{g-or C1 ... Ck})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \oplus_G \cdots \oplus_G \texttt{Ck}^{\mathcal{I}}(x)$

(C19) $(\texttt{l-or C1 ... Ck})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \oplus_{\text{L}} \cdots \oplus_{\text{L}} \texttt{Ck}^{\mathcal{I}}(x)$

(C20) $(\texttt{implies C1 C2})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \Rightarrow \texttt{C2}^{\mathcal{I}}(x)$

(C21) $(\texttt{g-implies C1 C2})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \Rightarrow_G \texttt{C2}^{\mathcal{I}}(x)$

(C22) $(\texttt{l-implies C1 C2})^{\mathcal{I}}(x) := \texttt{C1}^{\mathcal{I}}(x) \Rightarrow_{\text{L}} \texttt{C2}^{\mathcal{I}}(x)$

(C23) $(\texttt{m C})^{\mathcal{I}}(x) := \texttt{m}^{\mathcal{I}}(\texttt{C}^{\mathcal{I}}(x))$

(C24) $(\texttt{C [>= D]})^{\mathcal{I}}(x) := \begin{cases} \texttt{C}^{\mathcal{I}}(x), & \text{if } \texttt{C}^{\mathcal{I}}(x) \geq \texttt{D} \\ 0, & \text{otherwise} \end{cases}$

(C25) $(\texttt{C [<= D]})^{\mathcal{I}}(x) := \begin{cases} \texttt{C}^{\mathcal{I}}(x), & \text{if } \texttt{C}^{\mathcal{I}}(x) \leq \texttt{D} \\ 0, & \text{otherwise} \end{cases}$

(C26) $(\texttt{some t x })^{\mathcal{I}}(x) := \sup_{w \in \Delta^{\mathcal{D}}} \{\texttt{t}^{\mathcal{I}}(x,w) \otimes (w = \texttt{x}^{\mathcal{I}})\}$

(C27) $(\texttt{some t F })^{\mathcal{I}}(x) := \sup_{w \in \Delta_{\mathbf{D}}} \{\texttt{t}^{\mathcal{I}}(x,w) \otimes \texttt{F}^{\mathcal{I}}(w)\}$

(C28) $(\texttt{some t [>= x]})^{\mathcal{I}}(x) := \sup_{w \in \Delta^{\mathcal{D}}} \{\texttt{t}^{\mathcal{I}}(x,w) \otimes (w \geq \texttt{x}^{\mathcal{I}})\}$

(C29) $(\texttt{some t [>= F]})^{\mathcal{I}}(x) := \sup_{w,w' \in \Delta_{\mathbf{D}}} \{\texttt{t}^{\mathcal{I}}(x,w) \otimes (w \geq w') \otimes \texttt{F}^{\mathcal{I}}(w')\}$

(C30) $(\texttt{some t [<= x]})^{\mathcal{I}}(x) := \sup_{w \in \Delta^{\mathcal{D}}} \{\texttt{t}^{\mathcal{I}}(x,w) \otimes (w \leq \texttt{x}^{\mathcal{I}})\}$

(C31) $(\texttt{some t [<= F]})^{\mathcal{I}}(x) := \sup_{w,w' \in \Delta_{\mathbf{D}}} \{\texttt{t}^{\mathcal{I}}(x,w) \otimes (w \leq w') \otimes \texttt{F}^{\mathcal{I}}(w')\}$

(C32) $(\texttt{la s C})^{\mathcal{I}}(x) := \inf_{y \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(x,y) \Rightarrow \texttt{C}^{\mathcal{I}}(y)\}$

(C33) $(\texttt{tla s C})^{\mathcal{I}}(x) := \inf_{z \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(x,z) \Rightarrow \inf_{y \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(z,y) \Rightarrow \texttt{C}^{\mathcal{I}}(y)\}\}$

(C34) $(\texttt{lla s C})^{\mathcal{I}}(x) := \sup_{z \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(x,z) \otimes \inf_{y \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(z,y) \Rightarrow \texttt{C}^{\mathcal{I}}(y)\}\}$

(C35) $(\texttt{ua s C})^{\mathcal{I}}(x) := \sup_{y \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(x,y) \otimes \texttt{C}^{\mathcal{I}}(y)\}$

(C36) $(\texttt{tua s C})^{\mathcal{I}}(x) := \inf_{z \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(x,z) \Rightarrow \sup_{y \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(z,y) \otimes \texttt{C}^{\mathcal{I}}(y)\}\}$

(C37) $(\texttt{lua s C})^{\mathcal{I}}(x) := \sup_{z \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(x,z) \otimes \sup_{y \in \Delta^{\mathcal{I}}} \{\texttt{s}^{\mathcal{I}}(z,y) \otimes \texttt{C}^{\mathcal{I}}(y)\}\}$

(C38) $(\texttt{w-sum (w1 C1) ... (wk Ck)})^{\mathcal{I}}(x) := @^{\text{ws}}_{[\texttt{w1},\ldots,\texttt{wk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

(C39) $(\texttt{w-sum-zero (w1 C1) ... (wk Ck)})^{\mathcal{I}}(x) := @^{\text{wsz}}_{[\texttt{w1},\ldots,\texttt{wk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

(C40) $(\texttt{owa (w1 ... wk) (C1 ... Ck)})^{\mathcal{I}}(x) := @^{\text{owa}}_{[\texttt{w1},\ldots,\texttt{wk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

(C41) $(\texttt{q-owa Q C1 ... Ck})^{\mathcal{I}}(x) := @^{\text{ws}}_{[\texttt{w1},\ldots,\texttt{wk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

(C42) $(\texttt{choquet (w1 ... wk) (C1 ... Ck)})^{\mathcal{I}}(x) := @^{\text{ci}}_{[\texttt{w1},\ldots,\texttt{wk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

(C43) $(\texttt{sugeno (r1 ... rk) (C1 ... Ck)})^{\mathcal{I}}(x) := @^{\text{si}}_{[\texttt{r1},\ldots,\texttt{rk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

(C44) $(\texttt{q-sugeno (r1 ... rk) (C1 ... Ck)})^{\mathcal{I}}(x) := @^{\text{qsi}}_{[\texttt{r1},\ldots,\texttt{rk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

(C45) $(\texttt{w-max (r1 C1) ... (rk Ck)})^{\mathcal{I}}(x) := @^{\text{wmax}}_{[\texttt{r1},\ldots,\texttt{rk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

(C46) $(\texttt{w-min (r1 C1) ... (rk Ck)})^{\mathcal{I}}(x) := @^{\text{wmin}}_{[\texttt{r1},\ldots,\texttt{rk}]}(\texttt{C1}^{\mathcal{I}}(x), ..., \texttt{Ck}^{\mathcal{I}}(x))$

The notion $\mathcal{I}$ *satisfies* an axiom τ, denoted $\mathcal{I} \models \tau$, is defined as follows:

(A1) $\mathcal{I} \models$ `(instance o C D)` if $\texttt{C}^{\mathcal{I}}(\texttt{o}^{\mathcal{I}}) \geq \texttt{D}$

(A2) $\mathcal{I} \models$ `(related o1 o2 R D)` if $\texttt{R}^{\mathcal{I}}(\texttt{o1}^{\mathcal{I}}, \texttt{o2}^{\mathcal{I}}) \geq \texttt{D}$

(A3) $\mathcal{I} \models$ `(implies C1 C2 D)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{C1}^{\mathcal{I}}(x) \Rightarrow \texttt{C2}^{\mathcal{I}}(x) \geq \texttt{D}$

(A4) $\mathcal{I} \models$ `(define-primitive-concept A C D)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{A}^{\mathcal{I}}(x) \Rightarrow \texttt{C}^{\mathcal{I}}(x) \geq \texttt{D}$

(A5) $\mathcal{I} \models$ `(equivalent-concepts C1 ... Ck)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{C1}^{\mathcal{I}}(x) = \cdots = \texttt{Ck}^{\mathcal{I}}(x)$

(A6) $\mathcal{I} \models$ `(define-concept A C)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{A}^{\mathcal{I}}(x) = \texttt{C}^{\mathcal{I}}(x)$

(A7) $\mathcal{I} \models$ `(domain R C)` if $\forall x \in \Delta^{\mathcal{I}}.$ `(some R *top*)`$^{\mathcal{I}}(x) \leq \texttt{C}^{\mathcal{I}}(x)$

(A8) $\mathcal{I} \models$ `(range R C)` if $\forall x \in \Delta^{\mathcal{I}}.$ `(all R C)`$^{\mathcal{I}}(x) = 1$

(A9) $\mathcal{I} \models$ `(disjoint-concepts C1 ... Ck)` if $\min\{\texttt{Ci}^{\mathcal{I}}(x), \texttt{Cj}^{\mathcal{I}}(x)\} = 0$, $\forall 1 \leq i < j \leq k$

(A10) $\mathcal{I} \models$ `(disjoint-union C1 ... Ck)` if $\texttt{C1}^{\mathcal{I}}(x) =$ `(or C2 ... Ck)`$^{\mathcal{I}}(x)$, and $\mathcal{I} \models$ `(disjoint-concepts C2 ... Ck)`

(A11) $\mathcal{I} \models$ `(g-implies C1 C2 D)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{C1}^{\mathcal{I}}(x) \Rightarrow_G \texttt{C2}^{\mathcal{I}}(x) \geq \texttt{D}$

(A12) $\mathcal{I} \models$ `(kd-implies C1 C2 D)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{C1}^{\mathcal{I}}(x) \Rightarrow_{KD} \texttt{C2}^{\mathcal{I}}(x) \geq \texttt{D}$

(A13) $\mathcal{I} \models$ `(l-implies C1 C2 D)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{C1}^{\mathcal{I}}(x) \Rightarrow_{\mathrm{L}} \texttt{C2}^{\mathcal{I}}(x) \geq \texttt{D}$

(A14) $\mathcal{I} \models$ `(z-implies C1 C2)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{C1}^{\mathcal{I}}(x) \Rightarrow_Z \texttt{C2}^{\mathcal{I}}(x) = 1$

(A15) $\mathcal{I} \models$ `(implies-role R1 R2 D)` if $\forall x, y \in \Delta^{\mathcal{I}}.\texttt{R1}^{\mathcal{I}}(x, y) \Rightarrow \texttt{R2}^{\mathcal{I}}(x, y) \geq \texttt{D}$

(A16) $\mathcal{I} \models$ `(transitive R)` if $\forall x, y \in \Delta^{\mathcal{I}}.\texttt{R}^{\mathcal{I}}(x, y) \geq \sup_{z \in \Delta^{\mathcal{I}}} \texttt{R}^{\mathcal{I}}(x, z) \otimes \texttt{R}^{\mathcal{I}}(z, y)$

(A17) $\mathcal{I} \models$ `(inverse R1 R2)` if $\forall x, y \in \Delta^{\mathcal{I}}.\texttt{R1}^{\mathcal{I}}(x, y) = \texttt{R2}^{\mathcal{I}}(y, x)$

(A18) $\mathcal{I} \models$ `(functional r)` if $\forall x \in \Delta^{\mathcal{I}}$ there is a unique $y \in \Delta^{\mathcal{I}}$ for which $\texttt{r}^{\mathcal{I}}(x, y)$ is defined

(A19) $\mathcal{I} \models$ `(inverse-functional R)` if $\forall x \in \Delta^{\mathcal{I}}$ there is a unique $y \in \Delta^{\mathcal{I}}$ for which $\texttt{R}^{\mathcal{I}}(y, x)$ is defined

(A20) $\mathcal{I} \models$ `(symmetric R)` if $\forall x, y \in \Delta^{\mathcal{I}}.\texttt{R}^{\mathcal{I}}(x, y) = \texttt{R}^{\mathcal{I}}(y, x)$

(A21) $\mathcal{I} \models$ `(reflexive R)` if $\forall x \in \Delta^{\mathcal{I}}.\texttt{R}^{\mathcal{I}}(x, x) = 1$

(A22) $\mathcal{I} \models$ `(functional t)` if $\forall x \in \Delta^{\mathcal{I}}$ there is a unique $w \in \Delta^{\mathbf{D}}$ for which $\texttt{t}^{\mathcal{I}}(x, w)$ is defined

(A23) $\mathcal{I} \models$ `(range t *integer* k1 k2)` if the range of $\texttt{t}^{\mathcal{I}}$ is the set of integers in $[\texttt{k1}, \texttt{k2}]$

(A24) $\mathcal{I} \models$ `(range t *real* k1 k2)` if the range of $\texttt{t}^{\mathcal{I}}$ is $[\texttt{k1}, \texttt{k2}]$

(A25) $\mathcal{I} \models$ `(range t *string*)` if the range of $\texttt{t}^{\mathcal{I}}$ is the set of strings

(A26) $\mathcal{I} \models$ `(range t *boolean*)` if the range of $\texttt{t}^{\mathcal{I}}$ is $\{0, 1\}$

(A27) $\mathcal{I} \models$ `(a1 * x1 + ... + ak * xk` $\bowtie$ `a0)` if $\texttt{a1} \cdot \texttt{x1}^{\mathcal{I}} + \cdots + \texttt{a1} \cdot \texttt{xk}^{\mathcal{I}} \bowtie \texttt{a0}$

(A28) $\mathcal{I} \models$ `(binary x)` if $\texttt{x}^{\mathcal{I}} \in \{0, 1\}$

(A29) $\mathcal{I} \models$ `(free x)` if $\texttt{x}^{\mathcal{I}} \in (-\infty, \infty)$

(A30) $\mathcal{I} \models$ `(define-fuzzy-concept name d)` if $\texttt{name}^{\mathcal{I}} = \texttt{d}^{\mathcal{I}}$

Since the number of degrees of truth is finite, fuzzy interpretations $\mathcal{I}$ are always *witnessed* [43]. That is, whenever a supremum or infimum is involved in the semantics of a concept expression,[23] there is at least one element in the range of the relation for which the semantic value is attained. For instance, for (C7), besides the condition

$$\texttt{(some R C)}^{\mathcal{I}}(x) := \sup_{y \in \Delta^{\mathcal{I}}} \texttt{R}^{\mathcal{I}}(x, y) \otimes \texttt{C}^{\mathcal{I}}(y)$$

[23] Conditions (C7)–(C10), (C12), (C13), (C26)–(C37).

in witnessed interpretations we have also that

$$\text{there is } y \in \Delta^{\mathcal{I}} \text{ such that } (\texttt{some R C})^{\mathcal{I}}(x) = \texttt{R}^{\mathcal{I}}(x,y) \otimes \texttt{C}^{\mathcal{I}}(y) \quad .$$

Finally, an interpretation *satisfies* (is a *model of*) a fuzzy ontology if it satisfies each axiom in it.

B.1.4 Reasoning Tasks Syntax

The concrete *fuzzyDL* syntax for the reasoning tasks is the following one:

(Q1)	`(sat?)`	Consistency
(Q2)	`(min-sat? C [o])`	Minimal Satisfiability Degree of a concept
(Q3)	`(max-sat? C [o])`	Best Satisfiability Degree of a concept
(Q4)	`(min-instance? o C)`	Best Entailment Degree of a concept assertion
(Q5)	`(max-instance? o C)`	Maximal Entailment Degree of a concept assertion
(Q6)	`(min-related? o1 o2 R)`	Best Entailment Degree of a role assertion
(Q7)	`(max-related? o1 o2 R)`	Maximal Entailment Degree of a role assertion
(Q8)	`(min-subs? C D)`	Best Entailment Degree of a GCI
(Q9)	`(max-subs? C D)`	Maximal Entailment Degree of a GCI
(Q10)	`(min-g-subs? C D)`	BED of a GCI using Gödel implication
(Q11)	`(max-g-subs? C D)`	MED of a GCI using Gödel implication
(Q12)	`(min-l-subs? C D)`	BED of a GCI using Łukasiewicz implication
(Q13)	`(max-l-subs? C D)`	MED of a GCI using Łukasiewicz implication
(Q14)	`(min-kd-subs? C D)`	BED of a GCI using Kleene-Dienes implication
(Q15)	`(max-kd-subs? C D)`	MED of a GCI using Kleene-Dienes implication
(Q16)	`(all-instances? C)`	Instance retrieval
(Q17)	`(max-var? var)`	Variable maximisation
(Q18)	`(min-var? var)`	Variable minimisation
(Q19)	`(defuzzify-lom? C o t)`	LOM defuzzification
(Q20)	`(defuzzify-som? C o t)`	SOM defuzzification
(Q21)	`(defuzzify-mom? C o t)`	MOM defuzzification
(Q22)	`(bnp? F)`	Best Non-Fuzzy Performance

Note that in (Q2) and (Q3) we may specify optionally an individual `o`, while this is mandatory in defuzzification tasks.

B.2 Using the Protégé Plug-in

Figure 3 (a) shows the options of the plug-in to represent fuzzy ontologies, as illustrated in Figure 3 (b). In particular, the available options are:

- *fuzzyDL reasoner query* allows to export ontologies into *fuzzyDL* syntax and to reason using *fuzzyDL* as we will discuss in Section B.1.1;
- *ontology* allows to specify a fuzzy logic of types (FL1)-(FL3);
- *fuzzy modifier* defines modifiers of types (M1)–(M2);

- *fuzzy nominal* is currently not supported in *fuzzyDL*;
- *fuzzy modified concept* defines concepts of type (C23);
- *weighted single concept* and weighted complex concept define concepts of types (C38)–(C39);
- *aggregation and integral concept* define concepts (C40) and (C42)–(C46);
- *quantifier-guided OWA concept* defines concepts of type (C41) and quantifiers of types (Q1)–(Q2);
- *fuzzy modified role* is currently not supported in *fuzzyDL*;
- *fuzzy datatype* defines datatypes of types (D1)–(D5);
- *fuzzy modified datatype* defines datatypes of type (D6);
- *fuzzy axioms* defines axioms of types (A1)–(A21).

Reasoning queries can be submitted using the option *fuzzyDL reasoner query*. The queries must be written in the *fuzzyDL* syntax presented in Section B.1.4, as illustrated in Figure 4.

B.3 Using fuzzyDL API

This sections shows two examples illustrating how to use *fuzzyDL* API. Example 2.1 shows how to create a simple fuzzy ontology with two axioms. The next example creates a less simple axiom involving a complex concept. Then, another example shows how to answer user queries.

Example B.8 *The following code creates an empty fuzzy ontology, adds an axiom* `(g-implies HumanWithSon (and Human (some hasChild Male)))`*, and saves it into a file called "ontology.fdl".*

```
// Creates a new fuzzy ontology
KnowledgeBase kb = new KnowledgeBase();

// Adds a concept definition
Concept c1 = new Concept("HumanWithSon");
Concept c2 = Concept.and( new Concept("Human"),
                  Concept.some("hasChild", new Concept("Male")) );

d = Degree.getDegree(1.0);
kb.gImplies(c1, c2, d);

// Saves file
kb.saveToFile("ontology.fdl")
```

Example B.9 *The following fragment of code shows how to submit a concept satisfiability query to the fuzzy ontology "ontology.fdl":*

```
// Load a fuzzy KB
KnowledgeBase kb = Parser.getKB("ontology.fdl");

// Define queries
Concept conc = new Concept("C");

Query q = new MinSatisfiableQuery(conc);

// Load options for the reasoner, using file "CONFIG"
ConfigReader.loadParameters("CONFIG", new String[0]);

// After having created KB and queries, start logical inference
kb.solveKB();

// Solve a query q
Solution result = q.solve(kb);

// Print the result
if (result.isConsistentKB())
    System.out.println(q.toString() + result.getSolution());
else
    System.out.println("KB is inconsistent");
```

B.4 Advanced Details of fuzzyDL Reasoning Algorithm

In this section we discuss some advanced notions about reasoning with fuzzy ontologies using *fuzzyDL*. At first, Section B.4.1 briefly explains the basics of the reasoning algorithm with more details than Section 4. Next, Section B.4.2 explains the implemented reduction of reasoning tasks. Finally, Section B.4.3 details the parameters used by the reasoner.

B.4.1 Overview of the Algorithm

This section extends the details of the algorithm already given in Section 4.1. In order to determine *bed*($\mathcal{K}$, `(instance o C)`) with respect to $\mathcal{K}$, we consider an expression of the form `(instance o (not C) 1 - x)`), where `x` is a $\mathcal{D}$-valued variable. Informally, this means that `o` belongs to `C` with degree less or equal than `x`. Then, we construct a tableaux for $\mathcal{K}' = \mathcal{K} \cup \{$ `(instance o (not C) 1 - x)` $\}$ in which the application of tableaux rules generates new fuzzy assertion axioms together with *inequations* over variables. For example, the restriction $x_1 \otimes_{\mathrm{L}} x_2 \geq q$ can be encoded using the set of constraints $\{y \leq 1-q, x_1+x_2-1 \geq q-y, y \in \{0,1\}\}$. The tableau rules preserve both the satisfiability of the fuzzy ontology and the value of the variable that is being optimised. The rules are deterministic and only one optimisation problem is obtained. Indeed, in the previous example the two possibilities $y = 0$ and $y = 1$ encode the non-deterministic choice implicit in Łukasiewicz t-norm. After applying the tableaux rules, we *minimise* the original variable `x` in such a way that all constraints are satisfied. If the optimisation problem has a solution, the optimised value is the

solution to the BED problem. Otherwise, the fuzzy ontology is inconsistent and the BED is 1 since an inconsistent ontology entails anything.

In *fuzzyDL*, we end up with a MILP problem, which consists in minimising a linear function with respect to a set of constraints that are linear inequations in which rational and integer variables can occur. In our case, MILP problems will be *bounded* with rational variables ranging over a subset of $[0, 1]$ and integer variables ranging over $\{0, 1\}$.

Example B.10 *Consider a very simple fuzzy KB* $\mathcal{K} = \{$`(instance o (g-and B C 0.8))`$\}$ *and suppose that we want to compute the minimal membership of* `o` *to* `C`*. i.e., bed*($\mathcal{K}$,`(instance o C)`)*. To this end, we consider* $\mathcal{K}' = \{$`(instance o (g-and B C 0.8))`,`(instance o (not C) 1 - x)`$\}$ *and minimise the variable* `x`*. It is easy to see that the first axiom implies* $\mathtt{B}^{\mathcal{I}}(\mathtt{o}^{\mathcal{I}}) \geq 0.8$ *and* $\mathtt{C}^{\mathcal{I}}(\mathtt{o}^{\mathcal{I}}) \geq 0.8$ *for any model* $\mathcal{I}$ *of* $\mathcal{K}$*, since the minimum t-norm is used. Furthermore, the second axiom implies that* $(\mathtt{not\ C})^{\mathcal{I}}(\mathtt{o}^{\mathcal{I}}) = 1 - \mathtt{C}^{\mathcal{I}}(o) \geq 1 - \mathtt{x}$ *and, thus,* $\mathtt{x} \geq \mathtt{C}^{\mathcal{I}}(\mathtt{o}^{\mathcal{I}}) \geq 0.8$*. Hence, the minimal value that is consistent with these restrictions is* $\mathtt{x} = 0.8$*.*

B.4.2 Reduction of Reasoning Tasks

We will show now that indeed all the reasoning tasks can be reduced to the variable minimisation, that is, to minimise a $\mathcal{D}$-variable `x` given a fuzzy KB $\mathcal{K}$.[24] Let `n` denote a new individual not occurring in $\mathcal{K}$. Then:

- The BED of a concept assertion *bed*($\mathcal{K}$, `(instance o C)`) is equivalent to minimise x given the fuzzy KB $\mathcal{K}' = \mathcal{K} \cup \{$`(instance o (not C) 1 - x)`$\}$.
- The maximisation of the variable `x` w.r.t. $\mathcal{K}$ is equivalent to minimising a variable `z` such that `z = 1 - x` w.r.t. $\mathcal{K}$.
- $\mathcal{K}$ is consistent iff the maximisation of a new variable `x` is 1. Otherwise, *fuzzyDL* reports that $\mathcal{K}$ is inconsistent. Note that since `x` is a new variable its maximal value would be 1 unless the optimisation problem does not have a solution, which is the case if the ontology is inconsistent.
- The Best Satisfiability Degree of a concept `C` can be computed as the maximisation of `x` given $\mathcal{K}' = \mathcal{K} \cup \{$`(instance n C x)`$\}$.
- The Minimal Satisfiability Degree of `C` can be computed as the minimisation of `x` given $\mathcal{K}' = \mathcal{K} \cup \{$`(instance n (not C) 1 - x)`$\}$.
- `C` is `D`-satisfiable iff the Best Satisfiability Degree of `C` is greater or equal than `D`.
- *bed*($\mathcal{K}$, `(related o1 o2 R)`) = *bed*($\mathcal{K}$, `(instance o1 (some R o2))`).

[24] Please note that in the reductions of the BED and the Minimal Satisfiability Degree, in the construct `(not C)` the negation is Łukasiewicz negation.

- $bed(\mathcal{K}, \texttt{(implies C1 C2)}) = bed(\mathcal{K}, \texttt{(instance n (implies C1 C2))})$. The cases `g-implies`, `l-implies`, and `kd-implies` are similar.
- The Maximal Entailment Degree of an axiom can be computed as follows:
  - For a concept assertion, maximise x given $\mathcal{K}' = \mathcal{K} \cup \{\texttt{(instance o C x)}\}$.
  - For a role assertion, maximise x given $\mathcal{K}' = \mathcal{K} \cup \{\texttt{(related o1 o2 R x)}\}$.
  - For a GCI, maximise x given $\mathcal{K}' = \mathcal{K} \cup \{\texttt{(implies C1 C2 x)}\}$. The cases `g-implies`, `l-implies`, and `kd-implies` are similar.
- A fuzzy KB $\mathcal{K}$ entails an axiom of the form:
  - `(instance o C D)` iff $bed(\mathcal{K}, \texttt{(instance o C)}) \geq \texttt{D}$,
  - `(related o1 o2 R D)` iff $bed(\mathcal{K}, \texttt{(related o1 o2 R C)}) \geq \texttt{D}$,
  - `(implies C1 C2 D)` iff $bed(\mathcal{K}, \texttt{(implies C1 C2)}) \geq \texttt{D}$.
- `C2` `D`-subsumes `C1` w.r.t. a fuzzy KB $\mathcal{K}$ iff the BED of the corresponding GCI $bed(\mathcal{K}, \texttt{(implies C1 C2)}) \geq \texttt{D}$.
- The instance retrieval of `C` with respect to $\mathcal{K}$ can be solved by computing $bed(\mathcal{K}, \texttt{(instance i C)})$ for every individual `i` occurring in $\mathcal{K}$.
- `(defuzzify-lom? C o t)` is computed by maximising the value of the (internal) variable representing the value $\texttt{t}^{\mathcal{I}}(\texttt{o}^{\mathcal{I}}, \texttt{suc}^{\mathcal{I}})$ given $\mathcal{K}' = \mathcal{K} \cup \{\texttt{(instance o C z)}\}$, where `suc` is the unique `t`-successor of `o`, and `z` is the result of the query `(max-sat? C o)`.
- Similarly, `(defuzzify-som? C o t)` is computed by minimising the value of the (internal) variable representing the value $\texttt{t}^{\mathcal{I}}(\texttt{o}^{\mathcal{I}}, \texttt{suc}^{\mathcal{I}})$ given $\mathcal{K}' = \mathcal{K} \cup \{\texttt{(instance o C z)}\}$.
- Finally, the MOM is computed as the average of the LOM and the SOM.

B.4.3 Parameters

fuzzyDL uses a configuration file that specifies the values of 5 parameters:

- `debugPrint`: boolean value indicating whether to display debugging information or not.
- `epsilon`: real number indicating the tolerance of the reasoner. The default value is 0.001.
- `maxIndividuals`: integer value indicating the maximum number of new individuals created during the reasoning before aborting it. The default value is infinite.
- `optimisation`: integer value where 0 disables optimisations and a positive value enables them. There are plans for defining several configurations of optimisations, but this will be part of the future work.
- `showVersion`: boolean value used to display *fuzzyDL*'s version or not.