mathematics

MDPI

Article

Algorithms for Instance Retrieval and Realization in Fuzzy Ontologies

Ignacio Huitzil [1,*], Jorge Bernad [1,2] and Fernando Bobillo [1,2]

[1] Department of Computer Science and Systems Engineering, University of Zaragoza, 50009 Zaragoza, Spain; fbobillo@unizar.es (F.B.); jbernad@unizar.es (J.B.)
[2] Aragon Institute of Engineering Research (I3A), 50018 Zaragoza, Spain
* Correspondence: ihuitzil@unizar.es

Received: 31 December 2019; Accepted: 19 January 2020; Published: 22 January 2020

Abstract: Fuzzy description logics, the formalism behind fuzzy ontologies, are an important mathematical method with applications in many artificial intelligence scenarios. This paper proposes the first specific algorithms to solve two reasoning tasks with respect to a fuzzy ontology: the instance retrieval and the realization problem. Our algorithms are based on a reduction of the number of optimization problems to solve by merging some of them. Our experimental evaluation shows that the novel algorithm to solve the instance retrieval outperforms the previous algorithm, and that in practice it is common to be able to solve a single optimization problem.

Keywords: fuzzy ontologies; fuzzy description logics; semantic reasoning

1. Introduction

Description Logics (DLs for short) [1] are a popular family of logics to represent structured knowledge, particularly in ontologies. They make it possible to capture the relevant knowledge in a domain field and to infer implicit knowledge, providing a good balance between the expressiveness of the language and the practical computational complexity of the reasoning. For example, the current standard language OWL 2 (Web Ontology Language) is based on the DL $\mathcal{SROIQ}(\mathbf{D})$ [2,3].

Fuzzy DLs are an extension of classical DLs to deal with imprecise knowledge (see [4] for an overview). The main idea is that axioms can be partially satisfied (typically measured using a degree of truth in $[0,1]$). Fuzzy DLs have proved to be a useful mathematical model in many artificial intelligence applications, including the Semantic Web and the Internet [5], recommendation systems [6,7], medicine [8], image interpretation [9], computational perception [10], robotics [11], gait recognition [12], aerospace industry [13], transportation [14], Internet of Things-based healthcare monitoring [15], automatic hotel reservation [16], and web content classification [17].

One of the most important reasoning tasks in classical ontologies is the instance retrieval problem, which consists of retrieving all the individuals i that are known to belong to a given concept C. In the fuzzy setting, this reasoning task can be extended to retrieving pairs $\langle i, \alpha \rangle$ such that each individual i belongs to a given (possibly fuzzy) concept C with degree greater or equal than $\alpha > 0$.

However, there are no specific reasoning algorithms to solve this problem in fuzzy ontologies. Instead, one needs a best entailment degree test for each individual i in the fuzzy ontology, retrieving a lower bound for its membership to C. For example, this is the algorithm implemented in the fuzzy ontology reasoner fuzzyDL [18], which, to the best of our knowledge, is the only software supporting instance retrieval with respect to a fuzzy ontology. Clearly, running several entailment tests is not an optimal solution, and may take a dramatic increase in the running time for hard ontologies.

A similar problem happens with the realization problem. In classical ontologies, realization consists of retrieving all the concepts C that a given individual i is an instance of. In fuzzy ontologies,

it requires retrieving pairs $\langle C, \alpha \rangle$ such that i belongs to C with degree greater or equal than $\alpha > 0$. Because this can be computed using a best entailment degree test for each concept C in the fuzzy ontology, no specific algorithms have been designed.

In this paper, we propose two specific algorithms to solve the instance retrieval and the realization problem with respect to a fuzzy ontology. Such algorithms are based on an extension of an optimization technique called optimization partitioning, originally proposed in [19] and extended in [20]. This optimization can be applied in a family of algorithms to reason with fuzzy DLs that are based on a collaboration of classical tableaux rules and mathematical programming [21], as it is the case of the algorithm implemented by fuzzyDL. In such cases, it is possible to compute a partition of the single optimization problem into smaller optimization problems, called constraint group optimization problems (CGO problems). The idea is to optimize independently, these CGO problems (perhaps optimizing the same CGO problem several times, if necessary) rather than optimizing the whole original problem several times.

The rest of this paper is organized as follows. Section 2 recalls some preliminaries on fuzzy DLs. Then, we present two novel algorithms to solve the realization (Section 4) and the instance retrieval (Section 3) given a fuzzy ontology. Next, Section 5 discusses an experimental evaluation of both algorithms. Finally, Section 6 sets out some conclusions and ideas for future work.

2. Background on Fuzzy DLs

This section overviews some results on fuzzy DLs that will be used in this paper. We assume that the reader is familiar with fuzzy logic [22,23] and with classical DLs [1]. We will define a fuzzy DL (syntax and semantics), the main reasoning tasks, and some reasoning algorithms. For further details, we refer the interested reader to [4,18]. Although our approach does not depend on the particular fuzzy DL, this paper will consider a relatively simple language, fuzzy $\mathcal{ALC}$, to make the text more concrete.

2.1. Syntax

This section recalls the syntax of fuzzy $\mathcal{ALC}$ as originally proposed in [21]. Fuzzy DLs have three elements: individuals, fuzzy concepts, and fuzzy properties (or roles). Fuzzy concepts are fuzzy sets of individuals, and fuzzy properties are fuzzy binary relations between individuals.

Concepts (denoted C) of the language can be built inductively from atomic concepts (A), top concept $\top$, bottom concept $\bot$, properties (R), and individuals (a) as follows:

$$C_1, C_2 \quad \rightarrow \quad \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \forall R.C \mid \exists R.C.$$

A Fuzzy Ontology (or fuzzy knowledge base) $\mathcal{O} = \langle \mathcal{A}, \mathcal{T} \rangle$ is composed of a fuzzy ABox $\mathcal{A}$, including axioms about individuals, and a fuzzy TBox $\mathcal{T}$, including axioms about concepts.

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

- Concept assertions of the form $\langle a{:}C \geq \alpha \rangle$, with $\alpha \in [0, 1]$. The intuitive idea is that individual a is an instance of concept C with degree greater than or equal to α.
- Property assertions of the form $\langle (a_1, a_2){:}R \geq \alpha \rangle$, $\alpha \in [0, 1]$. The intuition here is that the pair of individuals (a_1, a_2) is an instance of property R with degree greater than or equal to α.

A fuzzy TBox consists of a finite set of fuzzy general concept inclusions (fuzzy GCIs). A fuzzy GCI is an expression of the form $\langle C_1 \sqsubseteq C_2 \geq \alpha \rangle$, $\alpha \in [0, 1]$. The intuition here is that the degree of concept C_1 being subsumed by C_2 is greater than or equal to α.

Two important types of axioms that will be mentioned in the rest of this paper are disjoint axioms of the form $\langle C_1 \sqcap C_2 \sqsubseteq \bot \geq 1 \rangle$, and range axioms of the form $\langle \top \sqsubseteq \forall R.C \geq 1 \rangle$.

2.2. Semantics

This section recalls the semantics of fuzzy $\mathcal{ALC}$ as originally proposed in [21]. The semantics is defined using a fuzzy interpretation and a fuzzy logic, composed of a t-norm $\otimes$, t-conorm $\oplus$, negation function $\ominus$, and implication function $\Rightarrow$. Table 1 summarizes the definitions of those fuzzy operators in the main four fuzzy logics; namely, Gödel, Łukasiewicz, Product, and Zadeh.

Table 1. Combination functions of various fuzzy logics [24].

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

A fuzzy interpretation (or model) $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty set $\Delta^{\mathcal{I}}$ (the domain) and of a fuzzy interpretation function $\cdot^{\mathcal{I}}$ that assigns:

- To each individual a an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.
- To each fuzzy concept C a function $C^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \to [0, 1]$.
- To each fuzzy property R a function $R^{\mathcal{I}}\colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$.

Table 2 shows how to extend the interpretation function to complex concepts and fuzzy axioms.

Table 2. Semantics of fuzzy concepts and axioms [21].

Concept		**Semantics**
$(\top)^{\mathcal{I}}(x)$	=	1
$(\bot)^{\mathcal{I}}(x)$	=	0
$(A)^{\mathcal{I}}(x)$	=	$A^{\mathcal{I}}(x)$
$(C_1 \sqcap C_2)^{\mathcal{I}}(x)$	=	$C_1^{\mathcal{I}}(x) \otimes C_2^{\mathcal{I}}(x)$
$(C_1 \sqcup C_2)^{\mathcal{I}}(x)$	=	$C_1^{\mathcal{I}}(x) \oplus C_2^{\mathcal{I}}(x)$
$(\neg C)^{\mathcal{I}}(x)$	=	$\ominus C^{\mathcal{I}}(x)$
$(\forall R.C)^{\mathcal{I}}(x)$	=	$\inf_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x, y) \Rightarrow C^{\mathcal{I}}(y)\}$
$(\exists R.C)^{\mathcal{I}}(x)$	=	$\sup_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x, y) \otimes C^{\mathcal{I}}(y)\}$
Axiom		**Semantics**
$(a{:}C)^{\mathcal{I}}$	=	$C^{\mathcal{I}}(a^{\mathcal{I}})$
$((a_1, a_2){:}R)^{\mathcal{I}}$	=	$R^{\mathcal{I}}(a_1^{\mathcal{I}}, a_2^{\mathcal{I}})$
$(C_1 \sqsubseteq C_2)^{\mathcal{I}}$	=	$\inf_{x \in \Delta^{\mathcal{I}}} \{C_1^{\mathcal{I}}(x) \Rightarrow C_2^{\mathcal{I}}(x)\}$

Let $\phi \in \{a{:}C, (a_1, a_2){:}R, C \sqsubseteq D\}$. A fuzzy interpretation $\mathcal{I}$ satisfies (is a model of) a fuzzy axiom $\tau = \langle \phi \geq \alpha \rangle$, denoted $\mathcal{I} \models \tau$, iff $\phi^{\mathcal{I}} \geq \alpha$. An interpretation satisfies (is a model of) a fuzzy ontology if it satisfies each axiom in it. A fuzzy ontology $\mathcal{O}$ entails an axiom τ, denoted $\mathcal{O} \models \tau$, if any model of $\mathcal{O}$ satisfies τ.

2.3. Reasoning Tasks

Common reasoning tasks on DLs include consistency (checking if an ontology has a model), concept satisfiability (checking if a concept can have instances), entailment (checking if an ontology necessarily entails an axiom), realization (retrieving all the concepts an individual belongs to), and instance retrieval (retrieving all the instances of a concept). In fuzzy DLs, we have extensions

of those tasks and some new ones, such as the best entailment degree (BED). The BED of an axiom $\phi \in \{a{:}C, (a_1, a_2){:}R, C \sqsubseteq D\}$ with respect to a fuzzy ontology $\mathcal{O}$ is defined as [24]:

$$\mathsf{bed}(\mathcal{O}, \phi) = \sup \{\alpha \mid \mathcal{O} \models \langle \phi \geq \alpha \rangle\} .$$

These reasoning tasks are usually inter-definable (depending on the expressivity of the language and the semantics of the fuzzy operators).

Some popular fuzzy ontology reasoners are fuzzyDL [18], Fire [25], FPLGERDS [26], YADLR [27], DeLorean [28], GURDL [19], FRESG [29], LiFR [30], and SMT-based solver [31]. To the best of our knowledge, fuzzyDL, FRESG and YADLR are the only ones supporting instance retrieval, while FRESG, and YADLR are the only ones supporting realization.

The fuzzyDL reasoner reduces the previous reasoning services to the variable minimization problem, that is, to minimize a $[0,1]$-variable x, given a consistent fuzzy ontology $\mathcal{O}$ [18]. For instance, $\mathsf{bed}(\mathcal{O}, a{:}C)$ is computed as:

$$\min\{x \mid O \cup \{\langle a : \neg_{\text{Ł}} C \geq 1 - x \rangle\} \text{ is consistent }, x \in [0,1]\ , \tag{1}$$

where $\neg_{\text{Ł}}$ denotes Łukasiewicz negation. Then, fuzzyDL solves the instance retrieval of C with respect to $\mathcal{O}$ by computing $\mathsf{bed}(\mathcal{O}, a{:}C)$ for every individual $a \in \mathcal{O}$.

Regarding FRESG and YADLR, no specific reasoning algorithms to solve those tasks are reported. FRESG computes instance retrieval and realization by reducing to several tableaux algorithm tasks, and YADLR to several BEDs. (Strictly speaking, YADLR checks if an individual is a member of a given concept with an unknown degree of truth, represented using a variable.)

That is, if the fuzzy ontology has n_i individuals and n_c atomic concepts, existing algorithms require n_i tests to solve the instance retrieval and n_c tests to solve the realization problem.

2.4. The fuzzyDL Reasoning Algorithm

In the rest of this section, we will give some more details about fuzzyDL reasoning algorithm [18], as we will extend it in the next sections. The algorithm is based on a combination of a tableaux algorithm and an optimization problem with respect to a constraint set $\mathbf{C}$.

- Firstly, there is some pre-processing. For example, in Łukasiewicz and Zadeh fuzzy DLs, each concept is transformed into its negation normal form [18], where the negation only appears before atomic concepts.
- Then, for each concept assertion $\langle a : C \geq \alpha \rangle \in \mathcal{O}$, it considers a node v_a, ensures that $\mathcal{L}(v_a) \leftarrow \mathcal{L}(v_a) \cup \{C\}$, and sets $\mathbf{C} \leftarrow \mathbf{C} \cup \{x_{v_a:C} \geq \alpha\}$.
- Similarly, for each property assertion $\langle (a_1, a_2) : R \geq \alpha \rangle \in \mathcal{O}$, it considers two nodes v_{a_1}, v_{a_2}, creates an edge $\langle v_{a_1}, v_{a_2} \rangle$ if it does not exist, and sets $\mathcal{L}(\langle v_{a_1}, v_{a_2} \rangle) \leftarrow \mathcal{L}(\langle v_{a_1}, v_{a_2} \rangle) \cup \{R\}$ and $\mathbf{C} \leftarrow \mathbf{C} \cup \{x_{(v_{a_1}, v_{a_2}):R} \geq \alpha\}$.
- Next, it applies some tableau rules. As usual in classical DLs, rules transform complex concept expressions into simpler ones, but in the fuzzy case they also create a set of linear in-equation constraints. Table 3 shows the rules for $\mathcal{ALC}$, and the encoding of the fuzzy operators is shown in Table 4 for Łukasiewicz (Ł), Gödel (G), and Zadeh (Z) fuzzy logics, where $\epsilon > 0$. These inequations have to hold in order to respect the semantics of the DL constructors.
- When no more rules can be applied, the reasoner solves an optimization problem with respect to $\mathbf{C}$. This problem has a solution iff the fuzzy ontology is consistent [18]. To solve the optimization problem, fuzzyDL uses Gurobi (http://www.gurobi.com) optimization problem solver.

Remark 1. *For simplicity, Table 3 assumes that standard Łukasiewicz negation is representable, so that concepts can be represented in negation normal form (NNF). This is obviously the case in Zadeh and Łukasiewicz fuzzy DLs, but in Gódel $\mathcal{ALC}$it requires extending the logic with standard negation, as fuzzyDL does. Similar rules can be defined when concepts cannot be represented in NNF (e.g., in Gódel DLs); see, e.g., [4]. Please also note*

that in Zadeh DLs it is usual to consider two different fuzzy implications, one for universal restriction concepts $\forall R.C$ and another one for subclass axioms $C_1 \sqsubseteq C_2$ [32].

In Łukasiewicz, Gödel, and Zadeh fuzzy DLs, we obtain a bounded mixed integer linear programming (MILP) problem [21]; in other fuzzy DLs more complex optimization problems can be obtained. A MILP problem consists of minimizing a linear function with respect to a set of constraints $\mathbf{C}$ that are linear inequations in which rational and integer variables can occur [24]. In particular, a constraint set $\mathbf{C}$ can contain linear equations $w_1x_1 + \cdots + w_nx_n \bowtie w_0$ or restrictions of the form $x_i \in \{0,1\}$ (forcing x_i to be a binary value), where x_i denotes a variable taking values in $\mathbb{R} \cap [0,1]$, w_i denotes a rational number, and $\bowtie \in \{\leq, \geq, =\}$.

Table 3. Rules of the tableaux algorithm combined with an optimization problem [33].

Rule	Preconditions	Actions
$(\bot)$	$\bot \in \mathcal{L}(v)$	$\mathbf{C} = \mathbf{C} \cup \{x_{v:\bot} = 0\}$
$(\top)$	$\top \in \mathcal{L}(v)$	$\mathbf{C} = \mathbf{C} \cup \{x_{v:\top} = 1\}$
$(\neg)$	$\neg A \in \mathcal{L}(v)$	$\mathbf{C} = \mathbf{C} \cup \{x_{v:\neg C} = \ominus x_{v:C}\}$
$(\sqcap)$	$C_1 \sqcap C_2 \in \mathcal{L}(v)$	$\mathcal{L}(v) = \mathcal{L}(v) \cup \{C_1, C_2\}$ $\mathbf{C} = \mathbf{C} \cup \{x_{v:C} \otimes x_{v:D} = x_{v:C_1 \sqcap C_2}\}$
$(\sqcup)$	$C_1 \sqcup C_2 \in \mathcal{L}(v)$	$\mathcal{L}(v) = \mathcal{L}(v) \cup \{C_1, C_2\}$ $\mathbf{C} = \mathbf{C} \cup \{x_{v:C} \oplus x_{v:D} = x_{v:C_1 \sqcup C_2}\}$
$(\exists)$	$\exists R.C \in \mathcal{L}(v)$	create a new node w $\mathcal{L}(\langle v,w\rangle) = \mathcal{L}(\langle v,w\rangle) \cup \{R\}$, and $\mathcal{L}(w) = \mathcal{L}(w) \cup \{C\}$, and $\mathbf{C} = \mathbf{C} \cup \{x_{(v,w):R} \otimes x_{w:C} = z, z \geq x_{v:\exists R.C}\}$
$(\forall)$	$\forall R.C \in \mathcal{L}(v)$ $R \in \mathcal{L}(\langle v,w\rangle)$	$\mathcal{L}(w) = \mathcal{L}(w) \cup \{C\}$ $\mathbf{C} = \mathbf{C} \cup \{x_{w:C} \geq z, z = x_{v:\forall R.C} \otimes x_{(v,w):R}\}$
$(\rightarrow)$	$C_1 \rightarrow C_2 \in \mathcal{L}(v)$	$\mathcal{L}(v) = \mathcal{L}(v) \cup \{\neg C_1, C_2\}$ $\mathbf{C} = \mathbf{C} \cup \{x_{v:C_1} \Rightarrow x_{v:C_2} = x_{v:C_1 \rightarrow C_2}\}$

Table 4. Encoding of some popular fuzzy logic operators [33].

Restriction	Logic	Encoding
$\ominus x = z$	G	$\{z \leq 1 - x, x + z \geq \epsilon, z \in \{0,1\}\}$
	Ł, Z	$\{1 - x = z\}$
$x_1 \otimes x_2 = z$	G, Z	$\{z \leq x_1, z \leq x_2, x_1 \leq z + y, x_2 \leq z + (1-y), y \in \{0,1\}\}$
	Ł	$\{x_1 + x_2 - 1 \leq z, x_1 + x_2 - 1 \geq z - y, z \leq 1 - y, y \in \{0,1\}\}$
$x_1 \oplus x_2 = z$	G, Z	$\{z \geq x_1, z \geq x_2, x_1 + y \geq z, x_2 + (1-y) \geq z, y \in \{0,1\}\}$
	Ł	$\{x_1 + x_2 \leq z + y, y \leq z, x_1 + x_2 \geq z, y \in \{0,1\}\}$
$x_1 \Rightarrow x_2 = z$	G	$\{2y + x_1 \geq x_2 + \epsilon, x_1 \leq x_2 + (1-y), y + x_2 \geq z, x_2 \leq z + y, z \geq y, y \in \{0,1\}\}$
	Ł	$\{1 - x_1 + x_2 \leq z + y, y \leq z, 1 - x_1 + x_2 \geq z, y \in \{0,1\}\}$
	Z	$\{z \geq 1 - x_1, z \geq x_2, (1 - x_1) + y \geq z, x_2 + (1-y) \geq z, y \in \{0,1\}\}$

Let the connection relation $\rightsquigarrow_{\mathbf{C}}$ between two variables z_1, z_2 be defined as follows: define $z_1 \rightsquigarrow_{\mathbf{C}} z_2$ if $\exists \psi \in \mathbf{C}$ with a term w_1z_1 and a term w_2z_2, and then, extend $\rightsquigarrow_{\mathbf{C}}$ to its transitive closure. The following result can be shown:

Lemma 1 ([20]). *A constraint set $\mathbf{C}$ can be partitioned into a set of constraint sets $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m\}$ verifying the following conditions:*

(OP1) *If z occurs in $\mathbf{C}_i$, then z does not occur in $\mathbf{C}_j$, $\forall i, j \in \{1, \ldots, m\}, i \neq j$;*
(OP2) $\bigcup_{i \in \{1,\ldots,m\}} \mathbf{C}_i = \mathbf{C}$;
(OP3) *$\forall z_j, z_k$ occurring in $\mathbf{C}_i, z_j \rightsquigarrow_{\mathbf{C}} z_k$. $\forall i \in \{1, \ldots, m\}$.*

$\mathbf{C}$ has a solution iff the $\mathbf{C}_i$ has a solution for every $i \in \{1, \ldots, m\}$. Furthermore, given the MILP problem of minimizing an objective variable z with respect to $\mathbf{C}$, the solution is the same as the solution of minimizing with respect to $\mathbf{C}_j$, where $\mathbf{C}_j$ is the optimization problem where z appears.

Computing the partition can be reduced to the problem of computing all the connected subgraphs of a graph. Given a constraint set $\mathbf{C}$, it is possible to build an undirected graph with as many nodes as variables in $\mathbf{C}$ and an edge linking two nodes n_{z_1}, n_{z_2} for every pair of variables z_1 and z_2 appearing in the same constraint $\psi \in \mathbf{C}$. The connected subgraphs indicate the contraint sets; i.e., if a variable z is in the i-th connected component, then the constraints where z occurs should be placed in $\mathbf{C}_i$ [20].

3. Realization in Fuzzy Ontologies

The intuition behind our algorithm is to reduce the number of optimization problems to be solved by merging them. Clearly, optimization problems cannot be merged in general, as Example 1 shows.

Example 1. *Consider an ontology $\mathcal{O}$ with the axiom*

$$\langle \mathsf{johnSmith} : \mathsf{DemocratVoter} \sqcup \mathsf{RepublicanVoter} \geq 1 \rangle$$

stating that citizen JohnSmith either voted for the Democratic Party or for the Republican Party in the last USA elections. If we want to retrieve all the concepts johnSmith *belongs to, the answer should be an empty set, because we cannot infer that he is a* DemocratVoter *and we cannot infer that he is a* RepublicanVoter*. In Łukasiewicz fuzzy DLs, the axiom in $\mathcal{O}$ leads to a constraint*

$$x_{johnSmith:DemocratVoter} + x_{johnSmith:RepublicanVoter} \geq 1 \tag{2}$$

We can indeed compute the realization problem by (i) adding $\langle \mathsf{johnSmith} : \neg C \geq x_C \rangle$ to $\mathcal{O}$, where x_C is a new variable, for one of the atomic concepts $C \in KB$, (ii) minimizing x_C, and (iii) repeating the process for each of the atomic concepts in $\mathcal{O}$. For example, adding $\langle \mathsf{johnSmith} : \neg \mathsf{DemocratVoter} \geq x_{DemocratVoter} \rangle$ leads to a constraint

$$x_{johnSmith:DemocratVoter} \leq x_{DemocratVoter} \tag{3}$$

and the minimum value of $x_{DemocratVoter}$ with respect to Equations (2) and (3) is 0; i.e., there is a solution to the MILP problem such that $x_{DemocratVoter} = 0$ (and $x_{johnSmith:RepublicanVoter} = 1$). However, if we also added $\langle \mathsf{johnSmith} : \neg \mathsf{RepublicanVoter} \geq x_{RepublicanVoter} \rangle$, we would have a constraint

$$x_{johnSmith:RepublicanVoter} \leq x_{RepublicanVoter} \tag{4}$$

Now, in any of the solutions of the optimization problem in Equations (2)–(4), $x_{DemocratVoter} > 0$ or $x_{RepublicanVoter} > 0$ hold, it is incorrect.

However, we can merge optimization problems as long as the involved variables are independent. Based on this idea, Algorithm 1 solves the realization problem of an individual a given a fuzzy ontology $\mathcal{O}$. The output is a (possibly empty) list of pairs of the form $\langle A, \alpha \rangle$, where $A \in \mathcal{O}$ is an atomic concept, $\alpha > 0$ and $\mathcal{O} \models \langle a : A \geq \alpha \rangle$.

Lines 1–6 add a new assertion (involving a new variable) for each named concept in the ontology, create an empty list of results L, and apply some reasoning rules that create a set of MILP constraints. The new assertions are similar to those in the previous algorithm implemented in fuzzyDL [18], but now we create them in the same step.

Lines 7–10 partitions the single constraint set with respect to **C** into a set of constraint sets $\mathbf{C}_i$. This is similar to the approach in [20]. However, we also compute the number of variables to be minimized $x_{a:C}$ in each of the problems so that we can consider three cases:

- Constraint sets without an objective variable (so that we only need to check if they have a solution);
- Constraint sets with exactly one objective variable of the form $x_{a:C}$;
- Constraint sets with more than one objective variable of the form $x_{a:C}$ (so they are dependent variables).

Lines 11–14 address the first case. Constraint sets are merged and we check if there is a solution to the merged constraint set to guarantee that there are not inconsistencies. We could also solve the problems independently, but some experiments showed empirically that it is faster to solve a single problem [20].

Algorithm 1 Algorithm to compute the realization problem given a fuzzy ontology $\mathcal{O}$.

Input: A fuzzy ontology $\mathcal{O}$ and an individual a

Output: A set of pairs individual–membership degree $\{\langle A_1, \alpha_1\rangle, \ldots, \langle A_n, \alpha_n\rangle\}$

1: **for** each atomic concept $A \in \mathcal{O}$ **do**
2: $x_{a:A}$ = new variable
3: $\mathcal{O} \cup \langle a : \neg A, 1 - x_{a:A}\rangle$
4: **end for**
5: $L \leftarrow \emptyset$
6: $\mathbf{C} \leftarrow ApplyReasoningRules(\mathcal{O})$
7: $\mathbf{C}_i \leftarrow Partition(\mathbf{C})$
8: **for** each $\mathbf{C}_i$ **do**
9: $v[i] \leftarrow$ number of variables $x_{a:A} \in \mathbf{C}_i$
10: **end for**
11: $\mathbf{C}_{zero} \leftarrow \bigcup \mathbf{C}_i$ s.t. $v[i] = 0$
12: **if** $\mathbf{C}_{zero}$ does not have a solution **then**
13: **return** $\emptyset$
14: **end if**
15: $\mathbf{C}_{one} \leftarrow \bigcup \mathbf{C}_i$ s.t. $v[i] = 1$
16: **if** $\mathbf{C}_{one}$ does not have a solution **then**
17: **return** $\emptyset$
18: **end if**
19: Minimize z s.t. $\mathbf{C}_{one} \cup \{z = \sum x_{a:A}\}$
20: **for** each solution $x_{a:A}$ in the model of the solution **do**
21: **if** $x_{a:A} > 0$ **then**
22: $L \leftarrow L \cup \langle A, \alpha\rangle$
23: **end if**
24: **end for**
25: $\mathbf{C}_{two\_or\_more} \leftarrow (\mathbf{C} \setminus \mathbf{C}_{zero}) \setminus \mathbf{C}_{one}$
26: **if** $\mathbf{C}_{two\_or\_more}$ does not have a solution **then**
27: **return** $\emptyset$
28: **end if**
29: **for** each $x_{a:A} \in \mathbf{C}_{two\_or\_more}$ **do**
30: $\alpha \leftarrow \min x_{a:A}$ s.t. $\mathbf{C}_{two\_or\_more}$
31: **if** $\alpha > 0$ **then**
32: $L \leftarrow L \cup \langle A, \alpha\rangle$
33: **end if**
34: **end for**
35: **return** L

Lines 15–24 address the second case. Constraint sets are merged and we optimize with respect to a variable with a value equal to the sum of all the variables $x_{a:C}$ to be minimized. This is possible only because all variables $x_{a:C}$ are independent: in this case the minimum of the sum occurs when all the variables have its minimum value. The value of each $x_{a:C}$ is added to the list of results if it is greater than 0.

Finally, Lines 25–34 address the third case. In this case, constraint sets are merged, but the merged problem is optimized independently with respect to a single variable, and this is repeated for each variable $x_{a:C}$ introduced in Lines 1–6 that belongs to $\mathbf{C}_{two\_or\_more}$. The minimal value of each $x_{a:C}$ is added to the list of results if it is greater than 0.

An alternative to the third case is not to merge all the constraint sets into a single one and optimize them independently with respect to each $x_{a:C}$. Note that each independent constraint set would need to be optimized two or more times.

Another alternative is to merge the first and the second case to optimize a single optimization problem. This seems more promising in practice, as the evaluation in [20] showed that solving a problem is not more expensive than solving two simpler ones, disjoint subsets of the original one.

A particularly interesting case happens when $\mathbf{C}_{two\_or\_more}$ is empty. In this case, we can solve a single optimization problem (with the union of the first and the second case). However, in practice, those cases might not happen very often. For example, Example 2 shows that having disjoint concept axioms introduces dependencies.

Example 2. *If there is an axiom stating that two concepts C_1 and C_2 are disjoint, this introduces a constraint $x_{a:C_1} \otimes x_{a:C_2} = 0$ for each individual a in the ontology, so variables $x_{a:C_1}$ and $x_{a:C_2}$ are dependent. Computing the realization of any individual a requires adding two assertions $\cup\langle a : \neg_{\text{Ł}} C_1, 1 - x_1\rangle$ and $\cup\langle a : \neg_{\text{Ł}} C_2, 1 - x_2\rangle$, Then, the variables $x_{a:C_1}$ and $x_{a:C_2}$ are connected ($x_{a:C_1} \rightsquigarrow_{\mathbf{C}} x_{a:C_2}$) and so are x_1, x_2, so $\mathbf{C}_{two\_or\_more}$ is not empty (there is a partition that contains at least two objective variables, x_1, x_2).*

4. Instance Retrieval in Fuzzy Ontologies

Algorithm 1 can be easily adapted to the instance retrieval problem. Algorithm 2 solves the instance retrieval of a fuzzy concept C given a fuzzy ontology $\mathcal{O}$. The output is a (possibly empty) list of pairs of the form $\langle a, \alpha\rangle$, where $a \in \mathcal{O}$ is an individual, $\alpha > 0$ and $\mathcal{O} \models \langle a : C \geq \alpha\rangle$.

Lines 1–6 are similar to the same lines in Algorithm 1, but now we add a new assertion for each named concept in the ontology. Then, Lines 7–10 partitions the single constraint set into several sets $\mathbf{C}_i$. Next, we address the same cases: Lines 11–14 address the first case, Lines 15–24 address the second case, and Lines 25–34 address the third case.

We can also consider the same alternatives as in the realization problem: in the third case it is possible not to merge all the constraint sets, and the constraint sets in the first and the second cases can be merged.

As our experiments show, the particularly interesting case when $\mathbf{C}_{two\_or\_more}$ is empty happens relatively often. In this case, we can solve a single optimization problem (with the union of $\mathbf{C}_{zero}$ and $\mathbf{C}_{one}$). However, we have identified some cases where $\mathbf{C}_{two\_or\_more}$ is not empty, described in Examples 3 and 4.

Example 3. *Assume that a fuzzy ontology $\mathcal{O}$ has a range axiom (stating that the range of an object property R is C) and two object property assertions stating that individuals a_1, a_2 are related via R with an individual a_3. Assume also that we want to retrieve the instances of a concept D such that $\langle D \sqsubseteq C \geq \alpha\rangle \in \mathcal{O}$, so that we need to add assertions $\langle a_1 : \neg_{\text{Ł}} D, 1 - x_1\rangle$ and $\langle a_2 : \neg_{\text{Ł}} D, 1 - x_2\rangle$. Then, the variables $x_{a_1:D}$ and $x_{a_2:D}$ are connected ($x_{a_1:D} \rightsquigarrow_{\mathbf{C}} x_{a_2:D}$) and so are x_1, x_2, so $\mathbf{C}_{two\_or\_more}$ is not empty (there is a partition that contains at least two objective variables, x_1, x_2). Note that this does not happen if both a_1, a_2 are related via R to a_3 but there is not a range axiom. Although a_1, a_2, and a_3 belong to the same ABox partition in the sense of [19], they do not belong to the same optimization problem partitioning.*

Example 4. *Assume again that we want to retrieve the instances of a concept D. In more expressive languages with nominals, there is an additional rule called $\exists_a$ [34] that adds a constraint of the form $x_{a:\{a\}} \Rightarrow (x_{a_i:\exists R.\{a\}} \Rightarrow x_{(a_i,a):R}) \geq 1$ for each individual a_i related to a. If there are two individuals a_1, a_2 related via R to a_3, $x_{a_1:D} \rightsquigarrow_{\mathbf{C}} x_{a_2:D}$, because both of them are connected to $x_{a:\{a\}}$. Therefore, $\mathbf{C}_{two\_or\_more}$ is not empty.*

Algorithm 2 Algorithm to compute the instance retrieval problem given a fuzzy ontology $\mathcal{O}$.

Input: A fuzzy ontology $\mathcal{O}$ and a fuzzy concept C

Output: A set of pairs individual–membership degree $\{\langle a_1, \alpha_1 \rangle, \ldots, \langle a_n, \alpha_n \rangle\}$

for each individual $a \in \mathcal{O}$ **do**
  $x_{a:C}$ = new variable
  $\mathcal{O} \cup \langle a : \neg C, 1 - x_{a:C} \rangle$
end for
$L \leftarrow \emptyset$
$\mathbf{C} \leftarrow ApplyReasoningRules(\mathcal{O})$
$\mathbf{C}_i \leftarrow Partition(\mathbf{C})$
for each $\mathbf{C}_i$ **do**
  $v[i] \leftarrow$ number of variables $x_{a:C} \in \mathbf{C}_i$
end for
$\mathbf{C}_{zero} \leftarrow \bigcup \mathbf{C}_i$ s.t. $v[i] = 0$
if $\mathbf{C}_{zero}$ does not have a solution **then**
  return $\emptyset$
end if
$\mathbf{C}_{one} \leftarrow \bigcup \mathbf{C}_i$ s.t. $v[i] = 1$
if $\mathbf{C}_{one}$ does not have a solution **then**
  return $\emptyset$
end if
Minimize z s.t. $\mathbf{C}_{one} \cup \{z = \sum x_{a:C}\}$
for each solution $x_{a:C}$ in the model of the solution **do**
  if $x_{a:C} > 0$ **then**
    $L \leftarrow L \cup \langle a, \alpha \rangle$
  end if
end for
$\mathbf{C}_{two\_or\_more} \leftarrow (\mathbf{C} \setminus \mathbf{C}_{zero}) \setminus \mathbf{C}_{one}$
if $\mathbf{C}_{two\_or\_more}$ does not have a solution **then**
  return $\emptyset$
end if
for each $x_{a:C} \in \mathbf{C}_{two\_or\_more}$ **do**
  $\alpha \leftarrow \min x_{a:C}$ s.t. $\mathbf{C}_{two\_or\_more}$
  if $\alpha > 0$ **then**
    $L \leftarrow L \cup \langle a, \alpha \rangle$
  end if
end for
return L

5. Evaluation of the Instance Retrieval Algorithm

To evaluate our novel algorithms, we compare the new implementation of the instance retrieval algorithm with the previous algorithm implemented in the fuzzyDL ontology reasoner. In our implementation, we chose to merge $\mathbf{C}_{zero}$ and $\mathbf{C}_{one}$, so if $\mathbf{C}_{two\_or\_more}$ is empty, we only need to solve one optimization problem. Because fuzzyDL did not previously implement an algorithm for concept realization, the evaluation of that algorithm is left as future work.

5.1. Datasets

We firstly considered Fuzzy Beer, a fuzzy ontology with information about beers used in GimmeHop, a knowledge-based recommender system for mobile devices [7]. Fuzzy Beer ontology is able to represent concepts (e.g., beer types or breweries), object properties (e.g., between beers and breweries), data properties (e.g., alcohol level ABV, color, or bitterness), instances (e.g., beers and countries), and fuzzy datatypes. In particular, Fuzzy Beer has 15,317 beer individuals.

Fuzzy datatypes are the only fuzzy element of the ontology, and make it possible to associate linguistic labels to some data properties. For instance, it is possible to define the fuzzy datatypes VeryLowAlcohol, LowAlcohol, NeutralAlcohol, HighAlcohol, and VeryHighAlcohol, as illustrated in Figure 1. Those fuzzy datatypes values were obtained using Datil [35], a software to learn fuzzy datatypes from real data using different clustering algorithms.

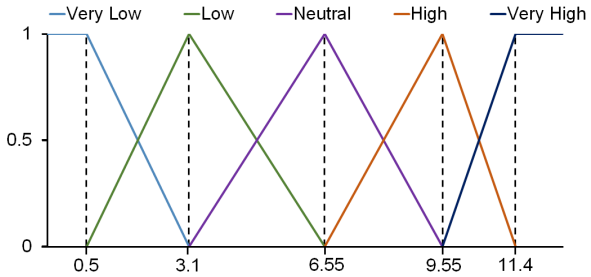

Figure 1. Linguistic labels for the alcohol level data property.

Fuzzy Beer was originally developed in the language Fuzzy OWL 2 [36], which extends classical OWL 2 with OWL 2 annotations encoding the fuzzy information (only those parts that cannot be represented in OWL 2 using an XML-like syntax). For the experiments in this paper, we firstly translated the ontology into FDL format, the native syntax supported by fuzzyDL, using an existing parser [36]. The parser discarded a few axioms that fuzzyDL was not able to support (one for each instance of the class Country), making it possible to deduce the country associated to a beer given the brewery associated to a beer, and the country associated to a brewery [7].

Because of the number of individuals, running time is very high. Indeed, the old approach takes several days to finish an instance retrieval query. Therefore, we restricted the run to several subsets of the Fuzzy Beer ontology, with different numbers of beers. In the following, we use $Beer_n$ to denote the subset of Fuzzy Beer with n beer individuals. Note that the number of individuals is higher than n, as there are also breweries and countries.

First, we solved $Beer_{500}$ 20 times and studied the standard deviation. In particular, we repeated, 20 times, the process of randomly selecting a subset of Fuzzy Beer with 500 beers and solved the instance retrieval problem. The average, standard deviation, and coefficient of variation (or CV, defined as the ratio of the standard deviation to the mean) are shown in Table 5 for both the old and the new algorithm. The new algorithm has a slightly higher CV (6.6% versus 4.3%) but it is still rather stable. Therefore, for the rest of the fuzzy ontologies $Beer_i$ we just solved once the instance retrieval problem in order to decrease the time needed to finish our experiments.

Table 5. Coefficient of variation for the Fuzzy $Beer_{500}$ ontology.

Measure	Old	New
Average (ms)	80,126.0	2047.7
Standard deviation	3438.7	135.3
Coefficient of variation	4.3%	6.6%

In Fuzzy Beer (and in its subsets Fuzzy $Beer_i$), $\mathbf{C}_{two\_or\_more}$ is empty, so it is possible to solve the instance retrieval with a single optimization problem. However, we have considered a harder version, Fuzzy $Beer^h$ (with its corresponding subsets $Beer_i^h$), with two additional axioms, stating the range of two object properties (brewedBy and country).

We also considered the 50 fuzzy ontologies in the Absorption dataset developed in [37]. Although there are several fuzzy versions of each crisp ontology, we considered here fuzzy ontologies of the form *l*.66, with a semantics given by Łukasiewicz fuzzy logic and 66% fuzzy axioms.

To make the comparison fair, we slightly optimized the previous algorithm implemented in fuzzyDL. The existing approach simply looped over all concept assertion entailment problems, and for each of them expanded both the original fuzzy ABox and the new fuzzy concept assertions. However, we made sure that the original fuzzy ABox is expanded only the first time and a cloned copy was shared by the next tests.

All experiments were performed on a laptop computer with Intel Core i7-8550U 1.8 GHz, 16 GB RAM under Windows 7 64-bits. We used Java 1.8 and Gurobi 8.1.0 build V8.1.0rc1 (Academic License). In general, we randomly selected an atomic concept to retrieve their instances, but for Fuzzy $Beer_i$ we also considered a complex concept (the list of queries can be found in the Appendix. During our experiments, we set a timeout of six hours to solve the instance retrieval problem using the new algorithm (as the old seems to take more time).

5.2. Results

Table 6 shows the results for 15 fuzzy ontologies of the Absorption dataset. For each ontology we include the number of individuals, the running time (in seconds) of the previous algorithm (denoted Old), the running time (in s) of the novel algorithm (denoted New), and some optional observations.

We can see that the new algorithm outperforms the previous one in the case of consistent ontologies. However, in inconsistent ontologies (process.l.66 and propreo.l.66), the old algorithm solves a simpler case (as it only needs to add a single axiom to find the inconsistency) and finishes faster. In general, all the partitions were independent, so $\mathbf{C}_{two\_or\_more}$ was empty and it was enough to solve a single MILP problem. There were only two exceptions: FuzzyWine.l.66, and lubm.l.66.

Table 6. Running time (s) for the Absorption dataset.

Ontology Name	#Individuals	Old (s)	New (s)	Comments
cancer_my.l.66	20	13	3	
earthrealm.l.66	167	6	0.8	
Economy.l.66	482	9	0.5	
fmaOwlDlComponent_1_4_0.l.66	98	1	0.6	
FuzzyWine.l.66	138	5165	384	9 objective dependent variables
goslim.l.66	79	1	0.2	
GRO.l.66	1	0.4	0.2	
lubm.l.66	115	9409	6027	719 objective dependent variables
people.fd.l.66	22	5	1	
pizza.l.66	5	0.3	0.2	
po.l.66	20	3	0.5	
process.l.66	167	0.68	1.40	Inconsistent ontology
propreo.l.66	46	20,402	20,544	Inconsistent ontology
thesaurus.l.66	8	5	2	
Transportation.l.66	181	4	0.3	

Table 7 shows some information about the fuzzy ontologies in the Absorption dataset that could not be considered: 29 ontologies did not have any individual and seven reached a timeout; in four cases the timeout was not surprising as there were more than 25,000 individuals.

Table 7. Problems found in the Absorption dataset.

Ontology Name	#Individuals	Problem
AirSystem.l.66	0	No individuals
amino-acid.l.66	0	No individuals
atom-common.l.66	0	No individuals
biochemistry-complex.l.66	0	No individuals
chebi.l.66	487,944	Timeout (many individuals)
chemical.l.66	0	No individuals
chemistry-complex.l.66	0	No individuals
cton.l.66	0	No individuals
EMAP.obo.l.66	0	No individuals
FBbt_XP.l.66	25,148	Timeout (many individuals)
FMA.l.66	94,228	Timeout (many individuals)
galen-ians-full-doctored.l.66	0	No individuals
gene_ontology_edit.obo.l.66	0	No individuals
heart.l.66	0	No individuals
legal-action.l.66	0	No individuals
matchmaking.l.66	0	No individuals
mosquito_insecticide_resistance.obo..l.66	0	No individuals
mygrid-moby-service.l.66	0	No individuals
NCI.l.66	0	No individuals
norm.l.66	0	No individuals
ontology.l.66	0	No individuals
organic-compound-complex.l.66	0	No individuals
pathway.obo.l.66	0	No individuals
periodic-table-complex.l.66	0	No individuals
photography.l.66	46	Timeout
PRO.l.66	277,804	Timeout (many individuals)
reaction.l.66	27	Timeout
relative-places.l.66	0	No individuals
SIGKDD-EKAW.l.66	0	No individuals
so-xp.obo.l.66	0	No individuals
spatial.obo.l.66	0	No individuals
subatomic-particle-complex.l.66	0	No individuals
teleost_taxonomy.obo.l.66	0	No individuals
time-modification.l.66	0	No individuals
worm_phenotype_xp.obo.l.66	0	No individuals
yowl-complex.l.66	79	Timeout

Table 8 shows the results for the Fuzzy $Beer_i$ and Fuzzy $Beer_i^h$ ontologies. In this case, we show the number of individuals, the running time (in s) for $Beer_i$ using the old algorithm and the new one, the running time (in s) for $Beer_i^h$ using the new algorithm, and the number of objective dependent variables to solve $Beer_i^h$. The table does not include the number of variables to solve $Beer_i$ because $\mathbf{C}_{two\_or\_more}$ was always empty and it was enough to solve a single MILP problem. Also, the table does not show the time needed by the old algorithm to solve $Beer_i$ because it is very similar to the time to solve the harder version $Beer_i^h$. We show the results for a query involving an atomic concept, but we also tried a complex concept obtaining a similar trend (see Appendix for details).

Similarly as for the Absorption dataset, we can observe that the new algorithm outperforms the previous one, and the improvement gets more spectacular as the number of individuals grows. This is illustrated in Figure 2. The three functions exhibit quadratic growth, but the new algorithms grow in a notably slower way. We can also observe that when it is possible to solve a single MILP problem (in $Beer_i$), the running time of the new algorithm is much smaller than in the harder case ($Beer_i^h$).

Next, we did some experiments with inconsistent versions of the $Beer_i$ and $Beer_i^h$ fuzzy ontologies, obtained by asserting that one the beer instances has two different alcohol levels. The results are shown in Table 9. We can see that the old algorithm is faster than the new one, as with the Absorption dataset. Furthermore, we can observe that the new algorithm performs similarly for $Beer_i$ and $Beer_i^h$. The reason is that although the hard versions of the ontologies require solving several optimization problems; after one of them is found to be inconsistent there is no need to solve the remaining ones. Furthermore, the problems solved by $Beer_i^h$ are smaller than the single problem solved by $Beer_i$.

Table 8. Running time (s) for subsets of the Fuzzy Beer ontology.

	Fuzzy $Beer_i$		Fuzzy $Beer_i^h$	
#Individuals	**Old (s)**	**New (s)**	**New (s)**	**Comments**
500	80	2	15	110 objective dependent variables
1000	432	6	125	212 objective dependent variables
2000	2719	23	1371	427 objective dependent variables
3000	8197	68	5155	641 objective dependent variables
4000	20,434	158	11,184	866 objective dependent variables
5000	36,128	263	18,329	1093 objective dependent variables

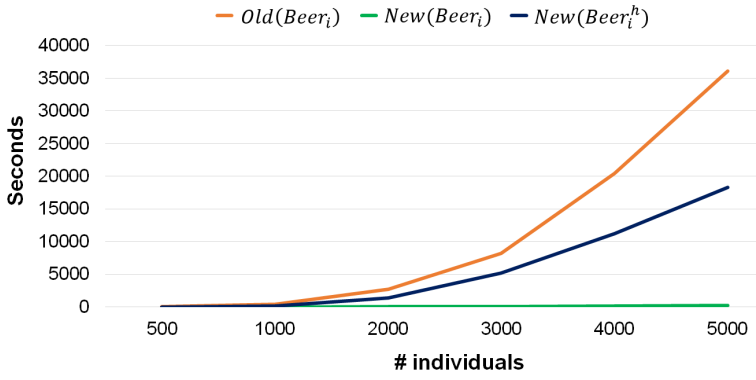

Figure 2. Running time for subsets of the Fuzzy Beer ontology.

Table 9. Running time (s) for subsets of the inconsistent Fuzzy Beer ontology.

	Fuzzy $Beer_i$		Fuzzy $Beer_i^h$
#Individuals	**Old (s)**	**New (s)**	**New (s)**
500	1	2	2
1000	4	5	5
2000	13	22	21
3000	33	62	58
4000	75	121	130
5000	160	222	220

6. Conclusions and Future Work

Fuzzy description logics, the formalism behind fuzzy ontologies, are an important mathematical method for many artificial intelligence scenarios requiring knowledge representation and automate reasoning. In this paper, we have proposed two specific algorithms to solve two important reasoning services given a fuzzy ontology, the instance retrieval and the realization problems. To the best of our knowledge, this is the first work not repeating a (best entailment degree) test for all individuals or concepts of the ontology.

Our approach is based on merging the optimization problems into three optimization problems according to the number of variables to be optimized: zero, one, or more than one. The key is that the first two problems can be solved just once. Furthermore, our experience shows that in practice,

the latter problem is often empty, i.e., instance retrieval often leads to independent optimization problems that can be merged to be solved as a single problem.

It is worth noting that the results of [20] involve reasoning tasks where just one optimization problem needs to be solved. In such cases, splitting the optimization problem into several does not decrease, in general, the running time. In this paper, however, we address problems that require solving several (n) optimization problems. In realization, the basic algorithm requires as many tests as atomic concepts in the ontology; in instance retrieval, as many tests as individuals in the ontology. With our novel algorithm, we decrease the number of optimization problems, and in some particular cases we are able to solve a single one.

Our approach has been implemented in the fuzzyDL fuzzy ontology reasoner and we performed an empirical evaluation with several fuzzy ontologies, some of them with an important number of individuals. Our experiments confirm that our novel algorithm to compute instance retrieval outperforms the previous implementation in all cases involving consistent ontologies, and the reduction of the reasoning time is more important as the number of individuals in the ontology grows. Furthermore, in almost all cases, it was enough to solve a single optimization problem. However, in inconsistent ontologies, the basic algorithm finds the inconsistency faster.

In future work we would like to test the instance retrieval with more fuzzy ontologies, either real or artificial, with more dependent variables. In such cases, we would like to study the best strategy to solve the optimization problems (i.e., merging all problems in $\mathbf{C}_{two\_or\_more}$, solving all of them independently, or using a hybrid approach). Furthermore, we would like to implement and evaluate the realization algorithm as well.

Author Contributions: Conceptualization, I.H. and F.B.; methodology, I.H., J.B., and F.B.; software, I.H. and F.B.; validation, I.H., J.B., and F.B.; investigation, I.H.; writing—original draft preparation, I.H. and F.B.; writing—review and editing, I.H., J.B., and F.B.; visualization, I.H.; supervision, F.B. All authors have read and agreed to the published version of the manuscript.

Funding: I. Huitzil was partially funded by Universidad de Zaragoza—Santander Universidades (Ayudas de Movilidad para Latinoamericanos—Estudios de Doctorado). I. Huitzil and F. Bobillo were partially supported by the projects TIN2016-78011-C4-3-R and JIUZ-2018-TEC-02.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A. List of Queries

Ontology	Query
cancer_my.l.66	WomanUnderIncreasedBRCRisk
earthrealm.l.66	IgeneousRock
Economy.l.66	ElectricDevice
fmaOwlDlComponent_1_4_0.l.66	Right_humerus
FuzzyBeer	Lager
FuzzyBeer	$\exists$ hasABV.LowABV
FuzzyWine.l.66	SweetWine
goslim.l.66	Cytoskeleton
GRO.l.66	BindingToProtein
lubm.l.66	Employee
people.fd.l.66	cat_liker
pizza.l.66	SpicyPizza
po.l.66	Person
process.l.66	Communications
propreo.l.66	HPLC_experimental_data_collection
thesaurus.l.66	astric_Body_Carcinoma
Transportation.l.66	Waterway

References

1. Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P.F. *The Description Logic Handbook: Theory, Implementation, and Applications*; Cambridge University Press: Cambridge, UK, 2003.
2. Cuenca-Grau, B.; Horrocks, I.; Motik, B.; Parsia, B.; Patel-Schneider, P.F.; Sattler, U. OWL 2: The Next Step for OWL. *J. Web Semant.* **2008**, *6*, 309–322.
3. Horrocks, I.; Kutz, O.; Sattler, U. The Even More Irresistible $\mathcal{SROIQ}$. In Proceedings of the 10th International Conference of Knowledge Representation and Reasoning (KR 2006), Lake District, UK, 2–5 June 2006; pp. 452–457.
4. Bobillo, F.; Cerami, M.; Esteva, F.; García-Cerdaña, À.; Peñaloza, R.; Straccia, U. Fuzzy Description Logics. In *Handbook of Mathematical Fuzzy Logic Volume III*; Cintula, P., Fermüller, C., Noguera, C., Eds.; College Publications: London, UK, 2015; Volume 58, Chapter XVI, pp. 1105–1181.
5. Sanchez, E. (Ed.) Capturing Intelligence. In *Fuzzy Logic and the Semantic Web*; Elsevier: Amsterdam, The Netherlands, 2006; Volume 1.
6. Carlsson, C.; Brunelli, M.; Mezei, J. Decision making with a fuzzy ontology. *Soft Comput.* **2012**, *16*, 1143–1152.
7. Huitzil, I.; Alegre, F.; Bobillo, F. GimmeHop: A Recommender System for Mobile Devices using Ontology Reasoners and Fuzzy Logic. *Fuzzy Sets Syst.* **2020**, doi:10.1016/j.fss.2019.12.001.
8. El-Sappagh, S.H.A.; Alonso, J.M.; Ali, F.; Ali, A.; Jang, J.; Kwak, K.S. An Ontology-Based Interpretable Fuzzy Decision Support System for Diabetes Diagnosis. *IEEE Access* **2018**, *6*, 37371–37394.
9. Dasiopoulou, S.; Kompatsiaris, I.; Strintzis, M.G. Investigating Fuzzy DLs-Based Reasoning in Semantic Image Analysis. *Multimed. Tools Appl.* **2010**, *46*, 331–370.
10. Martínez-Cruz, C.; van der Heide, A.; Sánchez, D.; Triviño, G. An approximation to the computational theory of perceptions using ontologies. *Expert Syst. Appl.* **2012**, *39*, 9494–9503.
11. Eich, M.; Hartanto, R.; Kasperski, S.; Natarajan, S.; Wollenberg, J. Towards Coordinated Multirobot Missions for Lunar Sample Collection in an Unknown Environment. *J. Field Robot.* **2014**, *31*, 35–74.
12. Huitzil, I.; Dranca, L.; Bernad, J.; Bobillo, F. Gait Recognition using Fuzzy Ontologies and Kinect Sensor Data. *Int. J. Approx. Reason.* **2019**, *113*, 354–371.
13. Rodger, J.A. A fuzzy linguistic ontology payoff method for aerospace real options valuation. *Expert Syst. Appl.* **2013**, *40*, 2828–2840.
14. Ali, F.; Kwak, D.; Khan, P.; El-Sappagh, S.H.A.; Ali, A.; Ullah, S.; Kim, K.; Kwak, K.S. Transportation sentiment analysis using word embedding and ontology-based topic modeling. *Knowl.-Based Syst.* **2019**, *174*, 27–42.
15. Ali, F.; Islam, S.M.R.; Kwak, D.; Khan, P.; Ullah, N.; Yoo, S.; Kwak, K.S. Type-2 fuzzy ontology-aided recommendation systems for IoT-based healthcare. *Comput. Commun.* **2018**, *119*, 138–155.
16. Ali, F.; Kim, E.K.; Kim, Y. Type-2 fuzzy ontology-based opinion mining and information extraction: A proposal to automate the hotel reservation system. *Appl. Intell.* **2015**, *42*, 481–500.
17. Ali, F.; Khan, P.; Riaz, K.; Kwak, D.; AbuHmed, T.; Park, D.; Kwak, K.S. A Fuzzy Ontology and SVM-Based Web Content Classification System. *IEEE Access* **2017**, *5*, 25781–25797.
18. Bobillo, F.; Straccia, U. The fuzzy ontology reasoner fuzzyDL. *Knowl.-Based Syst.* **2016**, *95*, 12–34, doi:10.1016/j.knosys.2015.11.017.
19. Haarslev, V.; Pai, H.I.; Shiri, N. Optimizing Tableau Reasoning in $\mathcal{ALC}$ Extended with Uncertainty. In Proceedings of the 20th International Workshop on Description Logics (DL 2007), Brixen/Bressanone, Italy, 8–10 June 2007; Volume 250, pp. 307–314.
20. Bobillo, F.; Straccia, U. On Partitioning-Based Optimisations in Expressive Fuzzy Description Logics. In Proceedings of the 24th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015), Istanbul, Turkey, 2–5 August 2015.
21. Straccia, U. Description Logics with Fuzzy Concrete Domains. In Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005), Edinburgh, UK, 26–29 July 2005.
22. Zadeh, L.A. Fuzzy Sets. *Inf. Control* **1965**, *8*, 338–353.
23. Klir, G.J.; Yuan, B. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1995.
24. Bobillo, F.; Straccia, U. Fuzzy Description Logics with General T-norms and Datatypes. *Fuzzy Sets Syst.* **2009**, *160*, 3382–3402.
25. Stoilos, G.; Simou, N.; Stamou, G.; Kollias, S. Uncertainty and the Semantic Web. *IEEE Intell. Syst.* **2006**, *21*, 84–87.

26. Habiballa, H. Resolution Strategies for Fuzzy Description Logic. In Proceedings of the 5th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2007), Ostrava, Czech Republic, 11–14 September 2007; Volume 2, pp. 27–36.
27. Konstantopoulos, S.; Apostolikas, G. Fuzzy-DL Reasoning over Unknown Fuzzy Degrees. In Proceedings of the 3rd International Workshop on Semantic Web and Web Semantics (SWWS 07), Albufeira, Portugal, 25–30 November 2007; Volume 4806, pp. 1312–1318.
28. Bobillo, F.; Delgado, M.; Gómez-Romero, J. DeLorean: A Reasoner for Fuzzy OWL 2. *Expert Syst. Appl.* **2012**, *39*, 258–272.
29. Wang, H.; Ma, Z.M.; Yin, J. FRESG: A Kind of Fuzzy Description Logic Reasoner. In Proceedings of the 20th International Conference on Database and Expert Systems Applications (DEXA 2009), Linz, Austria, 31 August–4 September 2009; Volume 5690, pp. 443–450.
30. Tsatsou, D.; Dasiopoulou, S.; Kompatsiaris, I.; Mezaris, V. LiFR: A Lightweight Fuzzy DL Reasoner. In Proceedings of the 11th Extended Semantic Web Conference (ESWC 2014), Anissaras, Crete, Greece, 25–29 May 2014.
31. Alsinet, T.; Barroso, D.; Béjar, R.; Bou, F.; Cerami, M.; Esteva, F. 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), Washington, DC, USA, 16–18 September 2013; Volume 8078, pp. 325–330.
32. Bobillo, F.; Delgado, M.; Gómez-Romero, J. Crisp Representations and Reasoning for Fuzzy Ontologies. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **2009**, *17*, 501–530.
33. Bobillo, F.; Straccia, U. Generalizing Type-2 Fuzzy Ontologies and Type-2 Fuzzy Description Logics. *Int. J. Approx. Reason.* **2017**, *87*, 40–66.
34. Bobillo, F.; Straccia, U. A MILP-based decision procedure for the (Fuzzy) Description Logic $\mathcal{ALCB}$. In Proceedings of the 27th International Workshop on Description Logics (DL 2014), Vienna, Austria, 17–20 July 2014; Volume 1193, pp. 378–390.
35. Huitzil, I.; Straccia, U.; Díaz-Rodríguez, N.; Bobillo, F. Datil: Learning Fuzzy Ontology Datatypes. In Proceedings of the 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2018), Cádiz, Spain, 11–15 June 2018; pp. 100–112.
36. Bobillo, F.; Straccia, U. Fuzzy Ontology Representation using OWL 2. *Int. J. Approx. Reason.* **2011**, *52*, 1073–1094.
37. Bobillo, F.; Straccia, U. Optimising Fuzzy Description Logic Reasoners with General Concept Inclusions Absorption. *Fuzzy Sets Syst.* **2016**, *292*, 98–129.