TESIS DE LA UNIVERSIDAD DE ZARAGOZA
2023
10

Ignacio Huitzil Velasco

Advanced Management of Fuzzy Semantic Knowledge

Director/es
Bobillo Ortega, Fernando

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

ADVANCED MANAGEMENT OF FUZZY SEMANTIC KNOWLEDGE

Autor

Ignacio Huitzil Velasco

Director/es

Bobillo Ortega, Fernando

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Ingeniería de Sistemas e Informática

2022

Universidad Zaragoza
1542

Tesis Doctoral

Advanced Management of Fuzzy Semantic Knowledge

Autor

Ignacio Huitzil Velasco

Director

Fernando Bobillo Ortega

Universidad de Zaragoza

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura

2022

Universidad Zaragoza

"Advanced Management of Fuzzy Semantic Knowledge"

This thesis dissertation is submitted to the Department of Informatics and Systems Engineering at University of Zaragoza, Spain in partial fulfilment of the requirements for the degree of Doctor

Ignacio Huitzil Velasco

Zaragoza, 2022

Departamento de Informática e Ingeniería de Sistemas

Universidad Zaragoza

To my mother, Silvia

"When the student is ready the teacher shall appear"

–Zen proverb

"Annus mirabilis" "wonderful year"

–John Dryden (poem 1667)

Acknowledgments

All my gratefulness and admiration to my advisor and great human, Fernando Bobillo. For his patience, support, and time to show me how to do a rigorous research. For teaching me to get better results and go on to the next target. His direction was very important for this project, without his help this would have not been possible.

Thank you to Eduardo Mena to allow me to join the Distributed Information System (SID) group. Thanks to every SID mate for showing me their professional work, a friendly way to collaborate, and promoting the meetings and share with you the tapas time.

Special thanks to Miguel Molina from Imperial College London for allowing me to enjoy a four months research stay at the laboratory (Data Science Institute), for making part of our research group, and have a new experience with a new project. I do not forget the city and people I met. The coffee time and the research talks of Luis Baca (Tronco). Thank you to my friend and English teacher George from Acton.

I appreciate the incredible job of the University Staff, Pilar Enguita and Ana Gimeno for their help in the research management process. Also, thanks to the International Staff, Asun Moreno, Araceli Bravo and Eva Pastor for their good job in the scholarship process and all my gratitude to Universidad de Zaragoza and Santander Universidades 2017-2018 who funded part of my PhD studies.

I do not forget my friends from Colegio Mayor Miraflores for their support in my personal life and that show me the value of the life. Thank you to Enrique Cuesta, Dani Alavedra, D. Antonio Schlatterr, Pablo López, Jesús Milan, Paco Baltar. Thanks for shared their time in activities like mountain bike, walks, meditations, discussions, travels, beers or tacos time.

A particular thank you to my friends Kuda and Caeh for reminding me not lose the ground and go on to my dreams. Thank you to each of my friends that share and enjoy the coffee break, lunchtime, beers, great talks, friends from laboratory and that known in the way: Emanuele, Gadiel, Ilaria, Ciro, David, Mariapia, Heide, Bettina, Daniel, Marta, Migue, Estefanía, Giselle, Toño, and more.

To my friends from México who ask about me and help me in the other side of the world: Pilar, Vero, Rube, Tere, Julio, Orlando, Fer, Jesús, Roberto (Torta), Humberto and Doña Isabel.

Thanks to my lovely Mexican family, to my sisters (Anabel, Paloma, Xochitl, and Engracia) and my nephews (Ángel, Gus, and Santi) little devils. Thank you to my

Parents (Silvia and Lázaro) by teaching me that the effort, honesty, dedication, daily work is the success key in the professional work.

Finally, thanks to all the people that do not appear here. I am sorry, you are in my mind.

"¡Gracias totales!" –Gustavo Cerati

Abstract

In recent years, Semantic Web technologies (in particular, ontologies) have become a de facto standard for knowledge representation. Managing uncertain semantic knowledge is a challenging topic in many Artificial Intelligence applications. Indeed, there are many scenarios and real-world domains where one must manage the imprecision and noise of data collected from sensors, the vagueness of perceptual data, the absence or incompleteness of data, the fuzziness due to ill-defined concepts, etc. In this thesis, we will concentrate on the particular case of fuzzy semantic knowledge, i.e., in the subfield of fuzzy ontologies. Although there has been a significant amount of previous work, there are still many open problems.

Firstly, there are not enough examples of publicly available fuzzy ontologies, suggesting that new techniques to build fuzzy ontologies are needed. Secondly, in order to solve some reasoning tasks, some algorithms were proposed to prove that some inference service is decidable, but there are no optimized reasoning algorithms. Thirdly, no attention has been paid to the support of the increasingly important and ubiquitous mobile devices. Last but not least, many of the developed fuzzy ontologies are toy examples, and there is a notable lack of applications to real-world problems.

In this thesis, we develop some advanced strategies, algorithms, and tools to enhance the management of fuzzy ontologies and fuzzy ontology reasoners. In particular, we present new algorithms to learn fuzzy ontologies, novel reasoning algorithms, new methods to manage imprecise knowledge on mobile devices, and the development of real-world applications as a proof of concept of our contributions.

Resumen

En los últimos años, las tecnologías de la Web Semántica (en particular, las ontologías) se han convertido en un estándar de facto para la representación del conocimiento. La gestión del conocimiento semántico incierto es un tema complejo en muchas aplicaciones de Inteligencia Artificial. De hecho, existen múltiples escenarios y dominios del mundo real en los que se debe gestionar la imprecisión y el ruido de los datos recopilados por los sensores, la vaguedad de los datos percibidos, la ausencia o incompletitud de los datos, la existencia de conceptos mal definidos, etc. En esta tesis nos concentraremos en el caso particular del conocimiento semántico difuso, es decir, en el subcampo de las ontologías difusas. Aunque ha habido una cantidad significativa de trabajo previo, todavía existen muchos problemas abiertos.

En primer lugar, no hay suficientes ejemplos de ontologías difusas disponibles públicamente, lo que sugiere que se necesitan nuevas técnicas para construir ontologías difusas. En segundo lugar, para resolver algunas tareas de razonamiento, se han propuesto algoritmos que permiten demostrar que una tarea es decidible, pero no existen algoritmos de razonamiento optimizados para resolverla. En tercer lugar, no se ha prestado atenciónal soporte de los cada vez más importantes y omnipresentes dispositivos móviles. Por último, pero no por ello menos importante, muchas de las ontologías difusas desarrolladas son ejemplos académicos y existe una notable falta de aplicaciones a problemas del mundo real.

En esta tesis, desarrollamos estrategias, algoritmos y herramientas avanzados para mejorar la gestión de ontologías difusas y los razonadores para ontologías difusas. En particular, presentamos nuevos algoritmos para aprender ontologías difusas, novedosos algoritmos de razonamiento, nuevos métodos para gestionar conocimiento impreciso en dispositivos móviles y el desarrollo de aplicaciones del mundo real como prueba de concepto de nuestras contribuciones.

Contents

Chapter 1

Introduction

Semantic Web Technologies, or simply semantic technologies, is a term that includes a wide range of technologies such as ontologies, linked data, or knowledge graphs. The name comes from the fact that these technologies are part of the architecture proposed for the Semantic Web [BHL01], an extension of the current Web proposed in 1991, understandable not only by humans but also by machines. These technologies have become a de facto standard for knowledge representation and have been successfully used in numerous applications in different domains, often unrelated to the Web.

In this thesis, our interest will focus on ontologies, since in our understanding, they are the ones which truly allow defining the semantics of knowledge, making the annotation of data (which can be part of the Linked Data cloud or a knowledge graph) possible. An ontology is defined as "an explicit specification of a conceptualization" [Gru93]. Essentially, ontologies allow defining the vocabulary of interest in a domain of interest in a formal way, understandable by machines, and allowing automatic reasoning to discover implicit knowledge that necessarily follows from explicitly represented knowledge.

The current expansion of Artificial Intelligence allows us to anticipate a high interest in these technologies in coming years. On the one hand, the use of knowledge is an important feature of many intelligent systems; for example, ontologies have been used extensively to store the knowledge in expert systems [YDNG18]. On the other hand, semantic technologies can be used as a common exchange format between different systems or intelligent agents; for example, ontologies have been used in multi-agent systems [HWDC09]. Both approaches are actually not disjoint: for example, ontologies can be used in robotics to make local decisions by a robot and to promote communication between robots [AS21]. Finally, the possibility of using these technologies for eXplainable Artificial Intelligence (XAI) has been raised, for example, by extending the current machine learning systems based on a black box model (such as neural networks or deep learning) with ontologies that promote justifications of the

deductions [CWBM21].

Shortly after the birth of ontologies, the inability to adequately handle knowledge affected by imprecision or vagueness was noticed. Such knowledge is inherent to many real-world domains and is necessary to deal with notions such as expensive, large, or recent. To overcome this limitation, fuzzy ontologies were proposed [Str13], extending classical ontologies with elements from fuzzy set theory and fuzzy logic. Fuzzy sets allow to represent the partial membership of an element to a set, and fuzzy logic allows to manage propositions which are partially true and make deductions through approximate reasoning [Zad65]. It also worth to note that fuzzy systems have also proved to be very useful for XAI [CLDW21]. The emergence of fuzzy ontologies should not come as a surprise, since it is similar to what happened with fuzzy databases [PB96], fuzzy evolutionary computation [Ped97], or fuzzy neural networks [LL04], just to cite some examples. Since both ontologies and fuzzy logic are useful for XAI, fuzzy ontologies could be particularly useful in that task.

The most important formalism used as a theoretical foundation for ontologies are Descriptive Logics (DLs) [BHLS17], a family of logics to represent structured knowledge providing a good trade-off between expressivity and reasoning complexity. In fact, the standard language for ontology representation, Web Ontology Language (OWL), is equivalent to the DL $\mathcal{SROIQ}(\mathbf{D})$. The current version of OWL is OWL 2 [CGHM$^+$08], and it includes several profiles or sublanguages with less expressivity but better computational properties.

If we revisit the history of DLs, we can identify several phases [BHS07]:

- In phase 0, antecedents such as semantic networks or frames were developed.
- In phase 1, implementations of structural-type algorithms were developed, but they only supported inexpressive DL languages.
- In phase 2, implementations of tableaux algorithms were developed, which supported more expressive DLs.
- In phase 3, optimized implementations for very expressive logic were developed.
- In phase 4, "industrial strength DL systems employing very expressive DLs" were developed, together with implementations specifically focused on less expressive languages, such as OWL 2 profiles.

Fuzzy ontologies are based on fuzzy DLs [BCE$^+$15]. Although there has been considerable work in the field, we believe that there is still a long way to go until

they are as mature as classical ones. If we analyze the state of the art, we can conclude that phases 3 and 4 have not yet finished.

Regarding phase 3, although there are several implementations of fuzzy ontology reasoners, many of them focus on poorly expressive languages, and most of them do not implement any optimization techniques. Therefore, we find it necessary to continue delving into the design, implementation, and evaluation of optimized techniques for reasoning in fuzzy ontologies.

Regarding phase 4, there are few real world applications. Furthermore, many of the applications based on fuzzy ontologies (for example, [LJH05]) do not use a formal language based on logic, so it is not possible to perform any reasoning, e.g., it is not possible to automatically check that the knowledge is logically consistent. We believe that there are numerous real-world applications that could benefit from the use of fuzzy ontologies for knowledge representation and reasoning. In addition, these applications would allow a more reliable evaluation of present and future contributions to research in fuzzy ontologies.

A possible reason for the scarcity of real applications based on fuzzy ontologies is the lack of techniques that simplify their construction. Despite the existence of some tools such as fuzzy ontology editors or reasoners, there are few implemented tools easing the learning of fuzzy ontologies. The main exception is the FuzzyDL-Learner system [LS13], which uses inductive reasoning to learn a very specific type of axiom (axioms of inclusion between classes). We think that the existence of new techniques for learning fuzzy ontologies would increase the number of fuzzy ontologies, which in turn would increase the number and size of current datasets.

In addition to the above, in recent times we have witnessed a huge spread of mobile computing in our daily lives. In fact, most of today's web traffic comes from mobile devices. The progress and popularity of different mobile devices (smartphones, tablets, etc.) have attracted a huge developer community that is releasing numerous applications (or apps) continuously. There are currently many interesting applications that take advantage of the user's context (for example, their geographical location). Even if its usefulness is beyond any doubt, it seems interesting to develop semantic apps, incorporating semantic technologies to improve these applications by combining information from ontologies (or fuzzy ontologies) with data obtained from mobile sensors. Despite the importance of mobile computing, semantic techniques have had in general little regard for mobile device support. In the specific case of fuzzy ontologies, the interest has been virtually nonexistent. For example, although there are some semantic reasoners available for mobile devices, the implementations are limited to classical ontologies [BYBM15].

Given that mobile computing poses scenarios where uncertainty management is necessary, we believe in the need to promote the support of fuzzy ontologies on mobile devices, through the adaptation or development of fuzzy ontology management techniques and tools for their use. This would allow increasing the intelligence of existing apps and, at the same time, would result in an increase in the popularity of fuzzy ontologies. It is also interesting to note that mobile devices, which typically have limited resources compared to desktops or cloud servers, are a good example of the importance of developing reasoning optimization techniques with fuzzy ontologies.

1.1 Objectives

Based on the previous discussion, we propose the following research questions:

Q1. Is it possible to develop new techniques helping to create fuzzy ontologies that use machine learning algorithms or other intelligent techniques for learning some of the elements of fuzzy ontologies?

Q2. Is it possible to improve the current reasoning algorithms for fuzzy ontologies through the development of optimization techniques or the identification of new reasoning tasks?

Q3. Is it possible to propose techniques and/or implement tools that improve the support for fuzzy ontologies on mobile devices?

Q4. Is it possible to develop new real applications based on fuzzy ontologies that allow us to evaluate the rest of our contributions?

The general objective of this thesis is to design, implement, and evaluate new techniques for managing fuzzy semantic knowledge. According to this general objective, we can identify the next specific objectives (each of them corresponding to a research question):

O1. To propose innovative algorithms and strategies to help in the process of fuzzy ontology building through the automatic learning of their components.

O2. To develop advanced techniques for reasoning with fuzzy ontologies, including optimization techniques, design of new algorithms, and identification of novel reasoning tasks.

O3. To develop techniques for managing fuzzy ontologies and reasoning with them on mobile devices and to implement tools to promote the use of fuzzy ontologies on such devices.

O4. To develop applications with real-world data as proof of concepts to validate the proposed techniques.

1.2 Structure of the thesis

This dissertation is composed of eight chapters, including this introductory one. In Chapter 2, we present the main previous concepts needed to understand our work. We firstly provide some background on fuzzy logic and fuzzy set theory. Next, we overview both classical ontologies and fuzzy ontologies. We also consider Description Logics and reasoning services for both ontology types. Then, we describe some clustering methods as they will be used in some of our learning approaches. To finish that chapter, we discuss mobile computing and mobile app development.

The next four chapters detail our contributions. The contributions to learning fuzzy ontologies are in Chapter 3. There, we detail our novel strategies to learn fuzzy datatypes from the values of numerical data properties and from the definitions of different experts. We also describe two implementations, Datil and Fudge.

Our contributions to reasoning with fuzzy ontologies are presented in Chapter 4. On the one hand, we improve the existing solutions to solve some problems, namely instance retrieval, realization, similarity between individuals, and fuzzy matchmaking. On the other hand, we propose a new reasoning task, flexible faceted instance retrieval, and solve it by means of a new reasoning algorithms.

Our contributions to the support of fuzzy ontologies on mobile devices can be found in Chapter 5. This includes the identification of some techniques to manage ontologies on devices with hardware limitations, a comparison of local and remote strategies for reasoning, the development of a serializable and incremental version of a fuzzy ontology reasoner promoting hybrid reasoning, and the development of tools and applications. In particular, we develop a mobile app (GimmeHop) as a use case involving real-world fuzzy data and adapt our tools to learn fuzzy ontologies (Datil and Fudge) to mobile devices.

Chapter 6 compiles our practical contributions. On the one hand, we evaluate some of the approaches discussed in the previous chapters. On the other hand, we present several real use cases and applications to validate some of our contributions. These real-world applications include a gait recognition system to identify people from the way they walk, a mobile application to recommend beers considering the preferences of the user (GimmeHop), a system to generate blockchain smart contracts considering partial agreements, a system to categorize the lifestyle of people from real data obtained from wearable devices, and a system to answer flexible queries over Building Information

Modeling (BIM) data.

Finally, Chapter 7 summarizes the main conclusions drawn in this thesis and discusses some possible directions to follow on future researches.

Chapter 2

Background

In this chapter, we describe the necessary concepts for reading the rest of this dissertation. We start by defining fuzzy set theory and fuzzy logic, introduced by L. Zadeh in 1965 [Zad65]. Under this paradigm, we can represent imprecise knowledge and perform approximate reasoning in a similar way to human thinking. For a long time these proposals have been studied and developed with application in many fields, including control systems and information processing.

Another important ingredient for our work are ontologies, that make it possible to add semantics to the representation of the knowledge in an area of interest. Classical ontologies cannot manage imprecise or fuzzy knowledge, but fuzzy ontologies are considered as a good solution. With the help of special software tools called reasoners, it is possible to compute implicit knowledge from explicit knowledge. Reasoners offer many valuable reasoning services to the users.

Some of our contributions for fuzzy ontology learning will be based on clustering machine learning techniques. Therefore, we will overview some centroid-based algorithms. Additionally, because part of our developments are conceived for mobile devices, we will provide some background on mobile computing.

This chapter is organized as follows. In Section 2.1 we recall some key concepts from fuzzy set theory and fuzzy logic, including linguistic variables, logical and aggregation operators, modifiers, and defuzzification methods. Section 2.2 overviews classical ontologies, including Description Logics, the language OWL, and reasoning, covering both traditional tasks and classical reasoners. The extensions to the fuzzy case are recapped in Section 2.3, where we talk about fuzzy ontologies, fuzzy Description Logics, fuzzy reasoners, and the language Fuzzy OWL 2. Next, Section 2.4 presents a brief review of some selected clustering machine learning methods. Finally, Section 2.5 describes some key notions of mobile computing and mobile app development.

2.1 Fuzzy sets and fuzzy logic

2.1.1 Fuzzy sets

In classical set theory, an element x either belongs to (is a member of) a set A or not. Thus, we can define a membership function μ_A as follows:

$$\mu_A(x) = \begin{cases} 1 & \forall x \in A \\ 0 & \forall x \notin A \end{cases}$$

Hence, in a classical set (known as crisp or non-fuzzy) the membership degrees of the elements are in $\{0, 1\}$, which can be modeled using two-valued logic.

Example 1. *Let us show an example from a domain that will be further developed in Section 6.2: beers. A very important issue is selecting the best temperature for enjoying a beer. Some recommendations from the Brewery Consortium are that beers with high levels of alcohol can be warm, but beers with low levels should be cold. A cold temperature can be defined between 2.5 and 7 degrees Celsius*[1]*. Figure 2.1 shows the graphical example of the membership function for the classical set of cold beers. As a specific case, Ámbar Especial is a beer of pale lager style that is usually served in Zaragoza's bars. An Ámbar Especial at 6º C belongs to the class of cold beers, but the temperature raises to 7.2º C, it is not cold anymore (it is out of the class).* □

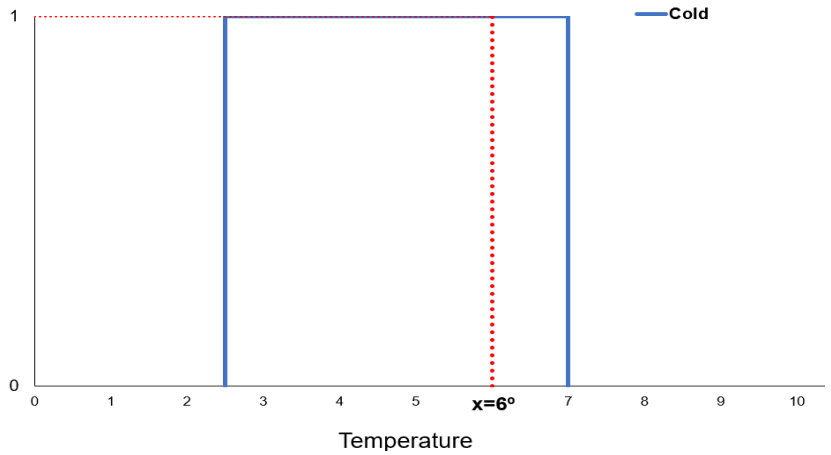


Figure 2.1: Crisp set about cold temperature (for a beer).

In fuzzy set theory elements can *partially* belong to a set, i.e., they can belong to some degree. Let X be a set of elements called the reference set. A *fuzzy subset* A of X is completely and uniquely characterized by a *membership function* $\mu_A(x)$, or simply $A(x)$, which assigns to every $x \in X$ a *degree of truth* [KY95]. Usually, degrees of truth

[1] `http://www.craftibeer.com/beer-focus/warm-or-cold-beer`

are real numbers in the unit interval, so the membership function is of the form

$$\mu_A(x) : X \to [0, 1]$$

When the membership function is evaluated, we get a membership degree in $[0, 1]$ or, equivalently, a degree of the truth for the sentence "x belongs to A". Larger values denote a higher degree of membership to the fuzzy set.

Example 2. *Coming back to the example of beer temperatures, for human thinking, the fact that Ámbar beer at 7.2º C is not cold at all, seems illogical. We can define a new boundary using the membership function in Figure 2.2. Here, temperatures in* $[2.5, 7]$ *fully belong to the set, but for temperatures in* $(7, 8.5)$ *there is a partial membership degree to the set. Therefore, Ámbar beer at 7.2º C is cold with a membership degree of* 0.86. □

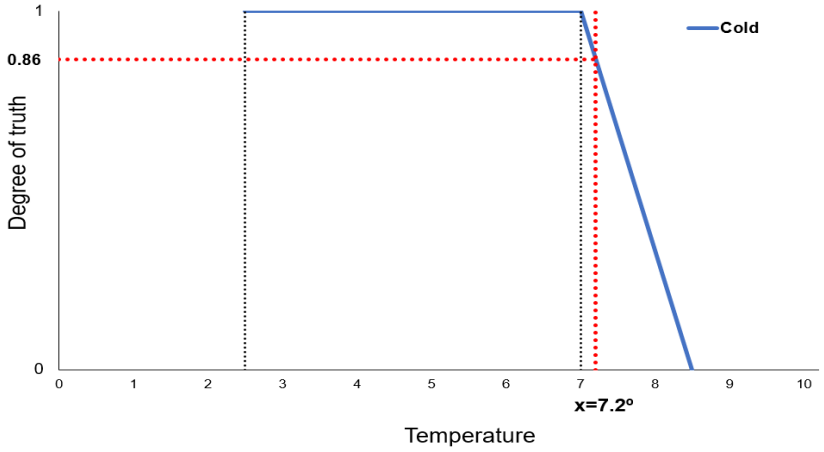


Figure 2.2: Fuzzy membership function of cold temperature (for a beer).

There are many ways to represent membership functions. In our work we will restrict ourselves to the trapezoidal, triangular, left-shoulder, and right-shoulder membership functions, although there are more possibilities such as the bell shaped function (e.g., Gaussian function). Figure 2.3 illustrates such functions, where the X axis corresponds to the universe of discourse and Y corresponds to the membership degree in $[0, 1]$. For the sake of completeness, we provide the definition of each function next:

Trapezoidal. The trapezoidal membership function $P = \mathbf{trap}(q_1, q_2, q_3, q_4)(x)$ is characterized by four real numbers such that $q_1 \leq q_2 \leq q_3 \leq q_4$. The definition of the trapezoidal function is the next one:

- $\mu_P(x) = (x - q_1)/(q_2 - q_1), \forall x \in [q_1, q_2]$

- $\mu_P(x) = 1, \forall x \in [q_2, q_3]$
- $\mu_P(x) = (q_4 - x)/(q_4 - q_3), \forall x \in [q_3, q_4]$
- $\mu_P(x) = 0, \forall x \in [K_1, q_1] \cup [q_4, K_2]$

Triangular. The triangular membership function $T = \mathbf{tri}(q_1, q_2, q_3)(x)$ is characterized by three real numbers where $q_1 \leq q_2 \leq q_3$ and is defined as:

- $\mu_T(x) = (x - q_1)/(q_2 - q_1), \forall x \in [q_1, q_2]$
- $\mu_T(x) = (q_3 - x)/(q_3 - q_2), \forall x \in [q_2, q_3]$
- $\mu_T(x) = 0, \forall x \in [K_1, q_1] \cup [q_3, K_2]$

Left-shoulder. The left-shoulder membership function $L = \mathbf{left}(q_1, q_2)(x)$ is characterized by two real numbers where $q_1 \leq q_2$ and defined as:

- $\mu_L(x) = 1, \forall x \in [K_1, q_1]$
- $\mu_L(x) = (q_2 - x)/(q_2 - q_1), \forall x \in [q_1, q_2]$
- $\mu_T(x) = 0, \forall x \in [q_2, K_2]$

Right-shoulder. The right-shoulder membership function $R = \mathbf{right}(q_1, q_2)(x)$ is characterized by two real numbers where $q_1 \leq q_2$ and defined as:

- $\mu_R(x) = 0, \forall x \in [K_1, q_1]$
- $\mu_R(x) = (x - q_1)/(q_2 - q_1), \forall x \in [q_1, q_2]$
- $\mu_R(x) = 1, \forall x \in [q_2, K_2]$

Notice that all the functions have been defined over an interval $[K_1, K_2]$ rather than over $(-\infty, \infty)$. Clearly, a triangular function $\mathbf{tri}(q_1, q_2, q_3)$ can be represented as a trapezoidal function $\mathbf{trap}(q_1, q_2, q_2, q_3)$. If the right-shoulder and left-shoulder functions are defined over a fixed range $[K_1, K_2]$, then they can also be represented using a trapezoidal function. For example, a left-shoulder function $\mathbf{left}(q_1, q_2)$ can be represented as $\mathbf{trap}(K_1, K_2, q_1, q_2)$. Note also that a crisp number x (a single real number) can be described as $\mathbf{tri}(x, x, x)$.

Finally, because in practice it is often difficult to define precisely the membership function of a fuzzy set, classical fuzzy sets (or *type-1* fuzzy sets) can be generalized: in a *type-2* fuzzy set, the membership function is a (type-1) fuzzy set [Zad75].

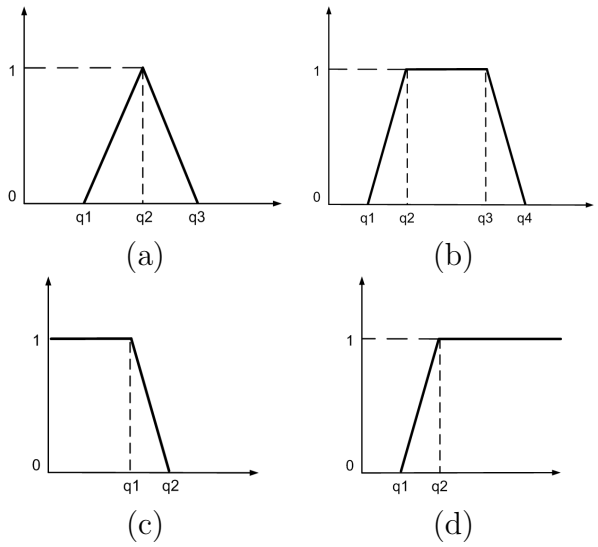


Figure 2.3: (a) Triangular function; (b) Trapezoidal function; (c) Left-shoulder function; (d) Right shoulder function.

2.1.2 Linguistic variables

The paradigm of fuzzy logic also includes a revolutionary concept: *linguistic variables*. In a linguistic variable, the values are not numerical but linguistic. The idea is to use linguistic labels or words, described using a membership function μ_A, rather than numerical values. In our beer scenario (Example 2), we can use the linguistic label *Cold*, defined as μ_{Cold}, as a possible value for the variable *temperature*.

Typically, the domain of a linguistic variable can be partitioned in several membership functions, each of them associated to a linguistic label that is perfectly understandable by humans. For example, Figure 2.4 shows how to partition the domain of the temperatures in the context of beers using a left-shoulder function, two triangular functions, a trapezoidal function and a right function, each of them with a specific linguistic label. A first possibility is using descriptions such as {*VeryCold*, *Cold*, *Cool*, *Warm*, *VeryWarm*}[2]. Another option is expressing how high or low a value is, e.g., {*VeryLow*,*Low*,*Neutral*,*High*,*VeryHigh*}.

Linguistic variables make it possible to treat the imprecision that naturally appears in natural language in a progressive mode.

Example 3. *Returning to the beer domain, we can say that a beer with a temperature of 6º C has:*

- *a* Cold *(or* Low*) temperature with a membership degree of* $\mu_{Cold}(x = 6) = (\mathbf{tri}(4, 5.5, 7))(x = 6) = 0.66$*, and*

[2] `http://www.craftbeerjoe.com/craft-beer-tips/craft-beer-temperature`

– *a* Cool *(or* Neutral*) temperature with a membership degree of* $\mu_{Cool}(x = 6) = (\mathbf{trap}(5.5, 7, 10, 11.5)(x = 6)) = 0.33 \quad \square$.

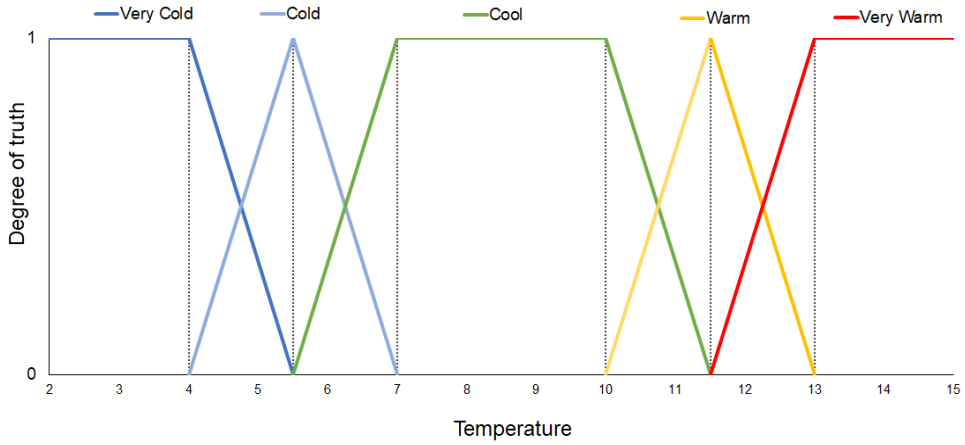


Figure 2.4: Fuzzy sets and linguistic labels about temperatures in the context of beers.

2.1.3 Fuzzy logical operators

In classic set theory, there are three main logical operators which are defined using a bivalued logic (Boolean logic). These operations are intersection (conjunction), union (disjunction), and complement (negation). A fourth logical operator is implication. These operations can be extended to fuzzy sets, as we will show next.

Intersection. The intersection of two fuzzy sets is defined as

$$\mu_{A \cap B}(x) = \mu_A(x) \otimes \mu_B(x)$$

where $\otimes$ denotes a triangular norm or t-norm. Figure 2.5 (a) shows the intersection (bold edge) of two triangular membership functions.

A t-norm is a function $\otimes : [0, 1] \times [0, 1] \to [0, 1]$ which satisfies the following properties:

– Associativity: $\alpha \otimes (\beta \otimes \gamma) = (\alpha \otimes \beta) \otimes \gamma, \forall \alpha, \beta, \gamma \in [0, 1]$

– Commutativity: $\alpha \otimes \beta = \beta \otimes \alpha, \forall \alpha, \beta \in [0, 1]$

– Monotonicity: if $\beta \leq \gamma$, then $\alpha \otimes \beta \leq \alpha \otimes \gamma, \forall \alpha, \beta, \gamma \in [0, 1]$

– Neutral element: $\alpha \otimes 1 = \alpha, \forall \alpha \in [0, 1]$

It can be proved that in every t-norm 0 is an absorbing element: $\alpha \otimes 0 = 0, \forall \alpha \in [0, 1]$.

Now we will define some important families of t-norms [KMP00]. A t-norm is:

– *Continuous* if it is continuous as a function in the $[0,1]^2$ interval.

– *Strict* if it is continuous and strictly monotone.

– *Nilpotent* if it is continuous and each $\alpha \in (0,1)$ is its nilpotent element, i.e., there is some natural number n such that $\alpha \otimes \alpha \otimes \ldots \alpha$ (n times) $= 0$.

Union. The union of two fuzzy sets is defined as

$$\mu_{A \cup B}(x) = \mu_A(x) \oplus \mu_B(x)$$

where $\oplus$ denotes a t-conorm (also called s-norm). Figure 2.5 (b) shows the union (bold edge) of two triangular membership functions.

A t-conorm is a function $\oplus : [0,1] \times [0,1] \rightarrow [0,1]$ which satisfies the following properties:

– Associativity: $\alpha \oplus (\beta \oplus \gamma) = (\alpha \oplus \beta) \oplus \gamma, \forall \alpha, \beta, \gamma \in [0,1]$

– Commutativity: $\alpha \oplus \beta = \beta \oplus \alpha, \forall \alpha, \beta \in [0,1]$

– Monotonicity: if $\beta \leq \gamma$, then $\alpha \oplus \beta \leq \alpha \oplus \gamma, \forall \alpha, \beta, \gamma \in [0,1]$

– Neutral element: $\alpha \oplus 0 = \alpha, \forall \alpha \in [0,1]$

Complement. The complement of a fuzzy set is defined as:

$$\mu_{\bar{A}}(x) = \ominus \mu_A(x)$$

where $\ominus$ denotes a negation function. Figure 2.5 (c) shows the complement (bold edge) of a triangular membership function.

A negation is a function $\ominus : [0,1] \rightarrow [0,1]$ which satisfies the following properties:

– Monotonicity: if $\alpha \leq \beta$, then $\ominus \alpha \geq \ominus \beta, \forall \alpha, \beta \in [0,1]$

– Boundary conditions: $\ominus 0 = 1$ and $\ominus 1 = 0$

Implication. An implication is a function $\Rightarrow : [0,1] \rightarrow [0,1]$ which satisfies the following properties:

– Antitonicity: if $\alpha \leq \beta$, then $\alpha \Rightarrow \gamma \geq \beta \Rightarrow \gamma, \forall \alpha, \beta, \gamma \in [0,1]$

– Monotonicity: if $\beta \leq \gamma$, then $\alpha \Rightarrow \beta \leq \alpha \Rightarrow \gamma, \forall \alpha, \beta, \gamma \in [0,1]$

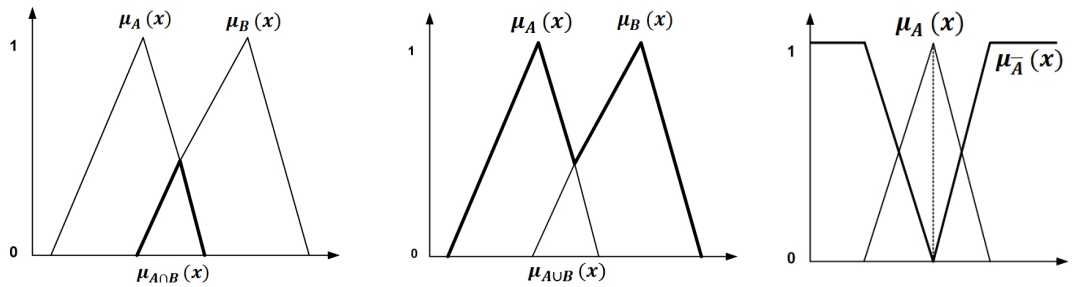


Figure 2.5: (a) Intersection; (b) union; (c) complement of fuzzy sets.

– Boundary conditions: $0 \Rightarrow \alpha = 1, \alpha \Rightarrow 1 = 1, 1 \Rightarrow 0 = 0$, and $\forall \alpha \in [0, 1]$

There are four main families (or fuzzy logics) commonly used to represent the logical operations over fuzzy sets, namely Gödel logic, Łukasiewicz logic, Product logic [Hájek98], and standard fuzzy logic or Zadeh logic [Zad65]. Table 2.1 shows the definition of the operators of these logics, There are other fuzzy logics and operators, but they are out of the scope of this thesis.

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

Table 2.1: Some popular families of fuzzy operators [BS09].

2.1.4 Fuzzy aggregation operators

Aggregation operators (AO) are mathematical operations that are employed to combine different pieces of information. Given a domain $\mathbb{D}$, an AO is a mapping $@ : \mathbb{D}^K \to \mathbb{D}$, aggregating K values of K different criteria into a single one. Some typical examples are the average, the weighted mean, the maximum, or the minimum. According to the previous definition, t-norms and t-conorms can be thought as AOs, but it is very common to restricts to AOs verifying the internality property, i.e.,

$$\min(x_1, \ldots, x_K) \leq @(x_1, \ldots, x_K) \leq \max(x_1, \ldots, x_K) \ . \tag{2.1}$$

Frequently, the AOs also use as a parameter a vector of weights $W = [w_1, \ldots, w_K]$ such that $w_i \in [0, 1]$ and $\sum_{i=1}^{K} w_i = 1$.

We will consider two cases for the domain $\mathbb{D}$, one in which we want to aggregate numbers, and another one in which we want to aggregate i.e., $\mathbb{D} = [0, 1]$. The second case happens when $\mathbb{D}$ is a set of linguistic labels.

AOs for numerical values. We will start by discussing the aggregation of numerical values. Without loss of generalization, we will assume $\mathbb{D} = [0, 1]$. We can see the information to aggregate as the partial degrees of fulfillment of K different criteria, represented using fuzzy sets. Therefore, $x_i \in [0, 1], i \in \{1, \ldots, K\}$.

We will focus on AOs parameterized with a vector of weights and denote them as $@([w_1, \ldots, w_K], [x_1, \ldots, x_K])$. The classical example of such an AO is *weighted mean* (WMEAN) or weighted sum, defined as:

$$\mathbf{WS}([w_1, \ldots, w_K], [x_1, \ldots, x_K]) = \sum_{i=1}^{K} w_i x_i \ . \tag{2.2}$$

A similar variant is the *Strict weighted sum* (SWS) which has 0 as an absorbing element, that is, the aggregated value is 0 if some criterion is 0, and the weighted sum of the arguments otherwise [BS16a]. Formally, SWS is defined as:

$$\mathbf{SWS}([w_1, \ldots, w_K], [x_1, \ldots, x_K]) = \begin{cases} 0 & \text{if } \prod_{i=1}^{K} x_i = 0 \\ \sum_{i=1}^{K} w_i x_i & \text{otherwise.} \end{cases} \tag{2.3}$$

A famous family of AOs are the Ordered Weighted Averaging (OWA) operators [Yag88]. An OWA operator is formally defined as:

$$\mathbf{OWA}([w_1, \ldots, w_K], [x_1, \ldots, x_K]) = \sum_{i=1}^{K} w_i x_{\sigma(i)} \tag{2.4}$$

where $\sigma(i)$ is 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 $x_1, \ldots, x_K$ to be aggregated.

Note that in this operator the criteria are sorted in decreasing order. So, the weight w_i is associated with a ordered position of the aggregate $x_{\sigma(i)}$ rather than with x_i. OWA operators verify the internality property: the extreme cases of OWA operators coincide with the minimum (a t-norm) and the maximum (a t-conorm) using $W = [0, \ldots, 0, 1]$ and $W = [1, 0, \ldots, 0]$, respectively. This can be seen as a degree of pessimism or optimism respectively: the closer to minimum, the more pessimistic. A measure of the optimism associated to a weight vector W is orness [Yag88], formally defined as.

$$orness([w_1, \ldots, w_K]) = \frac{1}{K-1} \sum_{i=1}^{K} (K - i) w_i \ . \tag{2.5}$$

Clearly, $orness([w_1, \ldots, w_K]) \in [0, 1]$. The orness of the minimum t-norm is 0 e.g., $orness([0, \ldots, 0, 1]) = 0$, and the orness of the maximum t-conorm is 1 e.g, $orness([1, 0, \ldots, 0]) = 1$.

An inconvenient with OWA operators is how to compute the weights. In the literature one can find two popular solutions: *(i)* using quantifier-based aggregation [Yag96] and *(ii)* applying recursive OWA [TY05].

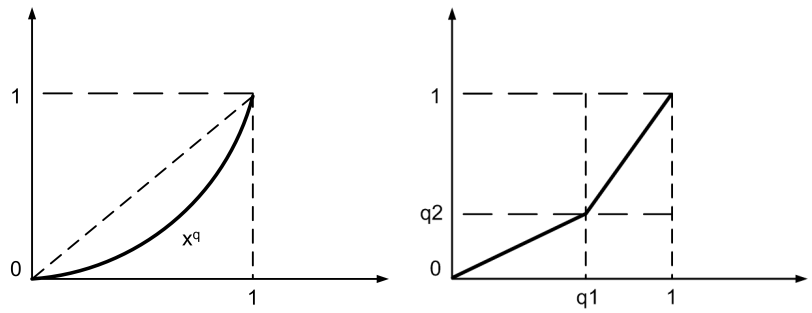
1. ***Quantifier-based aggregation.*** The idea is to use fuzzy quantifiers (e.g., all, most, at many possible, etc.) to compute the vector of weights. A proportional fuzzy quantifier $Q : [0, 1] \to [0, 1]$ is a fuzzy subset such that for each $r \in [0, 1]$, the evaluation of $Q(r)$ indicates the degree to which the proportion r satisfies a linguistic quantifier Q. In [Yag91], Yager proposed *Regular Increasing Monotone* (RIM) quantifiers, which satisfy the boundary conditions $Q(0) = 0$ and $Q(1) = 1$, and are monotone increasing, i.e., $Q(x_1) \leq Q(x_2)$ when $x_1 \leq x_2$. A RIM quantifier Q can be used to define an OWA weighting vector W_Q of dimension K, where each weight w_i is computed as follows:

$$w_i = Q(\frac{i}{K}) - Q(\frac{i-1}{K}), i = 1, ..., K \tag{2.6}$$

We can verify that $w_i \in [0, 1]$ and $\sum_{i=1}^{K} w_i = 1$.

For details about RIM quantifier families, we refer the reader to [XS08]. In this thesis, we will consider the following functions to build RIMS:

- *Right-shoulder* function: $Q(x) = \mathbf{right}(q_1, q_2)$, with $q_1, q_2 \in [0, 1]$ (defined in Section 2.1.1).
- *Power* function: $Q(x) = x^q$, with $q \in (0, \infty)$, as illustrated in Figure 2.6 (a).



(a) (b)

Figure 2.6: (a) Power function; (b) Linear function.

– *Linear* function: $Q(x) = \mathbf{lin}(q_1, q_2)$, with $q_1, q_2 \in [0, 1]$, as shown in Figure 2.6 (b). The linear function can be defined using a single parameter $q_3 \geq 0$, e.g $\mathbf{lin}(q_3)$, defined as follows:

  – $\mathbf{lin}(q_3)(x) = (q_2/q_1)x, \forall x \in [0, q_1]$

  – $\mathbf{lin}(q_3)(x) = 1 - (x-1)(1-q_2)/(1-q_1), \forall x \in [q_1, 1]$

  where $q_1 = q_3/(q_3+1)$ and $q_2 = 1/(q_3+1)$.

2. Recursive OWA. The idea is starting from a desired value for the orness of the OWA operator and calculate the weights in two recursive ways, a *Left Recursive Form* (LRF) or a *Right Recursive Form* (RRF). The original formulation of the OWA (Eq. 2.4) can be rewritten to obtain the LRF as follows:

$$\begin{gathered}\mathbf{OWA}([w_1^K, \ldots, w_K^K], [x_1, \ldots, x_K]) = \\ v_L^K \cdot \mathbf{OWA}([w_1^{K-1}, \ldots, w_{K-1}^{K-1}], [x_{\sigma(1)}, \ldots, x_{\sigma(K-1)}]) + (1 - v_L^K) \cdot x_{\sigma(K)} \ ,\end{gathered} \tag{2.7}$$

where the weights are defined as:

$$\begin{aligned} v_L^K &= \frac{(K-1) \cdot \text{orness}}{(K-2) \cdot \text{orness} + 1} \\ w_i^K &= v_L^K \cdot w_i^{K-1}, \ i \in \{1, \ldots, K-1\} \\ w_K^K &= 1 - v_L^K \ . \end{aligned} \tag{2.8}$$

The base case happens when $K = 2$:

$$\mathbf{OWA}([w_1^2, w_2^2], [x_1, x_2]) = v_L^2 \cdot x_{\sigma(1)} + (1 - v_L^2) \cdot x_{\sigma(2)} \tag{2.9}$$

In the same way, the original equation of OWA (Eq. 2.4) can be rewritten to obtain the RRF as follows:

$$\begin{gathered}\mathbf{OWA}([w_1^K, \ldots, w_K^K], [x_1, \ldots, x_K]) = \\ (1 - v_R^K) \cdot x_{\sigma(1)} + v_R^K \cdot \mathbf{OWA}([w_2^{K-1}, \ldots, w_K^{K-1}], [x_{\sigma(2)}, \ldots, x_{\sigma(K)}]) \ ,\end{gathered} \tag{2.10}$$

where the weights are defined as:

$$\begin{aligned} v_R^K &= \frac{(K-1)(1 - \text{orness})}{(K-2)(1 - \text{orness}) + 1} \\ w_i^K &= v_R^K \cdot w_i^{K-1}, \ i \in \{1, \ldots, K-1\} \\ w_K^K &= 1 - v_R^K \ . \end{aligned} \tag{2.11}$$

AOs for fuzzy linguistic values. In this case the AOs aggregate fuzzy linguistic numbers $d_1, \ldots, d_K$ rather than numerical values. In the work [Xu08] several methods to aggregate fuzzy linguistic numbers are reviewed. We are interested in four of them: convex combination, linguistic OWA, weighted mean (WMEAN), and fuzzy OWA (FOWA). Again, we will assume a vector of weights W and a permutation σ over fuzzy membership functions or numerical values such that $d_{\sigma(1)} \geq d_{\sigma(2)} \geq \cdots \geq d_{\sigma(K)}$.

1. ***Convex combination.*** This operator was proposed in [DVV93]. It assumes a fixed vector of possible values for the linguistic variables $\mathcal{L} = \{l_1, \ldots, l_L\}$ such that $l_i < l_j$ if $i < j$.

   The convex combination has a recursive definition. Indeed, we will call it CONV–RRF, to make it explicit that it corresponds to a Right Recursive Form. Let us start with the base case where we want to aggregate $K = 2$ fuzzy values $d_i \in \mathcal{L}$. The CONV–RRF of a vector $[d_1, d_2]$ given a vector of weights $W = [w_1, w_2]$, is defined as

$$\mathbf{CONV}^{\mathbf{RRF}}([w_1, w_2], [d_1, d_2]) = l_c \tag{2.12}$$

   where $d_{\sigma(1)} = l_j, d_{\sigma(2)} = l_i, c = i + round(w_{\sigma(1)} \cdot (j - i))$. Note that the permutation is applied both to the weights and to the values to be aggregated, so each w_i is associated to the value d_i. Now, if $K > 2$, then:

$$\begin{gathered}\mathbf{CONV}^{\mathbf{RRF}}([w_1, \ldots, w_K], [d_1 \ldots, d_K]) = \\ \mathbf{CONV}^{\mathbf{RRF}}\Big([w_{\pi(1)}, 1 - w_{\pi(1)}], \\ \Big[d_{\pi(1)}, \mathbf{CONV}^{\mathbf{RRF}}([\beta_2, \ldots, \beta_k], [d_{\pi(2)}, \ldots, d_{\pi(K)}])\Big]\Big) \ ,\end{gathered} \tag{2.13}$$

   where $\beta_h = w_{\pi(h)} \ / \ \sum_{j=2}^{K} w_{\pi(j)}, h = 2, \ldots, K$.

2. ***Linguistic OWA.*** The linguistic OWA aggregation operator is a variation of CONV–RRF using a ordering step as in standard OWA [HHVV96]. As with CONV–RRF, we will call it LOWA–RRF to make it explicit that it is a Right Recursive Form. LOWA–RRF is defined as:

$$\begin{gathered}\mathbf{LOWA}^{\mathbf{RRF}}([w_1, \ldots, w_K], [d_1 \ldots, d_K]) = \\ \mathbf{CONV}^{\mathbf{RRF}}([w_1, \ldots, w_K], [d_{\sigma(1)}, \ldots, d_{\sigma(K)}]) \ .\end{gathered} \tag{2.14}$$

   Now, as in classical OWA, each weight w_i is not associated to a value d_i.

3. ***WMEAN.*** It is an extension of the weighted mean aggregation operator but considering trapezoidal functions [DW87]. Given K trapezoidal functions $d_i = \mathbf{trap}(q_1^i, q_2^i, q_3^i, q_4^i)$ and a vector of weights W of size K, the result is:

$$\begin{gathered}\mathbf{WMEAN}([w_1, \ldots, w_K], [d_1, \ldots, d_K]) = \\ \mathbf{trap}\Big(\sum_{i=1}^{K} w_i q_1^i, \sum_{i=1}^{K} w_i q_2^i, \sum_{i=1}^{K} w_i q_3^i, \sum_{i=1}^{K} w_i q_4^i\Big) \ .\end{gathered} \tag{2.15}$$

   Now, the weights describe the importance of each number of K. An interesting property is that, in general, the result of the aggregation is not any of the original aggregated values.

4. ***FOWA.*** It is a variant of WMEAN with a reordering step [CC03], so that w_1 is associated to the largest trapezoidal function. Given a permutation σ such that $d_{\sigma(i)}$ denotes the i-th largest trapezoidal function, FOWA is defined as:

$$\mathbf{FOWA}([w_1,\ldots,w_K],[d_1,\ldots,d_K]) = \mathbf{trap}\Big(\sum_{i=1}^{K} w_i q_1^{\sigma(i)}, \sum_{i=1}^{K} w_i q_2^{\sigma(i)}, \sum_{i=1}^{K} w_i q_3^{\sigma(i)}, \sum_{i=1}^{K} w_i q_4^{\sigma(i)}\Big) \ . \tag{2.16}$$

It is important to mention that CONV–RRF, LOWA–RRF, and FOWA require an ordering between fuzzy linguistic labels. Delgado et al. [DHHVM98] propose the following one:

$$d_1 \geq d_2 \text{ iff } transform(d_1) \geq transform(d_2) \ , \tag{2.17}$$

where $transform(d)$ is a transformation function from a linguistic domain (represented by trapezoidal functions) to a numerical domain defined as:

$$transform(\mathbf{trap}(q_1,q_2,q_3,q_4)) = \frac{8(q_3+q_2)H + (q_4+q_1)H + 8(H+q_3q_4-q_1q_2)}{24H} \ , \tag{2.18}$$

where $H = q_4 + q_3 - q_2 - q_1$ and $q_1 = q_2 = q_3 = q_4$ is assumed not to hold. Other sorting methods are described in [SSG08, KV14].

2.1.5 Fuzzy modifiers

Fuzzy modifiers, commonly referred as fuzzy hedges, change the shape of a fuzzy set [HC76]. In other words, they transform the membership function into another one. Some examples of modifiers are *very*, *likely*, *more or less*, *few*, etc. By applying modifiers to the membership functions of the linguistic labels we can form new terms such as *very hot*, or *more or less cold*, which is useful to model natural language. Formally, a modifier m is defined using a function

$$f_m \colon [0,1] \to [0,1] \ .$$

For instance, we may define $f_{very}(x) = x^2$ and $f_{few} = \sqrt{x}$. Two famous families or fuzzy modifiers are *increasing*, if $f_m(x) \geq x$, and *weakening*, if $f_m(x) \leq x$. For instance, *very* is weakening and *few* is increasing.

Example 4. *For example, now it is possible to express the fuzzy set of* very cold *temperature in the context of beers. If we have an Ámbar Especial beer with a temperature of that $x = 6^o$ C, we compute the degree of being very cold as:*

$$\mu_{veryCold}(x) = f_{very}(\mu_{Cold}(x)) = \mu_{Cold}(x)^2 = (\mathbf{tri}(4,5.5,7)(x))^2 = (0.66)^2 = 0.43 \quad \square$$

.

Some typical examples of fuzzy modifiers are:

– Linear modifiers of the form $\mathbf{lin}(q_1, q_2)$, defined in Section 2.1.4.

– Triangular modifiers of the form $f_m(x) = \mathbf{tri}(q_1, q_2, q_3)(x)$, defined in Section 2.1.1, but restricting to the case $q_1, q_2, q_3 \in [0, 1]$.

2.1.6 Defuzzification

Fuzzification is the process to replace a real number with a fuzzy set. The inverse process is defuzzification, which converts a fuzzy set into a single crisp number. There are many techniques to defuzzify a fuzzy set. For an overview we refer the reader to [LK99, SVM02]. Here, we will focus on some of them.

To start with, let us define two important families of defuzzification techniques. A *distributed* technique treats the fuzzy set as a distribution for which the average value is evaluated, whereas a *maxima* technique strictly focus on the elements belonging with the highest membership degree [SVM02].

Let us recall that the membership function $\mu_A(x)$ is the evaluation of A with $x \in X$. The core of a fuzzy set A, denoted $core(A)$, is defined as the set of elements of the universe of domain with the highest degree of membership to A[3]:

$$core(A) = \{x \in X \mid \forall y \in X : \mu_A(y) \leq \mu_A(x)\}$$

Some common defuzzification techniques are the following ones:

Small of Maxima (SOM). The output is smallest element (value) with the maximum evaluation of the membership function.

$$\mathbf{SOM}(\mu_A) = min\ core(\mu_A) \tag{2.19}$$

Largest of Maxima (LOM). The output is largest element (value) with the maximum evaluation of the membership function.

$$\mathbf{LOM}(\mu_A) = max\ core(\mu_A) \tag{2.20}$$

Middle of Maxima (MOM). It is computed from the two previous values:

$$\mathbf{MOM}(\mu_A) = \frac{SOM + LOM}{2} \tag{2.21}$$

[3]Please note that the core is often defined as the set of the elements having degree of membership equal to 1.

Center of Gravity (COG). The output is a centroid of the fuzzy set, defined as:

$$\mathbf{COG}(\mu_A) = \frac{\sum_{i=1}^{n} \mu_A(x_i) x_i}{\sum_{i=1}^{n} \mu_A(x_i)} \tag{2.22}$$

Example 5. *Given the trapezoidal membership function shown in Figure 2.7, the core set is core* $= [a, b]$*. Therefore, we obtain* $SOM = a$*,* $LOM = b$*,* $MOM = m = (a+b)/2$ *and* $COG = c$ *(using a general formula to get the centroid in the Cartesian system).* □

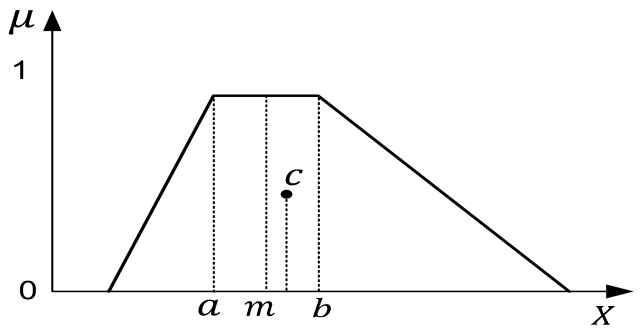


Figure 2.7: Some defuzzification methods [Ros10].

Note that if there is a unique maximum, SOM, LOM, and MOM give the same result. It is important to mention that COG is a distributed technique, whereas SOM, LOM and MOM are called maxima techniques.

2.2 Semantic Web technologies

2.2.1 Ontologies

Ontologies have a philosophical origin, specifically in the branch of metaphysics, with the meaning of the study of being in general. The first to treat it were Plato and Aristotle who created a hierarchical categorization of species as entities and features to group them (the tree of Porphyry, called scale of being). However, in computer science an ontology is widely used for Artificial Intelligence systems with a different meaning. Here the ontology is a representation that permits modeling the world. In the literature there are many ontology definitions. Some of them are the following ones:

- The most common definition of an ontology is by Gruber (1993) as "an explicit specification of a conceptualization" [Gru93].
- Guriano et al. (1995) define the ontology as "a logical theory which gives an explicit, partial account of a conceptualization" [GG95].
- Borst (1997) considered Gruber's definition too broad and modified it as "a formal specification of a shared conceptualization" [Bor97].

- In 1998 Studer et al. [SBF98] interpreted Gruber and Bort's definitions:
  - "Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon.
  - Explicit means that the type of concepts used, and constraints on their use are explicitly defined.
  - Formal refers to the fact that the ontology should be machine-readable.
  - Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group".
- Noy et al. (2001) defined an ontology as "a formal explicit description of concepts in a domain of discourse, properties of each concept describing various features and attributes of the concept, and restrictions on slots" [NM01].

We can see different perspectives of the ontologies, from philosophical to computational. In general, an ontology offers a common vocabulary, a terminology, a conceptual model, and reasoning capabilities. Ontologies offer advantages such as promoting knowledge share and reuse. They are easy to read and understand for machines (software applications) and humans (natural form). They also provide a division of the domain knowledge and the application level, particularly useful for intelligent applications.

The main ingredients of ontologies to model a domain are the following ones:

- *Concepts/classes*: They describe the concepts within the domain. Classes are unary predicates. For example, Beer class represents all beers. All classes are structured in a class hierarchy (superclass-subclass) or taxonomy. For example, Lager (a beer family made through a bottom fermentation process) is a subclass of Beer.
- *Instances/individuals*: They represent particular elements of the concepts in the ontology. For instance, the Sol beer is an instance of the Lager class.
- *Datatypes/concrete domains*: They are elements that do not belong to the represented domain, but rather to a different domain that is already structured and whose structure is already known to the machine (e.g. the machine already knows that $1 < 2$). Possible datatypes include numbers (real, rational, integer, non-negative, etc.), strings, Booleans, dates, times, or, more generally, XML literals [BCE+15].

– *Properties/roles*: They represent binary predicates, and represent the relationships between a pair of elements in the domain. There are two types of properties: *object properties* (or abstract roles) relate a pair of individuals, and *data properties* (or concrete roles)[4] link an individual with a data value (of a concrete datatype). For example, the object property brewedBy links a beer (such as Sol) and a brewery (such as Cuauhtémoc-Moctezuma-Heineken). Moreover, the data property ABV (Alcohol By Volume) links a beer (such as Sol) with a real number (4.5) denoting its alcohol degree.

– *Axioms*: They are constraints or restrictions to be verified by the elements of the ontology. The axioms define how to combine the previous elements to represent the knowledge of some concrete domain. They also make it possible to infer new knowledge implicitly represented in the ontology.

The traditional way to represent knowledge using databases has some analogies with ontologies but there are also notable differences. We will mention three of them:

1. Ontology axioms behave like inference rules rather than database constraints.

2. Ontologies follow an Open World Assumption, under which unknown information is actually treated as unknown. Databases follow a Closed World Assumption where unknown information is treated as false.

3. Unique Name Assumption (UNA) is common in databases, which require to identify each individual with a unique name. However, ontologies do not assume UNA, so individuals may have more than one name.

2.2.2 Description Logics

Description Logics (DLs) are "a family of knowledge representation languages that can be used to represent knowledge of an application domain in a structured and well-understood way" [BHLS17]. Their logic-based semantics makes it possible reasoning to discover implicit knowledge. Each DL is usually a subset of First Order Logic (FOL). In the following, we delineate the syntax, semantic and reasoning tasks.

Syntax. DLs have three elements: individuals (denoted a), concepts or classes (denoted C), and roles or properties (denoted R if they are object properties, or T if they are data properties). Complex concepts (e.g., MexicanBeer) can be built from

[4]In this thesis, we will also call them attributes or, if they are functional, features.

other concepts, roles, and individuals using different constructors, depending on the expressivity of the language.

A DL describes the knowledge using an ontology or Knowledge Base (KB) $\mathcal{O} = \langle \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, including an Assertional Box or ABox (denoted $\mathcal{A}$), a Terminological Box or TBox (denoted $\mathcal{T}$), and a Role Box or RBox (denoted $\mathcal{R}$). The concrete axiom types depend on the expressivity of the DL.

– An ABox includes a finite set of axioms about individuals (known as facts). ABox axioms include concept assertions and role assertions. A concept assertion of the form $a : C$ states than an individual a is part of concept C. A role assertion of the form $(a, b) : R$ denotes the relationship of two individuals a and b via the role R.

Example 6. *Examples of concept assertions are the axiom* $\mathsf{Sol} : \mathsf{Lager}$*, stating that* Sol *is a beer of the* Lager *family, and* $\mathsf{Cuauhtémoc–Moctezuma–Heineken} : \mathsf{Brewery}$*. The role assertion* $(\mathsf{Sol}, \mathsf{Cuauhtémoc–Moctezuma–Heineken}) : \mathsf{brewedBy}$ *says that* Sol *is brewed by* Cuauhtémoc-Moctezuma-Heineken *plant. The previous axioms can be represented in the following ABox:*

$$\mathcal{A} = \{\mathsf{Sol} : \mathsf{Lager},\ \mathsf{Cuauhtémoc–Moctezuma–Heineken} : \mathsf{Brewery};$$

$$((\mathsf{Sol}, \mathsf{Cuauhtémoc–Moctezuma–Heineken}) : \mathsf{brewedBy})\ \} \quad \square$$

– A TBox contains a finite set of axioms about concepts. Typically, they include *General Concept Inclusions* (GCI) axioms which are of the form $C \sqsubseteq D$, meaning that C is subsumed by D. Note that this implies that the individuals of C belong to D, but not vice versa. Another axiom type is the *concept equivalence* (or concept definition) $C \equiv D$ which is a shortcut for the pair of inclusions $C \sqsubseteq D$ and $D \sqsubseteq C$. Now, individuals in C also are in D and vice versa. TBoxes also include other useful syntactic sugar axioms (such domain/range axioms).

Example 7. *The axiom* $\mathsf{Lager} \sqsubseteq \mathsf{Beer}$ *states that* Lager *is a specific style of* Beer*, while the axiom* $\mathsf{HeinekenMexicanBeer} \equiv \mathsf{Beer} \sqcap \exists \mathsf{breweedBy}.\mathsf{Cuauhtémoc–Moctezuma–Heineken}$ *defines* HeinekenMexicanBeer *as those beers brewed by that Mexican plant. That TBox can be expressed as:*

$$\mathcal{T} = \{\mathsf{Lager} \sqsubseteq \mathsf{Beer};$$

$\mathsf{HeinekenMexicanBeer} \equiv \mathsf{Beer} \sqcap \exists \mathsf{breweedBy}.\mathsf{Cuauhtémoc–Moctezuma–Heineken}\}$ $\square$

– An RBox contains a finite set of axioms about roles. Typically, they include *subproperty axioms* and *property equivalences* or (or definitions). More expressive language can include *transitivity axioms*, *reflexivity axioms* ...

DLs also allow to represent concrete quantities for real-world applications, for example, the price of an item or the alcohol level of a beer. This is possible by using a *concrete domain* or *datatype theory* $\mathbf{D} = \langle \Delta_{\mathbf{D}}, \cdot_{\mathbf{D}} \rangle$, where $\Delta_{\mathbf{D}}$ is the datatype domain (e.g the set of positive integer numbers) and $\cdot_{\mathbf{D}}$ the set of datatype predicates $\mathbf{d}$. Every datatype predicate $\mathbf{d} \in \cdot_{\mathbf{D}}$ is associated with an arity n and an n-ary predicate $\mathbf{d}^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}^n$ [LAHS05]. In other words, $\cdot_{\mathbf{D}}$ assigns to each data value an element of $\Delta_{\mathbf{D}}$. For example, $\leq_1$ is a datatype unary predicate defined over the set of positive reals. Therefore, $\exists \mathsf{ABV}.\leq_1$ denotes the set of things with no more than 1 alcohol degree.

Typically, every DL is named by using a string of capital letters which identify the constructors of the logic and therefore characterize its complexity. The standard language for knowledge representation OWL 2 DL is equivalent to the DL $\mathcal{SROIQ}(\mathbf{D})$.

Table 2.2 shows the syntax of concepts, roles, and axioms in OWL 2 DL, where n is a natural number. There are some additional restrictions to ensure decidability of the language [CGHM$^+$08] (some roles are required to be simple, the use of universal role is restricted, RIAs must be regular, and inverse functionality of data properties is restricted), but we omit them here for simplicity. ABox axioms are (A1)–(A7), TBox axioms are (A8)–(A14), and RBox axioms are (A15)–(A30).

Semantics. The semantics of a DL is formally defined using an interpretation. Specifically, an interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where $\Delta^{\mathcal{I}}$ denotes a non-empty set (the domain of interpretation) and $\cdot^{\mathcal{I}}$ denotes a mapping function (or interpretation function) which maps:

– to each individual a an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$,

– to each atomic concept A a function $A^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \to \{0, 1\}$,

– to each object property R a function $R^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to \{0, 1\}$, and

– to each data property T a function $T^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \to \{0, 1\}$.

In summary, individuals are interpreted as objects, concepts as sets, and roles as binary relations. Note that either an individual belongs to an atomic concept A or not; and that a pair of individuals are either related via R or not. Table 2.3 shows the semantics of complex concepts, complex roles, and axioms in OWL 2 DL.

An interpretation $\mathcal{I}$ is a model of an ontology $\mathcal{O}$ (denoted $\mathcal{I} \models \mathcal{O}$) iff it satisfies all the axioms in the ABox $\mathcal{A}$, TBox $\mathcal{T}$ and RBox $\mathcal{R}$ in $\mathcal{O}$.

	Concept	
(C1)	A	Atomic concept
(C2)	$\top$	Top concept
(C3)	$\bot$	Bottom concept
(C4)	$C_1 \sqcap C_2$	Conjunction
(C5)	$C_1 \sqcup C_2$	Disjunction
(C6)	$\neg C$	Negation
(C7)	$\forall R.C$	Universal restriction
(C8)	$\exists R.C$	Existential restriction
(C9)	$\exists R.\{a\}$	Value restriction
(C10)	$\forall T.\mathbf{d}$	Concrete universal restriction
(C11)	$\exists T.\mathbf{d}$	Concrete existential restriction
(C12)	$\exists T.\{v\}$	Concrete value restriction
(C13)	$\{a_1, \ldots, a_n\}$	Nominal
(C14)	$\geq\ n\ R.C$	At-least cardinality restriction
(C15)	$\leq\ n\ R.C$	At-most cardinality restriction
(C16)	$=n\ R.C$	Exact cardinality restriction
(C17)	$\geq\ n\ T.\mathbf{d}$	Concrete at-least restriction
(C18)	$\leq\ n\ T.\mathbf{d}$	Concrete at-most restriction
(C19)	$=n\ T.\mathbf{d}$	Concrete exact restriction
(C20)	$\exists R.\texttt{Self}$	Local reflexivity
	Role	
(R1)	R_A	Atomic role
(R2)	U	Universal role
	Axiom	
(A1)	$a\!:\!C$	Concept assertion
(A2)	$(a_1, a_2)\!:\!R$	Role assertion
(A3)	$(a_1, a_2)\!:\!\neg R$	Negative role assertion
(A4)	$(a, v)\!:\!T$	Concrete role assertion
(A5)	$(a, v)\!:\!\neg T$	Negative concrete role assertion
(A6)	$a_1 = a_2$	Equality assertion
(A7)	$a_1 \neq a_2$	Inequality assertion
(A8)	$C_1 \sqsubseteq C_2$	General Concept Inclusion (GCI)
(A9)	$C_1 \equiv C_2$	Concept equivalence
(A10)	$\texttt{dis}(C_1, \ldots, C_n)$	Disjoint concepts
(A11)	$\texttt{dom}(R, C)$	Domain role axiom
(A12)	$\texttt{ran}(R, C)$	Range role axiom
(A13)	$\texttt{dom}(T, C)$	Domain concrete role axiom
(A14)	$\texttt{ran}(T, \mathbf{d})$	Range concrete role axiom
(A15)	$R_{11} \ldots R_{1n} \sqsubseteq R_2$	Role Inclusion Axiom (RIA)
(A16)	$R_1 \equiv R_2$	Role equivalence
(A17)	$T_1 \sqsubseteq T_2$	Concrete RIA
(A18)	$T_1 \equiv T_2$	Concrete role equivalence
(A19)	$\texttt{inv}(R_1, R_2)$	Inverse roles
(A20)	$\texttt{fun}(R)$	Functional role
(A21)	$\texttt{fun}(T)$	Functional concrete role
(A22)	$\texttt{invfun}(R)$	Inverse functional role
(A23)	$\texttt{invfun}(T)$	Inverse functional concrete role
(A24)	$\texttt{trans}(R)$	Transitive role
(A25)	$\texttt{dis}(R_1, \ldots, R_n)$	Disjoint roles
(A26)	$\texttt{dis}(T_1, \ldots, T_n)$	Disjoint concrete roles
(A27)	$\texttt{ref}(R)$	Reflexive role
(A28)	$\texttt{irr}(R)$	Irreflexive role
(A29)	$\texttt{sym}(R)$	Symmetric role
(A30)	$\texttt{asy}(R)$	Asymmetric role

Table 2.2: Syntax of concepts, roles, and axioms in OWL 2 DL.

Complex concept	
(C2)	$\Delta^{\mathcal{I}}$
(C3)	$\emptyset$
(C4)	$C_1^{\mathcal{I}}(x) \cap C_2^{\mathcal{I}}(x)$
(C5)	$C_1^{\mathcal{I}}(x) \cup C_2^{\mathcal{I}}(x)$
(C6)	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
(C7)	$\{x \in \Delta^{\mathcal{I}} : \forall y \in \Delta^{\mathcal{I}}, (x,y) \notin R^{\mathcal{I}} \text{ or } y \in C^{\mathcal{I}}\}$
(C8)	$\{x \in \Delta^{\mathcal{I}} : \exists y \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
(C9)	$R^{\mathcal{I}}(x, a^{\mathcal{I}})$
(C10)	$\{x \in \Delta^{\mathcal{I}} : \forall v \in \Delta_{\mathbf{D}}, (x,v) \notin T^{\mathcal{I}} \text{ or } v \in \mathbf{d_D}\}$
(C11)	$\{x \in \Delta^{\mathcal{I}} : \exists v \in \Delta_{\mathbf{D}}, (x,v) \in T^{\mathcal{I}} \text{ and } v \in \mathbf{d_D}\}$
(C12)	$T^{\mathcal{I}}(x, v_{\mathbf{D}})$
(C13)	$\{a_1^{\mathcal{I}}, \ldots, a_m^{\mathcal{I}}\}$
(C14)	$\{x \in \Delta^{\mathcal{I}} : \lvert\{y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}\rvert \geq n\}$
(C15)	$\{x \in \Delta^{\mathcal{I}} : \lvert\{y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}\rvert \leq n\}$
(C16)	$\{x \in \Delta^{\mathcal{I}} : \lvert\{y \in \Delta^{\mathcal{I}} : (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}\rvert = n\}$
(C17)	$\{x \in \Delta^{\mathcal{I}} : \lvert\{v \in \Delta_{\mathbf{D}} : (x,v) \in T^{\mathcal{I}} \text{ and } v \in \mathbf{d_D}\}\rvert \geq n\}$
(C18)	$\{x \in \Delta^{\mathcal{I}} : \lvert\{v \in \Delta_{\mathbf{D}} : (x,v) \in T^{\mathcal{I}} \text{ and } v \in \mathbf{d_D}\}\rvert \leq n\}$
(C19)	$\{x \in \Delta^{\mathcal{I}} : \lvert\{v \in \Delta_{\mathbf{D}} : (x,v) \in T^{\mathcal{I}} \text{ and } v \in \mathbf{d_D}\}\rvert = n\}$
(C20)	$\{x \in \Delta^{\mathcal{I}} : (x,x) \in R^{\mathcal{I}}\}$
Complex role	
(R2)	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Axiom	
(A1)	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
(A2)	$(a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \in R^{\mathcal{I}}$
(A3)	$(a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \notin R^{\mathcal{I}}$
(A4)	$(a^{\mathcal{I}}, v_{\mathbf{D}}) \in T^{\mathcal{I}}$
(A5)	$(a^{\mathcal{I}}, v_{\mathbf{D}}) \notin T^{\mathcal{I}}$
(A6)	$a_1^{\mathcal{I}} = a_2^{\mathcal{I}}$
(A7)	$a_1^{\mathcal{I}} \neq a_2^{\mathcal{I}}$
(A8)	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
(A9)	$C_1^{\mathcal{I}} = C_2^{\mathcal{I}}$
(A10)	$\forall i, j \in \{1, \ldots, n\}, i < j, C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \emptyset$
(A11)	$\{x \in \Delta^{\mathcal{I}} : \exists y \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$
(A12)	$\forall x, y \in \Delta^{\mathcal{I}}, (x,y) \notin R^{\mathcal{I}} \text{ or } y \in C^{\mathcal{I}}$
(A13)	$\{x \in \Delta^{\mathcal{I}} : \exists v \in \Delta_{\mathbf{D}}, (x,v) \in T^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$
(A14)	$\forall x \in \Delta^{\mathcal{I}}, \forall v \in \Delta_{\mathbf{D}}, (x,v) \notin T^{\mathcal{I}} \text{ or } v \in \mathbf{d_D}$
(A15)	$R_{11}^{\mathcal{I}} \circ \cdots \circ R_{1n}^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
(A16)	$R_1^{\mathcal{I}} = R_2^{\mathcal{I}}$
(A17)	$T_1^{\mathcal{I}} \subseteq T_2^{\mathcal{I}}$
(A18)	$T_1^{\mathcal{I}} = T_2^{\mathcal{I}}$
(A19)	$\forall x, y \in \Delta^{\mathcal{I}}, (x,y) \in R_1^{\mathcal{I}} \text{ iff } (y,x) \in R_2^{\mathcal{I}}$
(A20)	$\forall x, y, z \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \text{ and } (x,z) \in R^{\mathcal{I}} \text{ imply } y = z$
(A21)	$\forall x \in \Delta^{\mathcal{I}}, \forall v_1, v_2 \in \Delta_{\mathbf{D}}, (x,v_1) \in T^{\mathcal{I}} \text{ and } (x,v_2) \in T^{\mathcal{I}} \text{ imply } v_1 = v_2$
(A22)	$\forall x, y, z \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \text{ and } (z,y) \in R^{\mathcal{I}} \text{ imply } x = z$
(A23)	$\forall x, y \in \Delta^{\mathcal{I}}, \forall v \in \Delta_{\mathbf{D}}, (x,v) \in T^{\mathcal{I}} \text{ and } (y,v) \in T^{\mathcal{I}} \text{ imply } x = y$
(A24)	$\forall x, y, z \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \text{ and } (y,z) \in R^{\mathcal{I}} \text{ imply } (x,z) \in R^{\mathcal{I}}$
(A25)	$\forall i, j \in \{1, \ldots, n\}, i < j, R_i^{\mathcal{I}} \cap R_j^{\mathcal{I}} = \emptyset$
(A26)	$\forall i, j \in \{1, \ldots, n\}, i < j, T_i^{\mathcal{I}} \cap T_j^{\mathcal{I}} = \emptyset$
(A27)	$\forall x \in \Delta^{\mathcal{I}}, (x,x) \in R^{\mathcal{I}}$
(A28)	$\forall x \in \Delta^{\mathcal{I}}, (x,x) \notin R^{\mathcal{I}}$
(A29)	$\forall x, y \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \text{ implies } (y,x) \in R^{\mathcal{I}}$
(A30)	$\forall x, y \in \Delta^{\mathcal{I}}, (x,y) \in R^{\mathcal{I}} \text{ implies } (y,x) \notin R^{\mathcal{I}}$

Table 2.3: Semantics of complex concepts, complex roles, and axioms in OWL 2 DL.

2.2.3 Web Ontology Language (OWL)

The Semantic Web architecture requires a language to model knowledge using ontologies [BHL01]. The current standard is the Ontology Web Language (*OWL*), which is a recommendation of the World Wide Web Consortium (W3C) since 2004 [W3C04b].

OWL is designed on top of other important technologies in the Semantic Web stack, such as the eXtensible Markup Language (XML), the Resource Description Framework (RDF), and RDF Schema (RDFS).

- *XML* allows user to create their own tags and add arbitrary structure to documents, but it does not support semantics.

- *RDF* is a standard model for data interchange [W3C04a]. It uses a triple representation (s, p, o) where s is a subject (a resource), o an object (a resource or literal) and p is a predicate or property. For instance, *(Ignacio, likes, mexicanTacos)* is a triple with a straight-forward meaning: "Ignacio likes Mexican tacos". RDF have several syntaxes, and it is possible to represent triples using XML tags. Every element of a RDF triple is identified in the web by an Uniform Resource Identifier (URI). Sometimes, URI fragments are presented (omitting the prefix with the namespace) rather than complete URIS, as in our example. RDF triples form a graph of data.

- *SPARQL*[5] is the standard query language for RDF [PAG06], to get valuable information from data graphs.

- *RDFS* is an extension of RDF with a specific vocabulary to represent subclasses, subproperties, domain, and range restrictions.

However, OWL offers an additional vocabulary and a formal semantics to describe more complex ontologies. As already mentioned, OWL is based on DLs. For details about OWL syntax and semantics, we refer the user to [W3C04c].

The first version of OWL, sometimes called OWL 1, offers three sublanguages that can be used for specific implementations and requirements, namely OWL Full, OWL DL and OWL Lite. Each of them has a different expressivity, and therefore reasoning has a different complexity.

[5] `https://www.w3.org/TR/rdf-sparql-query/`

– OWL Full considers the maximum expressiveness. It was designed to be compatible with RDFS, but reasoning becomes undecidable. Therefore, this level is theoretically interesting but hardly interesting in practice.

– OWL DL offers a trade-off between expressiveness and complexity, as reasoning is decidable now. OWL DL is equivalent to the DL $\mathcal{SHOIN}(\mathbf{D})$.

– OWL Lite has the least expressiveness level, so reasoning with it has a smaller computational complexity. OWL Lite is equivalent to the DL $\mathcal{SHIF}(\mathbf{D})$.

The current OWL version is the standard ontology language called OWL 2 [W3C12a, CGHM$^+$08]. OWL 2 is based on the Description Logic $\mathcal{SROIQ}(\mathbf{D})$ has new features such as extra syntactic sugar, extended datatype support, simple meta-modelling, qualified cardinality constructors, more role axioms, and extended annotations.

There is an undecidable version of the language called OWL 2 Full, but in practice people usually mean the decidable version of the language OWL 2 DL when referring to OWL 2. W3C also recommends three sublanguages (tractable *profiles*) of OWL 2, namely OWL 2 EL, OWL 2 QL and OWL 2 RL [W3C12b]. These profiles have unique properties but all of them limit the expressiveness power for the sake of computational efficiency, having a polynomial time reasoning complexity.

– OWL 2 EL is particularly useful in applications requiring a large number of classes and/or properties. Here, reasoning requires polynomial time in relation with the ontology size. The EL acronym reflects that this profile is based on the DL $\mathcal{EL}^{++}$.

– OWL 2 QL is suitable for applications that manage a large amount of instance data, as they facilitate accessing and querying relational databases. The QL acronym reflects that this profile can be translated to a Query Language. OWL 2 QL is based on the DL-Lite family of Description Logics.

– OWL 2 RL is useful for domains needing scalable reasoning, which can be performed using rule-based reasoning engines. The RL acronym actually means Rule Language.

OWL 2 represents the axioms about classes, properties, and individuals of the domain using different syntaxes [W3C12a]:

– RDF/XML is the primary syntax, and is designed for data interchange.

– OWL/XML format is backwards compatible with XML tools.

– Functional syntax makes it easier to understand the formal structure of the ontologies.

– Turtle is a very concise syntax for RDF but can also be used for OWL [W3C14].

– Manchester syntax is one of the most compact and human-readable formats [HDG+06, W3C09a].

2.2.4 Reasoning

Reasoning is probably the main advantage of DLs, as it plays an important role both in the development of ontologies (to verify their correctness) and in the deployment of real-world applications (to discover knowledge).

A special computer program that solves reasoning tasks is called a *reasoner*. In other words, it infers implicit knowledge from explicit knowledge. For example, even if beer *Sol* is not explicitly related to *México*, we can infer that it is a Mexican beer because it is related to a brewery which is related to *México*.

Reasoning tasks. The most important reasoning tasks that ontology reasoners solve are:

– *Consistency/KB satisfiability*: check if the ontology $\mathcal{O}$ has a model (i.e., an interpretation that satisfies all the axioms in the ontology).

– *Entailment*: check if an axiom is a logical consequence of an ontology, i.e., if the ontology logically entails an axiom.

– *Concept satisfiability*: check if a concept C can have instances.

– *Subsumption*: check if a concept (resp. property) is more general than (subsumes) another concept (resp. property).

– *Classification*: compute a concept hierarchy and property hierarchy based on the subsumption relationships.

– *Instance retrieval*: get all the instances of a given concept C.

– *Realization*: get all the concepts that a given instance a belongs to.

Reasoning engines. There are many semantic reasoners. In this thesis, we mainly worked with three classical reasoners: JFact, *HermiT* [GHM+14], and *TrOWL* [TPR10].

- *JFact*[6] is an OWL DL reasoner and partially supports OWL 2 DL (it lacks support for key constraints[7] and some datatypes). It is developed on Java language and started as a port of FaCT++ reasoner. JFact implements a tableaux-based decision procedure.

- *HermiT*[8] is based on a hypertableau algorithm which is often more efficient than previous tableaux-based algorithms. It supports OWL 2 DL and is more robust for "hard" ontologies than other reasoners.

- *TrOWL*[9] is a tractable reasoning infrastructure for OWL 2 ontologies. It uses a syntactic approximation from OWL 2 DL to OWL 2 EL for TBox and ABox reasoning. Thus, TrOWL offers reasoning support to large ontologies.

These reasoners offer an OWL API for Java language [HB11] and a plug-in for the ontology editing tool Protégé [Mus15].

Other popular semantic reasoners are *CEL* [BLS06][10], *ELK* [KKS14][11], *FaCT++* [TH06][12], *Konclude* [SLG14][13], Pellet [SPC+07][14], *RacerPro* [HHMW12][15], and *SnoRocket* [LB10][16]. For more examples and details, the reader is referred to `http://owl.cs.manchester.ac.uk/tools/list-of-reasoners`.

2.3 Fuzzy extensions of Semantic Web technologies

2.3.1 Fuzzy ontologies

Several domains in the real world require managing a knowledge which is imprecise or vague, but crisp ontologies have limitations to deal with it. Consider for example the beer domain where there are imprecise notions such as *high* alcohol level, *cheap* item, and *low* temperature. Fuzzy ontologies are an extension of classical ontologies by

[6] `http://jfact.sourceforge.net`
[7] Keys are a restricted type of inverse functional data properties.
[8] `http://www.hermit-reasoner.com`
[9] `http://trowl.org/download-page`
[10] `http://tu-dresden.de/ing/informatik/thi/lat/forschung/software/cel`
[11] `http://liveontologies.github.io/elk-reasoner`
[12] `http://owl.man.ac.uk/factplusplus`
[13] `http://www.derivo.de/en/produkte/konclude.html`
[14] `http://clarkparsia.com/pellet`
[15] `http://franz.com/agraph/racer`
[16] `http://github.com/aehrc/snorocket`

considering several notions of fuzzy sets and fuzzy logic [Bob08, Str13]. Here, classes, properties, datatypes, and axioms are fuzzy.

The elements of a fuzzy ontology are the following ones:

- *Individuals* denote domain elements as in classical ontologies. i.e. Sol.
- *Fuzzy concepts* are interpreted as fuzzy sets of individuals (unary predicates), such as CheapBeer.
- *Fuzzy object properties* are interpreted as fuzzy binary relations (binary predicates), e.g., similarTo.
- Data properties (e.g., hasPrice) are instead usually assumed to be crisp and functional [BS09].
- *Fuzzy datatypes* extend crisp values (numerical values, textual, dates, etc.) by using a fuzzy membership function. For instance, in a fuzzy datatype, instead of using a precise number 4.5 as the value of ABV, now it is possible to use a linguistic term LowABV. Some typical fuzzy membership functions used to define fuzzy datatypes are the trapezoidal, the triangular, the left-shoulder and the right-shoulder functions (see Section 2.1.1).
- *Fuzzy axioms* are formal statements involving these ontology elements. In particular, fuzzy axioms express statements that are not either true or false but hold to some degree. Some typical axioms are:
  - *Fuzzy class assertions* state the membership of an individual to a concept. For example, we can state that Sol beer belongs to the concept of CheapBeer with degree greater or equal than 0.67, meaning that it is a quite cheap beer.
  - *Fuzzy object property assertions* describe the relation between two individuals, e.g., one can state that Sol and Corona are related via property isSimilarTo with degree greater or equal than 0.8, meaning that they are pretty similar.
  - *Fuzzy data property assertions* describe the (possibly partial) relation between an individual and a data value. For example, it is possible to express that a beer has a style rating of 10 by relating Sol and the number 10 via styleRating.
  - *Fuzzy subclass* axioms state that a concept is more specific (or partially more specific) than another one. For instance, we can state that CheapBeer is a subclass of ModerateBeer with degree greater or equal than 0.5.

– *Fuzzy subproperty* axioms state that a property, or a composition of properties, is (possibly partially) more specific than another one. For example, one can state that isVerySimilarTo is a subproperty of isSimilarTo with degree greater or equal than 0.5.

It is important to mention that classical ontologies are a special case of fuzzy ontologies, so crisp ontologies are backwards compatible.

Several methodologies to develop fuzzy ontologies from classical ones have been proposed in the literature, such as DOF [AZ21], IKARUSOnto [AWKA12], FODM [LMR16], and Fuzzy OWL 2 [BS11].

2.3.2 Fuzzy Description Logics

Fuzzy Description Logics are extensions of Description Logics [BCE+15] to manage imprecise knowledge. They are the logical formalism in which the language Fuzzy OWL 2, a de facto standard to represent fuzzy ontologies that we will consider in this thesis, is based [BS11]. Now, we will recap syntax and semantics, reasoning tasks, and the implemented reasoning engines.

Syntax. The main syntactic differences are fuzzy datatypes and fuzzy axioms. There can also be some new concept and role constructors, but we will only consider aggregation concepts.

In classical DLs, datatypes are based on concrete domains (integer, etc.). In fuzzy DLs, there is a generalization based on fuzzy sets. A fuzzy datatype theory is defined similarly as a classical datatype theory $\mathbf{D} = \langle \Delta_{\mathbf{D}}, \cdot_{\mathbf{D}} \rangle$ but now $\cdot_{\mathbf{D}}$ is a set of concrete fuzzy domain predicates $\mathbf{d}$ with a fixed interpretation $\mathbf{d}^{\mathbf{D}} : \Delta_{\mathbf{D}} \to [0, 1]$ [LS08].

It is typical to restrict to the case where the datatype domain is a dense total order [BCE+15], and to restrict to unary fuzzy predicates described using the following membership functions:

$$\mathbf{d} \;\to\; \mathbf{left}(q_1, q_2) \mid \mathbf{right}(q_1, q_2) \mid \mathbf{tri}(q_1, q_2, q_3) \mid \mathbf{trap}(q_1, q_2, q_3, q_4)$$

where $\mathbf{left}, \mathbf{right}, \mathbf{tri}, \mathbf{trap}$ stand for left-shoulder, right-shoulder, triangular and trapezoidal membership functions.

Example 8. *Let us define a fuzzy datatype predicate* LowABV*, which denotes the degree of membership to the fuzzy set of drinks with low alcohol, using a triangular function:* $LowABV(x) = \mathbf{tri}(0.5, 3.1, 6.55)(x)$*. Now, we can define the concept* LowAlcoholBeer *representing all beers with a low alcohol level:*

$$\mathsf{LowAlcoholBeer} \equiv \mathsf{Beer} \sqcap \exists \mathsf{ABV}.\mathsf{LowABV} \quad \square$$

A fuzzy ontology or fuzzy Knowledge Base is denoted as $\mathcal{O} = \langle \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ and contains a fuzzy ABox $\mathcal{A}$, with axioms about individuals, a fuzzy TBox $\mathcal{T}$, with axioms about concepts, and a fuzzy RBox $\mathcal{R}$, with axioms about roles.

Table 2.4 shows the syntax of Fuzzy OWL 2 DL. Fuzzy ABox axioms are (A1)–(A7), fuzzy TBox axioms are (A8)–(A14), and fuzzy RBox axioms are (A15)–(A30). Note that the only syntactic differences with classical OWL 2 DL are 1 concept (aggregation concepts, C21), and 6 axioms (A1–A3, A8, A15, and A17). In these axioms, the degree of truth can be omitted if it is equal to 1, e.g., an axiom of the form $\langle a : C \geq 1 \rangle$ can be shortened as $a : C$.

Notice also that aggregation concepts were not in the original version of the language but were proposed later [BS13]. We have not defined the aggregation concept as a binary concept to allow non-associative aggregation operators. It is trivial to extend the syntax in order to support different types of aggregation concepts (e.g., an OWA concept and a weighted sum concept).

Semantics. The semantics is specified with a fuzzy interpretation and a fuzzy logic composed of operators t-norm $\otimes$, t-conorm $\oplus$, negation function $\ominus$ and implication function $\Rightarrow$ which operate on a fuzzy logic, such as Gödel, Łukasiewicz, Product, or Zadeh (see some fuzzy operators in Section 2.1.3). In our case, we also need an aggregation operator @.

A fuzzy interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the the domain (a non-empty set) and $\cdot^{\mathcal{I}}$ is a fuzzy interpretation function that maps:

- 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 object property R a function $R^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$; and
- to each fuzzy functional data property T a partial function $T^{\mathcal{I}} \colon \Delta^{\mathcal{I}} \times \Delta_{\mathbf{D}} \to \{0, 1\}$.

Table 2.5 shows how the semantics of Fuzzy OWL 2 DL is extended to complex concepts, complex roles, and axioms. Let $\phi \in \{a{:}C, (a_1, a_2){:}R,\ C_1 \sqsubseteq C_2\}$. 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 $\phi^{\mathcal{I}} \geq \alpha$. Furthermore, $\mathcal{I} \models \langle a{:}C \leq \alpha \rangle$ iff $(a{:}C)^{\mathcal{I}} \leq \alpha$.

Example 9. *Assuming that the price of* Sol *beer is* 0.51*e and it has an ABV of* 4.5*, Table 2.6 shows how to evaluate* $\varphi =$ CheapBeer $\sqcap$ LowAlcoholBeer *in Łukasiewicz logic.* □

	Concept	
(C1)	A	Atomic concept
(C2)	$\top$	Top concept
(C3)	$\bot$	Bottom concept
(C4)	$C_1 \sqcap C_2$	Conjunction
(C5)	$C_1 \sqcup C_2$	Disjunction
(C6)	$\neg C$	Negation
(C7)	$\forall R.C$	Universal restriction
(C8)	$\exists R.C$	Existential restriction
(C9)	$\exists R.\{a\}$	Value restriction
(C10)	$\forall T.\mathbf{d}$	Concrete universal restriction
(C11)	$\exists T.\mathbf{d}$	Concrete existential restriction
(C12)	$\exists T.\{v\}$	Concrete value restriction
(C13)	$\{a_1, \ldots, a_n\}$	Nominal
(C14)	$\geq\ n\ R.C$	At-least cardinality restriction
(C15)	$\leq\ n\ R.C$	At-most cardinality restriction
(C16)	$=n\ R.C$	Exact cardinality restriction
(C17)	$\geq\ n\ T.\mathbf{d}$	Concrete at-least restriction
(C18)	$\leq\ n\ T.\mathbf{d}$	Concrete at-most restriction
(C19)	$=n\ T.\mathbf{d}$	Concrete exact restriction
(C20)	$\exists R.\texttt{Self}$	Local reflexivity
(C21)	$@(C_1, C_2, \ldots, C_n)$	Aggregation
	Role	
(R1)	R_A	Atomic role
(R2)	U	Universal role
	Axiom	
(A1)	$\langle a{:}C\{\geq, \leq\}\alpha\rangle$	Fuzzy concept assertion
(A2)	$\langle (a_1, a_2){:}R\{\geq, \leq\}\alpha\rangle$	Fuzzy role assertion
(A3)	$\langle (a_1, a_2){:}\neg R\{\geq, \leq\}\alpha\rangle$	Fuzzy negative role assertion
(A4)	$(a, v){:}T\{\geq, \leq\}$	Concrete role assertion
(A5)	$(a, v){:}\neg T\{\geq, \leq\}$	Negative concrete role assertion
(A6)	$a_1 = a_2$	Equality assertion
(A7)	$a_1 \neq a_2$	Inequality assertion
A8)	$\langle C_1 \sqsubseteq C_2 \geq \alpha\rangle$	Fuzzy General Concept Inclusion (GCI)
(A9)	$C_1 \equiv C_2$	Concept equivalence
(A10)	$\texttt{dis}(C_1, \ldots, C_n)$	Disjoint concepts
(A11)	$\texttt{dom}(R, C)$	Domain role axiom
(A12)	$\texttt{ran}(R, C)$	Range role axiom
(A13)	$\texttt{dom}(T, C)$	Domain concrete role axiom
(A14)	$\texttt{ran}(T, \mathbf{d})$	Range concrete role axiom
(A15)	$\langle R_{11} \ldots R_{1n} \sqsubseteq R_2 \geq \alpha\rangle$	Fuzzy Role Inclusion Axiom (RIA)
(A16)	$R_1 \equiv R_2$	Role equivalence
(A17)	$\langle T_1 \sqsubseteq T_2 \geq \alpha\rangle$	Fuzzy concrete RIA
(A18)	$T_1 \equiv T_2$	Concrete role equivalence
(A19)	$\texttt{inv}(R_1, R_2)$	Inverse roles
(A20)	$\texttt{fun}(R)$	Functional role
(A21)	$\texttt{fun}(T)$	Functional concrete role
(A22)	$\texttt{invfun}(R)$	Inverse functional role
(A23)	$\texttt{invfun}(T)$	Inverse functional concrete role
(A24)	$\texttt{trans}(R)$	Transitive role
(A25)	$\texttt{dis}(R_1, \ldots, R_n)$	Disjoint roles
(A26)	$\texttt{dis}(T_1, \ldots, T_n)$	Disjoint concrete roles
(A27)	$\texttt{ref}(R)$	Reflexive role
(A28)	$\texttt{irr}(R)$	Irreflexive role
(A29)	$\texttt{sym}(R)$	Symmetric role
(A30)	$\texttt{asy}(R)$	Asymmetric role

Table 2.4: Syntax of concepts, roles, and axioms in Fuzzy OWL 2 DL.

	Complex concept
(C2)	1
(C3)	0
(C4)	$C_1^{\mathcal{I}}(x) \otimes C_2^{\mathcal{I}}(x)$
(C5)	$C_1^{\mathcal{I}}(x) \oplus C_2^{\mathcal{I}}(x)$
(C6)	$\ominus C^{\mathcal{I}}(x)$
(C7)	$\sup_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)\}$
(C8)	$\inf_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)\}$
(C9)	$R^{\mathcal{I}}(x, a^{\mathcal{I}})$
(C10)	$\inf_{v \in \Delta_{\mathbf{D}}} \{T^{\mathcal{I}}(x,v) \Rightarrow \mathbf{d}_{\mathbf{D}}(v)\}$
(C11)	$\sup_{v \in \Delta_{\mathbf{D}}} \{T^{\mathcal{I}}(x,v) \otimes \mathbf{d}_{\mathbf{D}}(v)\}$
(C12)	$T^{\mathcal{I}}(x, v_{\mathbf{D}})$
(C13)	1 if $x \in \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$, 0 otherwise
(C14)	$\sup_{y_1,\ldots,y_n \in \Delta^{\mathcal{I}}} \{(\min_{i=1}^{n} \{R^{\mathcal{I}}(x,y_i) \otimes C^{\mathcal{I}}(y_i)\}) \bigotimes (\otimes_{j<k} \{y_j \neq y_k\})\}$
(C15)	$\inf_{y_1,\ldots,y_{n+1} \in \Delta^{\mathcal{I}}} \{(\min_{i=1}^{n+1} \{R^{\mathcal{I}}(x,y_i) \otimes C^{\mathcal{I}}(y_i)\}) \Rightarrow (\oplus_{j<k} \{y_j = y_k\})\}$
(C16)	$(\geq\ n\ R.C^{\mathcal{I}}(x)) \otimes (\leq\ n\ R.C^{\mathcal{I}}(x))$
(C17)	$\sup_{v_1,\ldots,v_n \in \Delta_{\mathbf{D}}} \{(\min_{i=1}^{n} \{T^{\mathcal{I}}(x,v_i) \otimes \mathbf{d}_{\mathbf{D}}(v_i)\}) \bigotimes (\otimes_{j<k} \{y_j \neq y_k\})\}$
(C18)	$\inf_{v_1,\ldots,v_{n+1} \in \Delta_{\mathbf{D}}} \{(\min_{i=1}^{n+1} \{T^{\mathcal{I}}(x,v_i) \otimes \mathbf{d}_{\mathbf{D}}(v_i)\}) \Rightarrow (\oplus_{j<k} \{y_j = y_k\})\}$
(C19)	$(\geq\ n\ T.\mathbf{d}^{\mathcal{I}}(x)) \otimes (\leq\ n\ T.\mathbf{d}^{\mathcal{I}}(x))$
(C20)	$R^{\mathcal{I}}(x,x)$
(C21)	$@(C_1^{\mathcal{I}}(x), C_2^{\mathcal{I}}(x), \ldots, C_n^{\mathcal{I}}(x))$
	Complex role
(R2)	1
	Axiom
(A1)	$C^{\mathcal{I}}(a^{\mathcal{I}}) \{\geq, \leq\} \alpha$
(A2)	$R^{\mathcal{I}}(a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \{\geq, \leq\} \alpha$
(A3)	$\ominus R^{\mathcal{I}}(a_1^{\mathcal{I}}, a_2^{\mathcal{I}}) \{\geq, \leq\} \alpha$
(A4)	$T^{\mathcal{I}}(a^{\mathcal{I}}, v_{\mathbf{D}})$
(A5)	$\ominus T^{\mathcal{I}}(a^{\mathcal{I}}, v_{\mathbf{D}})$
(A6)	$a_1^{\mathcal{I}} = a_2^{\mathcal{I}}$
(A7)	$a_1^{\mathcal{I}} \neq a_2^{\mathcal{I}}$
(A8)	$\inf_{x \in \Delta^{\mathcal{I}}} \{C_1^{\mathcal{I}}(x) \Rightarrow C_2^{\mathcal{I}}(x)\} \geq \alpha$
(A9)	$\forall x \in \Delta^{\mathcal{I}}, C_1^{\mathcal{I}}(x) = C_2^{\mathcal{I}}(x)$
(A10)	$\forall x \in \Delta^{\mathcal{I}}, \min_{1 \leq i < j \leq n} \{C_i^{\mathcal{I}}(x), C_j^{\mathcal{I}}(x)\} = 0$
(A11)	$\forall x, y \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,y) \leq C^{\mathcal{I}}(x)$
(A12)	$\forall x, y \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,y) \leq C^{\mathcal{I}}(y)$
(A13)	$\forall x \in \Delta^{\mathcal{I}}, \forall v \in \Delta_{\mathbf{D}}, T^{\mathcal{I}}(x,v) \leq C^{\mathcal{I}}(x)$
(A14)	$\forall x \in \Delta^{\mathcal{I}}, \forall v \in \Delta_{\mathbf{D}}, T^{\mathcal{I}}(x,v) \leq \mathbf{d}_{\mathbf{D}}(v)$
(A15)	$\inf_{x_1, x_{n+1} \in \Delta^{\mathcal{I}}} \{\sup_{x_2 \ldots x_n \in \Delta^{\mathcal{I}}} \{R_1^{\mathcal{I}}(x_1,x_2) \otimes \cdots \otimes R_n^{\mathcal{I}}(x_n,x_{n+1})\} \Rightarrow R^{\mathcal{I}}(x_1,x_{n+1})\} \geq \alpha$
(A16)	$\forall x, y \in \Delta^{\mathcal{I}}, R_1^{\mathcal{I}}(x,y) = R_2^{\mathcal{I}}(x,y)$
(A17)	$\inf_{x \in \Delta^{\mathcal{I}}, v \in \Delta_{\mathbf{D}}} T_1^{\mathcal{I}}(x,v) \Rightarrow T_2^{\mathcal{I}}(x,v) \geq \alpha$
(A18)	$\forall x \in \Delta^{\mathcal{I}}, \forall v \in \Delta_{\mathbf{D}}, T_1^{\mathcal{I}}(x,v) = T_2^{\mathcal{I}}(x,v)$
(A19)	$\forall x, y \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,y) = R^{\mathcal{I}}(y,x)$
(A20)	$\forall x, y_1, y_2 \in \Delta^{\mathcal{I}}$, if $\min\{R^{\mathcal{I}}(x,y_1), R^{\mathcal{I}}(x,y_2)\} > 0$ then $y_1 = y_2$
(A21)	$\forall x \in \Delta^{\mathcal{I}}, \forall v_1, v_2 \in \Delta_{\mathbf{D}}$, if $\min\{T^{\mathcal{I}}(x,v_1), T^{\mathcal{I}}(x,v_2)\} > 0$ then $v_1 = v_2$
(A22)	$\forall x_1, x_2, y \in \Delta^{\mathcal{I}}$, if $\min\{R^{\mathcal{I}}(x_1,y), R^{\mathcal{I}}(x_2,y)\} > 0$ then $x_1 = x_2$
(A23)	$\forall x_1, x_2 \in \Delta^{\mathcal{I}}, \forall v \in \Delta_{\mathbf{D}}$, if $\min\{T^{\mathcal{I}}(x_1,v), T^{\mathcal{I}}(x_2,v)\} > 0$ then $x_1 = x_2$
(A24)	$\forall x, y \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,y) \geq \sup_{z \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,z) \otimes R^{\mathcal{I}}(z,y)$
(A25)	$\forall x, y \in \Delta^{\mathcal{I}}, \min_{1 \leq i < j \leq n} \{R_i^{\mathcal{I}}(x,y), R_j^{\mathcal{I}}(x,y)\} = 0$
(A26)	$\forall x \in \Delta^{\mathcal{I}}, \forall v \in \Delta_{\mathbf{D}}, \min_{1 \leq i < j \leq n} \{T_i^{\mathcal{I}}(x,v), T_j^{\mathcal{I}}(x,v)\} = 0$
(A27)	$\forall x \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,x) = 1$
(A28)	$\forall x \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,x) = 0$
(A29)	$\forall x, y \in \Delta^{\mathcal{I}}, R^{\mathcal{I}}(x,y) = R^{\mathcal{I}}(y,x)$
(A30)	$\forall x, y \in \Delta^{\mathcal{I}}$, if $R^{\mathcal{I}}(x,y) > 0$ then $R^{\mathcal{I}}(y,x) = 0$

Table 2.5: Semantics of complex concepts, complex roles, and axioms in Fuzzy OWL 2 DL.

$\mathcal{I}$	Price	CheapBeer **left**$(50, 55)(0.51)$	ABV	LowAlcoholBeer **tri**$(0.5, 3.1, 6.55)(4.5)$	$\mathcal{I}(\varphi)$
Sol	0.51	0.8	4.5	0.59	$max(0, 0.8 + 0.59 - 1) = 0.39$

Table 2.6: An interpretation with the t-norm $\otimes$ Łukasiewicz operators.

It is also possible to combine operators from different fuzzy logics. For example, 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 $C_1 \sqsubseteq C_2$ [BDGR09].

2.3.3 Fuzzy OWL ontologies

Crisp ontologies are represented using the standard OWL 2 language, with a semantics based on a classical logic. Currently, it does not exist a standard for representing fuzzy ontologies, however Fuzzy OWL 2 is a popular option and can be seen as a de facto standard [BS11].

Fuzzy OWL 2 extends classical OWL 2 with OWL 2 annotations of the type fuzzyLabel encoding the fuzzy information using an XML-like syntax. The aim is start with an OWL 2 and add annotations to encode the fuzzy information that OWL 2 cannot directly encode.

Example 10. *Figure 2.8 shows an example of annotation to build a fuzzy axiom. Given the classical assertion* window001 : TallWindow*, stating that* window001 *is an instance of the class* TallWindow*, the annotation converts it into a fuzzy assertion* $\langle$window001 : TallWindow $\geq 0.67\rangle$*, ensuring that the axioms holds with degree at least* 0.67*.* □

The main features of the language are the following ones:

- It is possible to annotate fuzzy axioms adding a degree of truth, to represent fuzzy datatypes, and to define specific elements of fuzzy ontologies (e.g., modifiers, aggregation concepts, etc.).
- It provides a Protégé plug-in[17] that helps to add annotations while making the syntax of annotations transparent.
- It provides an API for Java applications to import Fuzzy OWL 2 ontologies and to translate them to fuzzyDL [BS16a] and DeLorean [BCFGR12] syntaxes. The API makes developing similar parsers easy.
- It is one of the syntaxes supported by fuzzyDL.

[17] `http://webdiis.unizar.es/~fbobillo/fuzzyOWL2`

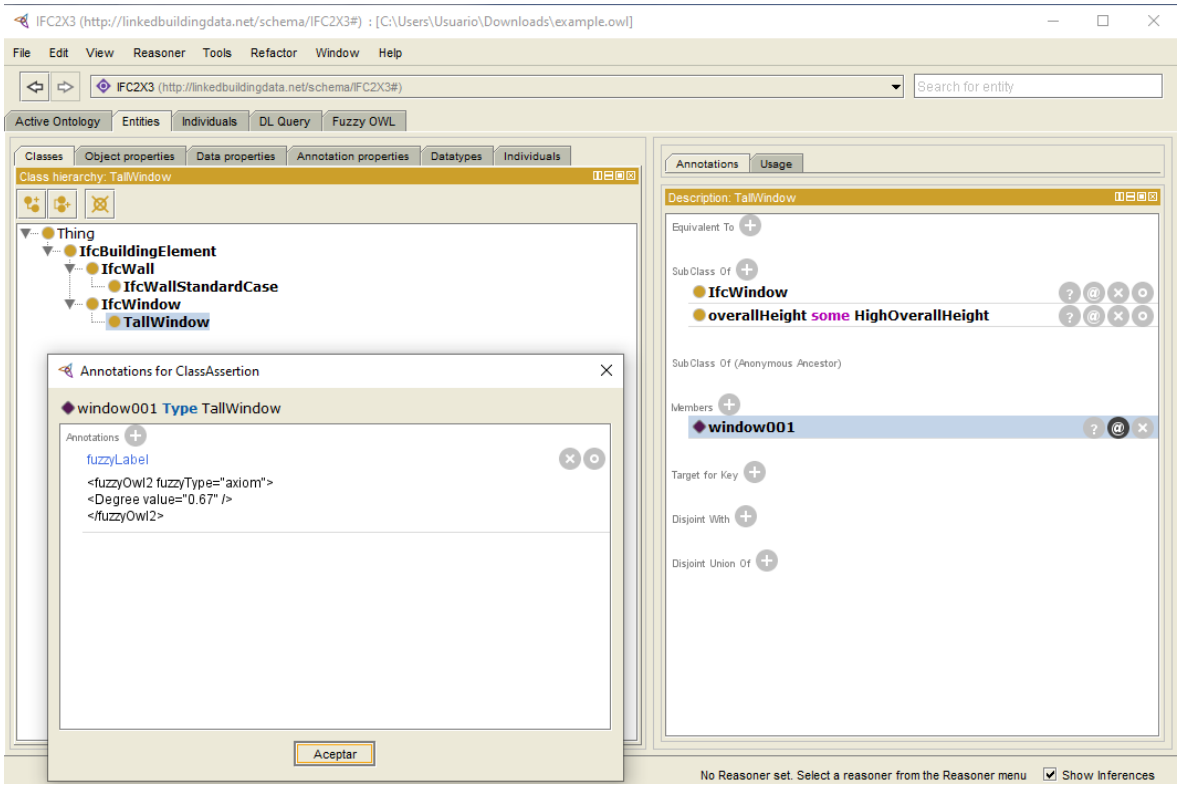


Figure 2.8: Snapshot of a Fuzzy OWL 2 fuzzy ontology edited in Protégé.

Example 11. *Figure 2.9 shows an example of the graphical interface of the plug-in. In particular, it shows how to build fuzzy datatypes. The user firstly choose the name of the fuzzy datatype, then selects the type of the membership function, and finally the value of the parameters.* □

2.3.4 Fuzzy reasoning

In this section we will enumerate the main reasoning tasks and the existing implemented reasoners. Then, we will give more details about fuzzyDL reasoning algorithm.

Fuzzy reasoning tasks. fuzzy DLs extend the reasoning tasks of classical DLs to manage the degrees of truth. Furthermore, some new reasoning tasks appear. The main reasoning tasks are the following ones [BS15]:

- *Fuzzy KB consistency/satisfiability*: check if there is a logical model in the fuzzy ontology $\mathcal{O}$.

- *Entailment:* a fuzzy axiom τ is a logical consequence of $\mathcal{O}$ (or $\mathcal{O}$ entails τ), denoted $\mathcal{O} \models \tau$, iff every model of $\mathcal{O}$ is a model of τ.

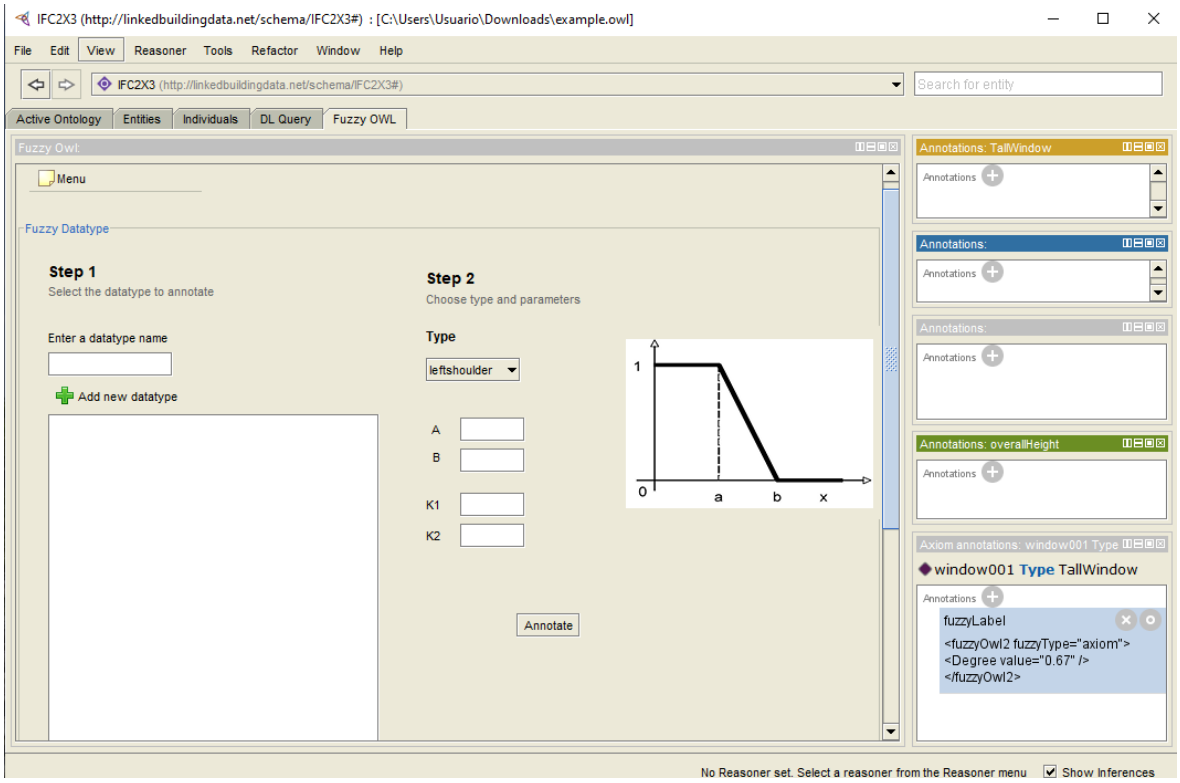


Figure 2.9: Snapshot of Fuzzy OWL 2 Protégé plugin.

– *Fuzzy concept satisfiability*: a fuzzy concept C is α-satisfiable w.r.t $\mathcal{O}$ iff there is a model $\mathcal{I}$ of $\mathcal{O}$ such that $C(x)^{\mathcal{I}} \geq \alpha$ for some element $x \in \Delta^{\mathcal{I}}$.

– *Fuzzy concept subsumption*: C_2 α-subsumes C_1 w.r.t $\mathcal{O}$ iff every model $\mathcal{I}$ of $\mathcal{O}$ satisfies $\forall x \in \Delta^{\mathcal{I}}, C_1^{\mathcal{I}}(x) \Rightarrow C_2^{\mathcal{I}}(x) \geq \alpha$. Fuzzy property subsumption can be similarly defined.

– *Classification*: compute a fuzzy concept hierarchy and fuzzy property hierarchy based on the subsumption relationships.

– *Best Entailment Degree (BED)*: the BED of $\phi \in \{a : C, (a_1, a_2) : R, C \sqsubseteq D\}$ is the maximal degree α such that every model of the fuzzy ontology entails $\langle \phi \geq \alpha \rangle$.

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

– *Best Satisfiability Degree (BSD)*: the BSD of a fuzzy concept C with respect to a fuzzy ontology $\mathcal{O}$ is defined as the maximal degree α such that C can have instances that belong to it with degree α, i.e.,

$$bsd(\mathcal{O}, C) = \sup_{\mathcal{I} \models \mathcal{O}} \sup_{x \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(x)$$

– *Instance retrieval*: given a fuzzy concept C, retrieve a set of pairs $\langle i, \alpha \rangle$ such that i belongs to C with degree greater or equal than $\alpha > 0$.

– *Realization*: given an individual i, retrieve a set of pairs $\langle C, \alpha \rangle$ such that i belongs to C with degree greater or equal than $\alpha > 0$.

Fuzzy reasoning engines. Fuzzy ontology reasoning is performed with semantic fuzzy reasoners. In this thesis, we will work with *fuzzyDL* [BS16a].

fuzzyDL supports $\mathcal{SHIF}(\mathbf{D})$ fuzzy DL and supports three semantics, namely Zadeh and Łukasiewicz fuzzy logics, but also Classical logic (for non-fuzzy ontologies) and Gödel operators. fuzzyDL has its own syntax for the fuzzy ontologies (FDL format) and also accepts Fuzzy OWL 2 ontologies. This fuzzy engine solves the previously mentioned reasoning tasks, but also variable maximization/minimization and defuzzification (using SOM, LOM and MOM methods). Indeed, it internally reduces all tasks to variable minimization. fuzzyDL also implements several optimizations to reduce the running time.

Other popular fuzzy reasoners are *DeLorean*[18] [BCFGR12], *Fire* [SSSK06], *FPLGERDS* [Hab07], *GURDL* [HPS07], *YADLR* [KA07], *FRESG* [WMY09], *LiFR* [TDKM14], and *SMT*-based solver [ABB+13]. It is also worth to mention that *MiniME* classical reasoner was extended to support fuzzy datatypes [RSS10].

fuzzyDL reasoning algorithm. In this section, we will give further details about fuzzyDL reasoning algorithm. The algorithm is based on a combination of a tableaux algorithm and an optimization problem. The idea is to build a *completion-forest* (a collection of nodes arbitrarily connected by edges) while generating some constraints, and then minimize a variable with respect to the constraint set. Each node v is labeled with a set $\mathcal{L}(v)$ of concept expressions, and each edge $\langle v_1, v_2 \rangle$ is labeled with a set $\mathcal{L}(\langle v_1, v_2 \rangle)$ of roles.

– Firstly, there is some preprocessing. For example, in Łukasiewicz and Zadeh fuzzy DLs, and in Classical DLs, each concept is transformed into its Negation Normal Form [BS16a], where the negation only appears before atomic concepts.

– Then, for each individual a in the ontology, it creates a new node v_a.

– For each concept assertion $\langle a : C \geq \alpha \rangle \in \mathcal{O}$, it ensures that $\mathcal{L}(v_a) \leftarrow \mathcal{L}(v_a) \cup \{C\}$, and sets $\mathbf{C} \leftarrow \mathbf{C} \cup \{x_{v_a:C} \geq \alpha\}$.

[18] `http://webdiis.unizar.es/~fbobillo/delorean`

– Similarly, for each property assertion $\langle (a_1, a_2) : R \geq \alpha \rangle \in \mathcal{O}$, it creates an edge $\langle v_{a_1}, v_{a_2} \rangle$ between the nodes v_{a_1}, v_{a_2} if it does not exist, 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 inequation constraints. For the sake of concrete illustration, Table 2.7 shows the rules for fuzzy $\mathcal{ALC}$ with respect to an empty TBox, and the encoding of the fuzzy operators is shown in Table 2.8 for Łukasiewicz (Ł), Gödel (G), and Zadeh (Z) fuzzy logics, where $\epsilon > 0$. In classical semantics, any of the previous encodings is possible, but we also need to set $x_{v_a:C}, x_{(v_{a_1}, v_{a_2}):R} \in \{0, 1\}$ for every variable in $\mathbf{C}$. These inequations need to hold to keep 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 [BS16a]. To solve the optimization problem, fuzzyDL uses Gurobi mathematical optimization solver[19].

Remark 1. *For simplicity, Table 2.7 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 Gödel $\mathcal{ALC}$ 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 Gödel DLs), see e.g., [BCE+15].*

In Łukasiewicz, Gödel, and Zadeh fuzzy DLs, we obtain a bounded Mixed Integer Linear Programming (MILP) problem [Str05]; in other fuzzy DLs more complex optimization problems can be obtained. A MILP problem consists in 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 [BS09]. In particular, a constraint set $\mathbf{C}$ can contain linear equations $w_1 x_1 + \cdots + w_n x_n \bowtie w_0$ 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 $\bowtie \in \{\leq, \geq, =\}$.

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 $w_1 z_1$ and a term $w_2 z_2$ and then, extend $\rightsquigarrow_{\mathbf{C}}$ to its transitive closure. The following result can be shown:

[19] `http://www.gurobi.com`

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 $\mathbf{C} = \mathbf{C} \cup \{x_{(v,w):R} \otimes x_{w:C} = z, z \geq x_{v:\exists R.C}\}$
$(\forall)$	$\forall R.C \in \mathcal{L}(v)$ $R \in \mathcal{L}(\langle v, w\rangle)$	$\mathcal{L}(w) = \mathcal{L}(w) \cup \{C\}$ $\mathbf{C} = \mathbf{C} \cup \{x_{v:\forall R.C} \geq z, z = x_{(v,w):R} \Rightarrow x_{w:C}\}$

Table 2.7: Reasoning rules for fuzzy $\mathcal{ALC}$ with an empty TBox (based on [BS17]).

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	$\{z \geq 1 - x_1, z \geq x_2, (1 - x_1) + y \geq z, x_2 + (1-y) \geq z, y \in \{0,1\}\}$

Table 2.8: Encoding of some popular fuzzy logic operators [BS17].

Lemma 1 ([BS15]). *A constraint set* $\mathbf{C}$ *can be partitioned into a set of constraint sets* $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}$ *verifying the following conditions:*

(OP1) If z *occurs in* $\mathbf{C}_i$ *then* z *does not occur in* $\mathbf{C}_j$, $\forall i, j \in \{1, \ldots, n\}, i \neq j$,

(OP2) $\bigcup_{i \in \{1,\ldots,n\}} \mathbf{C}_i = \mathbf{C}$,

(OP3) $\forall z_j, z_k$ *occurring in* $\mathbf{C}_i, z_j \rightsquigarrow_{\mathbf{C}} z_k$, $\forall i \in \{1, \ldots, n\}$.

$\mathbf{C}$ has a solution iff $\mathbf{C}_i$ has a solution for every $i \in \{1, \ldots, n\}$. 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 constraint sets, i.e., if a variable z is in the i-th connected component, then the constraints where z occurs should be placed in $\mathbf{C}_i$ [BS15].

2.4 Clustering

Clustering is an unsupervised technique in machine learning that discovers groups (known as clusters) of data points with similar properties or behavior. The aim of the clustering algorithm consists in finding the most similar (or dissimilar) data points between them to form clusters, according to some specific metrics such as distance measures (e.g., Euclidean, Manhattan, etc.).

An exhaustive analysis about clustering traditional methods and modern techniques applied to different domains is described in [XT15]. In this thesis, we will focus on *centroid-based* methods, where each cluster is characterized by a single value, the centroid.

More formally, clustering algorithms group n data values into k clusters. The set of data values is denoted $X = \{x_j\}, j = 1, \ldots, n$, and the set of clusters is denoted $C = \{C_i\}, i = 1, \ldots, k$. c_i denotes the centroid of the cluster C_i.

In the following, we will describe three clustering methods, namely k-means, fuzzy c-means and mean-shift.

k-means. It is one of the most famous clustering algorithms [Llo82]. k-means needs to fix the number of clusters k a priori. The main idea is to minimize the mean square error. The main steps are the following ones:

1. The k initial centroids c_i are randomly computed.
2. Each point x_j is assigned to its nearest cluster, denoted $C(x_j)$, according to Euclidian distance [SYR13]:

$$C(x) = C_k \text{ if } argmin_i \mid\mid x_j - c_i \mid\mid^2 = k \tag{2.23}$$

3. The centroids are recomputed:

$$c_i = \frac{\sum_{x_j \in C_i} x_j}{|C_i|} \tag{2.24}$$

4. Steps 2 and 3 are repeated until the algorithm reaches a maximum number of iterations or there are no more changes in the centroids after two complete iterations.

k-means is very sensitive to the initial randomly selected centroids and does not work well with outliers.

Fuzzy c-means. This algorithm is an extension of k-means where every point can belong to several clusters with different partial degrees in the unit interval $[0, 1]$ rather than being associated with just one cluster [Bez81]. Processes of initialization, iteration, and ending are the same as in k-means. Fuzzy c-means considers c fuzzy clusters[20] and a matrix of membership degrees μ, where $\mu_{ij} \in [0, 1]$ denotes the membership degree of the datum x_i to the j-th cluster [CDB86]. So, the new mathematical operations are:

- The location of the centroids are computed as:

$$c_i = \frac{(\sum_{j=1}^{n} \mu_{ij}^m x_j)}{\sum_{j=1}^{n} \mu_{ij}^m} \tag{2.25}$$

  where $m \geq 1$ is a parameter indicating a fuzziness degree.

- The membership degrees are updated as:

$$\mu_{ij} = (\sum_{k=1}^{c} \frac{||x_j - c_i||^{2/(m-1)}}{||x_j - c_k||^{2/(m-1)}})^{-1} \tag{2.26}$$

[20]The symbol c is used for historical reasons, although the number of clusters in k-means was denoted as k.

Mean-shift. It is a popular method used for clustering and image segmentation [Che95, CM02]. Mean-shift works with a sliding-window for each point, computes the mean of the data points in the sliding-window, and then moves its center to the mean. The aim of this method is to find modes or the local maxima of density in a data space. It requires a Kernel function such as a Gaussian Kernel K_g to keep track of the nearest neighbors of each x_i according to a bandwidth (or window size) h. The Gaussian Kernel function is represented as the follows:

$$K_g(x_j - x_i) = \frac{1}{\sqrt{2\pi}} exp(-\frac{1}{2}||\frac{x_j - x_i}{h}||^2) \tag{2.27}$$

The bandwidth may be computed with the rule of thumb in [Tur99]:

$$h = 1.06\sigma^2 n^{-1/5} \tag{2.28}$$

where σ^2 is the variance of n size data points.

Given a candidate point x_i for each iteration of the algorithm, the point is updated according to the next operation:

$$x_i = m(x_i) \tag{2.29}$$

where m is the mean-shift vector defined as:

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} N(x_j - x_i) x_i / h}{\sum_{x_j \in N(x_i)} N(x_j - x_i) 1/h} \tag{2.30}$$

where $N(x_i)$ is the neighborhood of data points within a given a local seeking distance such as $l = h/2$ around x_i.

Mean-shift vector converges into a set of centroids after removing data points at a too close distance. Note that this method does not need to create an initial number of clusters.

Table 2.9 shows some important features of the previous algorithms. All of them are centroid-based, many of them needs to fix the number of centroids (column "Fixed k"), and few works support partial membership to a cluster (column "Partial membership") or require additional parameters to define a sliding window ("Window").

Algorithm	Centroid-based	Fixed k	Partial membership	Window
k-means	•	•		
fuzzy c-means	•	•	•	
mean-shift	•			•

Table 2.9: Comparison of some clustering algorithms.

2.5 Mobile computing

In the last decade mobile devices (smartphone, tablets, etc.) have changed the behavior of the people in their lifestyles, including how to buy, travel, work out, do business, etc. According to the International Data Corporation (IDC), the smartphone market will reach 1.492 billion units in 2024[21]. Statista stats show that in 2021 there are 6.4 billion smartphone users worldwide[22], if the current world population[23] is 7.9 billion then 81% are smartphone owners.

In the first quarter of the year 2021, the average daily time people spent using mobile devices, specifically at mobile applications, was about 4 hours 12 minutes[24]. The pandemic changed consumer behavior on mobile devices and a daily time of 4 hours was reported for the first time in the USA, Turkey and México. In countries such as Brazil or South Korea, the daily time was more than 5 hours. At the start of 2021, the average smartphone data usage was 8.4GB per smartphone according to Ericsson[25]. The experts predict a grow up of traffic from mobile devices from 10 GB to 35 GB by 2026 as a result of user activity (mostly video streaming).

Mobile devices have many technical advantages as connection (wireless technologies GSM, Wi-Fi, NFC, 4G, 5G, etc.), mobility, sensors (e.g., geographic location) and increasing computation power. However, they have some limitations with respect to desktop computers, although the differences are being reduced, e.g., smaller screen size, memory, or CPU power, less stable connectivity, battery duration, etc.

Mobile applications are usually called *apps*. They typically make it possible to interact with the user in a different way than desktop applications. Furthermore, they can also interact with external services (e.g., cloud services). In order to develop apps, there are three main approaches, namely, web, native, and hybrid [Hel13]. We describe them in the following:

– *Web.* The development of this type of app integrates standard web technologies

[21] https://www.idc.com/promo/smartphone-market-share
[22] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
[23] https://www.worldometers.info/world-population/#growthrate
[24] https://www.appannie.com/en/insights/market-data/q1-2021-market-index
[25] https://www.businessofapps.com/data/app-statistics/#3.6

(HTML, CSS, and JavaScript) with libraries such as JQuery Mobile (based on JQuery) [Smu12]. The web application is uploaded to a conventional web server. Finally, the users interact with the app by means of the mobile web browser. Web app is independent of the hardware and may work with different browsers.

– *Native.* The developers use mobile software development kits (SDK's). The tools, APIs, and programming languages are native to a specific mobile Operating System (OS) [TH10]. The most installed operating systems for mobile devices are Android[26] (open source software sponsored by Google) and iOS[27] (developed by Apple). In 2021, StatCounter reported that the smartphone market share was 71.89% for Android and 27.34% for iOS[28]. Native apps have a higher performance and support a higher degree of customization thanks to the access to hardware resources and specific features by means of the APIs of each OS. The negative aspect is the development cost, as it requires different developments for different individual platforms.

– *Hybrid.* The development is a combination of the developments of web and native types. The hybrid apps are developed employing open source libraries and cross-platforms SDK's that permit the access to hardware devices (such as sensors or camera) and to the OS (e.g., to the file system) [LLNE16]. Here, mobile hardware access is limited compared to native apps but is better than in web apps. Web technologies and frameworks which permit converting web apps into native apps for different OS are commonly used. The benefits of the hybrid development are smaller cost and time than in native apps, but the app performance is worse.

Table 2.10 compares the previous types of app development according to the SDK, programming languages, execution mode, programming access to the OS and hardware, and time and cost of development.

In this thesis, we will develop mobile native apps for the Android platform for many reasons:

– Smartphone market demands: it is the most used platform nowadays, so there are more potential users.

[26] `https://developer.android.com/`

[27] `https://developer.apple.com/`

[28] `https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202101-202112-bar`

	Web	Native	Hybrid
SDK	JQuery mobile, JQtouch	Android Studio (Android) Xcode (iOS)	Flutter, Ionic, Phonegap, React Native, Titanium, Xamarin [MBHG17]
Programming language	HTML5, JavaScrip, CSS	Java, Kotlin (Android) Objctive-C, Swift (iOS)	HTML5, JavaScrip, CSS
Execution mode	Web browser	OS	Embedded web browser and part on OS
OS API access	Limited	Full	Partial
Hardware access	Limited	Full	Partial
Development time	Fast	Medium/High	Fast
Development cost	Low	Medium/High	Low/Medium

Table 2.10: Mobile development types.

- Most of the APIs that we will use to access some tools (e.g., reasoners, parsers, etc) are implemented in Java language, which is also a language to write Android applications.
- Access to real Android devices for debugging, testing and deploying is easy and cheap.
- There is a lot of information on Internet about Android development.

Another interesting point is reasoning with ontologies on mobile devices. Three different scenarios have been proposed: external reasoning, local reasoning and hybrid reasoning [BBMP17].

- *Local reasoning* means that all the reasoning is done on a mobile device. It requires that a reasoner is installed on the mobile device and it computes the reasoning tasks without any external intervention. This mode does not depend on a server, but reasoning can be challenging because of the limitations of mobile devices in terms of CPU power, memory, or battery, and the lack of optimization techniques for mobile devices (as ontology reasoners are typically optimized for desktop computers). Indeed, there is some evidence that reasoning time is only affordable in small or not very expressive ontologies [BYBM15].
- *Remote reasoning* (or external reasoning) needs a powerful hardware (e.g., a server on the cloud) where reasoning is actually performed. The mobile user makes petitions to the server and gets the answers from it. That architecture is suitable for large ontologies and complex axioms, as they typically require high CPU processing and memory size, and traditional ontology reasoners are optimized for desktop computers. Furthermore, one can consider a server which is as powerful as required by the application. However, in ubiquitous and mobile scenarios where context-awareness and privacy preserving play a crucial role,

sending sensitive data to a remote server might be an important privacy breach, and even sending non-sensitive data might be dangerous as it could enable the inference of sensitive information. Some disadvantages are data privacy, as the ontology could include confidential data, the fact that a high amount of data to transfer through the network, and the need for a good connectivity, but this requires assuming that the connectivity is fast and stable enough, which is not often the case in mobile computing environments.

– *Hybrid reasoning* does a part of the reasoning externally, and a part locally. Hybrid reasoning seems to be as a promising trade-off between the other options: on the one hand, it can benefit from the speed of an external device to preprocess the ontology, and on the other hand it can also add new sensitive information without compromising it (the information is added on the user's device). Moreover, this approach does not require to communicate with the server too many times (typically, only to download the preprocessed ontology). This latter aspect is interesting because in mobile computing environments connectivity is often unreliable and battery consuming.

Local reasoning can use *native* (developed for a specific platform) or *ported reasoners* (reusing the desktop versions). Bobed et al. made a rigorous study about the reuse of ontology reasoners on Android platform [BYBM15], showing that some reasoners could be imported directly on an Android project (e.g., TrOWL), making ported versions of reasoners (e.g., HermiT) available, and noting that a considerable number of reasoners are not compatible with the platform due special classes or third party libraries (such as fuzzyDL).

According to [BYBM15], there are 9 reasoners available for Android devices, namely *CB* [Kaz09], *ELK*, *HermiT*, *jcel* [Men12], *JFact*, *MORe* [ARCGHJR13], *Pellet*, *TReasoner* [GI13], and *TrOWL*. To date, there are no similar ports for iOS devices and none of the previously existing reasoners are implemented in the languages natively supported on iOS (Objective-C or Swift).

Regarding native reasoners, there are implementations for the following mobile operating systems:

– Android: *MiniME* [SRL+14][29] and the system in [WA18].

– iOS: *MiniME-Swift* [RSG+19].[30]

[29] `http://sisinflab.poliba.it/swottools/minime`
[30] `http://sisinflab.poliba.it/swottools/minime-swift`

- J2ME: COROR [TKO15], μ-OR [AK09], *Pocket KRHyper* [SK05], and the systems in [KDS+08, MHLN06].
- Windows Mobile: *mTableau* [SKG09] and MiRE4OWL [KPHL10].
- Multiplatform: *Tiny-ME* [RSB+22] works in Windows, Linux, macOS, Docker containers, Android, and iOS.
- Others:
  - *LiRoT* [BMSL22] is implemented in C for constrained devices such as Arduino Due or ESP32.
  - The system in [SS11] is implemented in CLIPS,
  - There is no information about *Delta* [MHK12].

Currently, three of these native reasoners support OWL 2 RL [SS11, TKO15, WA18], and the other ones cannot fully support OWL 2 or any of its profiles.

To implement hybrid reasoning on mobile devices, four strategies have been proposed [BBMP17]:

1. The server can send an expanded ontology (with all the inferences explicitly represented) back.
2. If the mobile device has a copy of the ontology, the server can send only a list of the inferences and the axioms can be integrated on the mobile device.
3. If the reasoner is serializable, the external server can send instead a copy of the reasoner. The mobile device avoids the cost of loading and preprocessing the ontology, but requires that both devices (the server and the mobile) use the same reasoner and version, and a common serialization strategy.
4. If the mobile device has a copy of the ontology, the external server can provide a serialized version of the reasoner but not including the original ontology, which will be locally integrated. This requires some additional time to add the axioms but reduces the transmission size.

Finally, we conclude this analysis by noting that to the best of our knowledge there does not exist any fuzzy ontology reasoner for mobile devices.

Chapter 3

Contributions to fuzzy ontology learning

In this chapter, we describe several novel methods to learn some elements of fuzzy ontologies. In particular, we will focus on learning fuzzy datatypes for numerical data properties. The first two sections focus on learning fuzzy datatypes from examples of numerical data. On the one hand, Section 3.1 focuses on local fuzzy datatypes, specific of one individual, and resulting in one triangular fuzzy membership function per each individual and data property. On the other hand, Section 3.2 studies global fuzzy datatypes that are learned using unsupervised clustering methods. Finally, Section 3.3 focuses on learning fuzzy datatypes from several experts: it describes a technique using linguistic aggregations operators to merge the definitions of different experts (represented using fuzzy datatypes) into a unique consensual fuzzy datatype. This approach is appropriate for scenarios without learning examples. While existing approaches assume that a single expert defines the fuzzy datatypes, we argue that having several experts is a better and more flexible option.

3.1 Learning local fuzzy datatypes

Motivation

As already stated in the introduction, it is difficult for ontologists to develop fuzzy ontologies. In particular, we will focus in the case of fuzzy datatypes, which are important to represent, for example, the fact that a product is cheap or a product is small.

The definitions of the linguistic labels are particularly subjective and context dependent. For instance, both Carrauntoohill (Ireland) and Fuji (Japan) are usually considered as high mountains by local people, although they have very different elevations (1041 m and 3776 m, respectively). To date, the only existing methods to

build fuzzy datatypes are agnostic to any information about the domain. Essentially, they either compute a uniform partitioning of the domain or assume that an expert provides the definitions (a more detailed discussion of the related work will be provided later). In the era of big data, very often there are a lot of data available for a given domain. Therefore, our aim is to use existing numerical data to learn fuzzy datatypes.

As a first step to solve the problem, we will focus on local fuzzy dataypes, which can be associated to a unique individual. For example, given different (possibly imprecise) measurements of the calories consumed by a human over several days, our objective is to learn automatically a fuzzy set representing the typical value of the calories consumed by that individual on a daily basis.

Contribution

We propose a first solution to learn the values of numerical fuzzy data properties describing them using local fuzzy datatypes. That is, each fuzzy datatype is related to one individual via some numerical data property. In our setting, the value will be described using a fuzzy set, characterized by its fuzzy membership function. In particular, we propose to represent the value using a different triangular membership function $\mathbf{tri}(q_1, q_2, q_3)$ for each individual i. Recall that the triangular function is one of the fuzzy datatypes supported in Fuzzy OWL 2 language.

To obtain the values of the parameters q_1, q_2, and q_3 of a fuzzy datatype d, we assume that we have an array of data with the values of the data property p for each individual i. Then, we compute the mean of the values (denoted x) and the standard deviation (denoted σ) and build the following fuzzy set:

$$\mathbf{tri}(x - \sigma, x, x + \sigma) \tag{3.1}$$

This can be represented in Fuzzy OWL 2 by adding an annotation to a datatype D (via the fuzzyLabel annotation property) of the form:

```
<fuzzyOwl2 fuzzyType="datatype">
   <Datatype type="triangular" a="x-σ" b="x" c="x+σ" />
</fuzzyOwl2>
```

Once the fuzzy ontology has been updated to define a fuzzy datatype d, it is associated to each individual i using an axiom of the form:

$$i : \exists p.d$$

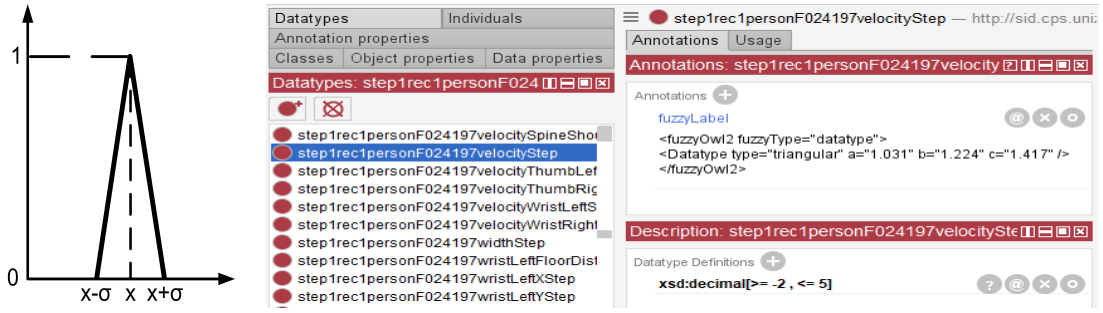
Example 12. *For some person (an individual)* personF024197 *we have different (imprecise) measures of the functional data property* velocityStep*. Assume that the mean of the values is* $x = 1.224$ *and the standard deviation* $\sigma =$

0.193. *Figure 3.1 (a) shows how to build a triangular fuzzy membership function* $\mathsf{step1rec1personF024197velocityStep} = \mathbf{tri}(1.031, 1.224, 1.417)$ *from those values. Figure (b) shows a fuzzy datatype annotation to represent this value of* velocityStep. *Finally, a concept assertion stating that* personF024197 *is an instance of the fuzzy concept* $\exists \mathsf{velocityStep} \,.\, \mathsf{step1rec1personF024197velocityStep}$ *is needed.* □

Note that although these fuzzy datatypes are local in principle, as they are computed from data corresponding to a single individual, there is not any problem if two different individuals share the same definition for the same data property.

Our solution is general and can be used in different domains. In Section 6.1, we will give more details about an implemented gait recognition system that uses sequences of data from walking people obtained using the Microsoft Kinect sensor. In particular, we represent the values of some biometric features of the people using fuzzy datatypes learned as explained in this section.

Before concluding this section, it is important to add a technical remark. Because fuzzy data properties are assumed to be functional, it is not possible to represent directly that an individual is linked to different values of a data property. However, it is possible to use an indirect representation. As we will see in Section 6.1, a person can be linked with several frames, and for each frame the value of the data property can be different.



a) b)

Figure 3.1: Fuzzy datatype representation: (a) Triangular function, (b) Fuzzy OWL 2 annotation.

Related work

Although some methodologies to build fuzzy ontologies have been proposed in the literature, such as DOF [AZ21], IKARUSOnto [AWKA12], FODM [LMR16], or Fuzzy OWL 2 [BS11], they mention the need to fuzzify some elements of an ontology, but do not actually detail how to fuzzify them. In other cases, there are some details but

not enough. Abulaish and Dey discuss how to build fuzzy hedges from a dissimilarity matrix, but they do not explain how to obtain such matrix [AD07]. Furthermore, how to learn fuzzy datatypes is not addressed, and fuzzy axioms are not allowed. Gu et al. pointed out the need to build a fuzzy matrix with the degrees of truth of the relationships between pairs of concepts (for example, fuzzy subclass axioms), but do not detail how to compute the degrees [GLGS07].

In the following, we will overview previous approaches to learn the elements of a fuzzy ontology, except fuzzy datatypes, which will be covered in Section 3.2.

Widyantoro and Yen described the construction of a fuzzy ontology and the application for query refinement in a search engine [WY01]. The fuzzy ontology contains two types of fuzzy relationships between pairs of terms: fuzzy narrower than and fuzzy broader than relations. The degrees of the truth of these relationships are computed from the frequency of co-occurrences of the terms. The authors also describe a mechanism to prune the fuzzy ontology by eliminating redundant, less meaningful, and unrelated term relations. This approach can be used to learn fuzzy object property assertions (if terms are represented as individuals) or fuzzy subclass axioms (if terms are represented as concepts).

The application of Fuzzy Formal Concept Analysis to build fuzzy ontologies has also been studied. Quan et al. proposed FOGA (Fuzzy Ontology Generation frAmework) [QHFC06, QHF06]. Their approach is based on fuzzy clustering and Fuzzy Formal Concept Analysis, and is able to build fuzzy ontologies with fuzzy relations between concepts (mainly, fuzzy subclass axioms) and fuzzy object property assertions (although the authors call it the "attribute value of an object"). It is also worth to note that they use clusters to group similar concepts, while we will use clustering to build fuzzy datatypes in Section 3.2. In a similar approach, Chen et al. built a fuzzy ontology with fuzzy subclass axioms using Fuzzy Formal Concept Analysis and fuzzy clustering to group concepts [CYZW09]. The approach was evaluated in an information retrieval scenario, showing that fuzzy ontologies improve the precision.

It is also possible to build fuzzy ontologies using information from crisp concept networks such as Wordnet or ConceptNet. Angryk et al. studied the automatic creation of fuzzy ontologies including a fuzzy concept hierarchy of terms [ADP06]. The fuzzy subclass axioms were computed using the semantic relations in Wordnet, and the degrees of truth are computed assuming that all the possible senses in WordNet are equiprobable. The authors also proposed an algorithm to find common hypernyms of two concepts and the generalization degree. Furthermore, Jai et al. discussed how to build a fuzzy concept ontology with fuzzy relations between concepts derived from ConceptNet (in particular, with fuzzy subclass axioms and fuzzy equivalence

axioms) [JSJ21]. The degrees of truth are computed using the weight values of the semantic relationships between the entities in ConceptNet. The approach is applied to semantic query enrichment in information retrieval.

Another possibility is to adapt classical learning algorithms. The works [LS13, LS15, SM15, CS21] use the *FuzzyDL-Learner*[1] software to learn fuzzy concept inclusion axioms using adaptations of learning techniques such as Inductive Logic Programming or boosting algorithms. This family of learning algorithms will be mentioned again in the next section, as they involve fuzzy datatypes.

In a series of works, researchers have also proposed to build fuzzy ontologies by exporting fuzzy databases models or fuzzy modeling diagrams. In particular, Ma et al. described a framework to build fuzzy ontologies from fuzzy relational databases [MLY08], while Zhang et al. built fuzzy ontologies from fuzzy entity-relationship models [ZMYC13], from fuzzy object-oriented data bases [ZMYW12], from fuzzy XML models [ZMY13], and from fuzzy UML models [MZYC11]. However, the only fuzzy elements in their approaches are fuzzy concept assertions and fuzzy role assertions.

3.2 Learning global fuzzy datatypes

Motivation

Section 3.1 studied how to learn local datatypes, i.e., fuzzy datatypes that are associated to only one individual. In particular, for each individual and each data property, we learned a unique fuzzy datatype. As an illustrating example, we could compute the value of the data property calories for each individual i (Calories_i) starting from data about the consumed calories of this specific individual over several days.

Another possibility is to learn global fuzzy datatypes, i.e., fuzzy datatypes that can be associated to more than one individual. For example, we can define the linguistic terms LowCalories and NeutralCalories as possible values of calories. Furthermore, we might want to associate and individual with both fuzzy datatypes with different degrees of truth, e.g., i consumes a number of calories which belongs to LowCalories with degree 0.8 and which belongs to NeutralCalories with degree 0.6.

Rather than computing a uniform partitioning of the domain, the idea is to compute the parameters of the fuzzy membership functions using numerical real data from more than one individual (e.g., using measures of the consumed calories of several people)

[1] `http://www.umbertostraccia.it/cs/software/FuzzyDL-Learner/index.html`

Contributions

In this section, we describe a novel approach to learn multiple fuzzy datatypes for a data property with a numerical range. The general idea is the following: for each data property we have a list of numerical values which are used to partition the range of the data property using fuzzy memberships functions. The partition can be computed using different centroid-based clustering algorithms. These algorithms compute a set of centroids that we use to create fuzzy membership functions. Furthermore, we propose to automatically create linguistic labels to identify the new fuzzy datatypes. Figure 3.2 shows a general view of our approach to learn fuzzy datatypes. First, we retrieve the array of values of each data property. Next, values of a property (calories) are clustered to compute some centroids $\{c1, c2, c3\}$, which are used to build the membership functions and their linguistic labels, e.g., left-shoulder function $\mathbf{left}(c1, c2)$ has a linguistic label LowCalories, triangular function $\mathbf{tri}(c1, c2, c3)$ has a linguistic label NeutralCalories, and right-shoulder function $\mathbf{right}(c2, c3)$ has HighCalories label. Example 13 shows how the fuzzy sets are computed from a numerical vector.

Example 13. *Given a vector with information about the consumed calories* $cal = [9, 11, 19, 21, 30]$*, a clustering algorithms computes a set of centroids* $c = \{10, 20, 30\}$*. From this set, we can build three fuzzy membership functions,* $\mathbf{left}(10, 20)$*,* $\mathbf{tri}(10, 20, 30)$ *and* $\mathbf{right}(20, 30)$*.* □

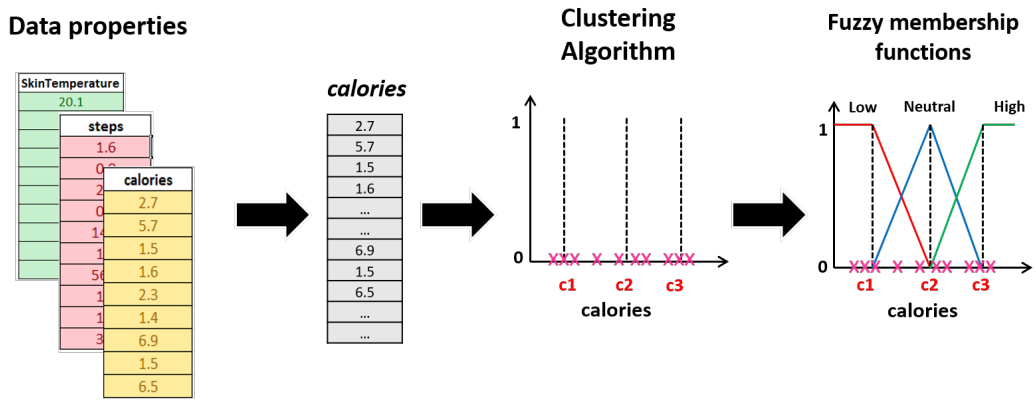


Figure 3.2: General outlook to learn fuzzy datatypes.

Learning algorithm. Algorithm 1 shows how to compute fuzzy datatypes for fuzzy ontologies. The first part (Lines 1–3) is an initialization that can be computed just once. Firstly, we retrieve all data properties (Line 2) and individuals (Line 3) of a fuzzy ontology O. The rest of the code (Lines 5–43) implements the proper learning algorithm. Lines 7–12 compute the array A of values of property p. Only numerical

values are accepted. The standard deviation σ of the array is calculated in Line 14. The fuzzy datatypes are defined over a range $[k_1, k_2]$ rather than over $[-\infty, \infty]$. For each data property p, our algorithm uses several strategies to compute such k_1, k_2 (Lines 15–21):

- First, checking if the range of p is of the form$[>= r1, <= r2]$, where $>=$ and $<=$ denote xsd:minInclusive and xsd:maxInclusive OWL 2 facets, respectively, that constrain the possible values of an OWL 2 numerical datatype.

- Otherwise, it computes the minimum (min) and the maximum (max) of the array of the data property p and defines $k_1 = min - \sigma$ and $k_2 = max + \sigma$.

Line 23 and Line 26 assume that a clustering algorithm provides a set of centroids C from the array of values of p. Some clustering methods do not require an initial number of clusters N but compute it automatically (Line 24). The centroids are used as the parameters to build fuzzy membership functions partitioning the domain (Lines 28–41). Assuming that $N \geq 2$, using a set of centroids $\{c_1, c_2, \ldots, c_N\}$, and an interval $[k_1, k_2]$, we create:

- a left-shoulder function with parameters c_1 and c_2 (Line 29),

- a right-shoulder function with parameters c_{N-1} and c_N (Line 31), and

- $k-2$ triangular functions, where the $j-$th triangular function has parameters c_j, c_{j+1}, and c_{j+2} (Line 34).

If the set of centroids has a unique centroid c_1, it creates one triangular function with parameters $c_1-\sigma$, c_1, $c_1+\sigma$, and the interval is recalculated (Line 39). The reason to use a ϵ to compute the range of k_1 and k_2 when there is a single fuzzy datatype is to ensure that the parameters of the triangular function are within the range of the fuzzy datatype, as we could have $c_1 - \sigma < k_1$ or $c_1 + \sigma > k_2$.

In all previous cases, when a fuzzy datatype d is created, the ontology is updated with new axioms encoding the annotations of the fuzzy datatypes. Finally, the fuzzy ontology is returned.

Some salient features of our algorithm are the following ones:

- It is domain independent and can be employed in different contexts where attributes are represented using numerical values (e.g integer, reals, etc.) and there is a large volume of data.

Algorithm 1 Algorithm to learn fuzzy datatypes for numerical data properties.

Input: A fuzzy ontology $\mathcal{O}$
Input: An optional number of centroids N
Input: A tolerance ϵ
Output: An updated version of $\mathcal{O}$

1: // Initialization
2: $P \leftarrow$ Retrieve all data properties p of $\mathcal{O}$
3: $I \leftarrow$ Retrieve all individuals i of $\mathcal{O}$
4: // Learning fuzzy datatypes
5: **for all** $p \in P$ **do**
6: $A \leftarrow \emptyset$
7: **for all** $i \in I$ **do**
8: $v \leftarrow$ Retrieve the value of the data property p for i in $\mathcal{O}$
9: **if** $v \neq$ NULL **and** v is numerical **then**
10: $A \leftarrow A \cup v$
11: **end if**
12: **end for**
13: **if** $A \neq \emptyset$ **then**
14: $\sigma \leftarrow standardDeviation(A)$
15: **if** $DataPropertyRange(p, [>= r_1, <= r_2]) \in \mathcal{O}$ **then**
16: $k_1 \leftarrow r_1$
17: $k_2 \leftarrow r_2$
18: **else**
19: $k_1 \leftarrow \min(A) - \sigma$
20: $k_2 \leftarrow \max(A) + \sigma$
21: **end if**
22: **if** $N =$ NULL **then**
23: $C \leftarrow clustering(A)$
24: $N \leftarrow size(C)$
25: **else**
26: $C \leftarrow clustering(A, N)$
27: **end if**
28: **if** $N \geq 2$ **then**
29: $d \leftarrow$ **left**(c_1, c_2) over the interval $[k_1, k_2]$
30: $\mathcal{O} \leftarrow \mathcal{O} \cup d$
31: $d \leftarrow$ **right**(c_{N-1}, c_N) over the interval $[k_1, k_2]$
32: $\mathcal{O} \leftarrow \mathcal{O} \cup d$
33: **for** $j \leftarrow 1$ to $N - 2$ **do**
34: $d \leftarrow$ **tri**(c_j, c_{j+1}, c_{j+2}) over the interval $[k_1, k_2]$
35: $\mathcal{O} \leftarrow \mathcal{O} \cup d$
36: **end for**
37: **else**
38: // unique centroid
39: $d \leftarrow$ **tri**$(c_1 - \sigma, c_1, c_1 + \sigma)$ over the interval $[k_1 - \epsilon, k_2 + \epsilon]$
40: $\mathcal{O} \leftarrow \mathcal{O} \cup d$
41: **end if**
42: **end if**
43: **end for**
44: **return** $\mathcal{O}$

– We can also create readable names for the fuzzy datatypes labels. For a small number of clusters ($N \leq 7$), the final name of a fuzzy datatype is the concatenation between a label with a linguistic prefix (e.g, *VeryLow*) and the property name, as shown in Table 3.1. For example, Figure 3.3 shows the partition of the domain in 7 clusters and their linguistic labels. For the property calories the algorithm creates 7 fuzzy datatypes VeryVeryLowCalories, VeryLowCalories, LowCalories, NeutralCalories, HighCalories, VeryHighCalories, and VeryVeryHighCalories. If the number of cluster was 6, NeutralCalories would be omitted. For an arbitrary number of clusters $N > 7$, label names can be formed by concatenating the name of the data property p and an integer number denoting the order of the fuzzy datatype according to an increasing value of the smaller centroid (e.g., calories8).

– It uses a classical reasoner to retrieve the values of the data properties (Line 8).

– It could be customized for diverse input and output file formats.

– It supports any method or clustering algorithm that returns a set of centroids.

– In the case of mean-shift clustering algorithm, we could define $\sigma = h/2$, where h is computed as in Equation 2.28.

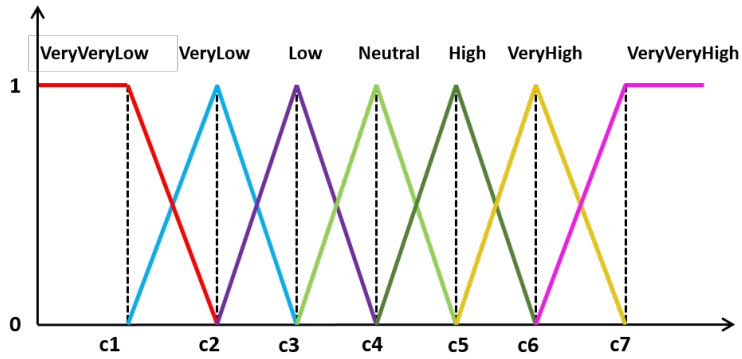


Figure 3.3: Fuzzy membership functions for seven linguistic labels.

Implementation: Datil. Datil (DATatypes with Imprecision Learner) is a software that automatically learns fuzzy datatypes for fuzzy ontologies from different types of inputs. Datil implements Algorithm 1 and several unsupervised clustering algorithms: k-means (Eq. 2.24), fuzzy c-means (Eq. 2.25), and mean-shift (Eq. 2.30). For k-means and fuzzy c-means, the number of clusters N (used to build K fuzzy datatypes) should

Number of labels	Label prefixes
2	{ *Low*, *High* }
3	{ *Low*, *Neutral*, *High* }
4	{ *VeryLow*, *Low*, *High*, *VeryHigh* }
5	{ *VeryLow*, *Low*, *Neutral*, *High*, *VeryHigh* }
6	{ *VeryVeryLow*, *VeryLow*, *Low*, *High*, *VeryHigh*, *VeryVeryHigh* }
7	{ *VeryVeryLow*, *VeryLow*, *Low*, *Neutral*, *High*, *VeryHigh*, *VeryVeryHigh* }

Table 3.1: Label prefixes for different number of labels.

be chosen by the end user. If we do not know the number of clusters, mean-shift is the right choice, as it does not require the number of clusters in advance. The tool is publicly available[2].

Input formats. Datil supports three possible input file formats: OWL, FDL, and CSV.

- OWL format (.owl) corresponds to ontologies in the standard language OWL 2. Files can be classical ontologies but also fuzzy ontologies in Fuzzy OWL 2; in the latter case previous annotations with the fuzzy information are discarded. Datil restricts itself to data property assertions and range restrictions. A semantic reasoner is used to retrieve both explicit and implicit axioms.

- FDL (.fdl) is the native syntax of fuzzyDL reasoner to define fuzzy ontologies [BS16a]. As in the previous case, Datil restricts itself to data property assertions and range restrictions and does not consider any fuzzy information (not even the degree of truth of the assertions).

- CSV (Comma-Separated Values, .csv) format consists of large data (numbers and text) in plain text. Each record (row) in the file contains one or more fields (columns) separated by commas. In this case, the clustering algorithm takes as an input all the values for a given column.

  To retrieve the data properties (Line 2), we assume that the first line of a CSV file is special and contains the column names, which correspond to the data properties in the ontology. Furthermore, rather than retrieving all individuals (Line 3), we chose to represent the data of each individual in a separate CSV file.

Output format. Datil supports two possible output file formats: OWL, and FDL. The output is a fuzzy ontology with some fuzzy datatype definitions that can be represented as OWL 2 annotations (as specified in Fuzzy OWL 2) or as fuzzyDL

[2] `http://webdiis.unizar.es/~ihvdis/Datil`

axioms (FDL). If the output is a FDL file, apart from the definition of the fuzzy datatypes, Datil adds further axioms required by fuzzyDL reasoner (functional and range data property axioms). If the input is an ontology (OWL or FDL), the output is an extension with the new elements. If the input is a CSV file, the output ontology is created from scratch.

Example 14. *Let us consider Figure 3.7. An excerpt of the output in FDL syntax is:*

```
(functional SkinTemperatureToWork)
(range SkinTemperatureToWork *real* 14 , 37)
% DataProperty: SkinTemperature SegmentType: ToWork
% Learned using k-means
% Centroids results: [15.19][22.02][34]
(define-fuzzy-concept LowSkinTemperatureToWork left-shoulder(14,37,15.19,22.02))
(define-fuzzy-concept NeutralSkinTemperatureToWork triangular(14,37,15.19,22.02,34))
(define-fuzzy-concept HighSkinTemperatureToWork right-shoulder(14,37,22.02,34))
```

An excerpt of the output (the definition of LowSkinTemperatureToWork*) in Fuzzy OWL 2 Manchester syntax is:*

```
Datatype: LowSkinTemperatureToWork
    Annotations:
      fuzzyLabel "<fuzzyOwl2 fuzzyType="datatype">
        <Datatype type="leftshoulder" a="15.19" b="22.02" />
         </fuzzyOwl2>"
    EquivalentTo:
     (xsd:decimal[>= "14"^^xsd:decimal] and xsd:decimal[<= "37"^^xsd:decimal])
```

□

Dependencies. Datil is implemented in Java and uses some external libraries:

- *OWL API* [HB11] is an ontology API to manage OWL 2 ontologies in Java applications and provides a common interface to interact with DL reasoners.
- *HermiT* is an OWL 2 ontology reasoner [GHM+14]. We use it to retrieve all the data property assertions, not only those explicitly represented in the ontology but also the implicit ones.
- *Java-ML* (Java Machine Learning Library)[3] is a collection of machine learning algorithms and a common Java interface for those algorithms. Although Java-ML provides an implementation of k-means, we have implemented our own version of the algorithm. However, we do use its Java data structures in all of our clustering algorithms.
- *fuzzyDL* is a fuzzy ontology reasoner [BS16a]. The possible input formats are Fuzzy OWL 2, its own syntax in FDL format, and a Java API. We use fuzzyDL to translate FDL fuzzy ontologies into Fuzzy OWL 2.

Configuration options. Datil requires several parameters:

[3] `http://java-ml.sourceforge.net`

– The input and output formats.

– The input file. The output file is not a parameter; Datil uses the same filename (with a different filename extension if there is a format change).

– The selected clustering algorithm.

– The properties for which to learn the fuzzy datatypes.

– The number of clusters (only for k-means and fuzzy c-means) for all the properties, or a different number for each property.

– Use of zeros (only for CSV files): zero values can be either taken into account or skipped (in practice, they are often used just to represent empty data).

– Use of segments (only for CSV files). Segments are special properties that make it possible to split the data. This is useful in applications where we have a lot of data but we do not want to consider all of them in a joint way. For example, we might have information about the time a person is walking, but we might want to differentiate whether s/he is walking to work or walking during the work. This way, we can learn fuzzy datatypes for the group of values of a data property that correspond to a specific segment value.

Example 15. *In Figure 3.4, we can restrict the values of the data property* calories *to the segment* atWork*, building a vector of data about calories consumed while working (*caloriesAtWork*) from which Datil can learn some fuzzy datatypes.* □

	A	B	C	D	E	F	G
1	segmentType	startTime	endTime	skinTemperature	galvanicSkin	heartRate	calories
2	wholeDay	-30	1440	15.61292517	2.68E-05	39.27619	1.5756215
3	morning	401	512	9.790178571	1.73E-05	29.223214	1.326982
4	toWork	512	597	29.88953488	5.20E-05	72.77907	1.9551664
5	atWork	597	1128	30.50093985	5.12E-05	63.740602	1.2718442
6	toHome	1128	1199	30.03472222	9.41E-05	79.361111	2.2555533
7	eveningAfterWork	1199	1438	28.58333333	9.31E-05	63.508333	1.1805533
8	morning	401	529	23.94573643	4.03E-05	59.55814	1.6450882
9	toWork	529	797	29.65427509	5.15E-05	66.624535	1.3543761
10	atWork	797	797	28	4.77E-05	63	1.17222
11	toHome	797	1095	30.0819398	5.14E-05	64.050167	1.17222
12	eveningAfterWork	1095	1438	29.06395349	0.0001002	65.247093	1.3960573

Figure 3.4: Example of an input CSV file.

User interface. Now we will describe the user interface for desktop computer; an interface for mobile devices will be described in Section 5.4.1.

Figure 3.5 shows a snapshot of the main user interface, where the user can configure most of the previously mentioned parameters: input and output formats, input file, use of zeros and segments, clustering algorithm, and global number of clusters. In case of CSV files, the user can select a folder with several files rather than a single one. By default, Datil learns fuzzy datatypes for all data properties with a numerical range.

It is also possible to use a *configuration file* to select a subset of the data properties and/or select a different number of clusters for each of them if the clustering algorithm is not mean-shift. Figure 3.6 shows how Datil supports the creation of the file by making its syntax transparent to the user. Thanks to the configuration file the user does not need to repeat the selection in future executions. If the system does not find it, it runs with the default values.

The user interface uses some strategies to obtain automatically the fuzzy datatype range $[k_1, k_2]$ for each data property of the input file. This automatic values can be customized by the user; for example, if there are outliers, or to reduce/amplify the range that will be considered to compute the fuzzy datatypes. To do so, there are two columns in the interface to specify the k_1 (Min) and k_2 (Max) values for each data property. For example, in Figure 3.6, the range of SkinTemperatureToWork could be modified to Min=0 and Max=42.

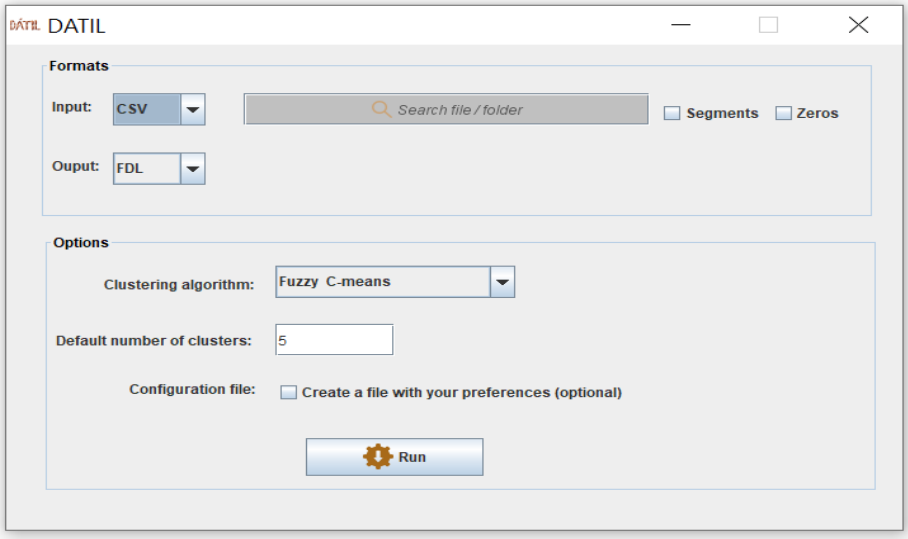


Figure 3.5: Snapshot of the main user interface of Datil.

In general, the main features of Datil software are the following ones:

– It makes an automatic partition domain using unsupervised clustering algorithms.

– It is domain independent.

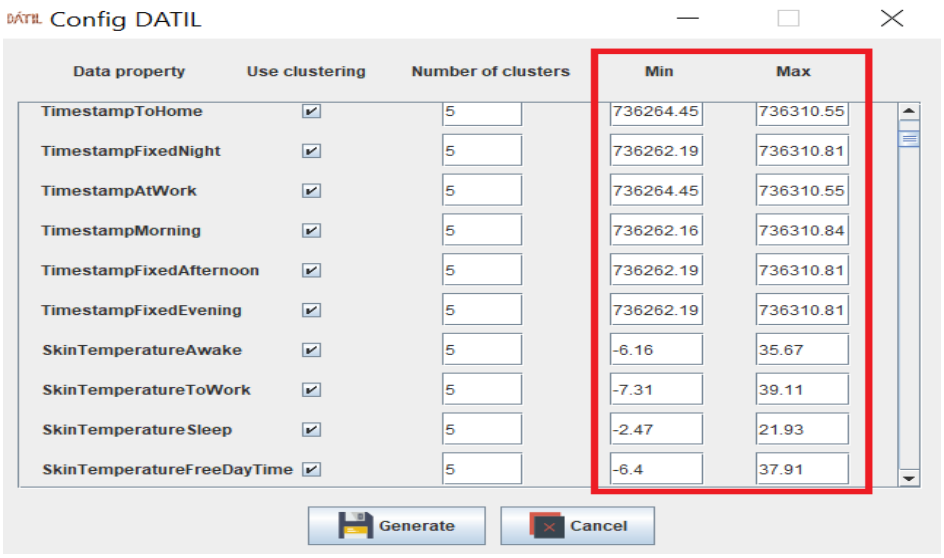


Figure 3.6: GUI to create a configuration file in Datil.

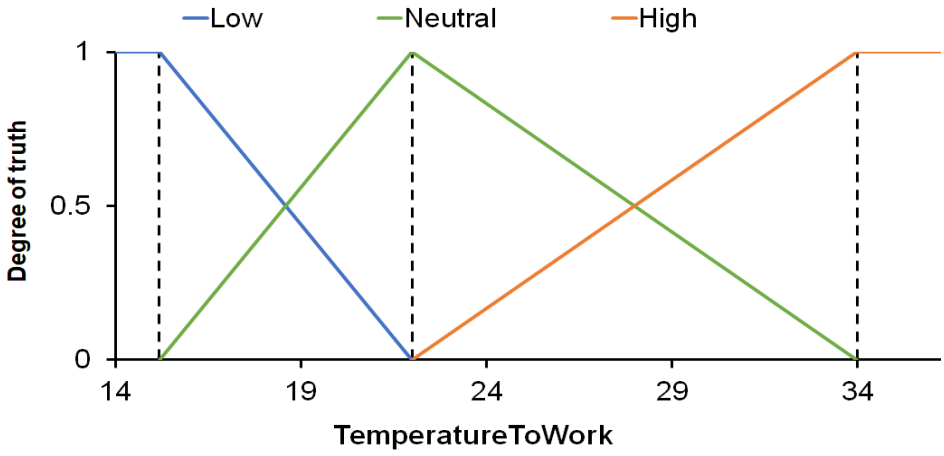


Figure 3.7: Learned fuzzy datatypes for the temperatureToWork data property.

– It not interfere in data acquisition and data prepossessing stages as shown in Section 6.6.2.

– It enriches the fuzzy ontologies with the fuzzy datatypes computed. In special cases, additional information needed by fuzzyDL is included (functional and range axioms data property axioms).

– It offers readable and interpretable way of fuzzy datatypes for humans and machines. For example, for an athlete it is easier to read and understand the label HighCalories than a simple amount of them.

– It uses reasoning (using HermiT reasoner) to get all the data property assertions explicit and implicit represented[4].

– Although 5 ± 2 is widely accepted as the optimal number of linguistic labels to make them more easily understandable by a person, Datil supports $n > 7$ number of labels.

– It implements a segmentation option for CSV format. For example, Figure 3.7 illustrates the fuzzy memberships and the linguistic labels computed using *k-means* algorithm, number of clusters $k = 3$ for skin temperature property values related to the segment *toWork*.

– It has an intuitive Graphic User Interface[5] and a mobile version for Android devices (described in Section 5.4.1).

– It supports different input formats (CSV, OWL 2, and Fuzzy OWL 2) and output formats (Fuzzy OWL 2 and FDL).

As we will see later, we have successfully evaluated Datil on some real use cases, such as lifestyle profiling (Section 6.6.2), gait recognition (Section 6.6.3), and beer recommendation (Section 6.2). Furthermore, Datil has also been independently used by Riali et al. to build the fuzzy datatypes in a medical decision support system that uses a fuzzy ontology and a fuzzy Bayesian network to improve the diagnosis of hepatitis C. diagnosis [RFIB].

Related work

There are many applications using fuzzy ontologies with fuzzy datatypes which, rather than explaining how to actually learn the fuzzy datatypes, assume that

[4]We use the method getDataPropertyValues.

[5]The first version of the tool, `https://github.com/NataliaDiaz/Ontologies/tree/master/Lifestyles-KG`, had a command-line interface

an expert defines them. We can find the examples in different fields such as recommender systems [CBM12], computational perception [MCvdHST12], ambient intelligence [DRPCLD14], diet recommendation [LWH10], matchmaking [RSB+08, RSS10], summarization [LJH05], robotics [EHK+14], aerospace industry [Rod13], diabetes diagnosis [ESAA+18], Alzheimer diagnosis [SRES+21], breast cancer diagnosis [OEA21], Internet of Things-based healthcare monitoring [AIK+18], automatic hotel reservation [AKK15], web content classification [AKR+17], interoperability of electronic health records [AESB+21], air quality assessment [GZ22], software design [DMS15], architectural design [NMNS19], or construction [GRBR+15].

In the following, we will focus on the very few works that actually involve other approaches to learn fuzzy datatypes.

As already mentioned in Section 3.1, a series of works have presented different algorithms to learn fuzzy general concept inclusion axioms by adapting classical learning techniques to the fuzzy case [LS13, LS15, SM15, CS21]. These algorithms are implemented in FuzzyDL-Learner reasoner. Because such axioms might include fuzzy datatypes, they must also be learned. In particular, the most recent work [CS21] learns the fuzzy datatypes using k-means with three possible numbers of clusters (3, 5, or 7). Our approach instead is more general, as it supports three different clustering algorithms and any number of clusters (at least 2). As we will see in Section 6.6.2, it is possible to adapt FuzzyDL-Learner to use fuzzy datatypes learned using our approach.

Lisi and Mencar have proposed a granular computing approach to compute granular views over individuals of a classical ontology [LM18]. Essentially, the idea is to compute fuzzy concepts of the form $C \sqcap \exists T.d$ representing the instances of a classical concept C that have a value of the data property T that is compatible with the fuzzy datatype d. Then, information granules can be quantified by evaluating some quantified sentences involving fuzzy quantifiers. The approach is implemented in the GranulO system. However, GranulO does not use an ontology reasoner but SPARQL queries, thus having limited inference capabilities. To learn the fuzzy datatypes, the authors suggest the use of fuzzy c-Means algorithm, although the implemented algorithm in GranulO is not clear. While the objective of the authors is to compute views to summarize a classical ontology, because several individuals could be replaced by information granules, our aim is to enrich a fuzzy ontology by adding some datatypes. Furthermore, our learning strategy is not restricted to the instance of a given concept, but considers all values of the data properties. Finally, our approach is more general as it supports more clustering algorithms, more input formats, more customization parameters, and has a mobile version.

El-Sappagh et al. proposed to build fuzzy datatypes from numerical data

by combining a clustering algorithm (k-means) and hierarchical fuzzy partitioning (HFP) [ESAA+18]. The partition is similar to ours, with fuzzy sets being represented using triangular and semi-trapezoidal (i.e., left-shoulder and right-shoulder) functions. In the evaluation of this work, the authors showed that this partitioning method outperforms uniform partitions (which should thus be used only when no training data are available) according to three objective measures, namely the partition coefficient (PC), the partition entropy (PE), and the Chen index (CI). In contrast, our work offers more than one clustering method, does not need to fix the number of clusters, addresses the naming of the linguistic labels, supports more input formats, proposes a learning algorithm that takes into account an input fuzzy ontology, discusses the format of the output fuzzy ontology, and provides an implementation with a mobile version.

3.3 Learning consensual fuzzy datatypes

Motivation

In the previous sections of this chapter, we have seen how to build fuzzy datatypes from real data. However, in some cases there are no real data to learn or they are not useful enough (there are not enough samples, data are noisy, etc.). In some cases, the typical solution is to require an human expert to provide the definitions of the fuzzy datatypes. Nevertheless, this also has some drawbacks, as the definitions might be biased to the opinion of the expert.

For example, most people across the world have different perceptions of what *strong coffee* means[6]. For example, Brazilians expect a coffee heavily roasted (dark roast is a synonym of strong). In Italy, many people associate a strong coffee with long notes of bitterness (a huge impact in the mouth), a low acidity, and reduced caffeine content. Citizens of Indonesia, however, consider a strong coffee to be dark, hot, bitter, and with an intense caffeine content. Therefore, providing a definition of strong coffee is not easy. Some coffee companies such as Nespresso[7] or Lavazza[8] have defined coffee intensity scales with discrete levels. However, the problem is the same, there is not a common intensity scale, e.g., Nespresso's rating 8 is different from Lavazza's rating 8. To reach a global understanding of the meaning of strong coffee between drinkers, baristas, café owners, and roasters, we propose not to use a single expert but several ones. That is, to learn a fuzzy datatype, the idea is to fuse the definitions provided by

[6] `http://perfectdailygrind.com/2020/08/strong-coffee-definitions-from-around-the-world`

[7] `http://www.nespresso.com/es/en/variedades-capsulas-cafe#!/by-intensity`

[8] `http://www.lavazza.com/en/magazine/coffee-culture/the-coffee-book/i-for-intensity.html`

different experts into a unique consensual definition.

Contributions

In this section we propose a novel approach to learn fuzzy datatypes for Fuzzy OWL 2 ontologies by using linguistic aggregation operators that merge a group of fuzzy datatype definitions provided by different experts into a unique fuzzy datatype.

We will start by providing an algorithm to build consensual fuzzy datatypes based on linguistic aggregation operators. Then, we will propose two novel aggregation operators and compare them with some existing ones. Finally, we will describe an implementation: *Fudge* software.

Algorithm to build fuzzy datatypes. At first, let us define formally the problem we will address. We assume that there is a group of experts $E_1, E_2, \ldots, E_N$ providing the definitions of the membership functions $F_1, F_2, \ldots, F_N$ characterizing several fuzzy datatypes of a fuzzy ontology. d_{ij} denotes the definition of the datatype F_i according to expert E_j. d_{ij} is assumed to be a linguistic value. Furthermore, there could be missing data, i.e., expert E_j might not provide his/her definition of some datatype F_i. Therefore, for each datatype F_i we have a number of definitions denoted K, with $K \leq N$. Our objective is to define each F_i as a consensus of the definitions $\langle d_{i1}, d_{i2}, \ldots, d_{iK} \rangle$. We will sometimes omit the subscript i when the particular F_i is not important.

We assume that all the definitions are given using trapezoidal functions of the form $\mathbf{trap}(q_1, q_2, q_3, q_4)$, as they are those supported by Fuzzy OWL 2. Note that triangular (denoted **tri**), right-shoulder (denoted **right**), and left-shoulder (denoted **left**) functions can be represented as trapezoidal fuzzy functions, provided that right-shoulder and left-shoulder functions are defined over a fixed range $[k_1, k_2]$ (see Section 2.1.1).

Now, for each fuzzy datatype F_i, we compute $@(\mathbf{W}, [d_{i1}, d_{i2}, \ldots, d_{iK}])$ as a consensual definition, for some aggregation operator $@$ taking as input a vector of numerical weights $\mathbf{W}$ and a vector of trapezoidal functions, returning as output a trapezoidal function, and satisfying internality. Possible choices for the aggregation operator include CONV (Eqs. 2.12 and 2.13), LOWA (Eq. 2.14), WMEAN (Eq. 2.15), and FOWA (Eq. 2.16).

An advantage of such a consensus process is that the individual opinions of the ontology builders are only used to build the final consensual values, thus respecting the privacy of the experts by hiding their individual opinions.

Example 16. *For the sake of illustrative purposes, let us consider the problem of paper*

reviewing. We assume that we want to build a general fuzzy ontology with the relevant definitions (e.g., a hierarchy of publication types, the steps of the reviewing process, the different roles that take part in the process, etc.) to reuse it in other applications, to enable interoperability, or to detect inconsistencies automatically.

We assume that there are 5 *possible decisions for a submission (*Reject*,* WeakReject*,* Borderline*,* WeakAccept*,* Accept*) and that we need to define them by aggregating the definitions given by* 4 *experts*[9]*. For example, these decisions could correspond to reject, reject and encourage re-submission, major revision, minor revision, and accept as it is, respectively. Table 3.2 shows the definitions of the fuzzy datatypes given by each expert.*

Table 3.3 shows instead the consensual aggregation for several aggregation functions. For CONVEX RRF and WMEAN, we used a vector of weights $\boldsymbol{W} = [0.2, 0.25, 0.25, 0.3]$ *taking into account the experience (years in academia) of the experts, while for LOWA–RRF and FOWA we used a fuzzy quantifier* $\mathbf{right}(0.3, 0.8)$*, leading to a vector of weights* $\boldsymbol{W} = [0, 0.4, 0.5, 0.1]$.

The resulting datatypes for each aggregation strategy are illustrated in Figure 3.8. It is worth to note that all methods result in a left-shoulder and a right-shoulder function, but there are differences for the rest of functions:

- *CONV–RRF results in* 3 *triangular and* 0 *trapezoidal functions,*
- *LOWA results in* 2 *triangular and* 1 *trapezoidal functions, and*
- *both WMEAN and FOWA result in* 3 *trapezoidal and* 0 *triangular functions.*

Note also that CONV–RRF and LOWA–RRF result in 4 *datatypes with the definitions given by Expert 3; CONV–RRF and LOWA–RRF coincide in* 3 *definitions out of* 5.

Now, let us discuss how to evaluate a given submission. We assume that each submission has a numerical score from 0 *to* 10 *that combines the evaluation of several criteria (e.g., originality, technical soundness, significance, presentation, and relevance) given by different reviewers*[10]*. For a paper* p_0 *with a score of* 7.5*, the membership degrees to each category are shown in Table 3.4.*

Therefore, the optimal decision depends on the aggregation strategy: for WMEAN, it is WeakAccept*, while for the other ones the optimal choice is* Accept*.* □

[9]The definitions were actually provided by the coauthors of [HBGRS20].

[10]This involves the aggregation of numerical values, which is a well-known problem and out of our scope.

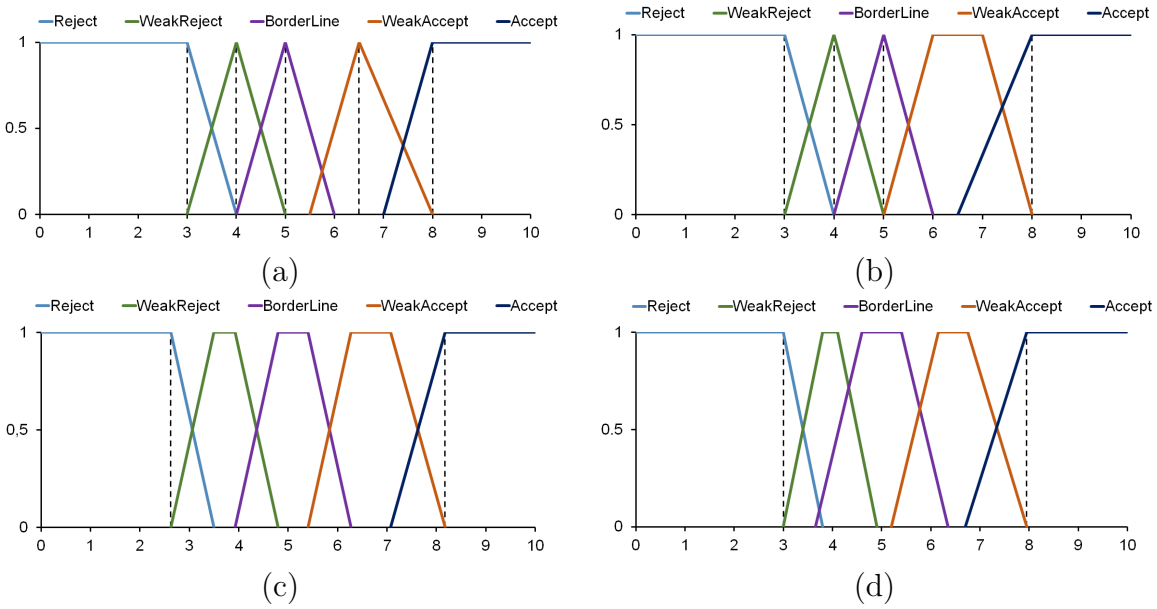


Figure 3.8: Consensual datatypes using (a) CONV; (b) LOWA; (c) WMEAN; and (d) FOWA.

Alternative	Expert 1	Expert 2	Expert 3	Expert 4
Reject	**left**$(3.5, 4.5)$	**left**$(3.5, 4)$	**left**$(3, 4)$	**left**$(1, 2)$
WeakReject	**tri**$(3.5, 4.5, 5.5)$	**trap**$(3.5, 4, 4.5, 5)$	**tri**$(3, 4, 5)$	**trap**$(1, 2, 3, 4)$
Borderline	**tri**$(4.5, 5.5, 6.5)$	**tri**$(4.5, 5, 5.5)$	**tri**$(4, 5, 6)$	**trap**$(3, 4, 6, 7)$
WeakAccept	**tri**$(5.5, 6.5, 8)$	**trap**$(5, 5.5, 6.5, 7.5)$	**trap**$(5, 6, 7, 8)$	**trap**$(6, 7, 8, 9)$
Accept	**right**$(6.5, 8)$	**right**$(6.5, 7.5)$	**right**$(7, 8)$	**right**$(8, 9)$

Table 3.2: Individual definitions of the decisions given by the experts.

Alternative	CONV–RRF	LOWA–RRF	WMEAN	FOWA
Reject	**left**$(3, 4)$	**left**$(3, 4)$	**left**$(2.63, 3.5)$	**left**$(3, 3.8)$
WeakReject	**tri**$(3, 4, 5)$	**tri**$(3, 4, 5)$	**trap**$(2.63, 3.5, 3.93, 4.8)$	**trap**$(3, 3.8, 4.1, 4.9)$
Borderline	**tri**$(4, 5, 6)$	**tri**$(4, 5, 6)$	**trap**$(3.93, 4.8, 5.4, 6.28)$	**trap**$(3.65, 4.6, 5.4, 6.35)$
WeakAccept	**tri**$(5.5, 6.5, 8)$	**trap**$(5, 6, 7, 8)$	**trap**$(5.4, 6.28, 7.075, 8.18)$	**trap**$(5.2, 6.15, 6.75, 7.95)$
Accept	**right**$(7, 8)$	**right**$(6.5, 8)$	**right**$(7.08, 8.175)$	**right**$(6.7, 7.95)$

Table 3.3: Consensual definitions of the decisions for 4 aggregation strategies.

Alternative	CONV–RRF	LOWA–RRF	WMEAN	FOWA
Reject	0	0	0	0
WeakReject	0	0	0	0
Borderline	0	0	0	0
WeakAccept	0.33	0.5	**0.61**	0.38
Accept	**0.5**	**0.67**	0.39	**0.64**

Table 3.4: Degree of satisfaction of each decision for a paper with score of 7.5.

The learning algorithm. Algorithm 2 computes the aggregation of fuzzy datatypes located in several fuzzy ontologies, each of them developed by a different expert. Our algorithm has two inputs: a group of files (fuzzy ontologies) SO and an array of weights.

Firstly, we build an output fuzzy ontology as a union of the input fuzzy ontologies excluding the annotation assertion axioms that provide the individual definitions of the fuzzy datatypes (Lines 2–9). Typically, all input fuzzy ontologies will share a common schema and only the definitions of the fuzzy datatypes will be different. If this is the case, the for loop in Line 2 could be restricted to just one of the input fuzzy ontologies.

Secondly, we create an associative array where the keys are fuzzy datatype names and the values are lists of the definitions given by the experts (Line 10). Next, each fuzzy datatype definition in each fuzzy ontology is represented using a trapezoidal datatype and then added to the associative array (Lines 11–20).

Thirdly, for each fuzzy datatype, a consensual definition is built using some aggregation operator and the result is added to the output fuzzy ontology (Lines 21–24). Finally, the algorithm returns the output fuzzy ontology.

Note that *aggregate* function can be implemented using diverse aggregation strategies: any linguistic aggregation operator aggregating trapezoidal membership functions given a vector of weights can be used. Table 3.5 contains six type of linguistic aggregation operators that can be integrated in our approach.

Computing the vector of weights. In LOWA or FOWA, we propose two different strategies to obtain the vector of weights. Namely:

- Quantifier-guided aggregation, using Eq. 2.6 as in standard OWA [Yag96]. In this case, we propose to use right-shoulder (Figure 2.3 (d)), power (Figure 2.6 (a)) and linear (Figure 2.6 (b)) functions as RIM quantifiers.

- A recursive procedure to compute a vector of weights with a given orness, using either Eq. 2.8 (to combine the lowest value and the aggregation of the other ones) or Eq. 2.11 (to combine the highest value and the aggregation of the other ones) [TY05].

Dealing with incomplete data. It could be the case that some of the experts do not provide his/her definition of some datatype. In this case, unavailable opinions are not taken into account during the aggregation, so a new vector of weights $\mathbf{W}$ is computed. Specifically, in CONV–RRF and WMEAN, we can normalize each weight dividing by the sum of the weights of the available experts. In LOWA–RRF or FOWA,

Algorithm 2 Algorithm to learn fuzzy datatypes using aggregation of definitions.

Input: A set of fuzzy ontologies SO
Input: An array of weights $\mathbf{W}$
Output: A fuzzy ontology $\mathcal{O}$

1: // Add background axioms
2: $\mathcal{O} \leftarrow \emptyset$
3: **for all** $o \in SO$ **do**
4: **for all** axiom a in o **do**
5: **if** a is not a fuzzy datatype definition **then**
6: $\mathcal{O} \leftarrow \mathcal{O} \cup a$
7: **end if**
8: **end for**
9: **end for**
 // Retrieve fuzzy datatypes
10: $listDefs \leftarrow$ new associative array
11: **for all** $o \in SO$ **do**
12: **for all** fuzzy datatype fd in o **do**
13: $trapFD \leftarrow trapezoidal(fd)$
14: **if** $listDefs[fd] = \emptyset$ **then**
15: $listDefs[fd] \leftarrow trapFD$
16: **else**
17: $listDefs[fd] \leftarrow listDefs[fd] \cup trapFD$
18: **end if**
19: **end for**
20: **end for**
 // Build consensual fuzzy datatypes
21: **for all** key fd of $listDefs$ **do**
22: $newFD \leftarrow aggregate(listDefs[fd], \mathbf{W})$
23: $\mathcal{O} \leftarrow \mathcal{O} \cup newFD$
24: **end for**
25: **return** $\mathcal{O}$

we can use the previously described strategies (quantifier-guided aggregation or a recursive procedure starting from the orness) to get a vector of smaller size.

Example 17. *Assume that there are* 4 *experts* E_1, E_2, E_3, *and* E_4, *but* E_3 *does not provide a definition for some datatype. To aggregate the other definitions using CONV–RRF or WMEAN, the initial vector of weights* $[w_1, w_2, w_3, w_4]$ *can be updated as*

$$[\frac{w_1}{w_1 + w_2 + w_4}, \frac{w_2}{w_1 + w_2 + w_4}, \frac{w_4}{w_1 + w_2 + w_4}] \ .$$

To aggregate the available definitions using LOWA-RRF or FOWA, let us firstly assume that the vector of weights was computed using a quantifier $Q = \mathbf{right}(0.3, 0.8)$. *Then, the initial vector* $[0, 0.4, 0.5, 0.1]$ *is replaced with* $[0.067, 0.667, 0.267]$.

Now let us assume that the vector of weights was computed from a desired orness 0.6 *using a left recursive procedure. Then, the initial vector* $[0.368, 0.245, 0.205, 0.182]$ *is replaced with* $[0.45, 0.3, 0.25]$. □

Left Recursive Form of CONV and LOWA. Now, we will propose some new linguistic aggregation operators. Inspired by the rewriting of classical OWA in two recursive forms LRF and RRF (see Eqs. 2.7 and 2.10), we may view the standard definition of CONV (Eqs. 2.12–2.13) as a right recursive form. From that, we propose a left recursive form (CONV–LRF).

Definition 1. *The* Left Recursive Form of the convex combination *(CONV–LRF) of* $K \geq 2$ *linguistic labels given a weighting vector* $[w_1, \ldots, w_K]$ *is defined as follows:*

- *if* $K = 2$, *then*

$$\mathbf{CONV^{LRF}}([w_1, \ldots, w_K], [d_1 \ldots, d_K]) \ = \ \mathbf{CONV^{RRF}}([w_1, \ldots, w_K], [d_1 \ldots, d_K])$$

- *if* $K > 2$, *then:*

$$\mathbf{CONV^{LRF}}([w_1, \ldots, w_K], [d_1 \ldots, d_K]) = \\ \mathbf{CONV^{LRF}}\Big([1 - w_K, w_K], \Big[\mathbf{CONV^{LRF}}([\beta_1, \ldots, \beta_{k-1}], [d_1, \ldots, d_{K-1}]), d_K\Big]\Big) \ , \tag{3.2}$$

where $\beta_h = w_h \ / \ \sum_{j=1}^{K-1} w_j, h \in \{1, \ldots, K-1\}$.

Example 18. 4 *experts provide definitions* [*VeryHigh*, *High*, *Low*, *VeryLow*] *and there is a weighting vector* $[0.45, 0.05, 0.1, 0.4]$. *Let us firstly aggregate using CONV–RRF:*
$\mathbf{CONV^{RRF}}([0.45, 0.05, 0.1, 0.4], [\textit{VeryHigh}, \textit{High}, \textit{Low}, \textit{VeryLow}]) =$
$\mathbf{CONV^{RRF}}\Big(\Big[0.45, 0.55\Big], \Big[\textit{VeryHigh},$
$\mathbf{CONV^{RRF}}([0.09, 0.91], [\textit{High}, \mathbf{CONV^{RRF}}([0.2, 0.8], [\textit{Low}, \textit{VeryLow}])\Big]\Big) =$

$$\mathbf{CONV}^{\mathbf{RRF}}\Big(\Big[0.45, 0.55\Big], \Big[\mathit{VeryHigh}, \mathbf{CONV}^{\mathbf{RRF}}([0.09, 0.91], [\mathit{High}, \mathit{VeryLow}])\Big]\Big) =$$
$$\mathbf{CONV}^{\mathbf{RRF}}\Big(\Big[0.45, 0.55\Big], \Big[\mathit{VeryHigh}, \mathit{VeryLow}\Big]\Big) = \mathit{Low}$$

Now, let us compute CONV–LRF, obtaining a different result:

$$\mathbf{CONV}^{\mathbf{LRF}}([0.45, 0.05, 0.1, 0.4], [\mathit{VeryHigh}, \mathit{High}, \mathit{Low}, \mathit{VeryLow}]) =$$
$$\mathbf{CONV}^{\mathbf{LRF}}\Big(\Big[0.6, 0.4\Big], \Big[\mathbf{CONV}^{\mathbf{LRF}}([0.83, 0.17],$$
$$[\mathbf{CONV}^{\mathbf{LRF}}([0.9, 0.1], [\mathit{VeryHigh}, \mathit{High}]), \mathit{Low}]), \mathit{VeryLow}\Big]\Big) =$$
$$\mathbf{CONV}^{\mathbf{LRF}}\Big(\Big[0.6, 0.4\Big], \Big[\mathbf{CONV}^{\mathbf{LRF}}([0.83, 0.17], [\mathit{VeryHigh}, \mathit{Low}]), \mathit{VeryLow}\Big]\Big) =$$
$$\mathbf{CONV}^{\mathbf{LRF}}\Big(\Big[0.6, 0.4\Big], \Big[\mathit{VeryHigh}, \mathit{VeryLow}\Big]\Big) = \mathit{High}$$

□

In Section 6.7, we perform an empirical evaluation of the CONV–RRF and CONV–LRF, showing the differences between both aggregation operators.

CONV–LRF can also be used to define a new version of the linguistic OWA based on the left recursive form of the convex combination.

Definition 2. *The* Left Recursive Form of the linguistic LOWA *(LOWA–LRF) of $K \geq 2$ linguistic labels given a weighting vector $[w_1, \ldots, w_K]$ is defined as follows:*

$$\begin{aligned} &\mathbf{LOWA}^{\mathbf{LRF}}([w_1, \ldots, w_K], [d_1 \ldots, d_K]) = \\ &\mathbf{CONV}^{\mathbf{LRF}}([w_1, \ldots, w_K], [d_{\sigma(1)}, \ldots, d_{\sigma(K)}]) \ , \end{aligned} \tag{3.3}$$

where σ is a permutation such that $d_{\sigma(1)} \geq d_{\sigma(2)} \geq \cdots \geq d_{\sigma(K)}$.

Example 19. *Let us revisit Example 16 considering CONV–LRF and LOWA–LRF. Compared to their right recursive forms, it turns out that LOWA–LRF produces exactly the same output and that CONV–LRF only differs in the consensuated definition of* Borderline*, which is now given by* $\mathbf{trap}(3, 4, 6, 7)$*, i.e., the definition given by Expert 4. The resulting datatypes for CONV–LRF are illustrated in Figure 3.9.*

Using CONV–LRF, the membership degree of paper p_0 to Borderline *is also* 0*, as it happens using CONV–RRF. Both CONV–LRF and LOWA–LRF return* 2 *triangular and* 1 *trapezoidal functions (CONV–RRF returns* 3 *triangular functions), CONV–LRF includes* 3 *datatypes defined by Expert 3 (CONV–RRF includes* 4*); and CONV–LRF and LOWA–LRF coincide in* 2 *definitions (the right recursive forms coincide in* 3*).* □

Some properties of the linguistic aggregation operators. The first thing to observe is that CONV–RRF and CONV–LRF (and hence LOWA–RRF and LOWA–LRF) do not care about the concrete definitions (e.g., if a trapezoidal function has some value of q_1 or another); only the relative ordering matters.

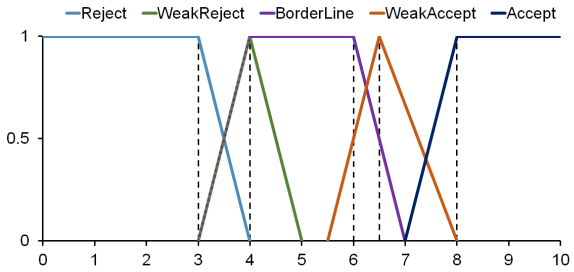


Figure 3.9: Consensual datatypes using CONV–LRF.

It is also important to stress that CONV–LRF and CONV-RRF are not commutative, as Example 20 shows.

Example 20. *Assume that* 3 *experts provide definitions* $[\mathsf{d}_1, \mathsf{d}_1, \mathsf{d}_4]$ *and we aggregate the values using CONV–LRF and a weighting vector* $[0.1, 0.1, 0.8]$*:*

$$\mathbf{CONV^{LRF}}([0.1, 0.1, 0.8], [\mathsf{d}_1, \mathsf{d}_1, \mathsf{d}_4]) =$$
$$\mathbf{CONV^{LRF}}\Big(\Big[0.2, 0.8\Big], \Big[\mathsf{d}_4,$$
$$\quad \mathbf{CONV^{LRF}}([0.5, 0.5], [\mathsf{d}_1, \mathsf{d}_1])\Big]\Big) =$$
$$\mathbf{CONV^{LRF}}\Big(\Big[0.2, 0.8\Big], \Big[\mathsf{d}_1, \mathsf{d}_4\Big]\Big) = \mathsf{d}_3$$

Now, let us swap the positions of the first and the third experts. Therefore, we want to aggregate $[\mathsf{d}_4, \mathsf{d}_1, \mathsf{d}_1]$ *given a weighting vector* $[0.8, 0.1, 0.1]$*. As we will see, the result is different:*

$$\mathbf{CONV^{LRF}}([0.8, 0.1, 0.1], [\mathsf{d}_4, \mathsf{d}_1, \mathsf{d}_1]) =$$
$$\mathbf{CONV^{LRF}}\Big(\Big[0.9, 0.1\Big], \Big[\mathsf{d}_1,$$
$$\quad \mathbf{CONV^{LRF}}([0.89, 0.11], [\mathsf{d}_4, \mathsf{d}_1])\Big]\Big) =$$
$$\mathbf{CONV^{LRF}}\Big(\Big[0.9, 0.1\Big], \Big[\mathsf{d}_4, \mathsf{d}_1\Big]\Big) = \mathsf{d}_4$$ □

In our case, when using CONV–RRF or CONV-LRF, rather than assuming a global vector of linguistic labels $\mathcal{L}$ (such as VeryLow, Low, Neutral, High, and VeryHigh), for each fuzzy datatype F_i we assume a different vector $\mathcal{L}_i = [l_{i1}, l_{i2}, \dots]$. On the one hand, it does not make sense to use the same label VeryHigh to define a coffee intensity or a temperature. On the other hand, the linguistic labels are not defined a priori, so we can only rely on the definitions given by the experts. Furthermore, the same set of labels can be different for each fuzzy datatype F_i, as some experts might not provide a definition. Thus, for each F_i, we define $\mathcal{L}_i = [d_{i\sigma(K)}, \dots, d_{i\sigma(1)}]$.

It is worth to note that a pair of labels $l_{ij}, l_{jk} \in F_i$ may have the same definition. That is, although it is probably not very common in practice, two experts may use the very same trapezoidal membership function to define a label. This case was not originally considered in [DVV93] for CONV–RRF, where the authors assumed that

$k > j$. However, we consider this case because combining the same definition is not trivial, because CONV-RRF and CONV-LRF are not associative, as Example 21 shows.

Example 21. *Assume that there are* 5 *experts and that* E_1 *and* E_2 *provide the same definition, so we have* $\mathcal{L} = [\mathit{Exp1\&2}, \mathit{Exp1\&2}, \mathit{Exp3}, \mathit{Exp4}, \mathit{Exp5}]$ *(for ease of presentation, we assume that the i-th expert provides the i-th largest value). Given a weighting vector* $[0.15, 0.15, 0.1, 0.5, 0, 1]$*, one may verify that the consensual definition using CONV–RRF is that of expert* E_3*:*

$$\mathbf{CONV}^{\mathbf{RRF}}\Big([0.15, 0.15, 0.1, 0.5, 0, 1], [\mathit{Exp1\&2}, \mathit{Exp1\&2}, \mathit{Exp3}, \mathit{Exp4}, \mathit{Exp5}]) = \mathit{Exp3}$$

Note that if we group the opinion of the two first experts and assign to this new value the sum of their weights, the result of CONV–RRF is different:

$$\mathbf{CONV}^{\mathbf{RRF}}\Big([0.3, 0.1, 0.5, 0, 1], [\mathit{Exp1\&2}, \mathit{Exp3}, \mathit{Exp4}, \mathit{Exp5}]) = \mathit{Exp4}$$

□

Table 3.5 summarizes some key features of the six operators that can help to choose one of them. In particular, the table shows if the output is always one of the inputs or not, and if weights are assigned to a specific expert or not.

Criterion	CONV–RRF/LRF	LOWA–RRF/LRF	WMEAN	FOWA
Output is always one of the input datatypes	Yes	Yes	No	No
Weights are assigned to a specific expert	Yes	No	Yes	No

Table 3.5: Comparison between different aggregation strategies.

Implementation: Fudge. We have developed an implementation of the consensual aggregation of fuzzy datatypes described in the previous paragraphs. Our tool is called *Fudge* (FUzzy Datatypes from a Group of Experts) and is available online[11]. The application has two versions for desktop computers and for mobile devices (see Section 5.4.2), and uses OWL API to manage (fuzzy) OWL 2 ontologies represented in Fuzzy OWL 2 language.

Fudge receives a folder containing the input ontology files and imports all the OWL files in the folder. We assume that each of the input files includes a Fuzzy OWL 2 ontology —specifically, an OWL 2 ontology where datatypes can have an OWL 2 annotation describing the parameters of the fuzzy function. As an output, Fudge creates a new ontology with the axioms included in the input files, except the

[11] `http://webdiis.unizar.es/~ihvdis/Fudge.html`

declarations of the datatypes, which are unique. That is, if two or more files have a datatype with the same name, it only adds a consensual one to the output ontology.

In theory, all input files should contain the same axioms (ontology schema and individuals), and only the datatype annotations may be different. In practice, it could happen that not all ontologies contain the same axioms. In such cases, there are several possible choices: adding to the output ontology axioms that are in all input ontologies, adding axioms that are in some of the ontologies, adding axioms that are in most of the ontologies, etc. Among them, we chose to add the axioms included in the input ontology with a larger number of logical axioms.

Fudge considers as a name of an entity its full URI (e.g., `http://sid.cps.unizar.es/engines.owl#HighTemperature`) rather than its fragment identifier (e.g., HighTemperature), as two experts could use the same fragment to denote two different entities (e.g., in the car domain, temperature of an engine and temperature of oil).

Note that some datatype may not be annotated in some of the input files. In such cases, only the existing annotations are taken into account, and a vector of weights of the appropriate size is computed, as already discussed in the previous section.

The declaration of the fuzzy datatypes must conform the specification of Fuzzy OWL 2, including an annotation (with the type of the membership function and the values of the parameters) and a range restriction to an interval $[k_1, k_2]$. The current implementation is restricted to trapezoidal, triangular, left-shoulder, and right-shoulder functions, accordingly to the Fuzzy OWL 2 specification.

So far, the supported aggregation operators are CONV–LRF, CONV–RRF, WMEAN, LOWA–LRF, LOWA–RRF, and FOWA. It is worth to stress that the application has been designed to ensure that adding more aggregation operators is very easy. Indeed, it is enough to *(i)* add a new class extending an existing one, *(ii)* implement a method computing the aggregation of K trapezoidal functions, and *(iii)* update the graphical interface by adding another item to a list of aggregation operators.

The following fragment of code illustrates the creation of the new class:

```
public class NewAO extends AggregationOperator
{
   @Override
   public TrapezoidalFuzzyNumber aggregate(
      ArrayList<TrapezoidalFuzzyNumber> values, Double[] weights)
      {
         ...
   }
}
```

To obtain the weights for LOWA–LRF, LOWA–RRF, and FOWA one may use quantifier-base aggregation (using right-shoulder, linear, and power functions) or two recursive procedures starting from a given orness.

User interface. Now we will describe the user interface for desktop computer; an interface for mobile devices will be described in Section 5.4.2.

Fudge is written in Java. A simple user interface allows to select the input ontologies, the type of consensus (aggregation operator) and the necessary parameters: a vector of weights for CONV–LRF, CONV–RRF, and WMEAN, and the type of fuzzy quantifier and its parameters or the orness value for LOWA–LRF, LOWA–RRF, and FOWA. Figure 3.10 shows the main tab of the user interface. Initially, the second and the third tab are disabled.

If the user selects CONV–LRF, CONV–RRF, or WMEAN as the aggregation operator, the fourth tab is enabled, as shown in Figure 3.11. In this case, Fudge checks that all values are positive and normalized (i.e., that the sum is equal to 1).

If the user selects LOWA–LRF, LOWA–RRF, or FOWA as an aggregation operator, the second and the third tab become enabled. The second tab allows to obtain the weights from a fuzzy quantifier. Figures 3.12–3.14 show how to select the type of fuzzy quantifier (right-shoulder, linear, and power functions) and their parameters. The user can see a general picture of the selected fuzzy quantifier and a customized picture with the values of the selected parameters (Figure 3.15). Fudge checks that all values are correct; for instance, in a right-shoulder function, $q_2 \geq q_1$. The third tab allows to use a recursive procedure to obtain the weights. Figure 3.16 shows that the user can select the type of recursive (left or right) and an orness value in $[0, 1]$.

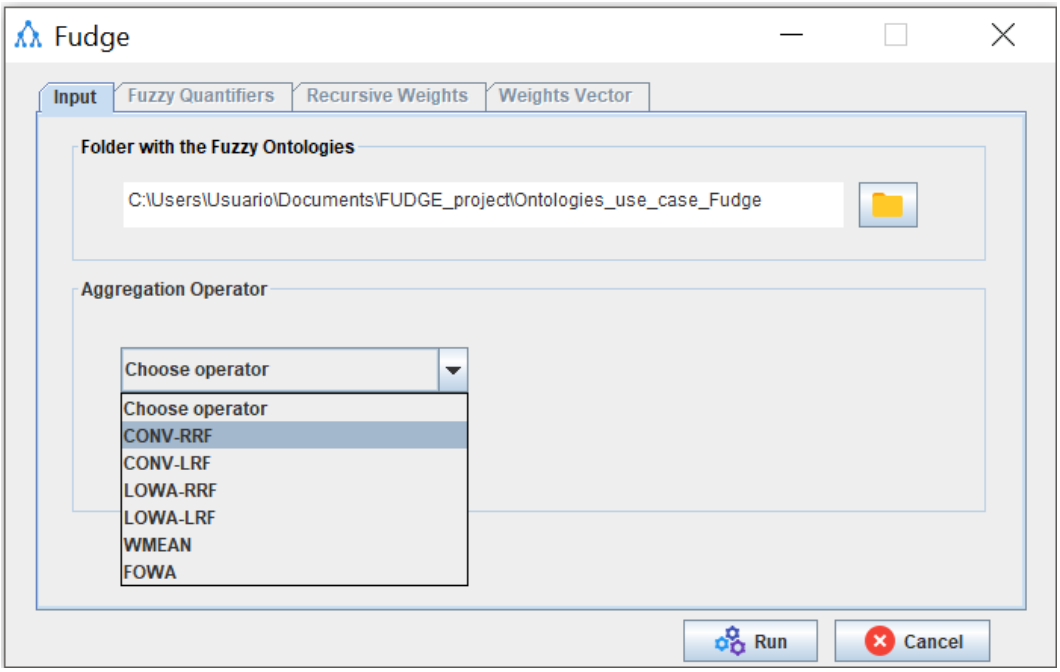


Figure 3.10: Snapshot of Fudge: selection of input files and aggregation operator.

The main features of Fudge are:

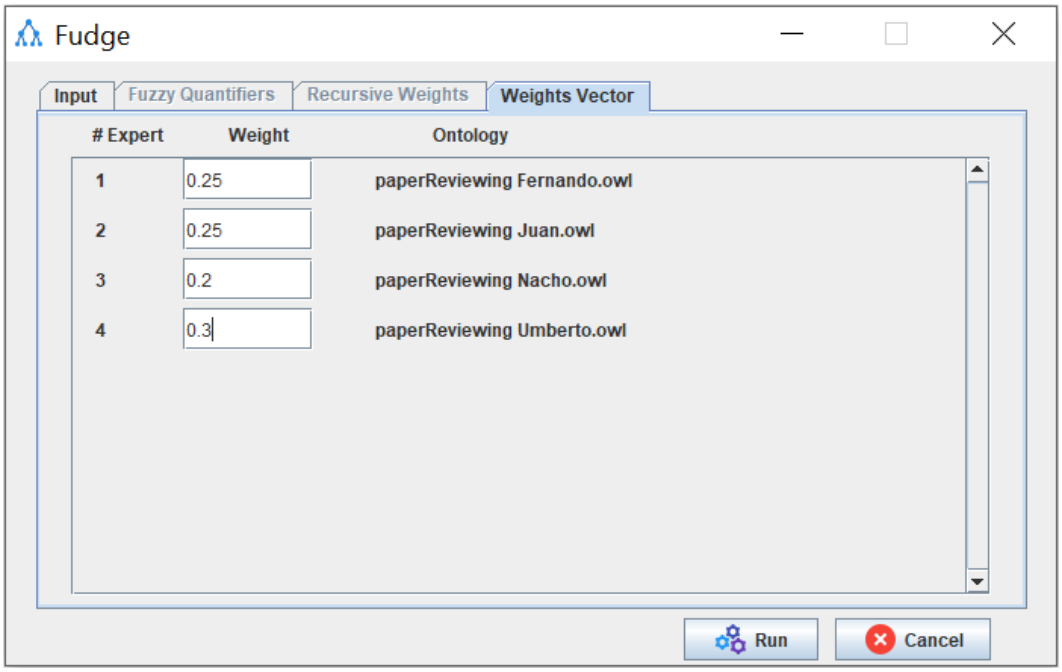


Figure 3.11: Snapshot of Fudge: vector of weights.

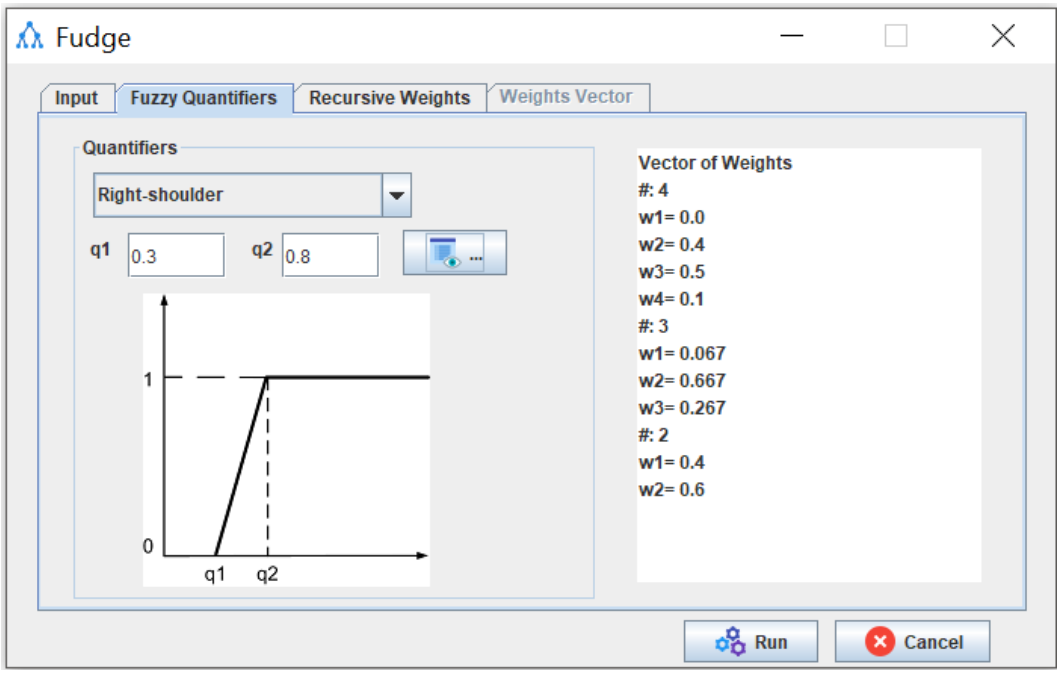


Figure 3.12: Snapshots of Fudge: selection of right-shoulder quantifier.

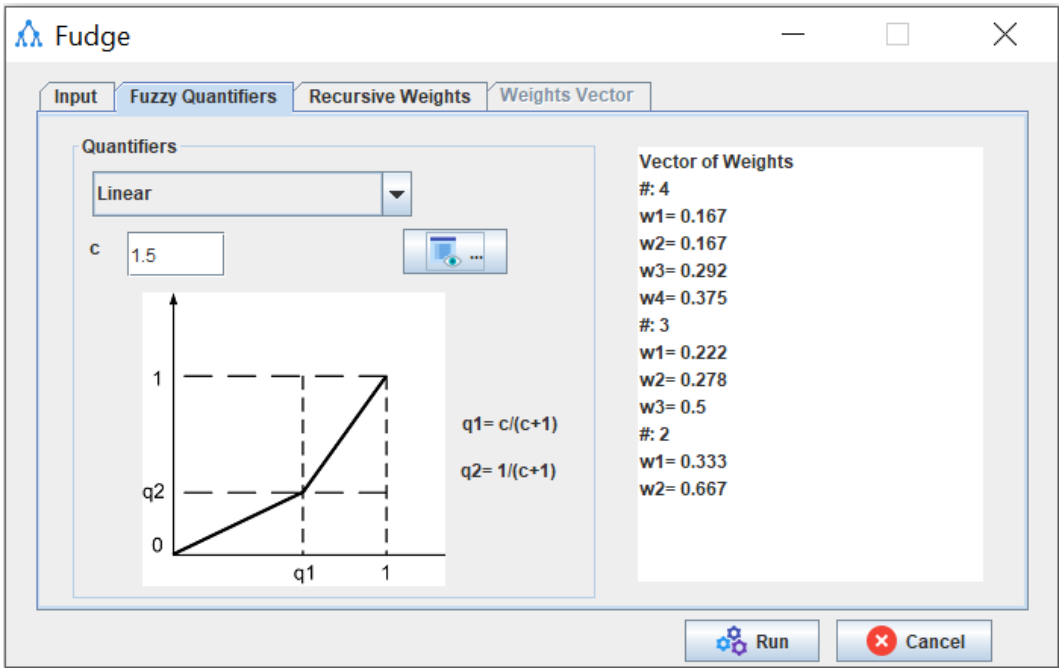


Figure 3.13: Snapshots of Fudge: selection of linear quantifier.

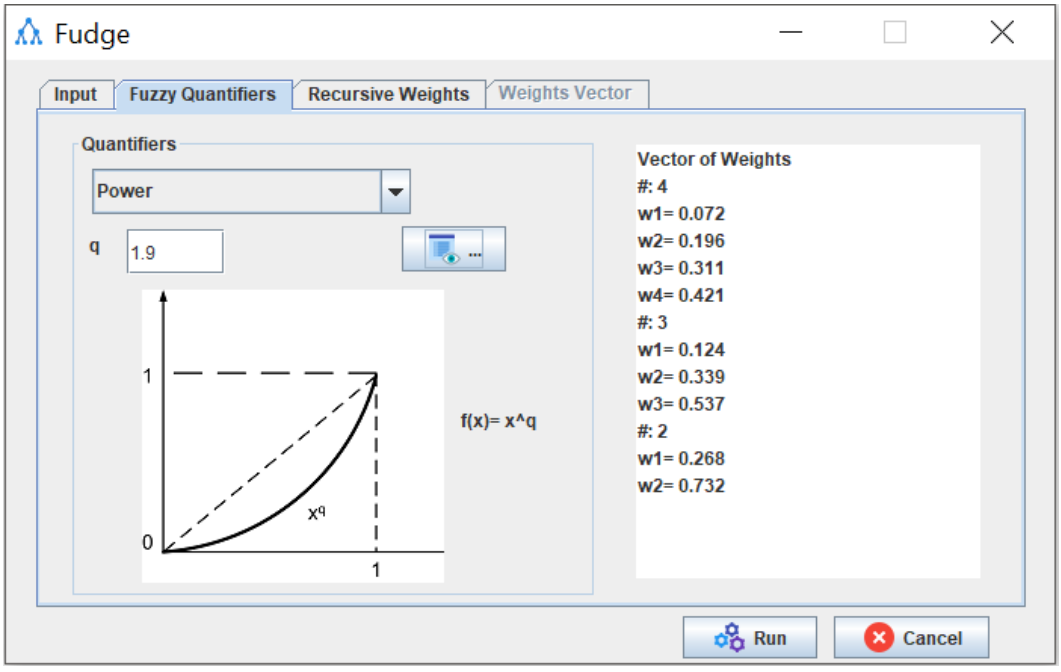


Figure 3.14: Snapshots of Fudge: selection of power quantifier.

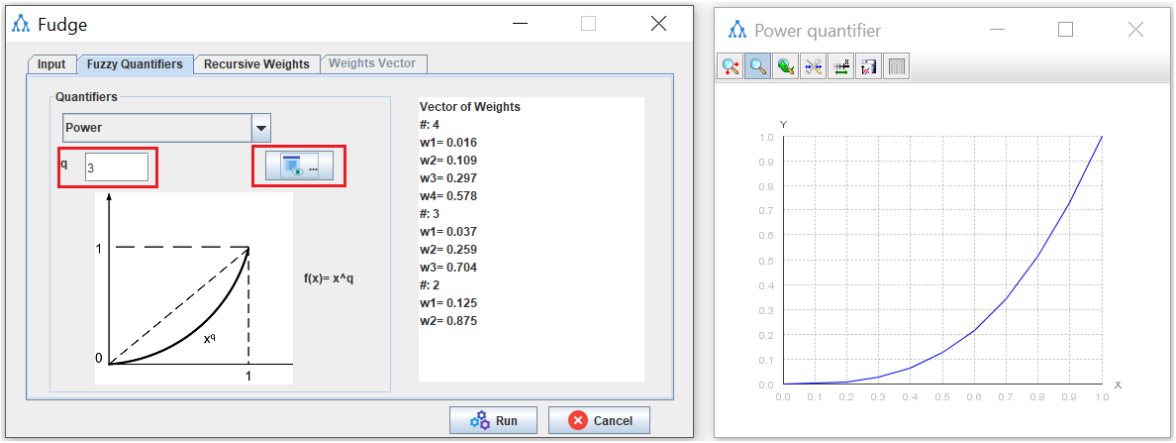


Figure 3.15: Snapshots of Fudge: customized picture for a power quantifier.

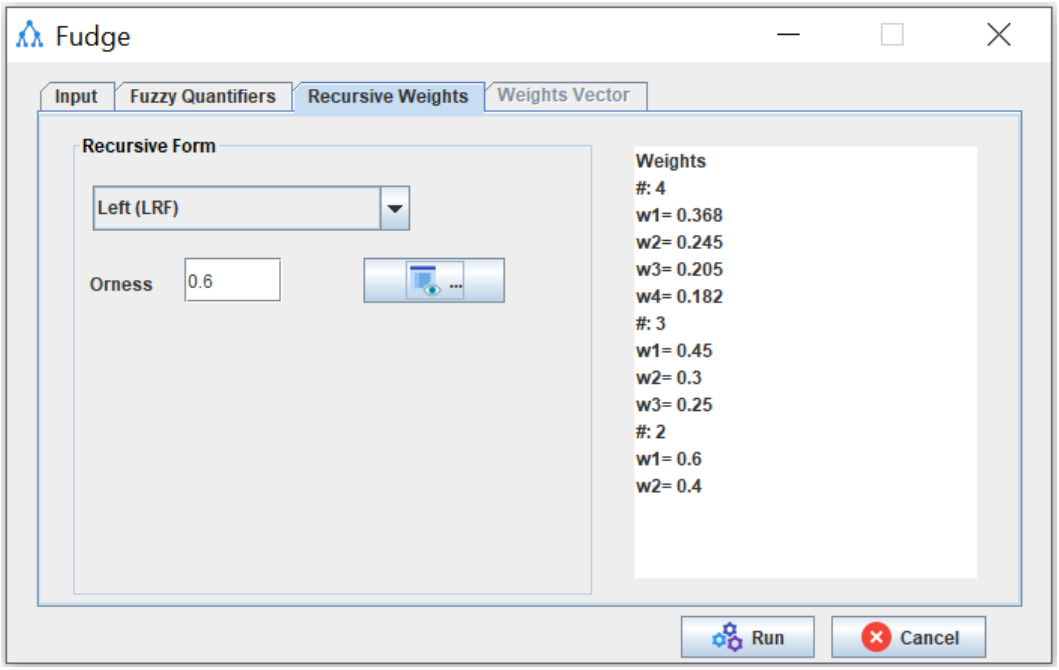


Figure 3.16: Snapshot of Fudge: computing the weights from an orness value.

– For fuzzy ontology developers, it could be used to generate a consensus in the definition of fuzzy datatypes in an automatic way.

– It is domain independent.

– It supports the case where some definitions are omitted by experts.

– It has a modular programming design that makes it very easy to add new aggregation methods.

– It has an intuitive Graphic User Interface and a mobile version for Android devices (described in Section 5.4.2).

Related work

Some works have previously considered aggregation operators in fuzzy ontology scenarios [BS13, CBM12, Voj07], but the aggregation was restricted to numerical degrees of truth. Furthermore, most of the existing work assumes a unique definition of the fuzzy membership functions that define the fuzzy datatypes. In the following we will focus only on the few exceptions.

In a series of papers, researchers studied the application of fuzzy ontologies to reach a consensus on decision making scenarios [MKC+17, PWM+13]; in particular, they used existing fuzzy ontologies as part of decision making processes. In related papers, they also studied the process of building a fuzzy ontology; for example, [MKP+19, MPUH15, MPUH16] discussed how to build a fuzzy ontology in scenarios of multi-granular linguistic information. There are significant differences with respect to our proposal. The approach in [MPUH15] does not take account the opinions of different ontology developers to build a fuzzy ontology. On the other hand, the approach in [MPUH16] does not focus on fuzzy datatype construction but on assigning a membership degree to individuals of a concept. Finally, the approach in [MKP+19] considers users' opinions in social networks and computes the fuzzy membership function from sentiment information (numbers of positive, neutral, and negative words). In contrast, we compute fuzzy membership functions by aggregating the fuzzy membership functions of each ontology builder, rather than directly obtaining the definitions of the fuzzy membership functions from the opinions of the different users. The latter strongly depends on the quality of the sentiment analysis and is more vulnerable to malicious users. In summary, none of the existing approaches uses the de facto standard for fuzzy ontology representation Fuzzy OWL 2, supports several aggregation operators, or provides a publicly available and extensible implementation.

It is also worth to mention a fuzzy ontology-based approach for diet recommendation where the fuzzy datatypes provided by each expert, described using standard fuzzy sets, are combined into a single fuzzy datatype described using a type-2 fuzzy set [LWH10]. Instead, we are concerned with building consensual type-1 fuzzy datatypes.

Chapter 4

Contributions to fuzzy ontology reasoning

In this section we will propose novel reasoning algorithms for fuzzy ontologies. The two first sections focus on the instance retrieval and the realization problems. On the one hand, Section 4.1 proposes two specific algorithms to solve these problems, rather than reducing them to another reasoning task as previous works did. On the other hand, Section 4.2 proposes and solves a novel reasoning problem, which we call flexible faceted instance retrieval, where the user does not only specify a fuzzy concept but also a list of values, described using fuzzy datatypes, for some data properties. As a solution we propose two minimalist reasoning algorithms for some restricted yet useful cases. The two latter sections focus on the similarity between fuzzy ontology elements: Section 4.3 describes a novel and general approach to compute the similarity between two individuals, whereas Section 4.4 focus on fuzzy matchmaking between two individuals, computing the best possible agreement satisfying their individual restrictions.

4.1 Algorithms for instance retrieval and realization

Motivation

As already mentioned in Section 2, two of the most important reasoning tasks in ontologies are the instance retrieval and the realization problems. In classical ontologies, instance retrieval consists of retrieving all the individuals i that are known to belong to a given a 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

to compute 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 [BS16a]. 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 in 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 $\alpha > 0$. This can be computed using a best entailment degree test for each concept C in the fuzzy ontology, but no specific algorithms have been designed.

Contributions

In this section we will describe two specific algorithms to solve the instance retrieval problem and realization problem for fuzzy ontologies. Such algorithms are based on an extension of an optimization technique called *optimization partitioning*, originally proposed at [HPS07] and extended at [BS15]. This optimization can be applied in a family of algorithms to reason with fuzzy DLs that are based on a combination of classical tableaux rules and mathematical programming [Str05], 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 solve independently these CGO problems (perhaps optimizing several times the same CGO problem if necessary) rather than optimizing several times the whole original problem. First, we will present the novel algorithm to solve the instance retrieval problem and then the realization problem given a fuzzy ontology.

It is worth noting that the results of [BS15] involve reasoning tasks where just one optimization problem needs to be solved. Their experiments show that, in such cases, splitting the optimization problem into several ones does not decrease, in general, the running time. In this thesis, however, we address problems that require solving several optimization problems. In instance retrieval, the basic algorithm requires as many tests as individuals in the ontology; in realization, as many tests as atomic concepts 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.

4.1.1 Instance retrieval 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 22 shows.

Example 22. *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

$$x_{johnSmith:DemocratVoter} + x_{johnSmith:RepublicanVoter} \geq 1 \tag{4.1}$$

We can indeed compute the realization problem by (i) *adding $\langle \mathsf{johnSmith} : \neg C \geq 1 - x_{CObj} \rangle$ to $\mathcal{O}$, where x_{CObj} is a new variable, for one of the atomic concepts $C \in \mathcal{O}$,* (ii) *minimizing x_{CObj}, and* (iii) *repeating the process for each of the atomic concepts in $\mathcal{O}$. For example, adding $\langle \mathsf{johnSmith} : \neg \mathsf{DemocratVoter} \geq 1 - x_{DemocratVoterObj} \rangle$ leads to a constraint*

$$x_{johnSmith:DemocratVoter} \leq x_{DemocratVoterObj} \tag{4.2}$$

and the minimum value of $x_{DemocratVoterObj}$ with respect to Equations 4.1–4.2 is 0, i.e., there is a solution to the MILP problem such that $x_{DemocratVoterObj} = 0$ (and $x_{johnSmith:RepublicanVoter} = 1$).

However, if we also added $\langle \mathsf{johnSmith} : \neg \mathsf{RepublicanVoter} \geq x_{RepublicanVoterObj} \rangle$, we would have a constraint

$$x_{johnSmith:RepublicanVoter} \leq x_{RepublicanVoterObj} \tag{4.3}$$

Now, in every solution of the optimization problem in Equations 4.1–4.3, $x_{DemocratVoterObj} > 0$ or $x_{RepublicanVoterObj} > 0$ hold, which is incorrect as we are interested in answers that hold in every model.

□

However, we can merge optimization problems as long as the involved variables are independent. Based on this idea, Algorithm 3 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 $\mathcal{O} \models \langle a : C \geq \alpha \rangle$.

Algorithm 3 Algorithm to compute the instance retrieval problem given a fuzzy ontology

Input: A fuzzy ontology $\mathcal{O}$

Input: 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**
  $x_{a:Obj}$ = new variable
  $\mathcal{O} \cup \langle a : \neg C, 1 - x_{a:Obj}\rangle$
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 $x_{a:Obj} \in \mathbf{C}_i$
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 w.r.t. $\mathbf{C}_{one} \cup \{z = \sum_{x_{a:Obj} \in \mathbf{C}_{one}} x_{a:Obj}\}$
for each $x_{a:Obj}$ in the model of the solution **do**
  if $x_{a:Obj} > 0$ **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 $x_{a:Obj} \in \mathbf{C}_{two\_or\_more}$ **do**
  $\alpha \leftarrow$ Minimize $x_{a:Obj}$ w.r.t. $\mathbf{C}_{two\_or\_more}$
  if $\alpha > 0$ **then**
    $L \leftarrow L \cup \langle a, \alpha\rangle$
  end if
end for
return L

Lines 1– 6 add a new assertion 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 the previous algorithm implemented in fuzzyDL [BS16a], but now we create them together, at the beginning of the algorithm.

Lines 7– 10 partition the single constraint set with respect to $\mathbf{C}$ into a set of constraint sets $\mathbf{C}_i$. This is similar to the approach in [BS15]. However, we also compute the number of variables to be minimized $x_{a:Obj}$ in each of the problems so that we can consider three cases:

$\mathbf{C}_{zero}$**:** constraint sets without an objective variable (so that we only need to check if they have a solution),

$\mathbf{C}_{one}$**:** constraint sets with exactly one objective variable of the form $x_{a:Obj}$, and

$\mathbf{C}_{two\_or\_more}$**:** constraint sets with more than one objective variable of the form $x_{a:Obj}$ (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 [BS15].

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 $x_{a:Obj}$ to be minimized. This is possible only because all variables $x_{a:Obj}$ are independent: in this case the minimum of the sum occurs when all the variables have its minimum value. The value of each $x_{a:Obj}$ 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 $x_{a:Obj}$ introduced in Lines 1– 6 that belongs to $\mathbf{C}_{two\_or\_more}$. The minimal value of each $x_{a:Obj}$ 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 $x_{a:Obj}$. 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 [BS15] showed that solving a problem is not more expensive than solving two simpler ones, disjoint subsets of the original one.

Example 23 illustrates our instance retrieval algorithm by showing some of the variables obtained and their interdependence.

Example 23. *Consider an input fuzzy ontology* $\mathcal{O} = \{a : A, \langle b : B \geq 0.3\rangle, \langle c : B \geq 0.1\rangle, \langle (d, e) : R \geq 0.8\rangle, d : B \sqcap \forall R.B\}$ *and let us compute the instance retrieval of the concept* B *in Łukasiewicz fuzzy logic. The reasoning algorithm computes a constraint set* $\mathbf{C}$*, which can be partitioned into three partitions:*

- $\mathbf{C}_{zero}$*, including variable* $x_{a:A}$*,*
- $\mathbf{C}_{one}$*, including variables* x_{aObj}*,* x_{bObj} *and* x_{cObj}*, and*
- $\mathbf{C}_{two\_or\_more}$*, including variables* x_{dObj} *and* x_{eObj}*.*

Figure 4.1 shows a graph including some variables in the constraint set $\mathbf{C}$ *and their partitions: orange denotes that the variable is in* $\mathbf{C}_{zero}$*, green denotes a variable in* $\mathbf{C}_{one}$*, and yellow denotes a variable in* $\mathbf{C}_{two\_or\_more}$*. It suffices to check that* $\mathbf{C}_{zero}$ *has a solution.* $\mathbf{C}_{one}$ *can be solved by minimizing a variable* z *defined as a sum of the three objective variables, i.e.,* $z = x_{aObj} + x_{bObj} + x_{cObj}$*. To solve* $\mathbf{C}_{two\_or\_more}$*, one needs to minimize separately* x_{dObj} *and* x_{eObj}*. The output of the algorithm is a set of pairs individual-membership degree, namely* $\langle b, 0.3\rangle, \langle c, 0.1\rangle, \langle d, 1\rangle, \langle e, 0.8\rangle$ □

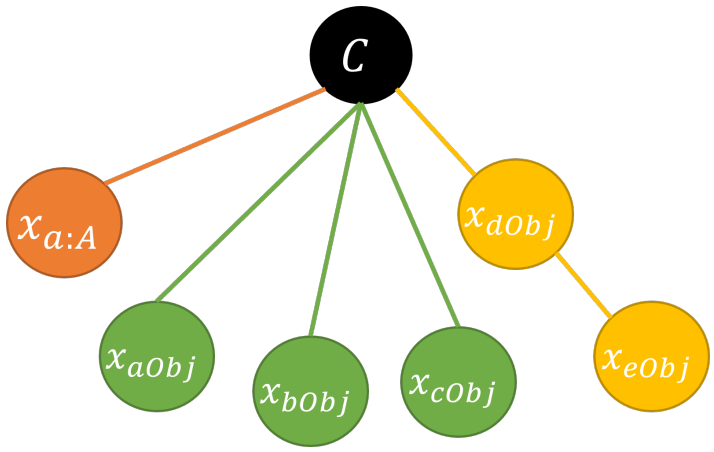


Figure 4.1: Fragment of a dependency graph to solve an instance retrieval problem.

As our experiments will show (see Section 6.4), 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 24 and 25.

Example 24. *Assume that a fuzzy ontology* $\mathcal{O}$ *has two domain and range axioms, stating that the domain and range of an object property* R *are* C_d *and* C_r*, respectively,*

and that there are two object property assertions relating individuals i_1 and i_2 with individual i_3, via R. Assume also that we want to retrieve the instances of C_d, so the algorithm adds (among others) the pair of assertions $\langle i_1 : \neg_{\text{Ł}} C_d, 1 - x_1 \rangle$ and $\langle i_2 : \neg_{\text{Ł}} C_d, 1 - x_2 \rangle$. The first assertion causes that the objective variable x_1 is connected to $x_{i_1:C_d}$. Because of the domain axiom, variables $x_{i_1:C_d}$ and $x_{(i_1,i_3):R}$ are connected. Because of the range axiom $x_{(i_1,i_3):R}$ and $x_{i_3:C_r}$ are connected, and so are $x_{i_3:C_r}$ and $x_{(i_2,i_3):R}$. Moreover, the domain axiom causes that $x_{(i_2,i_3):R}$ and $x_{i_2:C_d}$ are connected. Because of the assertion $\langle i_2 : \neg_{\text{Ł}} C_d, 1 - x_2 \rangle$, $x_{i_2:C_d}$ and the objective variable x_2 are connected. Therefore, $x_1 \rightsquigarrow_{\mathbf{C}} x_2$, so $\mathbf{C}_{two\_or\_more}$ is not empty (there is a partition that contains at least two objective variables, x_1, x_2). The same problem happens if we want to retrieve the instances of a subclass of C_d.

Note that this situation does not happen without the range axiom. Although a_1, a_2, and a_3 would belong to the same ABox partition in the sense of [HPS07], they would not belong to the same optimization problem partitioning. □

Example 25. *Consider again Example 24 without the range axiom. In expressive languages with nominals, there is an additional rule called $\exists_a$ [BS14] that adds a constraint of the form $x_{i:\{i\}} \Rightarrow (x_{i_j:\exists R.\{i\}} \Rightarrow x_{(i_j,i):R}) \geq 1$ for each individual i_j related to i via R*[1]*. Because individuals i_1, i_2 are related via R to i_3, variables $x_{(i_1,i_3):R}$ and $x_{(i_2,i_3):R}$ are connected via $x_{i_3:\{i_3\}}$. Therefore, x_1 and x_2 are connected, so $\mathbf{C}_{two\_or\_more}$ is not empty.* □

In Section 6.4 will describe our experiments and findings about implemented Algorithm 3 on fuzzyDL semantic reasoner. We compared the previous version of fuzzyDL with new version.

4.1.2 Realization in fuzzy ontologies

Algorithm 3 can be easily adapted to the realization problem. Algorithm 4 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 are similar to the same lines in Algorithm 3, but now we add a new assertion (involving a new variable) 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.

[1] If a fuzzy ontology does not use nominals, this rule does not need to be applied.

Algorithm 4 Algorithm to compute the realization problem given a fuzzy ontology

Input: A fuzzy ontology $\mathcal{O}$

Input: An individual a

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

for each atomic concept $A \in \mathcal{O}$ **do**
  x_{aObj} = new variable
  $\mathcal{O} \cup \langle a : \neg A, 1 - x_{aObj}\rangle$
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 $x_{aObj} \in \mathbf{C}_i$
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 w.r.t. $\mathbf{C}_{one} \cup \{z = \sum_{x_{aObj} \in \mathbf{C}_{one}} x_{aObj}\}$
for each solution x_{aObj} in the model of the solution **do**
  if $x_{aObj} > 0$ **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 $x_{aObj} \in \mathbf{C}_{two\_or\_more}$ **do**
  $\alpha \leftarrow$ Minimize x_{aObj} w.r.t. $\mathbf{C}_{two\_or\_more}$
  if $\alpha > 0$ **then**
    $L \leftarrow L \cup \langle A, \alpha\rangle$
  end if
end for
return L

We can also consider the same alternatives as in the instance retrieval 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.

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 26 shows that having disjoint concept axioms introduces dependencies.

Example 26. *If there is an axiom stating that two concepts C_1 and C_2 are disjoint, a constraint $x_{a:C_1} \otimes x_{a:C_2} = 0$ is created 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 $\langle a : \neg_L C_1, 1 - x_{C_1Obj}\rangle$ and $\langle a : \neg_L C_2, 1 - x_{C_2Obj}\rangle$, so variables x_{C_1Obj}, x_{C_2Obj} are connected as well, and thus $\mathbf{C}_{two\_or\_more}$ is not empty (there is a partition that contains at least two objective variables, x_{C_1Obj} and x_{C_2Obj}).* □

Related work

To the best of our knowledge, no specific reasoning algorithms to solve instance retrieval and realization problems given a fuzzy ontology are reported in the literature.

Table 4.1 shows a group of fuzzy reasoners that support the instance retrieval task or realization task. We can see that fuzzyDL, FRESG and YADLR are the only ones supporting instance retrieval, while FRESG and YADLR are the only ones supporting realization.

Reasoner	Instance Retrieval	Realization
fuzzyDL [BS16a]	•	
Fire [SSSK06]		
FPLGERDS [Hab07]		
YADLR [KA07]	•	•
DeLorean [BCFGR12]		
GURDL [HPS07]		
FRESG [WMY09]	•	•
LiFR [TDKM14]		
SMT-based solver [ABB+13]		

Table 4.1: fuzzy reasoners.

FRESG computes instance retrieval and realization by reducing to several tableaux algorithm tasks, and YADLR to several BEDs[2]. 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]Strictly speaking, YADLR checks if an individual is a member of a given concept with an unknown degree of truth, represented using a variable.

The previous version of fuzzyDL also required to compute several BED tests to compute the instance retrieval. In this work, we have proposed specific algorithms to solve both reasoning tasks and have implemented the instance retrieval algorithm in fuzzyDL.

It is worth to mention a recent work that solves the realization problem given a classical ontology (although only the most specific concepts are retrieved) [SDY22] using ontology partitioning, as we do. While the different queries to be solved are easier, as they involve smaller ontologies, the total number of tests is not optimized. Furthermore, our work [HBB20] was published two years before.

4.2 Minimalist algorithms for flexible faceted instance retrieval

Motivation

The instance retrieval problem is well known but might give too many results. For example, retrieving all instances of Ale in a beer recommender system would include the instances of all subclasses of Ale, which are too many. To reduce the number of results, one can restrict to the top-k instances or filter those individuals not exceeding a certain threshold. Another alternative is to specify the values of some data properties, e.g., retrieving all *Ale beers* with a *high alcohol* and a *high quality*.

Furthermore, although we have optimized in the previous section a reasoning algorithm to solve the instance retrieval problem, some further optimizations are possible if we restrict to some specific cases. In particular, if we can assume that only some parts of the fuzzy ontology are actually fuzzy, one can try to adapt the reasoning algorithms to such cases, and to reuse as much as possible existing crisp ontology reasoners. Specifically, a possible idea is to retrieve the instances of a concept using a crisp ontology reasoner, and then to compute the membership degree to some fuzzy sets (e.g., the fuzzy set of high alcoholic beers) and to combine the degrees using fuzzy logic operators.

Contributions

Our first contribution is the definition of a novel reasoning task that we call *flexible faceted instance retrieval*. The idea is to extend classical fuzzy instance retrieval to narrow down the query results by imposing some conditions on the attribute values. We will also propose two *minimalist* reasoning algorithms for some specific cases, where the term minimalist refers to the fact that the algorithm cannot support any element

of a fuzzy ontology, but only a selection of them useful for the task —namely, fuzzy datatypes and fuzzy concept assertions. This kind of queries is pervasive in real-world problems since they can be used to obtain the domain objects that satisfy imprecise (and probably complex) constraints defined over their data properties.

Flexible faceted instance retrieval. In particular, given a fuzzy ontology $\mathcal{O}$, our aim is to retrieve the instances of a fuzzy concept C such that the values of n functional data properties p_i with a numerical range are compatible with a fuzzy datatype D_i. Recall the previous example: retrieving all the instances of *Ale* beers such that their *alcohol* is *High* and their *quality* is *High*. Furthermore, the intermediate degrees of truth can be combined using a combination function f_c (e.g., an aggregation operator such as weighted mean or OWA, a t-norm, or a t-conorm), and the final degree can be modified using a modifier function f_h (a fuzzy hedge), e.g., the `very` modifier.

Definition 3 (Flexible faceted instance retrieval)**.** *Given the sextuple* $\langle \mathcal{O}, C, [p_1, \ldots, p_n], [D_1, \ldots, D_n], f_c, f_h \rangle$*, the solution to the flexible faceted instance retrieval is an ordered list of pairs* $\langle i_i, \alpha_i \rangle$ *such that*

$$\begin{aligned} &\mathcal{O} \models \langle i_i : C, \beta_i \rangle \ , \\ &\mathcal{O} \models (i_i, v_j) : p_j, j \in \{1, \ldots, n\} \ , \\ &\alpha_i = f_h(f_c(\beta_i, D_1(v_1), \ldots, D_n(v_n)) > 0 \ , \\ &\alpha_i \geq \alpha_j, j > i \ . \end{aligned} \tag{4.4}$$

Example 27. *Consider a beer ontology* $\mathcal{O}$ *where the class* Beer *have* 5 *sibling subclasses* Ale*,* Lager*,* Wheat*,* Sour*, and* Specialty*.* Ale *has some subclasses* Stout*,* Trappist*,* Belgian_Strong_Ale*, and* Indian_Pale_Ale*. Such subclasses are non-direct in general, for example,* Belgian_Strong_Ale *is a direct subclass of* BelgianAle*, which is a direct subclass of* Ale*. There are* 2 *data properties* alcohol *and* quality *representing the alcohol level and the quality of a beer, respectively, and* 5 *individuals (*Chimay Bleue*,* Pauwel Kwak*,* Delirium Tremens*,* BrewDog Punk IPA*, and* Guinness Draught*). The following table shows for each beer to which subclass of* Ale *it belongs to, the alcohol level, and the quality:*

Beer	*Type*	*alcohol*	*quality*
BrewDog Punk IPA	*Indian_Pale_Ale*	5.6	3.78
Chimay Bleue	*Trappist*	9	3.95
Delirium Tremens	*Belgian_Strong_Ale*	*8.5*	3.9
Guinness Draught	*Stout*	4.2	3.79
Pauwel Kwak	*Belgian_Strong_Ale*	8.4	3.81

We want to retrieve the instances of Ale *such the alcohol level is* HighAlcohol *and the quality is* HighQuality*, using as a combination function the minimum t-norm*

$f_c(x_1, \ldots, x_k) = \min\{x_1, \ldots, x_k\}$ *and using as a modifier function the fuzzy hedge* very *defined as* $f_h(x) = x^2$. *HighAlcohol is defined as a triangular fuzzy function* $\mathbf{tri}(7, 9, 11)$ *and HighQuality is defined as a triangular fuzzy function* $\mathbf{tri}(3.5, 4, 4.5)$. *Remember that* $\alpha_i = f_h(f_c(\beta_i, D_1(v_1), \ldots, D_n(v_n)) > 0$. *Thus:*

Beer	*HighAlcohol*	*HighQuality*	α_i
BrewDog Punk IPA	0	0.56	0
Chimay Bleue	1	0.9	0.81
Delirium Tremens	0.75	0.8	0.56
Guinness Draught	0	0.58	0
Pauwel Kwak	0.7	0.62	0.38

Therefore, the answer would be:

$$\Big\{\langle \textit{Chimay Bleue}, 0.81\rangle, \langle \textit{Delirium Tremens}, 0.56\rangle, \langle \textit{Pauwel Kwak}, 0.38\rangle\Big\} \qquad \square$$

Note that the previous general case could be simplified in different modes: C can be a crisp concept, there can be a single property (or even none), and f_h can be omitted assuming that it is the identity function. Also, it is trivial to extend the reasoning task to consider only the top-k results.

Specific cases. Our aim now will be to propose a reasoning algorithm for more specific, but still common in practice, cases:

Case 1. We assume that the only fuzzy elements that the fuzzy ontology can contain are fuzzy datatypes and fuzzy concept assertions of the form $\langle i : A \geq \beta\rangle$. Furthermore, we assume that if an individual is asserted to belong partially to a concept, it is not asserted to fully belong to any descendant of the concept for which we want to retrieve the instances.

Case 2. We assume that the only fuzzy elements are fuzzy datatypes.

Note that in Case 1 we only take into account those partial memberships that are stated via a fuzzy concept assertion $\langle i : C \geq \beta\rangle$ with $\beta > 0$ (that will also be propagated to the named concepts that are superclasses of C). Therefore, if $\mathcal{O} \models \langle i : C \geq \beta\rangle$ and $\beta < 1$, there is at least a fuzzy concept assertion of the form $\langle i : C' \geq \beta\rangle \in \mathcal{O}$, for some subclass C' of C (so $\mathcal{O} \models C' \sqsubseteq C$). Note in particular that the case $C' = C$ is possible. Example 28 shows an example of an implicit fuzzy concept assertion that is excluded.

Example 28. *Let us assume* $\{\{i\} \sqsubseteq (A \sqcup A)\} \in \mathcal{O}$ *under Łukasiewicz family of fuzzy operators. Therefore, for each element* x *of the domain,* $(\{i\})^{\mathcal{I}}(x) \leq (A \sqcup A)^{\mathcal{I}}(x)$ *holds. In particular,* $x = i^{\mathcal{I}}$ *implies* $1 \leq (A \sqcup A)^{\mathcal{I}}(i^{\mathcal{I}})$, *so* $A^{\mathcal{I}}(x) \oplus A^{\mathcal{I}}(x) = \min\{A^{\mathcal{I}}(i^{\mathcal{I}}) +$

$A^{\mathcal{I}}(i^{\mathcal{I}}), 1\} \geq 1$, *and thus* $2 \cdot A^{\mathcal{I}}(i^{\mathcal{I}}) \geq 1$, *so* $A^{\mathcal{I}}(i^{\mathcal{I}}) \geq 0.5$ *holds. Therefore, the fuzzy ontology entails a fuzzy concept assertion* $\langle i : A \geq 0.5 \rangle$ *that is not explicitly represented in* $\mathcal{O}$. □

The rationale behind forbidding that an individual fully belongs to another descendant of the concept for which we want to retrieve the instances if the individual partially belongs to a concept, is to avoid computing the membership degrees of individuals to classes using an ontology reasoner. We instead take one or more fuzzy concept assertions and propagate the membership degrees to the superclasses of the concepts. If there is more than one fuzzy concept assertion involving subclasses of C, we can take the maximum of the membership degrees $\max\{\beta_i\}$.

Example 29. *Assume that* beer1 *is a* Stout *with degree* 0.9 *and a* Trappist *with degree* 0.8. *Then, the membership degree to the common superclass* Ale *is* $\max\{0.9, 0.8\} = 0.9$. □

Note also that we do not restrict to a specific family of fuzzy operators (Zadeh, Gödel, Łukasiewicz, or Product). Because we only consider fuzzy datatypes and the propagation of fuzzy concept assertions to their superclasses, where the subclasses axioms are fully true, our algorithm does not depend on the choice of the fuzzy operators.

To efficiently retrieve the degrees β_i without using a reasoner, we retrieve them from the Fuzzy OWL 2 annotations and store them in an appropriate data structure (such as a NoSQL database storing triples) for an efficient data access. In particular, for each fuzzy concept assertion $\langle i : C \geq \beta \rangle \in \mathcal{O}$, we add a tuple $\langle i, C, \beta \rangle$ to the data structure. Note that it is possible to visit all fuzzy concept assertions in a Fuzzy OWL 2 ontology, by looping over all existing annotation assertions involving the fuzzyLabel property.

We avoid adding to the data structure individuals which fully belong to a concept. That is, for each classical concept assertion $i : C$ we do not add to the data structure a tuple $\langle i, C, 1 \rangle$. The reason is that it is not efficient to retrieve each concept C' such that $\mathcal{O} \models i : C'$ but $i : C' \notin \mathcal{O}$; in particular, we would need to use an ontology reasoner.

Note that if there is a classical assertion stating that an individual belongs to a class, there is no annotation assertion. Therefore, individuals fully belonging to a class are not stored in the data structure. Given a flexible faceted instance retrieval over a concept C, it is fine to have an individual partially belonging to several fuzzy concepts that are subclasses of C (i.e., appearing in more than one fuzzy concept assertion) and it is fine to have an individual fully belonging to several fuzzy concepts that are subclasses of C, but it is not possible to have both cases at the same time to propagate

a membership degree to a class C' to a (possibly non-direct) superclass C without having to check (using a reasoner) if the individual fully belongs to C.

A reasoning algorithm for Case 1. Algorithm 5 shows how to compute the flexible faceted instance retrieval of a fuzzy ontology under the restrictions enumerated in the previous section.

The first part (Lines 2–6) is an initialization that can be computed just once, and can be reused by future queries. Firstly, we load the ontology (Line 2), classify the ontology by computing the hierarchy of concept names that fully subsume their subclasses (Line 2), and store the degrees of the fuzzy concept assertions in an auxiliary data structure DS (Lines 3–6). The rest of the code (Lines 8–33) implements the proper query answering. The next steps retrieve all instances of C, noting that some of them might partially belong to C (Line 8), and retrieving all subclasses of C (Line 9). Then, we look in the data structure if each retrieved instance appears in the data structure (partial membership) or not (fully membership). In particular, Lines 12–22 compute the maximum of the degrees in the data structure (1 if there is none). The next step is to compute the satisfaction degrees of the linguistic labels associated to the attributes of the instance. Therefore, Lines 23–28 retrieve the values of each data property (which must be unique because the properties are functional) and compute the membership degrees to the respective fuzzy datatypes (except if the value of the property is unknown). All the obtained degrees are aggregated in Lines 29–31 using a combination functions and a fuzzy hedge, and then added to a list of solutions. Finally, the list is ordered and returned.

One of the key points of the algorithm is that Lines 2, 8, 9, and 24 can be obtained using a classical ontology reasoner and, therefore, rather efficiently.

An implementation of the algorithm will be discussed in Section 6.5, where we will describe a prototype implementation and the evaluation of our application on a real use case for Architecture, Engineering, and Construction (AEC).

A reasoning algorithm for Case 2. Algorithm 6 is more specific than Algorithm 5 for fuzzy ontologies where there are only fuzzy datatypes. Here, the query answering needs all instances of a specific class of an ontology and the functional data properties. For each instance, Lines 7–12 retrieve the values of each data property and compute the membership degrees to the respective fuzzy datatypes. The combination function and a fuzzy hedge compute all the degrees obtained (Lines 13–15), and then the pair $\langle$ instance, degree $\rangle$ is added to a list of solutions LS. Finally, the list is ordered and returned. Example 27 illustrates how this algorithm works.

Algorithm 5 Algorithm to compute the flexible faceted instance retrieval in Case 1.

Input: A fuzzy ontology $\mathcal{O}$
Input: A concept C
Input: A list of functional numerical data properties $[p_1, \ldots, p_n]$
Input: A list of fuzzy datatypes $[D_1, \ldots, D_n]$,
Input: A combination function f_c
Input: A fuzzy hedge f_h
Output: A list of pairs with individuals and membership degrees to C $\left\{\langle i_i, \alpha_i \rangle\right\}$

1: // Initialization
2: $O \leftarrow crispClassify(\mathcal{O})$
3: $DS \leftarrow \emptyset$
4: **for all** $\langle i : C \geq \beta \rangle \in O$ **do**
5: $\quad DS \leftarrow DS \cup \langle i, C, \beta \rangle$
6: **end for**
7: // Query answering
8: $I \leftarrow$ Retrieve all instances i of C in O
9: $S \leftarrow$ Retrieve all subclasses of C in O
10: $LS \leftarrow \emptyset$
11: **for all** $i \in I$ **do**
12: $\quad A \leftarrow \emptyset$
13: $\quad$ **for all** $s \in S$ **do**
14: $\quad\quad$ **if** $\langle i, s, \beta \rangle \in DS$ **then**
15: $\quad\quad\quad A \leftarrow A \cup \beta$
16: $\quad\quad$ **end if**
17: $\quad$ **end for**
18: $\quad$ **if** $A = \emptyset$ **then**
19: $\quad\quad \beta \leftarrow 1$
20: $\quad$ **else**
21: $\quad\quad \beta \leftarrow \max(A)$
22: $\quad$ **end if**
23: $\quad$ **for all** data property p_i **do**
24: $\quad\quad v \leftarrow$ Retrieve the value of the data property p_i for i in O
25: $\quad\quad$ **if** v $\neq$NULL **then**
26: $\quad\quad\quad D \leftarrow D \cup D_i(v)$
27: $\quad\quad$ **end if**
28: $\quad$ **end for**
29: $\quad auxDegree \leftarrow f_c(\beta, D)$
30: $\quad \alpha \leftarrow f_h(auxDegree)$
31: $\quad LS \leftarrow LS \cup \langle i, \alpha \rangle$
32: **end for**
33: $LS \leftarrow sort(LS)$ in decreasing order of degrees of truth
34: **return** LS

In Section 5.2 we will describe an Android app using Algorithm 6 in a beer recommender system that will be evaluated on Section 6.2.

Algorithm 6 Algorithm to compute the flexible faceted instance retrieval in Case 2.

Input: A fuzzy ontology $\mathcal{O}$
Input: A concept C
Input: A list of functional numerical data properties $[p_1, \ldots, p_n]$
Input: A list of fuzzy datatypes $[D_1, \ldots, D_n]$,
Input: A combination function f_c
Input: A fuzzy hedge f_h
Output: A list of pairs with individuals and membership degrees to C $\left\{\langle i_i, \alpha_i \rangle\right\}$

1: // Initialization
2: $O \leftarrow crispClassify(\mathcal{O})$
3: // Query answering
4: $I \leftarrow$ Retrieve all instances i of C in O
5: $LS \leftarrow \emptyset$
6: **for all** $i \in I$ **do**
7: **for all** data property p_i **do**
8: $v \leftarrow$ Retrieve the value of the data property p_i for i in O
9: **if** v $\neq$ NULL **then**
10: $D \leftarrow D \cup D_i(v)$
11: **end if**
12: **end for**
13: $auxDegree \leftarrow f_c(D)$
14: $\alpha \leftarrow f_h(auxDegree)$
15: $LS \leftarrow LS \cup \langle i, \alpha \rangle$
16: **end for**
17: $LS \leftarrow sort(LS)$ in decreasing order of degrees of truth
18: **return** LS

Related work

Different families of reasoning algorithms for fuzzy ontologies can be found in the literature [Str13]. However, most of them are focused on showing the existence of an algorithm rather than on the efficiency in practice. For example, some reasoning algorithms are based on computing an equivalent crisp ontology, with a blowup in the size of the ontology [BDGRS12]. DeLorean implements some of these algorithms. This is clearly not scalable and in appropriate to answer queries over real knowledge bases, with a very high number of individuals and axioms.

Because ontology languages provide a trade-off between expressive power and complexity of the reasoning, a first way to guarantee an efficient reasoning is to restrict the expressivity. In classical ontologies, the OWL 2 language has three sublanguages or profiles with tractable reasoning (i.e., the main reasoning tasks can be solved in a

polynomial time), namely OWL 2 EL, OWL 2 QL, and OWL 2 RL [W3C12b]. In the fuzzy case, it has been showed that fuzzy extensions of tractable languages are not tractable in general [Bob16], and they can even be undecidable [BCP17]. Despite this fact, some fuzzy extensions of tractable DLs have been investigated, including fuzzy extensions of the logics behind OWL 2 EL [BS18, MSS+12], OWL 2 QL [PSS+08], and OWL 2 RL [SVS15].

Another approach is to develop specific optimization techniques to make reasoning more efficient in some common cases in practice. While many optimization techniques are known for classical DLs, optimizations for fuzzy DLs have not received such attention, but there are some exceptions. Haarslev et al. [HPS07] proposed caching (to avoid repeating computations), lexical normalization (transforming concept expressions into a canonical form to detect inconsistencies earlier), simplifications of concept expressions, and ABox partitioning (splitting axioms about individuals—concept and property assertions—into disjoint sets). Simou et al. [SMSS10] proposed degrees normalization, to remove superfluous axioms when the same axioms is stated with different degrees of truth, and some optimizations of the algorithm to compute the best entailment degree of an axiom. Moreover, Bobillo and Straccia [BS16b] proposed lazy unfolding, to delay the expansion of subclass axioms as much as possible, and an absorption algorithm to increase the applicability of lazy unfolding. fuzzyDL reasoner implements these and other optimization techniques, such as using different blocking strategies (adapted to the expressivity of the ontology) to guarantee the termination of the reasoning [BS16a], or using some reasoning rules for some common special cases (such as n-ary conjunctions).

Finally, it is common to solve a reasoning task on fuzzy ontologies by reducing it to solving another task. However, developing a specific reasoning algorithm is often more efficient. For example, we can mention a specific algorithm for the classification problem [Str13].

In Section 4.1, we also proposed specific algorithms for the realization and instance retrieval problems. In this section, we provide a novel reasoning algorithm to solve the instance retrieval problem. The main difference is that we can reuse a classical ontology reasoner, but imposing some restrictions on the language (for example, we do not support fuzzy role assertions). Reusing classical reasoners is interesting because existing fuzzy ontology reasoners have limitations: most of them cannot completely support Fuzzy OWL 2 (e.g., fuzzyDL [BS16a]) and the only current exception, DeLorean, implements a non-scalable algorithm [BDGRS12].

4.3 Similarity between individuals

Motivation

Apart from the standard inference services enumerated in Section 2.3.4, many applications require alternative reasoning tasks. In particular, determining the similarity between individuals is a relevant operation in many intelligent applications such as knowledge-intensive case-based reasoning, clustering, or information retrieval [SOGP16]. Unfortunately, most of the research on similarity in fuzzy ontologies has been restricted to the similarity between fuzzy concepts, but the similarity between individuals of a fuzzy ontology has not received enough attention.

Contributions

Our main contribution is an algorithm (Algorithm 7) to compute the degree of similarity between two individuals from a fuzzy ontology based on the values of several fuzzy functional data properties, where the values of the data properties are described using fuzzy sets. We will assume that an individual i is characterized by the values of n fuzzy data properties p_i, i.e., $i = \langle p_1, p_2, \ldots, p_n \rangle$.

The first part (Lines 2–3) retrieves all the ontology data properties and initializes an empty list of degrees. The next part is computing the similarity between two individuals (Lines 5–18). Lines 6–7 get the value of a particular data property for the respective individuals, represented by means of fuzzy membership functions. Line 8 ensures that the data property is defined for both individuals. A key part of our algorithm is the similarity between the values v_{j1}, v_{j2} of a data property p_j for two different individuals i_1, i_2 (Line 10). We can think of different ways to compute its similarity $sim(v_{j1}, v_{j2})$. For example, we can compute the intersection of the two fuzzy sets and then defuzzify it into a single value, i.e., given the intersection set $d_{j1} \cap d_{j2}$ characterized by its membership function $\mu_{d_{j1} \cap d_{j2}}(x) = \mu_{d_{j1}}(x) \otimes \mu_{d_{j2}}(x)$, where $\mu_d(x)$ denotes the membership degree of x to the fuzzy set d (the value of a functional data property), we can compute:

$$sim(d_{j1}, d_{j2}) = \mu_{d_{j1} \cap d_{j2}}\Big(defuzzify(d_{j1} \cap d_{j2})\Big) \tag{4.5}$$

where $\otimes$ and $defuzzify$ are a t-norm operator [KMP00] and a defuzzification function [LK99], respectively. For example, $\otimes$ can be the minimum and def can be the mean of maxima MOM.

Example 30. *Figure 4.2 shows the intersection between two fuzzy datatypes d_1 and d_2 described using two triangular functions. For example, the output of method MOM*

is 2.29 *and* $\mu_{d_{j1} \cap d_{j2}}(2.29) = 0.791$*, while the output of the method COG is* 2.39 *and* $\mu_{d_{j1} \cap d_{j2}}(2.39) = 0.716$*.* □

Algorithm 7 Algorithm to compute the similarity between individuals.

Input: A fuzzy ontology $\mathcal{O}$
Input: Two individuals i_1 and i_2
Input: A tolerance $\epsilon > 0$
Output: A *degree* of similarity between two individuals

```
1: // Initialization
2: P ← Retrieve all data properties p_j of O
3: D ← ∅
4: // Similarity between two individuals
5: for all p_j ∈ P do
6:    d_j1 ← Retrieve the value of data property p_j for i_1 in O
7:    d_j2 ← Retrieve the value of data property p_j for i_2 in O
8:    if d_j1 ≠ NULL and d_j2 ≠ NULL then
9:       // Similarity between two data property values
10:      sim ← μ_{d_j1 ∩ d_j2}(defuzzify(d_j1 ∩ d_j2))
11:      // Aggregation of the new similarity
12:      if sim = 0 then
13:         D ← D ∪ ϵ
14:      else
15:         D ← D ∪ sim
16:      end if
17:   end if
18: end for
19: degree ← @(D)
20: return degree
```

Lines 12–16 store the similarities between data properties $sim(d_{j1}, d_{j2})$ in a list D. To avoid the fact the a single 0 value in the similarities between two data properties would turn the whole result into 0, in practice, Line 13 replaces a 0 value with ϵ.

Finally, Line 19 computes the similarity of a pair of individuals as an aggregation of the similarities between pairs of data property values:

$$sim(i_1, i_2) = @_{j=1}^{n} sim(d_{j1}, d_{j2}) \tag{4.6}$$

where $@ : [0,1] \times [0,1] \to [0,1]$ is a combination function (e.g., product t-norm or an aggregation operator). Finally, Line 20 returns the similarity degree between the two individuals.

Example 31. *Assume that there are two individuals,* ind1 *with* step1 *and* ind2 *with* step2*, and let us compute the similarity based on two properties* height *and* lengthStep*. Assume that the similarity for* height *(using MOM) is* = 0.791 *(see Figure 4.2, where*

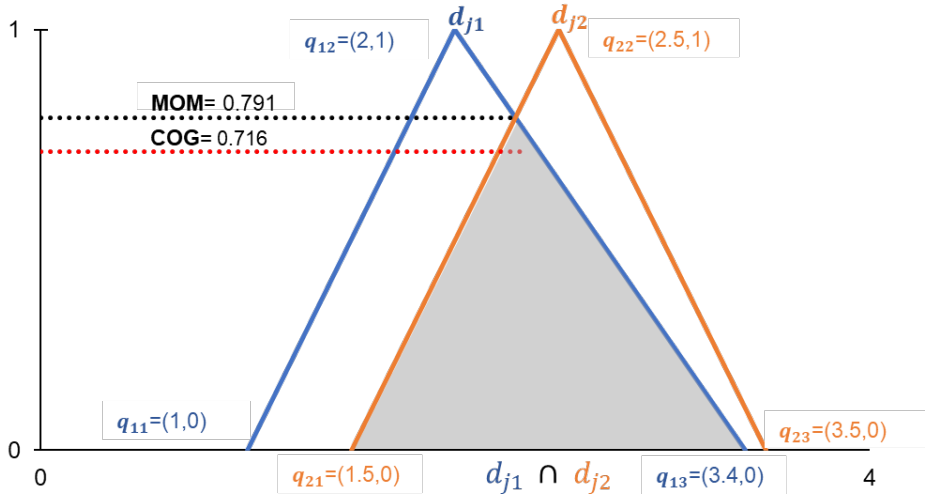


Figure 4.2: Similarity between two fuzzy sets computed using the defuzzification of the intersection.

the value for ind1 *is shown in blue, and the value for* ind2 *is shown in orange) and the similarity for* lengthStep *is* 0.8. *Those degrees can be aggregated using product, so the global result is* $@(0.791, 0.8) = 0.791 \cdot 0.8 = 0.633$. □

Now, we will add some interesting remarks:

- Note that if @ is not associative, it is possible to store all values computed in the for loop and to aggregate all of them after the loop.
- Furthermore, we only take into account those fuzzy data properties which are defined for both individuals, but it would be possible to state that if some property is undefined for some individual, the similarity corresponding to that fuzzy data property is amost zero ($sim(d_{j1}, d_{j2}) = \epsilon$).
- If the values of some data property are crisp values, the intersection is either an empty set or the crisp value; so the local similarity would be in $\{0, 1\}$. The algorithm could also be adapted to use similarity measures $sim(d_{j1}, d_{j2})$ between crisp values.

Finally, let us anticipate that this approach to compute the similarity between individuals has been evaluated on a real use case: gait recognition (Section 6.1).

Related work

In this section we describe some previous approaches focusing on the similarity between fuzzy ontology elements. Firstly, we will discuss other previous approaches to compute the similarity between individuals. Then, we will overview the similarity between concepts (although it is not directly related to our work).

Similarity between individuals. Armengol et al. studied similarities between individuals in fuzzy ontologies using a non-geometrical interpretation [ADG16]. They proposed to use a Similarity Box (SBox) in fuzzy DL languages, including axioms to represent the properties of the so-called fuzzy abstract properties that encode similarities between pairs of individuals. The authors mention that an SBox could also represent similarities of two objects with respect to their attributes, but without representing this information using ontology axioms. A local similarity is computed for each property, comparing the values of the property for the two compared individuals. As an example, a formula to compute the local similarities between two numerical values is shown. Then, global similarities are computed as an aggregation using a t-norm of the local similarities. The authors also mention that other aggregation operators could be used, such as uninorms or OWA. Instead, our work is able to compute local similarities between individuals when the values of the data properties are fuzzy datatypes rather than numerical values. Furthermore, our global similarity is more general as it supports any aggregation operator. Finally, we discuss the case where two individuals do not have the same properties.

Bobillo et al. proposed fuzzy similarity relationships between individuals in [BS12] to build fuzzy rough ontologies. In our work the aim is to calculate the similarity for individuals, while in that work, from the existing similarity relation, upper and lower approximations of rough concepts are computed.

Similarity between concepts. Bahri et al. considered four types of similarity relations between pairs of fuzzy concepts, namely *MoreGeneral*, *LessGeneral*, *Equivalent*, and *Disjoint* [BBG07]. The authors propose some formulae to compute the similarity degree between two concepts, based on the conjunction of two fuzzy sets. The authors propose to compute the conjunction degree using a possibility-based approach, and study the particular case of fuzzy attribute-based and fuzzy number restriction-based concepts. Their approach is restricted to Zadeh fuzzy logic.

Other works consider similarity between elements of different ontologies. Ying et al. proposed a similarity function between a pair of fuzzy concepts from different fuzzy ontologies [YRbJb09]. The global similarity measure is a weighted sum combining three local similarities: linguistic similarity (combining the edit distance between the label strings and the similarity between Wordnet synsets), fuzzy set similarity (using a sup-min set operation), and context similarity (called facet similarity by the authors, based on the common ancestors, successors, instances, and properties). It is worth to note that the authors categorize fuzzy concepts into three types, namely "fuzzy set concepts", "fuzzy linguistic terms", and "fuzzy numbers and linguistic qualifiers", but

fuzzy linguistic terms, fuzzy numbers, and fuzzy linguistic qualifiers are actually fuzzy datatypes rather than fuzzy concepts. Our approach to compute the global similarity is more general, as we do not restrict to a weighted sum. Furthermore, local similarities are computed in a different way.

Chandran and Crockett follow a very different path: rather than computing the similarity between fuzzy ontology elements, they use fuzzy ontologies to improve semantic similarity measures, as a fuzzy ontology makes it possible to compute the relatedness between pairs of concepts representing fuzzy words [CC16]. In particular, the authors study which structure is more appropriate for the fuzzy semantic similarity measure FAST.

Finally, it is worth to mention that the similarity between concepts has been used to define the semantics of threshold concepts [BG17]. Although they are crisp concepts used in crisp ontologies, the idea is very similar to that of fuzzy concepts: threshold concepts require that every individual belongs to a concept with a degree $\bowtie \alpha$, with $\bowtie \in \{<, \leq, \geq, >\}$ and $\alpha \in [0, 1]$.

4.4 Matchmaking between individuals

In the business world, transactions are done between several parts (typically, a buyer and a seller). Sometimes, an intermediary is used to support and facilitate the transactions (in particular, to achieve an agreement) [OBT04]. This intermediary uses information about the interests and preferences of the parts and computes an appropriate *match* of the alternatives, providing a compromise solution that can be acceptable for both parts. The process can be more complex, as parts can follow the negotiation by proposing more offers and counter-offers.

We are interested in computing a fair agreement between several parts, taking their preferences into account, without human intervention. To do so, a software agent would be responsible of computing an agreement between all parts. A possible way to so is representing the information using ontologies and formulating the problem as an ontology reasoning task. Furthermore, because imprecise or vague knowledge naturally appears in many real-world domains, fuzzy ontologies seem even more promising.

In some classical scenarios, such as blockchain smart contracts, either an agreement is found or not. However, sometimes one cannot find a solution that completely satisfies both parts, but it is often possible to find a partial agreement, where the terms are partially fulfilled in such a way that they are acceptable for everybody. Using fuzzy ontologies, a fuzzy matchmaking makes it possible to compute an optimal common satisfiability degree so that all involved parts are partially satisfied.

Contributions

Our contribution is a novel strategy (Algorithm 8) to compute a fuzzy matchmaking (a partial agreement) between individuals from a fuzzy ontology. Individuals represent the different parts that want to reach an agreement. For the sake of clarity, the algorithm is restricted to two individuals, but the generalization to more individuals is trivial. Our approach guarantees Pareto optimality of the solutions and is based on the computation of the Best Satisfiability Degree (BSD) of the combinations of the restrictions of each individual.

Algorithm 8 Algorithm to compute an agreement between two individuals.

Input: A fuzzy ontology $\mathcal{O}$
Input: Two individuals i_1 and i_2
Input: A Boolean parameter *and*
Input: An optional vector of weights $\mathbf{W}$
Output: A *degree* of agreement between two individuals
Output: A model of $\mathcal{O}$

1: // Initialization
2: $P \leftarrow$ Retrieve all data properties p_j of O
3: $C_1 \leftarrow \top$
4: $C_2 \leftarrow \top$
5: // Create local combinations of the restrictions
6: **for all** $p_j \in P$ **do**
7: $\quad d_{j1} \leftarrow$ Retrieve the value of data property p_j for i_1 in O
8: $\quad d_{j2} \leftarrow$ Retrieve the value of data property p_j for i_2 in O
9: $\quad$ **if** $d_{j1} \neq$ NULL **and** $d_{j2} \neq$ NULL **then**
10: $\quad\quad C_1 \leftarrow C_1 \sqcap \exists p_j.d_{j1}$
11: $\quad\quad C_2 \leftarrow C_2 \sqcap \exists p_j.d_{j2}$
12: $\quad$ **end if**
13: **end for**
14: // Compute the global satisfaction degree
15: **if** *and* = TRUE **then**
16: $\quad$ **return** $\langle bsd(\mathcal{O}, C_1 \sqcap C_2)$, model of $\mathcal{O}\rangle$
17: **else**
18: $\quad$ **return** $\langle bsd(\mathcal{O}, @_{\mathbf{W}}(C_1, C_2))$, model of $\mathcal{O}\rangle$
19: **end if**

The algorithm requires a common ontology $\mathcal{O}$, the individuals to be considered, a Boolean parameter determining if the user wants to combine the values using a conjunction or using an aggregation operator, and an optional vector of weights.

The algorithm is divided into three parts: an initialization (Lines 2–4), a local combination of the restrictions of each individual (Lines 6–13), and a global combination of all restrictions (Lines 15–19).

During the initialization, the ontology is loaded, a set of data properties is retrieved,

and two concepts are initialized to build inductively a combination of the restrictions of each individual.

Next, for each data property such that its value is known for both individuals, we retrieve its value, described using a fuzzy datatype. For each individual, all its restrictions of the form $\exists p.\mathbf{d}$ are combined locally, forming two concepts C_1 and C_2 (Lines 10–11). The following is an example of a concept definition:

$$C_1 \equiv \exists T_1.\mathbf{d}_{11} \sqcap \exists T_2.\mathbf{d}_{12} \sqcap \cdots \sqcap \exists T_m.\mathbf{d}_{1m} \tag{4.7}$$

Next, we compute the BSD of a global combination of the concepts C_1 and C_2 encoding the restrictions of each individual. According to the Boolean parameter, the combination can use a t-norm or an aggregation operator with respect to the input vector of weights.

Example 32. *Let us assume a car sale with a pair of agents (*Main *part, or seller, and* Secondary *part, or buyer) being involved. A buyer is looking for a car on a mobile app, s/he wants to pay less than 165 Ethers*[3] *but if there is a good car, s/he can afford to pay 185. S/he is not sure about speed of the car, but the app suggests around 240 km/h (at least 180 km/h and no more than 320 km/h). S/he needs the car in a range of days from 10 to 30. In a fuzzy ontology, this can be encoded as follows:*

$$\begin{aligned} \mathsf{Secondary} \;=\; & \exists \mathsf{unitPrice}.\mathbf{left}(0, 200, 165, 185) \\ & \sqcap \exists \mathsf{speed}.\mathbf{tri}(0, 500, 180, 240, 320) \\ & \sqcap \exists \mathsf{deliveryTime}.\mathbf{left}(0, 30, 10, 30) \end{aligned}$$

A seller posts in the app a new car with a negotiable price between 165 and 169 Ethers, and a maximum speed of 250 km/h. The seller proposes a delivery time of 14 days but might do it in 7 days. This can be encoded as:

$$\begin{aligned} \mathsf{Main} \;=\; & \exists \mathsf{unitPrice}.\mathbf{right}(0, 200, 165, 169) \\ & \sqcap (= \mathsf{speed}\ 250) \\ & \sqcap \exists \mathsf{deliveryTime}.\mathbf{right}(0, 30, 7, 14) \end{aligned}$$

In summary, there are two parts involved and three attributes. The Main *part has two soft restrictions (price and delivery time) and a hard restriction (speed of the car). The* Secondary *part has three soft restrictions.*

Using Łukasiewicz t-norm to compute the BSD we have:

$$bsd(\mathcal{O}, Main \sqcap_{\mathbf{L}} Secondary) = 0.475$$

An excerpt of the ontology model providing the optimal solution is the following:

[3] A cryptocurrency used in Ethereum network, as we will see in Section 6.3.

– *a* $\mathsf{unitPrice} = 169$*, that satisfies the buyer with degree* 0.8 *and the seller with degree* 1 *(Figure 4.3 (a)),*

– *a* $\mathsf{speed} = 250$*, that satisfies the buyer with degree* 0.875 *and the seller with degree* 1 *(Figure 4.3 (b)), and*

– *a* $\mathsf{deliveryTime} = 14$*, that satisfies the buyer with degree* 0.8 *and the seller with degree* 1 *(Figure 4.3 (c)).*

This model leads to the following satisfaction degrees:

– *the buyer is satisfied with degree* $0.8 \otimes 0.875 \otimes 0.8 = \max\{0.8+0.875+0.8-2, 0\} = 0.475$*,*

– *the seller is fully satisfied, with degree* $1 \otimes 1 \otimes 1 = \max\{1+1+1-2, 0\} = 1$*, so*

– *the BSD is* $0.475 \otimes 1 = \max\{0.475 + 1 - 1, 0\} = 0.475$*.* □

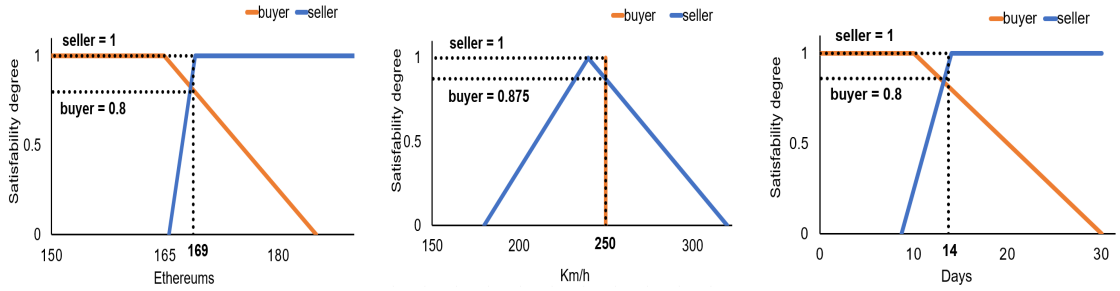


Figure 4.3: Partial agreements on (a) unit price, (b) speed, and (c) delivery time.

Example 33. *The agreement in Example 32 might be seen as a little bit unfair as one part is very satisfied and the other is not. To solve it, we can use Strict Weighted Sum to aggregate the constraints of the customer and the seller. Now, we get*

$$bsd(\mathcal{O}, @^{SWS}_{[0.25,0.75]}(\mathsf{Main}, \mathsf{Secondary}) = 0.6134$$

An excerpt of a possible ontology model is:

– *a* $\mathsf{unitPrice} = 169$*, that satisfies the buyer with degree* 0.8 *and the seller with degree* 1*,*

– *a* $\mathsf{speed} = 250$*, that satisfies the buyer with degree* 0.875 *and the seller with degree* 1*, and*

– *a* $\mathsf{deliveryTime} = 10$*, that satisfies the buyer with degree* 1 *and the seller with degree* 0.4286*.*

We can check that:

– *the buyer is satisfied with degree* $0.8 \otimes 0.875 \otimes 1 = 0.675$*,*

– *the seller is satisfied with degree* $1 \otimes 1 \otimes 0.4286 = 0.4286$*, so*

– *the global satisfaction is* $0.75 \cdot 0.675 + 0.25 \cdot 0.4286 = 0.6134$*.* □

Let us now discuss some important aspects of the algorithm:

– It is important to check (Line 9) that all compared individuals define the same attributes (using concrete existential restrictions). For example, assume that a buyer defines that he wants a pink product, but a seller does not define the color. Because of the Open World Assumption, the result could be satisfiable (there could be a model of the ontology satisfying it), but the seller has not confirmed that he has actually that product in store. This restriction should also affects other approaches to compare two fuzzy concepts, e.g., [RSB+08].

– fuzzyDL reasoner makes it possible to obtain not only the result of the BSD but also a (possibly not unique) model of the ontology built during the computation of the BSD.

– The restrictions of each individual are combined using a conjunction (Lines 10–11). We propose to use Łukasiewicz t-norm because it is Pareto optimal, and supported by fuzzyDL. More generally, these restrictions could be combined using an associative function $f_1 : [0,1]^m \to [0,1]$, where m is the number of data properties that are considered in the agreement. Note in particular that it could be possible not to use a t-norm, e.g., to modify the algorithm such that

$$C_1 \equiv @(\exists T_1.\mathbf{d}_{11}, \exists T_2.\mathbf{d}_{12}, \ldots, \exists T_m.\mathbf{d}_{1m})$$

for some associative aggregation operator @, such as the SWS.

– The global combination (Lines 15–19) could be computed using a function $f_2 : [0,1]^n \to [0,1]$, where n is the number of involved individuals. We propose to use Łukasiewicz t-norm or the Strict Weighted Sum because they are Pareto optimal, as we will see later, and supported by fuzzyDL.

Pareto optimality. An agreement is Pareto optimal if it is not possible to improve the satisfaction degree of one part without lowering the satisfaction degree of the opponent's one. That is, if the satisfaction degrees of the two parts are α and β, and the common satisfaction degree $\gamma = \alpha \otimes \beta > 0$ is optimal, there cannot be another $\alpha' > \alpha$ such that $\gamma = \alpha' \otimes \beta$.

Note that the BSD gets the maximum value over all models, so in general it does not provide a Pareto optimal solution, as the following example shows.

Example 34. *If there is a solution S_1 (a model of the fuzzy ontology) where the satisfaction degree of the seller is* 0.8 *and the satisfaction degree of the customer is* 0.6*, using Gödel t-norm the common satisfaction degree is* $\min\{0.8, 0.6\} = 0.6$*. However, there could be another solution S_2 where the satisfaction degree of the seller is* 0.9 *and the satisfaction degree of the customer is* 0.6*, with a common satisfaction degree* $\min\{0.9, 0.6\} = 0.6$*. Although the mutual satisfaction degree is the same, the latter solution is preferable.* □

Pareto optimality of the solution was already known for Łukasiewicz fuzzy DLs, but we will generalize it here, including among others Product t-norm and Strict Weighted Sum.

Proposition 1 ([RSB+08])**.** *In Łukasiewicz fuzzy DLs, if the maximum of $\alpha \otimes \beta$, with $\langle \alpha, \beta \rangle \in [0,1] \times [0,1]$, is positive then the maxima are also Pareto optimal.*

Proposition 2. *Let $f : [0,1]^2 \to [0,1]$ be a strictly increasing aggregation function having* 0 *as an absorbing element. If the maxima $\max_{\alpha,\beta \in [0,1]^2} f(\alpha, \beta) > 0$, then the maxima are also Pareto optimal.*

Proof. Firstly note that $f(\alpha, \beta) > 0$ implies $\beta > 0$, as 0 is an absorbing element. Now let us assume that there is a solution $\langle \alpha', \beta \rangle$ with $\alpha' > \alpha$. Since f is strictly increasing, $f(\alpha', \beta) > f(\alpha, \beta)$ follows, which contradicts the premise that $f(\alpha, \beta)$ was a maximum. □

Corollary 1. *If the maxima $\max_{\alpha,\beta \in [0,1]^2} f(\alpha, \beta) > 0$, then the maxima are also Pareto optimal in the following cases:*

- *if f is Product t-norm,*
- *if f is a strict t-norm (isomorphic to the Product), or*
- *if f is a Strict Weighted Sum.*

Note in particular that this not hold for the usual weighted sum, where it would be possible to have one the parts completely unsatisfied, but a positive aggregated value.

Łukasiewicz t-norm is supported by fuzzyDL reasoner. However, it is nilpotent and easily collapses to zero when aggregating several values. Therefore, in practice, it should be used when satisfaction degrees of the attributes are high or when the number of attributes is low.

Proposition 2 can be easily generalized to n involved parts, and to use two functions f_1 (to combine the constraints of each of the parts) and f_2 (to combine the local satisfaction degrees). For example, f_1 can be Łukasiewicz t-norm and f_2 can be the strict weighted sum.

Proposition 3. *Let* $f_1 : [0,1]^m \to [0,1]$ *and* $f_2 : [0,1]^n \to [0,1]$ *be strictly increasing aggregation functions having* 0 *as an absorbing element. If the maxima* $\max_{\alpha,\beta\in[0,1]^2} f_2\Big(f_1(x_{11}, x_{12}, \ldots, x_{1m}), f_1(x_{21}, x_{22}, \ldots, x_{2m}), \ \ldots, \ f_1(x_{n1}, x_{n2}, \ldots, x_{nm})\Big) > 0$*, then the maxima are also Pareto optimal.*

This approach to compute the similarity between concepts has been evaluated on a real use case: blockchain smart contracts (Section 6.3).

Related Work

In this section, we will discuss some previous work on matchmaking in fuzzy ontologies.

There are many works studying how to represent the restrictions in matchmaking scenarios and how to compute an agreement. For example, Ouksel et al. analyzed matchmaking scenarios where a seller is matched to a single buyer, and evaluate empirically different arbitration protocols (conflict resolution mechanisms) such as maximal (non-weighted) sum protocol, Nash function, preference arbitration, deterministic compensational arbitration (where positive weights are assigned to the parts), and probabilistic compensated arbitration [OBT04]. The authors also consider Pareto optimality of the solutions. However, weights are assigned (in some protocols) to the parts, while we associate them to the individual criteria (data properties). Furthermore, this approach does not consider background knowledge (ontologies) nor imprecision (fuzzy logic). Our arbitration protocol is thus very different.

Other works use ontologies to represent matchmaking scenarios:

– For example, Grimm and Hitzler proposed two methods (based on autoepistemic logic and on circumscription) to perform local close-world reasoning with OWL ontologies [GH07]. The authors show that semantic matchmaking can be computed using standard DL inference services such as concept satisfiability

or entailment. Instead, our model considers fuzzy ontologies and Open World Assumption, and we use a different reasoning task (Best Satisfiability Degree).

– Semantic matchmaking can also be understood as finding the best resources for a given request, with both resources and requests described as complex concepts and there is a background ontology. This has been explored in a series of papers using MiniME reasoner [SRL+14] to solve standard reasoning tasks (subsumption, satisfiability, or classification) but also non-standard ones (abduction and contraction). Some problems that have been formulated this way are computing Points of Interest in augmented reality explorers [SRL+14], possibly from mobile devices, Kinect-based posture and gesture recognition [RSdS+14], or finding the most suitable sensors in cooperative semantic sensor networks from the Semantic Web of Things [RSP+19].

To the best of our knowledge, only a few works incorporate management of imprecise knowledge.

– Ragone et al. proposed the use of fuzzy Description Logics to automate matchmaking in e-marketplaces and support imprecise preferences [RSB+08]. Numerical restrictions can be hard or soft (flexible). Their approach is also restricted to two parts (a buyer and a seller). Computing the best partial agreements between two parts consists in maximizing the degree of satisfaction of the conjunction (using Łukasiewicz logic) between all restrictions, which can be solved computing the Best Satisfiability Degree.

– Fuentemilla used a similar approach, using the Best Satisfiability Degree to find partial agreements in blockchain scenarios [Fue19]. His approach is restricted to two numerical restrictions. Instead, we accept a finite number of numerical restrictions rather than only two.

– Finally, Ruta et al. use MiniME to solve matchmaking queries in fuzzy $\mathcal{ALN}(\mathbf{D})$ ontologies, where the only fuzzy elements are fuzzy datatypes, and discuss the application to fire risk detection [RSS10]. On the contrary, our approach supports Fuzzy OWL 2 ontologies.

The main differences with these approaches are that we support more than two parts, take into account the Open World Assumption, and support more general operators than Łukasiewicz t-norm. Furthermore, these approaches require to encode the restrictions using fuzzy concepts, whereas we accept individuals as the input of the algorithm.

Chapter 5

Contributions to the support of fuzzy ontologies on mobile devices

People has increased the use of mobile phones with respect to desktop devices and mobile devices changed our lifestyle and the business logic, providing new opportunities to create, acquire, process, and share knowledge. In this chapter, we will contribute to the management of vague knowledge in mobile devices by means of fuzzy ontologies, proving reasoning services and developing native apps for mobile devices.

We start by mentioning some techniques that are useful for any ontology-based system regardless of the hardware device, but that turn to be particularly useful on mobile devices because of their limitations. Specifically, Section 5.1 discusses the importance of optimizing reasoning and distributing knowledge bases. Then, we develop some real-world applications or tools supporting the different ways to reason with (fuzzy) ontologies, namely local, remote, and hybrid reasoning. On the one hand, Section 5.2 describes GimmeHop, a fuzzy ontology-based recommender system that supports both local and remote reasoning. On the other hand, Section 5.3 discusses the implementation of a new version of fuzzyDL reasoner that is serializable and incremental, thus promoting a hybrid reasoning strategy. Finally, Section 5.4 reports the adaptation of some of the fuzzy ontology learning tools developed in the previous chapter (Datil and Fudge) to work on mobile devices.

5.1 Transversal techniques

Motivation

Before addressing solutions specifically conceived for mobile devices, it is useful to recall some techniques that are useful to improve the performance of managing fuzzy ontologies on any hardware device, but that are be particularly useful when reasoning on mobile devices because of their limitations.

Contributions

In this section we recall some techniques for the optimization of the reasoning (Section 5.1.1) and discuss the distribution of the files (Section 5.1.2).

5.1.1 Optimization of the reasoning

It is clear that the last new mobile devices have good hardware specifications but they are not yet comparable with the computing power of personal computers. So, our contributions to optimize the reasoning discussed in Chapter 4 are particularly helpful on mobile devices, where optimizing the reasoning is even more important.

- Our *minimalist algorithm* to solve the flexible faceted instance retrieval (Section 4.2) is appropriate for fuzzy ontologies where the only fuzzy elements are fuzzy datatypes. It makes it possible to reuse classical ontology reasoners, so the fact that there are no implementations of fuzzy ontology reasoners for mobile devices is not a problem. As we will see in Section 6.2, our experiments with a native beer recommender app show that it is possible to use fuzzy ontologies with 3000 individuals in a low-powered mobile.

- Our *instance retrieval and realization algorithms* for fuzzy ontologies (proposed in Section 4.1) reduce the number of tests needed to solve a query. Our implementation extended fuzzyDL fuzzy ontology reasoner for desktop computers. Unfortunately, we can not evaluate the performance of those algorithms on mobiles because fuzzyDL does not have yet a mobile version, but a future mobile version would benefit from our optimized version. Another possibility is that a native fuzzy ontology reasoner could implement our algorithms.

5.1.2 Distributed ontology files

In this section, we examine the use of a distributed ontology architecture on any computer, which is particularly interesting on mobile devices.

Using a centralized model, with a unique ontology file, can be problematic in real applications, where large ontologies are often needed. We found examples where ontologies could not be loaded on a desktop computer (a gait recognition system, see Section 6.1), and examples where the loading time on a mobile device was very high or the app crashed (a beer recommender system, see Section 6.2).

In such cases, we propose to use a distributes architecture where ontologies are split. If there are a lot of individuals but a relatively small number of classes and properties,

we propose to use a main ontology with the *schema* (including classes, properties, TBox axioms, and RBox axioms) and several *individual ontologies* which import the schema and populate it with instances, their membership to classes, the values of their properties, their relationships, and other ABox axioms. In other scenarios with large numbers of classes and/or properties, one would have to split the schema, possibly using ontology modules [SPS09].

In future chapters, we will apply this distributed architecture to three real-world domains, namely gait recognition, beer recommendation, and blockchain-based e-commerce.

- In Section 6.1, we will present a gait recognition system, using one ontology with the schema and many ontologies populating the schema. Such ontologies include gait data (step sequences) and biometric features of a human person, and there is one ontology for each individual. The distributed architecture leads to a better performance of our classification algorithm.

- In Section 6.2, we will present a recommender system in the beer domain, using one ontology schema and many ontologies populating the schema with beer and brewery data. This way, bars, breweries, stores, and other actors involved in the beer industry can create their own personal ontologies populating the schema. Furthermore, this makes it easier to develop ontologies with different sizes to evaluate the performance of the system.

- In Section 6.3 we will propose an e-commerce scenario, based on the blockchain, where partial agreements are possible. The system uses 4 ontologies: an ontology schema, one ontology for each of the involved parts in the commercial transaction (e.g., buyer and seller), and a common ontology. The personal ontologies of the involved parts import the schema and populate it with their specific preferences. The common ontology contains all the information required for the transaction, including the values of the attributes that lead to an agreement. Using this distributed architecture, only the agreement and not the personal preferences are stored in the blockchain. It is important to mention that the architecture is flexible and there could be more than two parts involved in the commercial transaction.

The main advantages of this model of distributed ontologies are the following ones:

- It leads to smaller files, and promotes using different subsets of the data (for example, ontologies with different numbers of beers). This is a good option for large ontologies.

– It leads to a better distribution of the knowledge, splitting schema and factual data.

– For humans, it might be easier to read and understand the representation of files.

– It is independent of the domain.

– It promotes local reasoning to work offline. This is crucial when Internet access is limited.

– It has a reasonable performance on mobile devices. Our experiments show that ontologies with an important number of individuals can be loaded and the running time of the local reasoning is acceptable.

5.2 GimmeHop app: Beer recommender system

Motivation

The mobile development statistics in Android about *Shopping* and *Food&Drink* are eighth and ninth position respectively from 49 categories in the distribution of apps by category[1]. In the *Food&Drink* category, we can notice a bigger number of free apps (129127) than paid apps (597). In summary, there is a great consumer interest/demand in food and beverage industry.

Beer market is a hot topic which is receiving a notable attention in the last years: more foreign beers are imported by many stores and bars, the number of artisan beers is growing significantly, new beer recipes are being investigated, and users are willing to try new beer styles, thanks in part to the phenomenon of home-brewing. Thus, software tools providing users with good recommendations about beers seem very interesting.

The already mentioned increasing importance of mobile computing on our daily lives invites us to develop a recommender system for mobile devices. Furthermore, given the imprecise nature of several terms in the beer domain (such as low-alcohol beer or dark beer), it seems convenient to use fuzzy logic as part of our solution.

As already mentioned in Section 2.5, semantic reasoning on mobile devices can be performed in local, remote, and hybrid ways. Hybrid reasoning is not an option yet (Section 5.3 will discuss some advances in that direction), but it seems crucial to develop apps that can support both local and remote reasoning strategies to compare their advantages and limitations in a practical scenario, evaluating with empirical data their feasibility and scalability.

[1] `https://42matters.com/stats`

Contributions

We have developed a beer app recommender system for Android platform called GimmeHop. The prototype has been developed in Java 1.8 using the IDE Android Studio 5.3.3. It is able to deal with user context (in particular, user location), user preferences, and supports both local and remote reasoning.

Query types. GimmeHop supports three types of queries or searches:

- *Basic search*: the input is the name (or part of the name) of a beer or a brewery, and the output is a list of beers or breweries matching syntactically.
- *Advanced search*: the input is a beer type, and optionally, values of some features, namely ABV (alcohol level) and/or IBU (bitterness). The output is an ordered list of beers together with the recommendation degrees (i.e., the satisfiability degree of the query). The list is decreasingly ordered by the recommendation degrees (see Section 6.2 for details). Moreover, the user is also able to select some user preferences (for example, which is the most important property for his/her). For this query type we implemented the minimalist reasoning solution in Algorithm 6. Recall that this algorithm assumes that the only fuzzy elements are fuzzy datatypes (e.g., LowABV).

  Note that the system also takes into account the rating of each beer, but this is not shown in the user interface because users are interested in items with the best possible rating.
- *Similarity search*: the input is a beer and the output is a ordered list of similar beers together with their similarity degrees.

The app supports two ways to solve the queries: *local* and *remote* reasoning. This involves using a local semantic reasoner or an external one, respectively. Local mode computes the reasoning in the mobile device, so Android versions of classical ontology reasoners are needed. Instead, the remote mode uses a client/server architecture where the server hosts an ontology reasoner to compute the solution.

Main features. The most relevant characteristics of GimmeHop app are:

- *It has an intuitive graphic user interface.* Figure 5.1 (a) shows the main view of the app, where the end user has two options, a basic search and an advanced search. Figure 5.1 (b) shows the form to submit an advanced search. The user must select the style. Optionally, s/he can set the degrees of alcohol and

bitterness, using linguistic labels rather than numerical values (it is also possible to select "Indifferent"). The user can also choose the most important property between ABV, IBU, style rating, or indifferent. We wanted to make the interface as simple as possible, a possible extension would be to ask for a complete ordering of the three properties in terms of importance. Figure 5.1 (c) illustrates the output of the advanced search with respect to the query specified in Figure 5.1 (b). We can see that the beers are decreasingly ordered and the numbers (the recommendation degrees) are colored to illustrate the importance of the recommendation (going from green to red as the quality of the recommendation decreases). Figure 5.2 (a) shows the output of a basic search, a list of beers relevant to the name "Ámbar" beer. The system retrieves all the coincidences with beer names in the ontology, without any numerical degree. If the user taps on any beer presented in a list of results, s/he navigates to another page displaying information about the beer (name, brewery, style, bitterness, alcohol, country, and rating). Figure 5.2 (b) illustrates this information when clicking on the "Ámbar especial" beer. If the user taps on the button "similar", a similarity search is performed. The result of the similarity search is a list of beers as shown for advanced search.

– *It supports two semantic reasoners.* We use classical semantic reasoners to discover implicit knowledge and then perform some fuzzy reasoning to solve the flexible queries. The app is integrated with Hermit 1.3.8 and TrOWL 1.5. HermiT is an adapted version for Android SO, summarized in [BYBM15]. TrOWL reasoner can be directly imported in Android projects, but some tasks are solved not supported by TrOWL and are solved using the OWL API (for example, retrieving the values of the data properties of each individual is supported by Hermit but not by TrOWL).

– *It runs as an Android service.* For long running operations such as loading a large ontology (fuzzy beer ontology is described in Section 6.2) classify it (using a classical reasoner), or computing the queries, we use the Service class[2]. This way, the app performance is better than using a local thread[3].

– It uses a group of *linguistic labels* rather than numbers in the GUI. That helps the end user to choose the preferences for each item. For example, in a hot day users typically prefer to select a Lager style beer with a Low degree of alcohol.

[2]`https://developer.android.com/guide/components/services`
[3]`https://www.geeksforgeeks.org/services-in-android-with-example`

– *It uses user context (location).* In general, context can be "any information that can be used to characterize the situation of an entity" [Dey01]. In our case, we consider the location of the user, so GimmeHop can be seen as a location-based service. The app obtains the location (country) to give a higher score to local beers. It gets the location via network or GPS using Android's Network Provider, which is faster and cheaper (in battery consumption) than GPS Provider[4]. The app uses the country code location (e.g., ES for Spain) rather than the GPS position because it does not require a lot of accuracy.

– *It works with user preferences.* Users can select the relative importance of the different criteria, the reasoning mode, the reasoner, the fuzzy quantifier type, and its parameters. For example, some preferences are depicted in Figure 5.2 (c), where the user selects Hermit reasoner and configure the type of fuzzy quantifier and its parameters. The complete list of user settings is:

  – Reasoners: TrOWL and HermiT.

  – Quantifiers: right-shoulder (Figure 2.3 (d)), power function (Figure 2.6 (a)) and linear (Figure 2.6 (b)).

  – Value parameters for the quantifiers: possible values are shown in Table 5.1.

  The default values are TrOWL reasoner and right-shoulder quantifier with the parameters $q_1 = 0.3$ and $q_2 = 0.8$.

  In the future, we will consider allowing to configure the maximum number of beers (200 so far).

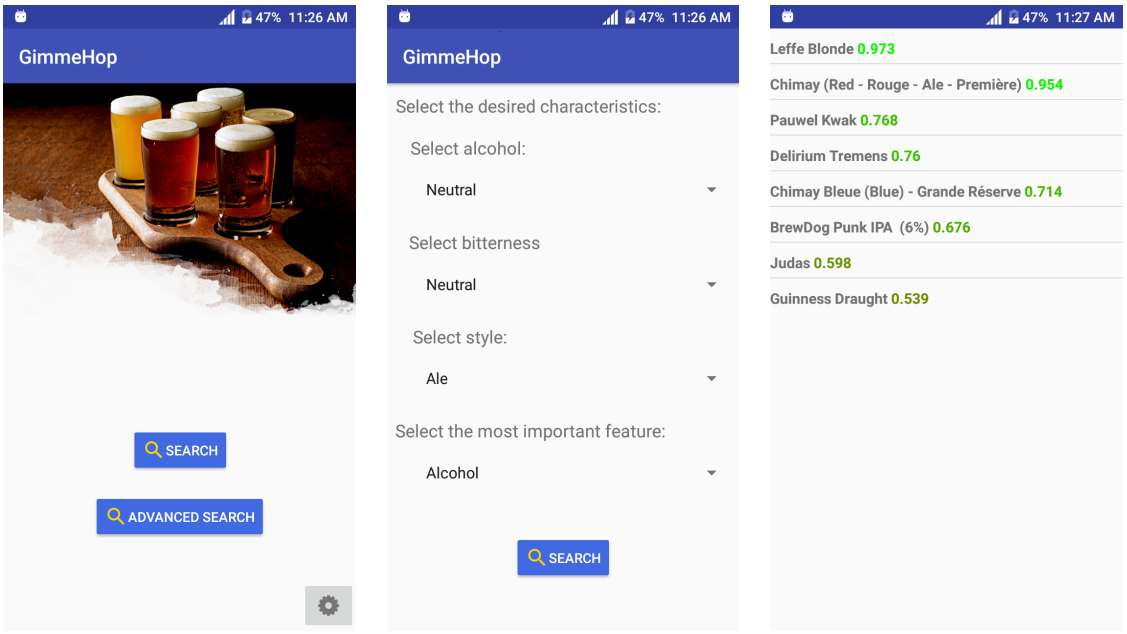
– *It manages weights internally.* To aggregate the information, the user does not need to specify the numerical values of the weights. If the user selects a feature as the most important one, the system automatically gives a higher weight to this feature. Otherwise, the user can just choose a fuzzy quantifier instead of the weight values.

– *It supports incomplete knowledge.* To deal with missing data, GimmeHop is able to adjust the weights in a transparent to the user way.

Example 35. *Given a right-shoulder fuzzy quantifier with* $q_1 = 0.3$ *and* $q_2 = 0.8$*, if the number of available features for a beer is* $n = 3$*, the vector of weights is* $[0.067, 0.667, 0.267]$*, and if* $n = 2$*, the vector of weights is* $[0.4, 0.6]$. □

[4] `https://developer.android.com/training/location/retrieve-current#java`

– *It respects the Android security measures using app permissions.* GimmeHop needs to save files, Internet access (only to visualize beer images), and location access, which requires some specific permissions, namely READ_EXTERNAL_STORAGE, INTERNET, and ACCESS_FINE_LOCATION. Sometimes, it is necessary to manually allow the permission on the mobile device.



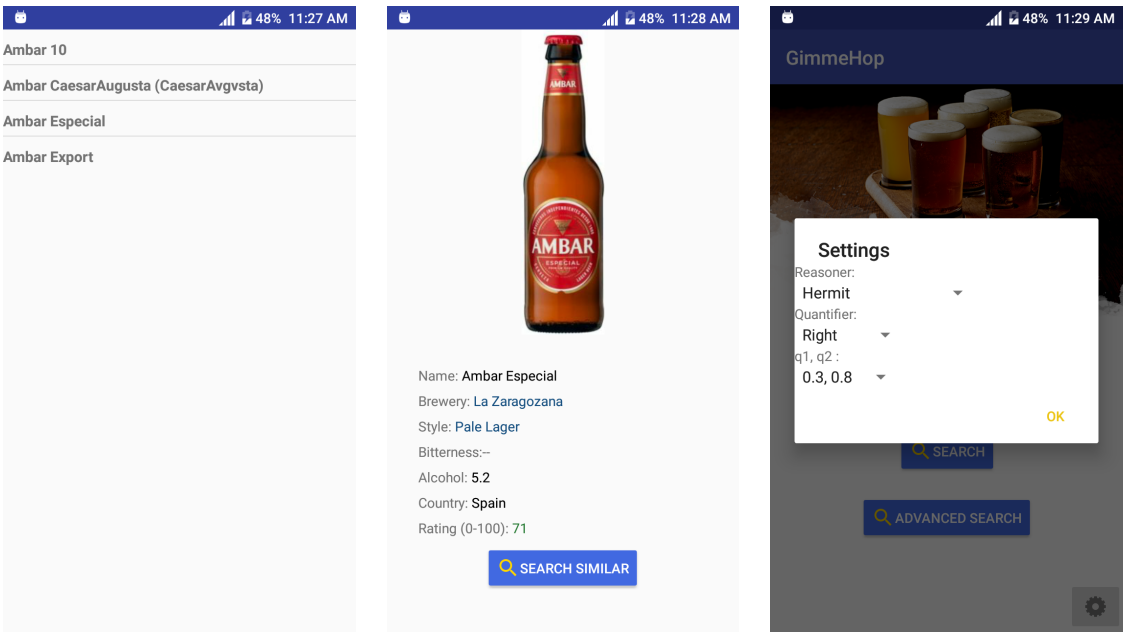
a) b) c)

Figure 5.1: Some screenshots of GimmeHop: (a) initial page, (b) advanced search form, and (c) result of an advanced search.

Right		**Power**	**Linear**
q_1	q_2	q	q_3
0.2	0.7	0.5	0.5
0.3	0.8	1.1	1.1
0.31	0.9	1.5	1.5

Table 5.1: Definitions of the fuzzy quantifiers.

Local reasoning. To perform local reasoning, GimmeHop uses Android versions of HermiT and TrOWL. To solve advanced searches, the app implements our Algorithm 6 for the flexible faceted instance retrieval problem given a fuzzy ontology. In particular, the inputs of the algorithm are:



a) b) c)

Figure 5.2: Snapshots of GimmeHop: (a) result of a basic search, (b) detailed information about a beer, and (c) settings

- A Fuzzy Beer ontology $\mathcal{O}$ (see Section 6.2.1 for details).
- A concept C representing a beer type.
- Three data properties p_n: ABV, IBU, and style rating.
- Three fuzzy datatypes D_n related to the data properties. The values of ABV and IBU are defined using Fuzzy OWL 2 datatypes. The rating is a normalization to $[0, 1]$ of the original percentage (e.g., $RatingDegree(70) = 0.7$).
- The combination function f_c is OWA (Eq. 2.4) or WMEAN (Eq. 2.2).
- The fuzzy hedge f_h is the increasing modifier $f_{few} = \sqrt{x}$ (Section 2.1.5).

To solve similarity searches, a specific beer is compared each other beer in the system. The local similarities between the styles, the values of the ABV, and the values of the IBU are computed and aggregated using an average.

Finally, to address location-based reasoning, we use a fuzzy hedge f_h to increase the recommendation degree if the country of the beer is the same as the user location.

Remote reasoning. Now we will describe the *remote* mode. As it is well known, the client/server architecture is an usual solution on Internet, and semantic reasoning is not an exception.

We developed on the client side a native Android app with an intuitive user interface (including location-based capabilities). The app is able to send request to answer queries and the GUI can show the responses processed by a third part.

The server is an instance of Amazon Web Service (AWS), where the reasoning algorithm to compute the flexible faceted instance retrieval is implemented. Classical semantic reasoners (Hermit and TrOWL) are running here to support the reasoning task. The server computes and filters the top–k results, and sends them to the client side. The base of the communication is the TCP protocol and a socket. All services are codified on the server, and all the project is developed on Java language. A fuzzy ontology is loaded and classified when the server is started. It is also possible to choose the reasoner when the server is loaded.

We found some advantages and disadvantages about the remote mode. The main advantages are the following ones:

- All the reasoning and query execution are computed on the server side.
- It offers high speed in the query answering system, with powerful resources such as memory and processing.
- It supports a multi-threaded mode to solve requests in parallel.
- It is able to work with large fuzzy ontologies. For example, in our experiments in Section 6.2 we used 15317 individuals.

But the remote mode also has some disadvantages:

- The free version of the AWS has limitations in the use of the hardware. Once such limits are reached, it is necessary to pay for a better service. Therefore, app developers need to evaluate if the management cost is convenient and affordable.
- General problems of Internet applications such as security, connection, or network quality.

In Section 6.2 we will present the results of our evaluation of GimmeHop, including the query answering time, the quality of queries, data traffic, and a comparison of local and remote reasoning.

Related work

In this section we overview some related work in two directions: semantic reasoning (inference engines) for mobile devices and mobile applications using web semantic technologies.

Semantic reasoners for mobile devices. There are several possibilities to do semantic reasoning on mobile devices [BBMP17]. The most direct way is to use an *external* solution, relying on servers on the cloud which would perform all the calculations. The main advantage is that one can consider a server which is as powerful as required by the application. However, in ubiquitous and mobile scenarios where context-awareness and privacy preserving play a crucial role, sending sensitive data to a remote server might be an important privacy breach, and even sending non-sensitive data might be dangerous as it could enable the inference of sensitive information. Furthermore, this requires assuming that the connectivity is fast and stable enough, but this is not often the case in mobile computing environments. It is also possible trying to use a *local* solution, which can be challenging because of the limitations of mobile devices in terms of CPU power, memory, or battery. Indeed, there is some evidence that reasoning time is only affordable in small or not very expressive ontologies [BYBM15]. For the sake of completeness, let us also mention that it is possible to develop *hybrid* strategies.

As already discussed in Section 2.5, there are two options to perform local semantic reasoning:

- To reuse or port previously existing reasoners. In particular, *Android Semantic* project made 9 reasoners available for Android devices [BYBM15] .

- To implement native reasoners for mobile devices. However, none of the existing native reasoners support the expressivity of our ontology (OWL 2 EL), so we needed to reuse existing reasoners not specifically designed for mobile devices.

Semantic apps. The roots of semantic apps can be traced back to knowledge mobilization (KMob), which consists of making knowledge available for real-time use in a form which is adapted to the context of use and to the needs and cognitive profile of the user [GR08]. One of the main requisites of KMob systems is that they should be ubiquitous and, in particular, accessible from mobile devices. Our system was indeed designed to meet some of the common requirements of KMob systems, such as being proactive, integrative, context-aware, declarative, concise, extensible, and easily maintainable.

Many of the so-called semantic applications do not actually use a semantic reasoner. For instance, Punya is an open source, web-based platform based on MIT App Inventor that helps to build mobile apps using Linked Data but there is no support for ontology reasoners [PWS+21].

Early examples of semantic application for mobile systems were very different to the more recent examples, and they typically relied on external servers. For example, *PDA2* system supports the diagnosis of psychological disorders on PDAs [DGRMPP05]. PDA2 uses an OWL 2 ontology to represent the relevant knowledge and accesses a semantic reasoner stored on an external server.

More recent examples using local semantic reasoners include location-based services providers (e.g., *Sherlock* can infer that both a cab and a tram are interesting for a certain mobile user, given the information obtained from sensors on his/her device such as the location and time [YMII14]), health apps (e.g., *Rafiki* infers possible diseases for a patient given his/her symptoms and context [PYJF14]), privacy control applications (e.g., *Faceblock* infers whether information about a user should be shared with other people or applications according to his/her context [YPD+14]), and navigation applications (e.g., *MAR* is a mobile augmented reality explorer to discover points of interest [SRL+14, RSG+19]).

It is also mandatory to give some credit to Alegre, who developed a first version of GimmeHop, restricted to remote reasoning version [Ale17]. A detailed comparison with this work is performed in Section 6.2.

The interested reader can find more semantic apps in a survey paper [YP15]. Unfortunately, most of the existing semantic apps do not use fuzzy logic (with the exceptions of [MMWHVC16] and [ST22] that we will discuss in Section 6.2) or supports both local and external reasoning, as our system does.

5.3 Serializable and incremental fuzzyDL

Motivation

A *serializable semantic reasoner* can clone the data structures that represent its internal object state, obtaining two or more independent instances of the reasoner that can evolve in parallel. We will also assume that they can be written into a file. Such file could be computed by a server and downloaded by a mobile device using hybrid reasoning. However, serializable reasoners depend on the version of the reasoner, and small changes in the code of the reasoner could require changes in the serialization. They also require a common serialization strategy (e.g., a Java virtual machine –on the server– does not serialize data in the same way as a Dalvik/ART virtual machine –on an Android device).

A less restrictive concept is that of persistent semantic reasoner. A *persistent semantic reasoner* can save its internal state together with some precomputed inferences and reload it (for a given ontology). If it receives as input a previously

considered ontology, it reuses previously computed inferences, avoiding to recompute them. For example, it can store the inferred class hierarchy obtained in an ontology classification.

An *incremental semantic reasoner* can manage changes in the ontology without restarting the reasoning from scratch: that is, avoiding reloading the ontology and repeating computations (such as reclassifying the ontology). Incremental reasoners are useful, for example, when we want to submit several queries to the same ontology.

As already mentioned in Section 2.5, semantic reasoning on mobile devices can be performed in three modes: local, remote, and hybrid. To implement hybrid reasoning on mobile devices, four strategies have been proposed. We are interested in the third one, where an external server can load and preprocess an ontology, and send a copy of the reasoner to the mobile device. This requires that the reasoner is serializable (to store a copy of the reasoner) and incremental (so that we can add new axioms while reusing the previous inferences). Therefore, to promote hybrid reasoning, serializable incremental semantic reasoners seem helpful. Unfortunately, although there are some serializable and some incremental semantic reasoners, there are no semantic reasoners yet that are both serializable and incremental.

Contributions

In this section, we will explain some changes to the fuzzy ontology reasoner fuzzyDL. Firstly, we detail how we converted it into a serializable reasoner. Secondly, we discuss how it was turned into an incremental reasoner.

Serializable fuzzyDL. In Java applications, to make a class serializable it has to implement an interface called Serializable. Furthermore, all other classes used by a serializable class must be serializable as well. This can be a problem if an application uses third-party libraries such that the source cannot be modified to implement the serializable class. Furthermore, serialization converts objects into bytes, but it does not convert class variables (static variables in Java).

fuzzyDL's main class KnowledgeBase encodes a reasoner state and a fuzzy ontology, not only with the original axioms but also with some inferred ones. In the serializable version there are two new methods:

- writeToFile makes it possible to save a KnowledgeBase object into a binary file.
- readFromFile obtains a KnowledgeBase object from a serialized binary file.

To make fuzzyDL serializable we needed to revise the code allowing to do three things:

– Ensure that all necessary classes (KnowledgeBase and the classes that it uses, e.g., class Individual) implement the Serializable interface.

– Encode class variables as object variables.

– Store the data using our own classes rather than Gurobi classes, acting as a wrapper. Thus, we have all the required data to create Gurobi objects when needed.

Incremental fuzzyDL. fuzzyDL applies some preprocessing that transform a given fuzzy ontology into an expanded version that can be reused to answer different queries. For simplicity, we will describe here the preprocessing when behaving like a classical semantic reasoner (i.e., without managing fuzzy logic operators or degrees of truth), which includes the following tasks (for more details, see [Str13, BS16a])

1. Determine the language of the fuzzy ontology, e.g., $\mathcal{ALC}$. This is useful to know which inference optimizations methods can be applied.

2. Convert strings into integers. For each data property assertion of the form $(i, s) : T$ where s is a string, replace s with an integer number. Integers are assigned in such a way that the lexicographic order of all strings in the ontology is preserved. This is needed in order to deal with string based operators within MILP.

3. Solve inverse roles. For each object property assertion of the form $(i_1, i_2) : R$, it adds an assertion $(i_1, i_2) : R^I$ if R^I is an inverse role of R.

4. Compute the property hierarchy. For example, if R_1 is a sub-property if R_2 and R_2 is a sub-property of R_3, add that R_1 is a sub-property of R_3.

5. Solve object property assertions. For example, for each object property assertion of the form $(i_1, i_2) : R_1$ and for each super property R_2 of R_1, we add an assertion $R(i_1, i_2) : R$. Furthermore, if there is a pair of assertions of the form $(i_1, i_2) : R$ and $(i_2, i_3) : R$ for a transitive role R, we add an assertion $(i_1, i_3) : R$.

6. Solve reflexive roles. For each reflexive role R and each individual i in the ontology, add an assertion $(i, i) : R$.

7. Solve functional roles. If there is a pair of assertions of the form $(i_1, i_2) : R$ and $(i_1, i_3) : R$ for a functional role R, then state that i_2 and i_3 must be the same individual.

8. Preprocess TBox. In the current version, there is an absorption algorithm that splits the TBox into an acyclic part and a general part [BS16b]. In the acyclic part, it is possible to reason using an optimization called lazy unfolding. The intuitive idea is that TBox axioms are not applied to every individual but only to those individuals that are known to belong to some classes, decreasing the number of applications of the rules. In the general part, harder reasoning rules are needed and even simple TBox axioms (where the left side of the axiom is a named concept) are represented as GCIs. In the future, we plan to implement a classification algorithm to expand the class hierarchy (see [Str13] for a preliminary version).

9. Compute blocking type. Depending on the language of the fuzzy ontology, different blocking strategies are needed: subset (of labels), simple equality (of labels), simple pairwise, anywhere subset, anywhere equality, and anywhere pairwise [GHM$^+$14]. Of course, one wants to use the simplest strategy that provides correct results for a given language.

10. Solve concept assertions. For each concept assertion $a : C$ in the ontology, we apply some tableau rules to decompose C into simpler concepts.

Once the preprocessing has been done, to solve a query, the reasoner reuses the expanded version of the fuzzy ontology, but creates a local copy. For instance, to check if an ontology entails a concept assertion of the form $a : C$, fuzzyDL adds a new assertion of the form $a : (\neg C)$. Since this new assertion is added to the local copy, it will not affect other future queries.

For the moment, we have only evaluated the serializable and incremental version of fuzzyDL on a laptop computer (see details in Section 6.8), with good results. Although a mobile version is not implemented yet (it would require to have a mobile version of Gurobi, or another alternative working on Android), we are optimistic and confident on the possibility to use fuzzyDL on Android and get reasonable results.

Related work

In this section, we will describe some semantic reasoners that are serializable or incremental, but none of them is serializable and incremental, as shown in Table 5.2:

– JFact is serializable in the versions 3.5.* and 4.0.*. It also takes advantage of the Java mechanisms for serialization and is able to save a binary file. However, incremental reasoning is not implemented in those versions, so if one adds new axioms it is necessary to start from scratch.

– FaCT++ is claimed to be incremental (only in the non-buffered mode) and persistent, although not serializable. Indeed, it is able to save a text file with a representation of the ontology (with some changes, e.g., URIs are encoded as integers), the reasoner state, etc. Being persistent could be acceptable sometimes, but we have checked that incremental reasoning using a restored version of the reasoner over a serialized ontology does not always give the correct results.

– Pellet is incremental and persistent. As in Fact++, it uses Java serialization to save a binary file with the reasoner state. It is also worth to remark that in Pellet 2.2 the incremental version of the reasoner does not support datatypes [BBIM14], and the situation seems similar in the most recent version 2.3.

– Finally, other semantic reasoners, such as CEL, ELK, and SnoRocket, implement some kind of incremental reasoning but, to the best of our knowledge, do not support serialization.

Reasoner	**Serializable**	**Incremental**
JFact 3.5.* and 4.0.*	•	
Fact++n [TH06]		•
Pellet [SPC+07]		•
CEL [BLS06]		•
ELK [KKS14]		•
SnoRocket [LB10]		•

Table 5.2: Set of reasoners serializable or incremental.

It is also worth to note that the support for incremental reasoning is usually restricted to the non-buffered mode. In the buffered mode, ontology changes are stored in a buffer and are only taken into account when the user invokes a flushing method. In the non-buffered mode, ontology changes are processed as soon as they are received. Currently, fuzzyDL does not implement a buffered mode.

Another possible application of serializable and incremental reasoners is the implementation of a semantic reasoner managing volatile information [BBMP17]. The idea was not to develop a new reasoner from scratch, but to build a *metareasoner* using a serializable and incremental semantic reasoner. In particular, the authors discuss a Java implementation using the OWL API that would be able to use any serializable and incremental ontology reasoner accessible via the OWL API. Currently, fuzzyDL does not implement the OWL API.

It is also worth to note that Fact++ only has a partial support of OWL 2 datatypes, which are crucial in mobile and dynamic scenarios [BBMP17], as well as in fuzzy ontologies [Str13, BS11].

5.4 Learning fuzzy ontologies on mobile devices

Motivation

To promote the use of fuzzy ontologies in mobile computing, it is necessary to have tools that are able to run on mobile devices, including fuzzy ontology editors, reasoners, and other examples of application software. In particular, it is convenient to adapt the tools that we have developed in this thesis implementing our novel algorithms to learn fuzzy ontologies to mobile devices. This way, it would be possible to create fuzzy ontologies from mobile devices, so that other apps can use them, or to enrich fuzzy ontologies developed from other apps with new fuzzy datatypes.

Contributions

In this section, we show the contributions to create fuzzy datatypes for fuzzy ontologies using mobile applications. The aim is to make Datil and Fudge compatible with mobile devices by adapting their user interfaces. In particular, we will discuss Datil app in Section 5.4.1 and Fudge app in Section 5.4.2, both of them for Android OS.

5.4.1 Datil app

Datil version for desktop computers was presented in Section 3.2. Now, we will describe a mobile version for Android platform. The app learns fuzzy datatypes locally, avoiding external servers.

The main features of the Datil app version are:

- To the best of our knowledge, it is the first app that learns fuzzy datatypes for fuzzy ontologies.

- It uses OWL API, Fuzzy OWL 2, and fuzzyDL API. fuzzyDL API requires Gurobi library to solve MILP problems but there is no version for mobile devices. Therefore, the Gurobi library was excluded in our project from fuzzyDL API because we only need the FuzzydlToOwl2 method to convert FDL files into Fuzzy OWL 2.

- Mobility makes it possible to learn fuzzy datatypes anytime and anywhere, and to take into account in the learning the user context, such as sensor data obtained from the sensors of a smartphone.

- Learnt ontologies could be used later on another device.

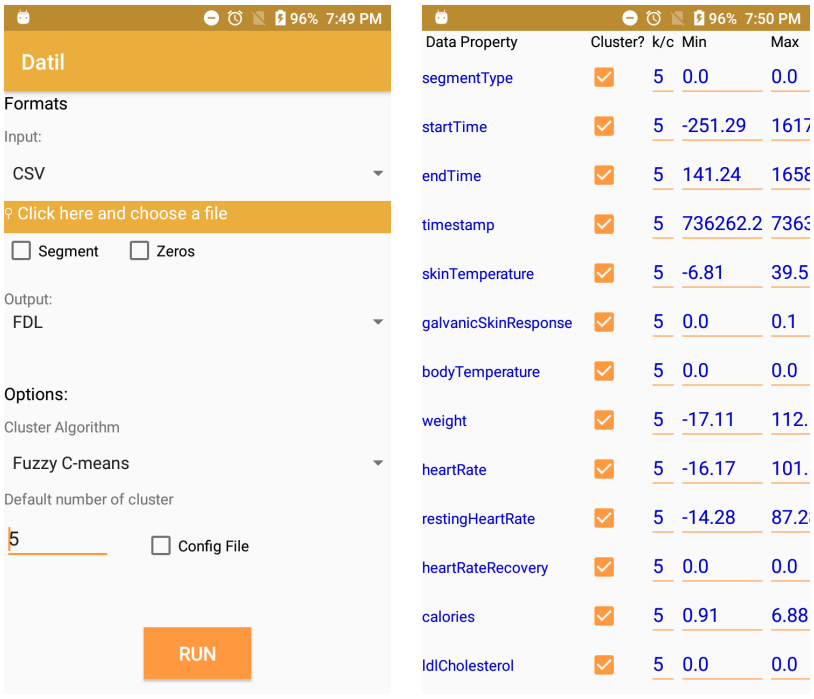
– Easy portability from desktop version to mobile app. Datil app reuses classes and methods from the desktop version because Android supports the Java programming language. However, we had to develop a new GUI.

– The GUI uses Android components and usability techniques to make it a friendly app. We use elements that provide an easy and intuitive way to use the app. e.g,. help to search for files and to create configuration files.

– It uses the API of an adapted version of HermiT 1.3.8 for Android [BYBM15]. On mobile devices, there were some conflict names (in methods and classes) between HermiT, fuzzyDL API, and OWL API. After a meticulous review, we removed duplicate folders from fuzzyDL API.

– It respects the Android security measures using app permissions. Datil needs to read and save files, so we needed READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE permissions.

Figure 5.3 (a) shows the main screen of the mobile version, while Figure 5.3 (b) shows how to compute a configuration file for all the data properties, including the Minimal (Min) and Maximal (Max) limits for each property.



a) b)

Figure 5.3: Datil app: (a) main screen, (b) options for the data properties.

Section 6.6 describes the evaluation of Datil app versus the desktop version.

5.4.2 Fudge app

In Section 3.3 we described the desktop version of Fudge. In this section, we describe the features of Fudge as a mobile application to compute fuzzy datatypes without numerical values, using instead a set of fuzzy ontology files, each of them including the definitions of an expert on a common domain.

The main features of the Fudge app are:

- To the best of our knowledge, it is the first app that creates a consensus from fuzzy datatypes definitions for fuzzy ontologies.
- It uses OWL API and Fuzzy OWL 2. Note that it does not require a reasoner.
- Mobility makes it possible to learn fuzzy datatypes anytime and anywhere.
- Learnt ontologies could be used later on another device.
- Easy portability from the desktop version to the mobile app. In our experience, the development time for Fudge app was minimal except for the need to develop a new GUI.
- It respects security and uses the same store permissions as Datil app (Section 5.4.1).
- Usability techniques were used to improve user interaction. Notably, we included in our project Google chart library[5] to customize the quantifiers chart to the parameters selected by the user.

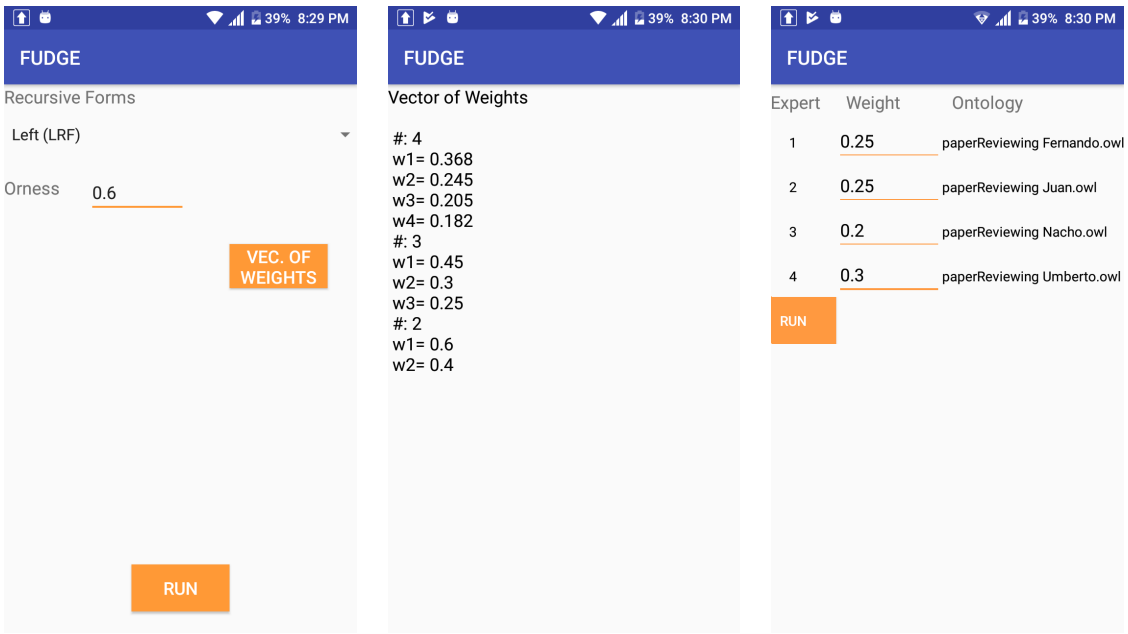
Fudge app which contains all the elements of the desktop version. Figure 5.4 (a) the main screen where user can select the input fuzzy ontologies and the aggregation operator. Depending of the aggregation operator, s/he will be able to select different ways to compute the weights:

- Figure 5.4 (b) shows how to obtain the weights for LOWA–LRF, LOWA–RRF, and FOWA from a quantifier; a customized picture of the quantifier is shown in Figure 5.4 (c).
- Figure 5.5 (a) shows how to obtain the weights using the recursive procedure; an example of vector is shown in Figure 5.5 (b).
- Figure 5.5 (c) shows how to manually set the vector of weights for CONV–LRF, CONV–RRF, and WMEAN.

[5] `https://developers.google.com/chart`

a) b) c)

Figure 5.4: Fudge app: (a) selection of input fuzzy ontologies and aggregation operator, (b) quantifiers interface, and (c) customized picture of a power quantifier with $q = 3$.



a) b) c)

Figure 5.5: Fudge app: (a) interface for recursive learning of the weights, (b) LFR vector of weights, and (c) vector of weights for CONV and WMEAN.

Chapter 6

Practical contributions: real-world applications and evaluation

In this chapter of the thesis, we will discuss the evaluation of the techniques presented in previous sections. In some cases, we have developed a real-world application in different domains (all of them related whit the management of fuzzy knowledge) as a proof of concept. In other cases, we perform an empirical evaluation and discuss the main results and findings.

To start with, we evaluate the developed reasoning algorithms. Firstly, Section 6.1 describes a gait recognition system taking advantage of the similarity between individuals. Secondly, Section 6.2 describes GimmeHop, a beer recommender system using our flexible faceted instance retrieval algorithm, and both local and remote reasoning. Thirdly, Section 6.3 details a blockchain system taking advantage of the matchmaking between individuals. Fourthly, Section 6.4 evaluates our novel instance retrieval algorithm. Next Section 6.5 discusses a Building Information Modeling application using our faceted instance retrieval algorithm.

Then, we evaluate the two developed tools to learn fuzzy datatypes. On the one hand, Section 6.6 presents an evaluation of Datil, comparing the desktop and the mobile versions, and discussing two use cases, namely life style profiling and data summarization of gait recognition data. On the other hand, Section 6.7 evaluates the novel aggregation strategies implemented in the Fudge tool.

To conclude this chapter, we evaluate the techniques for mobile devices that were not included in none of the previous sections. In particular, Section 6.8 evaluates the serializable and incremental version of fuzzyDL reasoner.

Table 6.1 summarizes the contributions described in previous sections and explains in which chapter we can find an evaluation of them.

Section	Technique	Application/evaluation
3.1	Learning local data properties	Section 6.1.1
3.2	Datil tool	Section 6.6
3.3	Fudge tool	Section 6.7
4.1	Instance retrieval algorithm	Section 6.4
4.2	Faceted instance retrieval algorithm	Sections 6.2.2, 6.5
4.3	Similarity between individuals	Section 6.1.6
4.4	Matchmaking between individuals	Section 6.3.2
5.1.1	Optimization of the reasoning	Sections 6.2.2, 6.4, 6.5
5.1.2	Distribution of the files	Sections 6.1.3, 6.2.1, 6.3.1
5.2	Local and remote reasoning	Section 6.2.2
5.3	fuzzyDL serializable and incremental	Section 6.8
5.4	Datil app	Section 6.6.1

Table 6.1: Subsections of this chapter where we can find the evaluation or applications of each of the previous sections.

6.1 Gait recognition system

Motivation

The problem of gait recognition consists of automatically classifying human people by analyzing data about their movement patterns. Gait recognition has many applications, including security (e.g. authentication and surveillance) and medicine (e.g. automatic support for the diagnosis of neurological diseases). Compared to other biometric measures for human recognition, gait has several advantages: it is non-intrusive, does not require any collaboration from the subject, involves less confidential data than other techniques (such as face recognition), and is relatively difficult to manipulate (e.g., by simulating a different walking style that does not seem unnatural).

In the last years we have witnessed an increase in the number of low cost sensors to capture pose sequences. An example is Microsoft Kinect [Dav12], a motion sensing input device originally conceived as a peripheral for video game consoles but used in many other applications, such as diagnosis of Parkinson's disease stages [DLG+18]. Pose sequences provided by the sensor could be used to compute biometric measures related to the human gait which could eventually be used for human gait recognition

Although there is a notable effort in the gait recognition using Microsoft Kinect, existing approaches have some limitations in terms of reuse and interpretability of the collected data, as well as in the management of imprecise and incomplete data.

Existing approaches generate big amounts of data which are difficult to understand by a non-expert or to reuse between different applications. For this reason, we advocate

for the combination of Semantic Web technologies to represent human Microsoft Kinect data and the biometric features for human gait motion analysis computed using them.

Rather than using classical ontologies, and because of the intrinsic imprecision of the sensor data, we propose to use fuzzy ontologies. This way, it is possible to replace precise values with more flexible fuzzy sets.

Contributions

In this section we propose gait recognition system based on Kinect data and fuzzy ontologies. A key aspect of the system is that we propose to divide each sequence of gait data frames into steps, as in conditions with incomplete data, step data would be easier to be managed. We also propose and evaluate a novel recognition algorithm of a sequence (i.e., recognizing its author) basing on the similarity between two steps (a step coming from training data and a step obtained at production stage of the system).

This section starts by describing the architecture of our gait recognition system. It has four main components as illustrated in Figure 6.1:

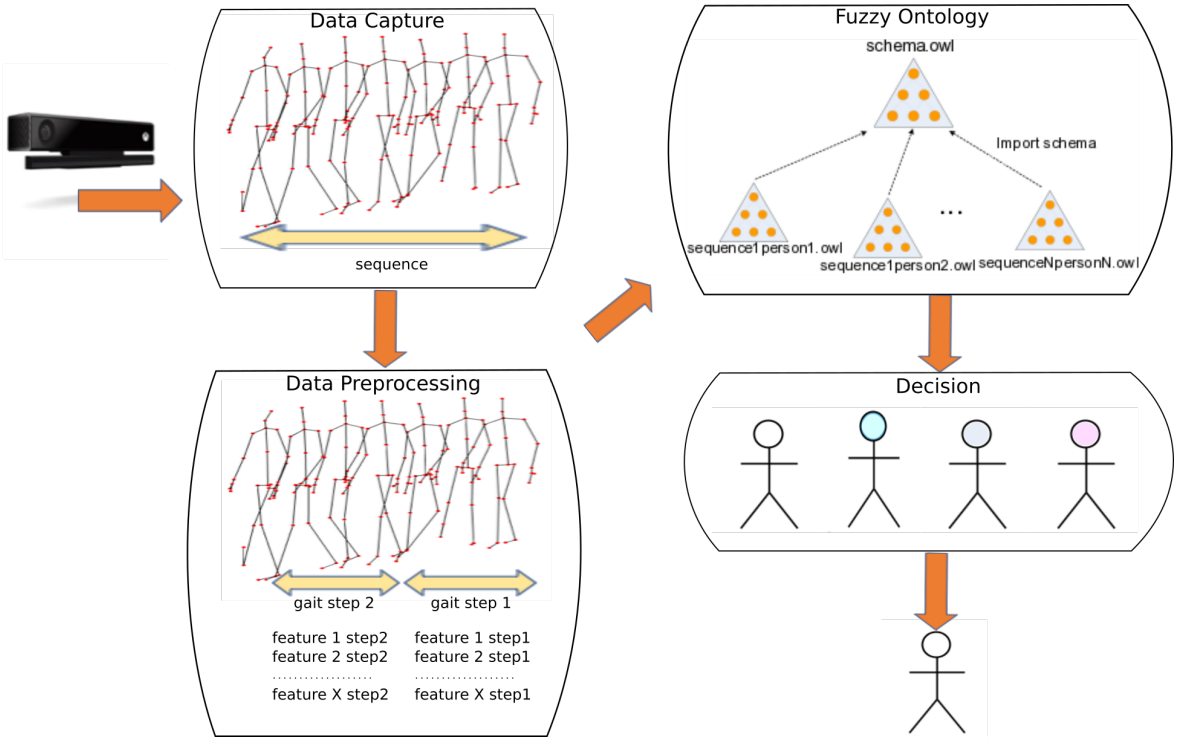


Figure 6.1: Architecture of the system.

– A data capture phase that uses a Kinect sensor to obtain the skeleton data of a person walking. The settings used to build a new dataset are described in Section 6.1.1.

– A preprocessing phase, in charge to remove the data that were incorrectly obtained by the sensor or that contain too many inferred values, to segment the gait steps from a sequence of skeletons, and compute the associated features (Section 6.1.2).

– A fuzzy ontology to represent the imprecision of the values (Section 6.1.3).

– A decision phase to classify a sequence of gait steps, where each gait step includes several features described using triangular fuzzy membership functions (Section 6.1.4)

After a discussion of the four main components of the architecture of our system (Sections 6.1.1–6.1.4), Section 6.1.5 describes existing datasets and a new dataset that we ourselves have built, and Section 6.1.6 discusses an empirical evaluation.

6.1.1 Data capture

The Data Capture module interacts directly with the sensor and collects raw data from a Microsoft Kinect sensor. The Kinect sensor actually integrates several sensors (e.g., RGB camera, depth sensor, or infrared sensor) from which several joint points of the human skeleton are obtained. These joint points are retrieved as points in a 3D-space where the coordinate origin is located at the center of the Kinect sensor. For each joint point, the sensor also shows if the value is tracked or inferred.

As an example, Figure 6.1 (Data Capture) shows how a sequence of frames (skeletons) captured by a Kinect sensor could be. The skeletons capture the pose of a person during a walk. The joints of each skeleton are represented in the figure as red dots.

Kinect sensor has two versions: v1 (with 20 joint points) and v2 (with 25 joint points, as a superset of v1). In this module we use Kinect v2. It is worth to mention a more recent effort, Nuitrack,[1] with 19 joint points which are a subset of those in Kinect v2.

To capture the data, a sensor was placed in a hall at 1 meter above the ground. We designed a square-like path and asked the participants to walk in a normal way (as natural as possible) in a straight line direction and facing to the sensor. This is illustrated in Figure 6.2, where the solid line shows the path fragment that was actually captured by the sensor. A longer segment is not possible as the sensor is not able to capture closer or farther objects. The path was repeated 10 times, so we obtained 10 sequences for each individual, with each sequence including between 69 and 163 frames.

[1] `http://download.3divi.com/Nuitrack/doc`

Each sequence was physically stored as a different recording. The resulting dataset will be described more deeply in Section 6.1.5.

6.1.2 Data preprocessing

Next, the data captured in the previous phase are preprocessed. The Data Preprocessing module contains several algorithms for identification of gait steps, noise reduction, data alignment, and feature extraction. This step does not require any human intervention.

Identification of gait steps Our system uses a gait step-based identification approach rather than using entire sequences as other approaches do (for example [AdRMA15, YDL+16]). Sequence means in this context a Kinect register of a person walking towards the camera. We consider a gait step as the sequence of frames (skeletons) from the moment when one foot strikes the floor to the other foot striking the floor. Usually, sequences contain 3-4 gait steps. In order to detect the gait steps in a sequence, a strategy based on local maximums of the distance between the feet time series in the X-axis (horizontal) and Z-axis (depth) is used. The frames (skeletons) registered from one local maximum (included) until the next local maximum (excluded) were considered as a gait step. Figure 6.1 (Data Preprocessing) shows an example of two steps segmentation, where each gait step is represented by a set of skeletons.

Noise reduction Each detected gait step is considered valid if only one foot is detected as moved. The movement of each foot is calculated based on its position in the last frame of the considered gait step with respect to its position in the first frame of the considered gait step. The frames corresponding to invalid gait steps were not considered for the next analysis steps. At this stage, frames with a large difference in length between limbs of the same type were also removed for further analysis. Specifically, frames, where the length of a bone differs by more than a third part of the average length calculated for that bone, considering all the frames of the gait step, are eliminated.

Some other strategies have been used for noise reduction purposes: based on height of a person (we considered that a person can not measure more than 2.7 m), based on the distance in the frontal plane between legs (the separation between the ankles in the frontal plane should not exceed 1 m for a person who walks), and based on the variation of the movement direction in a gait step (the position of the spine base joint follows a fairly straight trajectory during one gait step; frames whose spine base

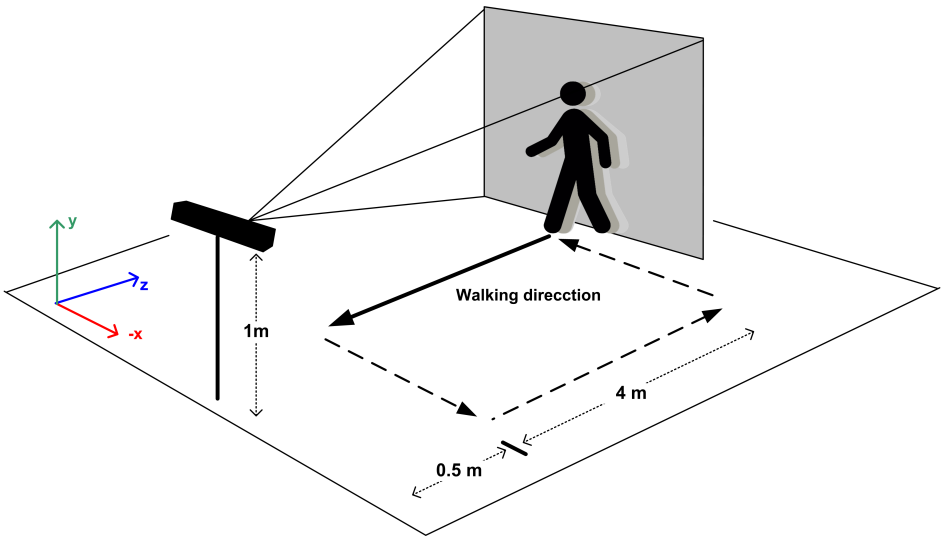


Figure 6.2: Our data capture scenario.

joint is more than 0.2 radians of that trajectory have been considered noisy). Another possibility could be taking into account the percentage of inferred values in a frame, but this was not used in this version of our system.

Data alignment Since the walking direction of a person may vary and is not always straight to the location of the camera, an angular rotation over the Y-axis (vertical axis) has been applied to each frame in order to align the skeleton joints with the hips position in the frontal (XY) plane. This kind of alignment it is widely used in the related work [CSM14, KTT+15, KNJ18]). A few features to be calculated (see below) need a different alignment of the skeletons. Specifically, for the calculation of the length and width of the gait step, a rotation over the Y-axis (vertical axis) has been applied in order to align the skeletons with the direction of the displacement (direction computed based on the variation of the spine base joint).

Some of the related works use also a second rotation over the X-axis (horizontal axis) in order to align each skeleton with the position of the spine. This rotation may be useful when the Kinect camera is placed with a certain inclination toward the ground [CSM14]. As in the recordings that we have used in this work the camera is straight, we have not applied that second rotation on the skeletons.

Feature extraction Then, the features extraction is performed for each of the gait steps identified previously. Studying the related work, we have identified 8 types of features:

1. *Spatiotemporal features*: step length, step width, velocity, and cadence. These

kind of features were used in related works as [AdRMA15].

2. *Anthropometric features*: height of the subject and length of each of the bones of the skeleton (head, up and low spine, left and right humerus, forearm, thigh, shin, foot, and hand), computed as Euclidean distance of the corresponding adjacent joints. These kind of features are commonly used in the related work on person recognition using Kinect, to compare with or to reinforce the performance of recognition based on gait kinematic features [AdRMA15, JWZS15, YDL+16].

3. *Bones angles features*: angles of the projections on the XY, XZ and YZ plane of each of the bones corresponding to upper and lower limbs. Some of these features, generally related to the limbs position, were previously used in [AdRMA15, JWZS15, KTT+15]. Additionally, we have computed the spine projection angles.

4. *Vertical distance features*: distance between a skeleton joint and the ground; e.g. left knee distance to the ground: $kneeLeft(Y)$. These kind of features were proposed in [AAJS14].

5. *Relative distance features*: distances according to the X,Y and Z-axis between symmetrical skeleton joints (e.g. right and left knees according to the X-axis: $abs(kneeLeft(X) - kneeRight(X))$) or between head and symmetrical skeleton joints; e.g. the position of the head joint with respect to both foots joints according to the X-axis: $abs(head(X) - (footLeft(X) + footRight(X))/2))$. These kind of features where used in [AAJS14] (X-axis distances) and [YDL+16] (some X, Y and Z-axis distances, selected with a greedy procedure - forward selection). In this work we have calculated the distances in all the axis, for each combination of joints proposed in the mentioned works.

6. *Relative joint position features*: position of each skeleton joint (25 for Kinect V2, 20 for Kinect V1) in X, Y, and Z-axis with respect to a fixed joint of the skeleton, namely the spine base. These kind of features have been proposed in [CSM14, KNJ18].

7. *Connected joints angles features*: angles formed by different connected joints (e.g., the angle between the upper and lower left leg, formed by $hipLeft(X, Y, Z)$, $kneeLeft(X, Y, Z)$ and $ankleLeft(X, Y, Z)$). While this type of joint angles seems natural to take into account, it is not a type of features that has been commonly used for Kinect based gait recognition. Note that, to the best of our

knowledge, these features had never been considered before in gait recognition using Kinect.

8. *Relative joint movement*: Euclidean distance showing the joint movement from frame to frame, with respect to a fixed joint of the skeleton, namely the spine base. These features have been proposed in [KNJ18].

   Example 36. *The movement of the left knee is computed as*

$$EuclideanDistance\big(\begin{array}{l}(kneeLeft(X_i,Y_i,Z_i)-spineBase(X_i,Y_i,Z_i)),\\(kneeLeft(X_{i-1},Y_{i-1},Z_{i-1})-spineBase(X_{i-1},Y_{i-1},Z_{i-1}))\end{array}\big),$$

   where i and $i-1$ represent two consecutive frames. □

We consider all the features used in the bibliography on purpose, as we want to propose a backwards compatible model. There could be some dependencies between features, but we still consider them separately as computing some of them from another features is not possible in general using standard ontology reasoning.

Table 6.4 summarizes the number of features of each type. In total, there are 211 biometric features. All these measures, except the first type, may vary from frame to frame, thus we compute mean and standard deviation of them, for each detected gait step. It should be clarified that the mean aggregation usually characterizes the individual's pose while walking (except the anthropometric features where the average is a way to compute a bone length from different noisy values). For example, the mean of the relative position of the right foot joint on each of the axes would mean the average of the right foot pose during one step. On the other hand, the standard deviation aggregation usually characterizes here the amplitude of the movement. For instance, low standard deviations of the relative position of the right hand joint on each of the axes would mean that the individual does not move his right arm while walking. These kind of aggregations are commonly used in the related work at different levels: [YDL+16] considers them at a sequence level of the feature time series, [AdRMA15] at a mixture between a sequence level and peak and valley level of the feature time series, [AAJS14] at a gait cycle level, and [CSM14] at different fractions of a gait cycle level.

In this system we propose the aggregation of the feature values at the level of a gait step, since even in conditions with short gait recordings, a gait step data would be easier to be managed than a complete sequence or a complete gait cycle with two steps (a gait step with right foot moving and another one with left foot moving). In

an incomplete sequence scenario and splitting the gait cycle in four divisions, [CSM14] has observed that the subjects achieved one or two divisions in 100% of the cases (one gait step would correspond to two divisions here).

6.1.3 Fuzzy ontologies for gait recognition

Our fuzzy ontology is able to represent raw Kinect data about the movement of a person but also biometric features computed from them. The former type of knowledge is represented using a classical ontological semantics, as we want to represent sensor data almost as they come from the sensor (only after some preprocessing, described in the previous section).

Because of the huge amount of data we are trying to manage, we propose a distributed architecture (see Section 5.1.2) formed by a general schema ontology, with the definitions of the classes and properties, and a collection of instantiated ontologies that populate the schema ontology with individuals and their attribute values. This architecture is illustrated in Figure 6.3.

The cornerstone of our architecture is the schema ontology. It contains 4 classes, 9 object properties, 391 data properties and 1409 logical axioms. The expressivity of the ontology is that of the Description Logic $\mathcal{ALCRIF}(\mathbf{D})$. To represent the Kinect data, we consider 4 mutually disjoint classes. For each instance of Human, there are several recordings. Every recording obtained using Kinect is represented as an instance of Sequence and each sequence is composed of several instances of Frame. After some preprocessing, we can also divide a sequence in several instance of Step, so that each step is related to a unique sequence. Each step contains several frames, but each frame is associated at most to one step (if the human stops walking at some point of the video, there might be frames not associated to any step).

We also have object and data properties with their corresponding domain, range, and functionality restrictions. Relationships between classes are modeled using object properties personIsInRecording, recordingHasFrame, recordingHasStep, and stepIsInFrame, together with their inverses recordingHasPerson, frameIsInRecording, stepIsInRecording, and frameHasStep, respectively. Figure 6.4 shows the classes and their relationships. We use subproperty chains to infer missing information. For example, the chain frameIsInRecording $\circ$ recordingHasStep is a subproperty of frameHasStep.

Figure 6.5 shows a fragment of the data property hierarchy in the schema ontology. For example, we can see that stepAttribute contains the 8 types of biometric features enumerated in Section 6.1.2 plus some additional attributes (otherAttributes, with the moving leg and some IDs), all the four type 1 features, and both a comment and a range restriction on the functional property velocityStep.

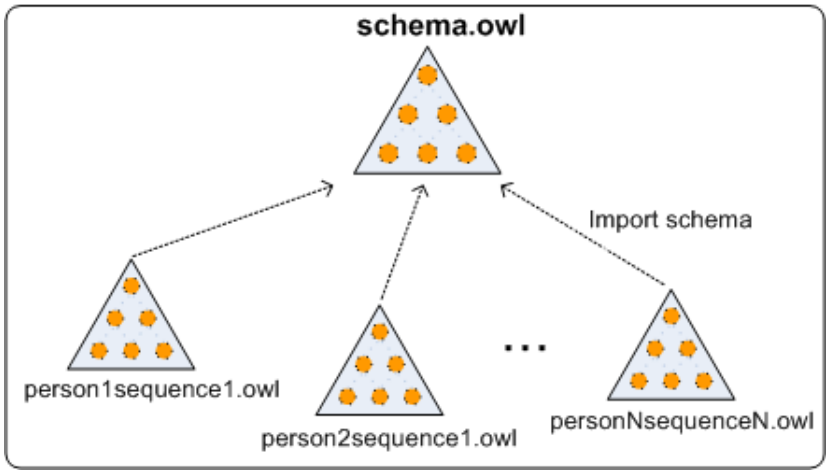


Figure 6.3: Organization of our fuzzy ontologies: a schema ontology and several instantiated ontologies.

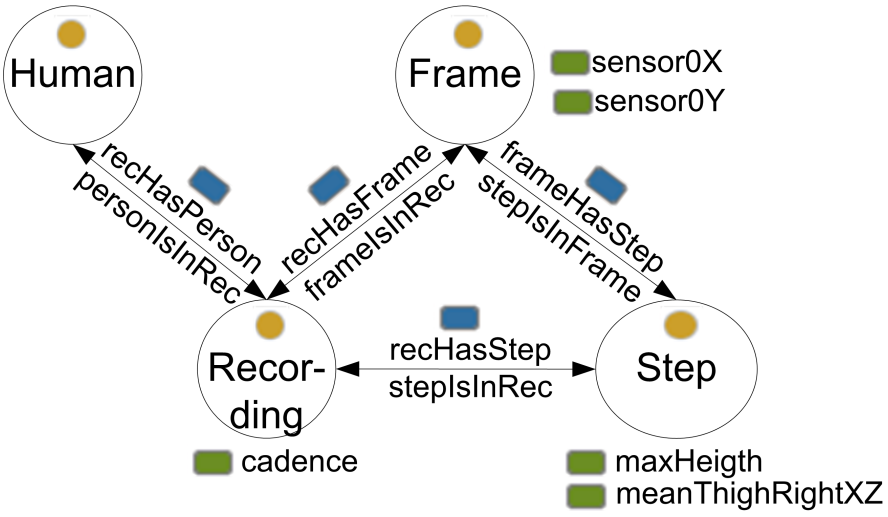


Figure 6.4: An excerpt of our ontology scheme.

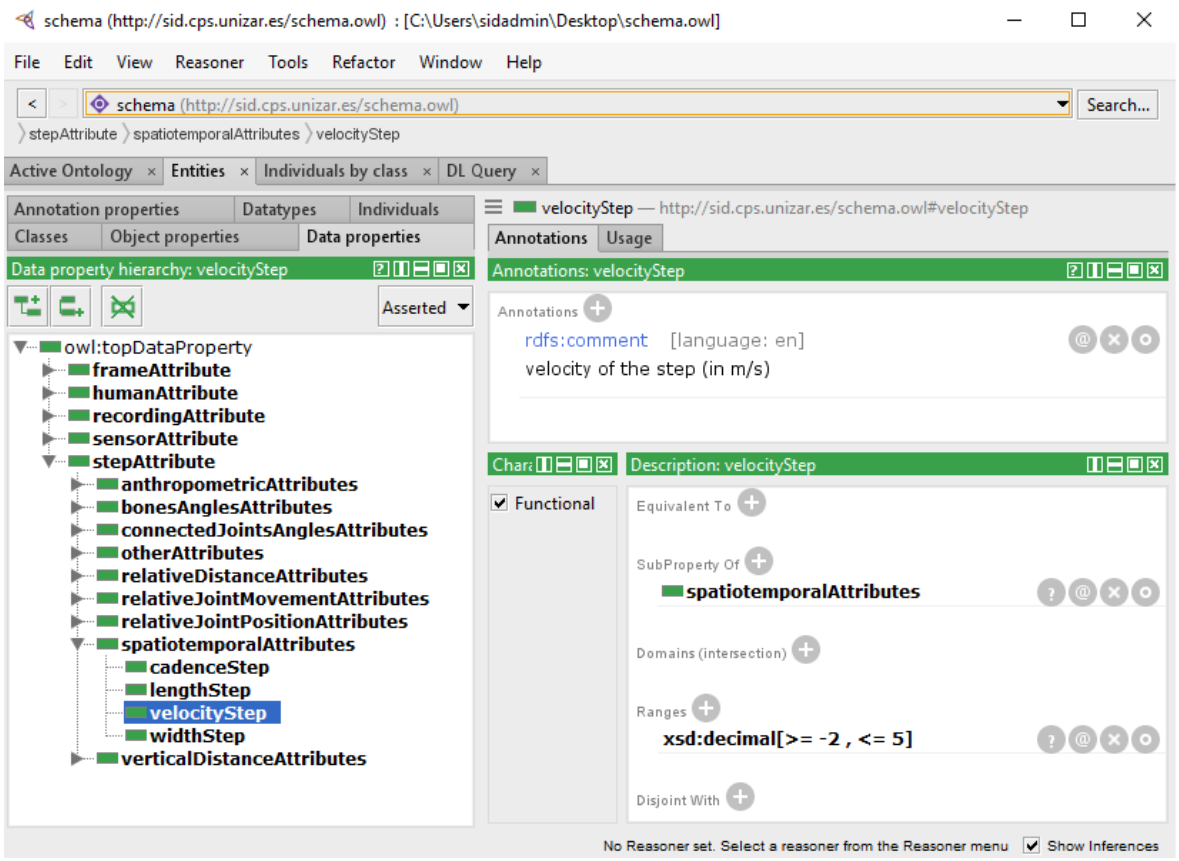


Figure 6.5: Some properties (type 1) of our schema ontology.

We have an instantiated ontology for each gait sequence of each individual. Instantiated ontologies can be logically classified in a dataset, as we will describe in the next section. Sequences to be classified are also represented as an instantiated ontology. An instantiated ontology firstly imports the schema ontology and then populates it with individuals (persons, recordings, frames, and steps), and axioms (class/property assertions).

We worked with sequences (recordings) of walking people recorded using the Microsoft Kinect V2 sensor. While one could consider using a better sensor, we aim at using low cost devices and thus must deal with such imprecision. Therefore, the values of the data properties are fuzzy datatypes. In our scenario, fuzzy datatypes replace precise crisp values with a more general fuzzy membership function, as seen in Section 3.1. Recall that the values of the biometric features are computed for each step.

Each frame has 25 data properties linking it with each of the 25 joints identified by the Kinect V2 sensor (see Figure 6.6). For example, sensor0 related a frame with a real number (xsd:decimal). The names of these data properties use the common numeration of the joints, but the ontology schema also has 25 equivalent data properties with more

readable names. For example, spineBase is equivalent to sensor0. To represent Kinect V1 or Nuitrack data, one would only use the relevant 20 or 19 (respectively) properties.

Regarding the biometric features, each step has several data properties, such as meanHeight (average value of the height of a person) or meanThighRightXZ (average value of the angles formed by the right thigh). Similarly, our ontology also allows representing biometric features of a sequence (although we do not currently use them) such as the total cadence (cadenceTotal, number of steps per unit of time).

To the best of our knowledge, none of the existing ontologies is suitable for our needs, so we needed to develop our ontology from scratch. It has been developed using Protégé ontology editor. Classes, properties, individuals, and most of the axioms are represented as usual. To represent the fuzzy datatypes, we have used Protégé plug-in called *Fuzzy OWL 2* that can be used to create and edit fuzzy ontologies.

The main version of the ontologies is encoded in OWL 2 Manchester syntax, but we also experimented with another version in RDF using Turtle syntax (TTL format). RDF triples are appropriate to answer SPARQL queries, but this is not necessary in our gait recognition scenario.

Before concluding this section, let us illustrate how our approach leads to more readable datasets than the related work. Figure 6.7 (a) shows a fragment of a file included in the dataset built by Kastanioitis et al. [KTT+15], whereas Figure 6.7 (b) shows how one of our instantiated fuzzy ontologies represent some of the same data (the first four rows), corresponding to the first frame of the first recording of the twenty-first person.

6.1.4 Decision: gait recognition algorithm

The final step of our system is a decision module to classify a sequence of steps. Algorithm 9 shows how to identify the individual that is the author of the sequence or to recognize that it is an unknown individual. The inputs are a sequence of steps r_1, a fuzzy ontology $\mathcal{O}$ with the dataset, and a threshold Θ.

The first part is the initialization (Lines 2–4), where an array to store the number of votes of each individual is created; initially every individual has zero votes.

The next part computes the similarity of steps (Lines 6 –16). For each data property assertion relating an step s_1 with the input sequence r_1, and for each step s_2 of the dataset, the algorithm computes the similarity between the pair of steps using Algorithm 7 (Line 9). Next, it compares this similarity with the highest one previously found, in order to store the individual which is the author of the step with the highest value of similarity (Lines 10–13). When all steps in the dataset are compared to s_1, the individual which is the author of the step with the highest value of similarity to s_1

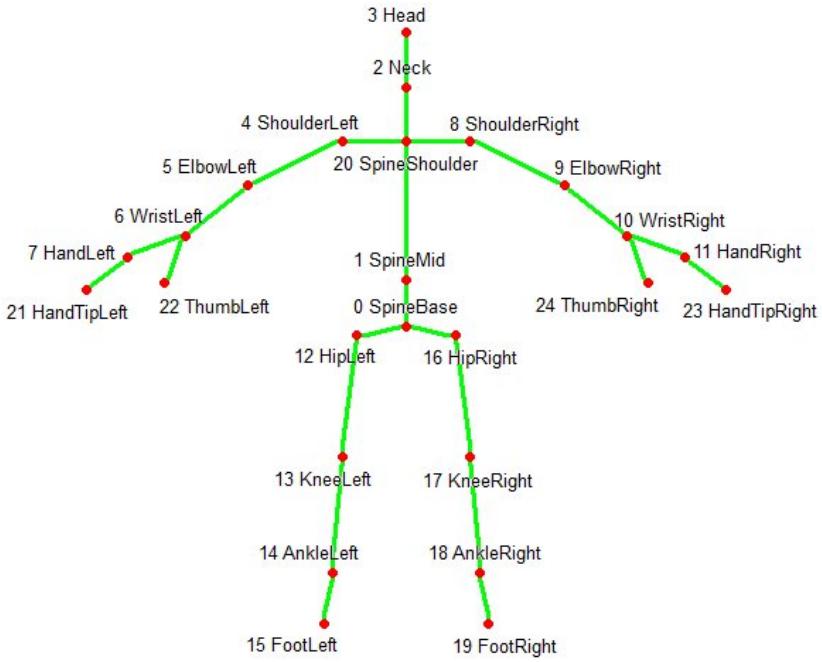


Figure 6.6: 25 skeleton joints captured by Microsoft Kinect V2 sensor[OVB19].

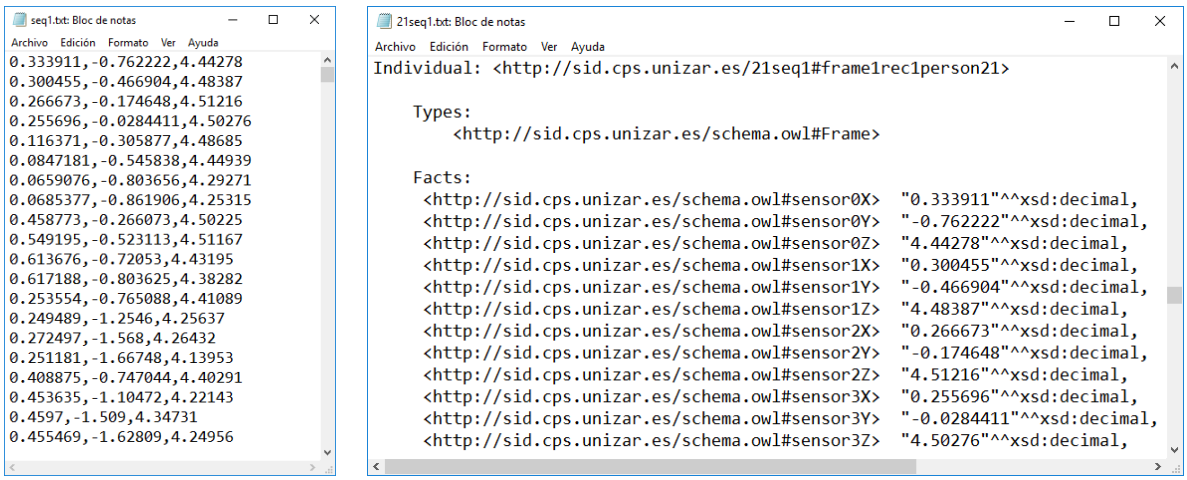


Figure 6.7: (a) Original representation; (b) Fuzzy ontology representation.

receives one vote (Line 15).

Next, the algorithm computes the most probable individual. In particular, when all steps s_1 are processed, the algorithm retrieves the individual x with the highest number of votes (Line 17).

Finally, the algorithm returns the result of the classification (Lines 19–23). It retrieves the individual with more votes x only if it received a percentage of the votes greater or equal than an electoral threshold $\Theta \in [0, 1]$, otherwise the system assumes that the individual is unknown (i.e., it is none of the individuals in the training dataset) and returns a new individual.

Algorithm 9 Algorithm to compute the classification of a sequence.

Input: A sequence r_1,
Input: A fuzzy ontology $\mathcal{O}$ with the dataset,
Input: A threshold Θ
Output: An individual x

```
1: // Initialization
2: for all individual i in O do
3:    votes[i] ← 0
4: end for
5: // Similarity of steps
6: for all data property assertion (s1,r1):stepIsInRec in O do
7:    maxSim ← −1
8:    for all instance of Step s2 in O do
9:       sim ← similarity(s1, s2, O, ϵ) using Algorithm 7
10:      if sim > maxSim then
11:         r2 ← Retrieve the value of the data property stepIsInRec for s2 in O
12:         i ← Retrieve the value of the data property recHasPerson for r2 in O
13:      end if
14:   end for
15:   votes[i] ← votes[i] + 1
16: end for
17: x ← argmax_i votes[i]
18: // Comparison with threshold
19: if votes[x]/∑_i votes[i] > Θ then
20:   return x // Individual more similar or voted
21: else
22:   return new individual
23: end if
```

Let us now discuss some important issues of our algorithm or implementation details.

– For an easier presentation of Algorithm 9, we assumed a single fuzzy ontology, but we actually work with sequences r_i, each of them represented by a different

fuzzy ontology (see Section 6.1.3). Similarly, to compute the similarity between two steps we need a pair of ontologies with the values of the data properties described using fuzzy datatypes.

– Regarding Algorithm 7, the values of the features f are now described using triangular fuzzy membership functions (for example, $f_1 = \mathbf{tri}(a_{11}, a_{12}, a_{13})$ and $f_2 = \mathbf{tri}(a_{21}, a_{22}, a_{23})$). To compute its similarity, we restricted to the minimum t-norm to make the computation easier. The minimum of two triangular fuzzy membership functions depend on several cases, as illustrated in Figure 6.8. As defuzzification methods, we have tried the Middle of Maxima (MOM, Eq. 2.21) and the center of gravity (COG, Eq. 2.22). Note that in this case there is a single maximum (possibly zero), so the well-known smallest (SOM, Eq. 2.19), largest (LOM, Eq. 2.20), and middle of maxima defuzzification methods coincide. We have also tried with several other measures that we call the average (AVG):

$$sim(p_1, p_2) = 1 - \frac{\mu_{p_1}(a_{22}) + \mu_{p_2}(a_{12})}{2} \tag{6.1}$$

and the minimum (MIN):

$$sim(p_1, p_2) = \min\left\{\mu_{p_1}(a_{22}), \mu_{p_2}(a_{12})\right\} \tag{6.2}$$

As the aggregation operator @, we use a t-norm, namely the product. Łukasiewicz t-norm collapses to zero very easily, and the minimum of a lot of aggregated values is not as discriminant as the product. We work with very small values but we checked in practice that they can be successfully managed as Java double numbers.

– In the voting-based scheme, in case of a tie we retrieve the step with the highest value of similarity. It would also be possible to associate a fuzzy degree $\alpha \in [0, 1]$ to the classification result, and retrieve not only the winner individual but also a numerical value given by the ratio between the number of votes obtained by the winning individual and the total number of steps. This would also make it possible to retrieve the top–k individuals, decreasingly ordered by the classification degree.

– To avoid the fact that a single $similarity(s_1, s_2) = 0$ in Algorithm 7, in Line 9 we replace the 0 value with a very small $\epsilon > 0$.

– Finally, an important problem is how to select the value of Θ. A possible approach is to compute the percentages of correct classifications of the known people and of the new people, and find a trade-off between both values using the ROC curve.

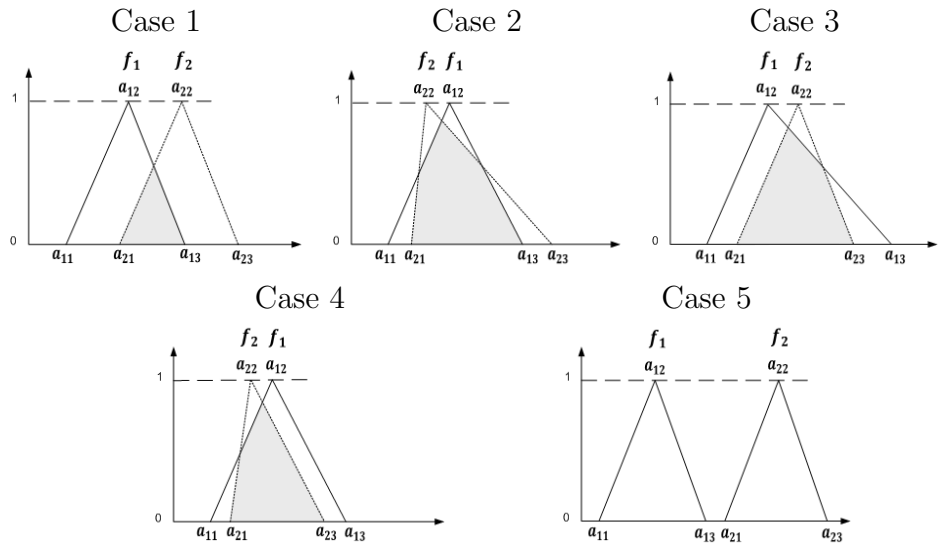


Figure 6.8: Different cases to compute the minimum of two triangular functions.

6.1.5 Zaragoza dataset: OWL and RDF representation

Before presenting our evaluation of the system, we will describe existing datasets and a new one developed by us.

Recall that each dataset is a collection of instantiated fuzzy ontologies, with respect to a schema ontology. To the best of our knowledge, there are only three existing datasets using Kinect to record people walking on straight lines or semi-circle paths, so we built a new one.

Kastanioitis et al. proposed the two first ones, denoted *Kas1* [KTT+15] and *Kas2* [KTT+15] , where individuals walk a straight line. V. O. Andersson et al. proposed a dataset with lateral recordings using a moving camera, denoted *And1* [AdRMA15]. For the purpose of this thesis, we have built a new dataset, denoted *Zar2*. More details about the datasets can be found online, subject to legal compliance[2].

- *Kas1* contains 30 individuals (15 males and 15 females). Each person contain 5 sequences and each sequence has between 1 and 5 steps. Data were acquired using a Microsoft Kinect V1 sensor.

- *Kas2* includes 30 individuals (17 males and 13 females). There are 10 sequences for every person. Each sequence has between 1 and 5 steps. Data were obtained using a Microsoft Kinect V2 sensor.

- *And1* includes 164 individuals. For each of them there are 5 or 6 sequences with a number of gait steps detected by our algorithm between 6 and 62. Data were

[2] `http://webdiis.unizar.es/~ihvdis/gait_recognition`

recorded using a Microsoft Kinect V1 sensor.

– *Zar2* was built to increase the small number of individuals walking a straight line using Kinect V2 in *Kas2*. We had help of 91 volunteers walking a straight line, 34 women and 57 men with ages between 18 and 60. For each volunteer, 10 sequences were recorded in the experimental scenario, but a total of 8 sequences were discarded because they wore reflective clothes (such as leather jackets) that provided too many errors. Every sequence contains between 1 and 6 steps. Data were captured using a Microsoft Kinect V2 sensor, as described in Section 6.1.1.

Figure 6.9 shows the number of steps in the different datasets. We can see that most of the sequences in *Zar2* and *Kas2* contain between 3 and 5 steps, while most of the sequences in *Kas1* have between 2 and 3 steps. Because of the moving camera, sequences in *And1* have a much larger number of steps.

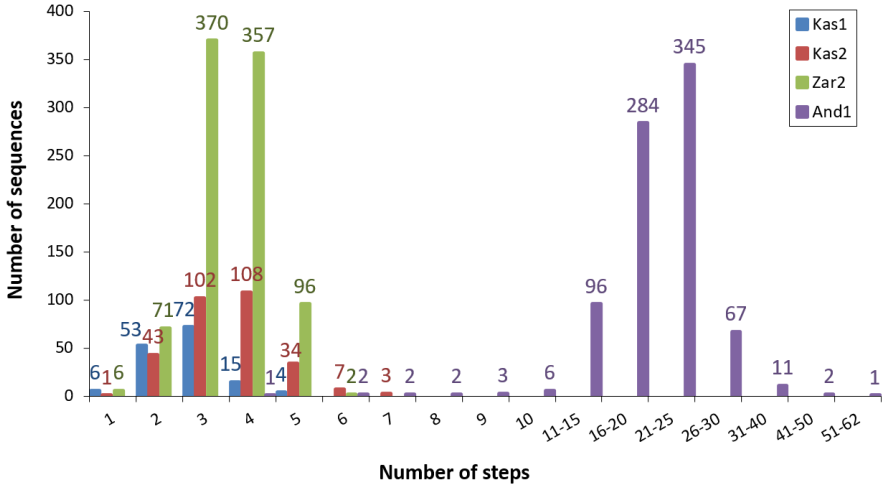


Figure 6.9: Number of steps in each dataset.

Let us conclude with some statistical information. Table 6.2 shows for each dataset (Dataset) the Kinect version (Kinect) and the total numbers of People (#People), Sequences (#Sequences), and Steps (#Steps). The table also shows the average number of Individuals (I), Frames (F), Fuzzy Datatypes (FD), Class Assertions (CA), Object Property Assertions (OPA), and Data Property Assertions (DPA) for each sequence of the respective dataset. Overall, there are 2172 OWL instantiated files (so, 2172 sequences).

6.1.6 Results and discussion

In this section we evaluate the accuracy of the classification, the accuracy of each type of features, the size of the datasets, and the running time.

Table 6.2: Statistics of OWL ontologies

Dataset	Kinect	#People	#Sequences	#Steps	I	F	FD	CA	OPA	DPA
Kas1	V1	30	150	408	91	86	574	664	101	5180
Kas2	V2	30	298	1058	107	101	749	856	122	7611
Zar2	V2	91	902	3178	107	102	743	850	123	7619
And1	V1	164	822	16745	621	598	4298	4919	746	35914

Classification analysis. To evaluate the performance of the classification, we considered 4 datasets (Kas1, Kas2, Zar2, And1), 4 defuzzification strategies (COG, MOM, AVG, MIN), and 2 training setups (20/80 and 80/20). For example, 20/80 means that we took 20% of the sequences of each individual as training data, and the remaining 80% was used as test data. Experiments were repeated 20 times and for each of them the selection of the sequences was random. We also include as baselines the best results computed by other authors: for Kas1 and Kas2 we include the results of the EK+RH method [KTEF16], and for And1 we include the results of the method in [KNJ18]; these are the best methods so far for straight lines and lateral recordings, respectively. We also include the results of a previous version of our work [BDB17], using a different classification algorithm (nearest neighbor with Euclidean distance, restricting triangular functions to the average value) and only 12 biometrical features.

Table 6.3 shows the percentage of correct classification of the methods (average and, if available, standard deviation) for all feature types; please recall that details such as Kinect version, number of individuals, and number of sequences can be found in Table 6.2.

In datasets where individuals walk a straight line (Kas1, Kas2, and Zar2), our method outperforms EK+RH, the best method so far. For the best defuzzification strategy, the new method obtained an increment in the precision of 3–4%. In particular, our method was able to classify precisely all individuals of Zar2 for the case 80/20. However, for the dataset with lateral recordings using a moving camera, our system achieves worst classification results. This shows that our algorithm would require some changes to manage such datasets, but we leave this as future work. In particular, we would need different preprocessing techniques, as the type of noise is different, and possibly more flexible fusion operators than a t-norm (because every t-norm produces a value which is smaller or equal than the minimum of the aggregated values, a single small value is enough to produce a global small value).

The best defuzzification strategy depends on the dataset. AVG is usually the best one (in Kas1, Kas2, and And1), but MOM is the best one in Zar2. In datasets where individuals walk a straight line there are not significant differences between the different

Table 6.3: Classification results for various methods and training sizes using feature types 1-8.

Dataset	Method	Correct Classification Rate	
	Training/Test Size (%)	20/80	80/20
Kas1	EK+RH [KTEF16]	No data	95.67
	COG	88.37±1.88	**99.33±1.37**
	MOM	89±1.61	**99.33±1.37**
	AVG	86.88±1.01	97.17±1.22
	MIN	84.38±1.85	95.67±2.44
Kas2	EK+RH [KTEF16]	92.27	97.05
	[BDB17]	No data	89.03
	COG	93.51±1.07	99.83±0.53
	MOM	94.01±1.05	99.66±0.71
	AVG	96.83±1.35	**100±0**
	MIN	94.92±1.26	99.66±0.71
Zar2	COG	98.34±0.54	**100±0**
	MOM	98.45±0.57	**100±0**
	AVG	97.86±0.95	**100±0**
	MIN	97.39±1.09	**100±0**
And1	[KNJ18]	No data	**97.0±0.5**
	COG	31.74±2.34	66.25±6.33
	MOM	33.75±2.80	68.56±4.97
	AVG	69.07±5.54	95.66±2.53
	MIN	61.54±6.17	93.84±3.57

defuzzifications, but in And1 the differences between AVG and MIN, on the one side, and COG and MOM, on the other side, is really important (close to 30%).

Note that the sensor Kinect V2 (datasets Kas2 and Zar2) makes it possible to obtain very good results for the case 20/80, almost comparable to the case 80/20. This means that only 20 % of the data (i.e., 2 sequences for each individual) is enough as training data. However, for Kinect V1 (datasets Kas1 and And1), it seems preferable to use a 80/20 configuration (i.e., 4 sequences for each individual).

Note also that our system gets better results in Zar2 with 91 individuals than in Kas2 with 30 individuals. It has to be taken into account that Kas2 has been recorded with a sensor angle of 30 degrees on the direction of displacement. Therefore, the worst results in Kas2 might be caused by a problem of body self-occlusion, a problem previously reported in [MIGL17]. These results suggest that 3D sensor based gait recognition could be more appropriate in scenarios where the sensor is located frontally or combining information from different side placed sensors, that have vision on both sides of the body.

Features analysis. We considered the Zar2 dataset, 4 defuzzification strategies (COG, MOM, AVG, MIN), and a 20/80 scenario. Table 6.4 shows the results of each type of features (as defined in Section 6.1.2), and the number of features of each type. Experiments were repeated 20 times and we show the average and the standard deviation.

We can see that using all the feature types provides the best results. It is worth to stress that a feature selection step to reduce the number of variables was proposed in [BDB17], and some preliminary experiments showed that they did not have an influence in the results. However, the results with a high number of individuals show that when all variables are considered, the precision of the classification is higher. Indeed, feature selection is more common in classification scenarios with a fixed number of labels to be classified, but in our case the number of classes increases every time a new individual enters the system.

Table 6.4: Results of each type of attributes on Zar2 dataset (20/80).

	Type								
	1	2	3	4	5	6	7	8	1-8
Size	4	16	27	21	33	60	25	25	211
COG	1.47±0.33	68.96±3.10	52.72±3.45	52.54±2.58	92.83±1.65	95.54±1.43	83.89±2.37	4.65±0.41	98.34±0.54
MOM	1.53±0.40	72.43±3.63	61.30±3.33	56.37±3.54	93.79±1.40	95.94±1.45	84.46±2.72	7.10±0.94	**98.45±0.57**
AVG	0.87±0.28	59.57±2.70	35.70±3.11	44.29±2.83	71.14±3.11	94.72±1.23	67.75±3.47	6.29±0.97	97.86±0.95
MIN	0.70±0.23	50.92±2.52	20.99±2.47	36.13±2.27	62.36±3.30	90.80±1.67	59.83±4.44	5.33±0.98	97.39±1.09

We can also see that choice of the defuzzification strategy is quite significant. In general, when using only some type of features, COG and MOM perform better.

Using just some type of variables might provide good enough results. For example, type 6 has a higher precision than the method EK+RH (for the case of MOM). Then, if we add more attributes to the model, precision grows. Indeed, we can obtain simpler models if needed for scalability reasons, although as we will see later the running time of our system seems very reasonable. Consider for example the case of MOM. While the full model has 211 attributes and a precision of 98.5%, a model based on type 6 has a precision only 2.5% smaller with between a third and a quarter of the attributes (60). Then, a model based on type 5 has a precision only 2% smaller than the model based on type 6 with about half of the attributes (33). Type 7 also provides a good precision (it is the third best type) with only 25 features, despite not having being widely used in the literature. Because type 7 features are those with less dependency of the height, this confirms again that how a person walks is actually useful to identify him/her.

Space and time analysis. Table 6.5 shows the space required by the OWL (OWL 2 ontologies in Manchester syntax) and TTL (RDF triples in Turtle syntax) versions of our fuzzy ontologies. We can see that OWL files require a much smaller size than TTL files. Indeed, OWL Manchester syntax is known to be a rather succinct syntax.

Table 6.5: Size (in MB) for each input format and dataset.

Format	Dataset			
	Kas1	Kas2	Zar2	And1
OWL	111	304	939	4190
TTL	205	557	1750	8080

Next, we will detail some experiments to measure the running time. Firstly, we measured the starting time of the system; in particular, the loading time of the datasets in memory. This step is only performed once, during the initialization of the system. To manage TTL files, we used the server Apache Jena Fuseki 3.14 and Jena Java API[3]. As a baseline, we also considered a CSV (text files with comma-separated values) version of the fuzzy ontologies that only stores data about the biometric features and not the raw Kinect data (as they are not used during the classification). The CSV version represents triangular fuzzy functions by separately storing the average x and the standard deviation σ.

[3] `http://jena.apache.org`

Table 6.6 shows the results of the time (in seconds) needed to load the different datasets for different input formats: CSV, OWL, and TTL. Of course, the time depends on the number of individuals and steps, and so on the dataset. TTL requires much more time than OWL, as managing RDF triples and SPARQL queries using Apache Jena Fuseki Server was more costly than retrieving OWL 2 individuals and their data property values using the OWL API. Managing CSV is also significantly faster than OWL, so if the system is going to be restarted often, one could consider having a CSV copy of the datasets to speed the initialization up. Another suggestion is to split each OWL/TTL file into two files: one with the raw Kinect data, and another one with data about the biometric features, so that only the second one is loaded during the initialization of the system.

Table 6.6: Loading time (in s) for each input format and dataset.

Format	Dataset			
	Kas1	Kas2	Zar2	And1
CSV	0.14	0.30	0.97	3.39
OWL	15	47	172	654
TTL	62	418	4267	16224

Secondly, we analyze the time (in seconds) needed to classify a new sequence using only features of type 6 (the type with a better precision) and using all of them. The experiment is repeated for each dataset and the results are shown in Table 6.7. On average, type 6 requires about a third of the total time, making it possible to have a good trade-off between running time and classification accuracy. It is worth to note that for very big datasets, such as And1, the difference between the defuzzification methods is relatively significant, with AVG and MIN being the faster ones.

Table 6.7: Classification time (s) of one individual.

Type	Method	Dataset			
		Kas1	Kas2	Zar2	And1
6	COG	0.03	0.07	0.10	5.49
	MOM	0.02	0.07	0.09	5.21
	AVG	0.02	0.06	0.12	4.42
	MIN	0.02	0.06	0.12	4.44
1-8	COG	0.07	0.22	0.30	17.37
	MOM	0.07	0.21	0.27	16.52
	AVG	0.06	0.18	0.35	14.59
	MIN	0.06	0.18	0.36	14.68

Evaluation of the discovery of new individuals. We have used Zar2 dataset to find an estimation of Θ. First, Zar2 dataset was split into a training set (80 %) and a test set (20 %). then, we computed for any gait sequence of any individual p in the test set, the rate of votes obtained by p when there are other sequences of p in the training set, and the maximum rate of votes obtained by an individual q ($\neq p$) when all sequences of p are removed from the training set. The experiment was repeated 20 times with different training/test sets.

Figure 6.10 shows the ROC curve and the AUC (area under the curve). It is shown that the best founded electoral threshold Θ is equal to 0.917, and AUC = 0.881, that is, with this method it can be correctly classified the 88.1% of the sequences with a 95% confidence interval $(0.874, 0.889)$.

Related Work

This section summarizes the previous work on gait recognition using the Kinect sensor on the use of ontologies to represent Kinect data.

Gait recognition and Kinect. There is a long history of research on the recognition of people through the gait using video recordings [BHP05]. The release in 2011 of the Kinect depth camera has led to another approach to the gait recognition. Since then, several research papers have addressed the gait analysis for human recognition with skeleton data captured by a Kinect sensor.

We can find in the literature some prospective works that use their own datasets, with few individuals (usually less than ten), to investigate the possibilities of this technology. In this sense we can mention the approach of [PKWLP12], where the authors observed promising results concerning person recognition using a Naive Bayes classifier and a simple set of features obtained from 9 people. Another work to mention here is the one in [JWZS15]. In that case the features are characterized as static (height, length of bones) or dynamic (angles of joints). Several distances where used between these features and finally a K-nearest neighbor (KNN) classifier [FH51] obtained around 80% accuracy for ten people. In [AAJS14] the authors use their own dataset with lateral recordings from 20 participants. They extract some vertical and horizontal features, use a KNN based method and obtain 92% recognition rate.

Other kind of proposals use broader datasets with 30 people recordings [KTEF16, KTT$^+$15]. In [KTT$^+$15] a framework for gait-based recognition is proposed. The authors provide a new dataset captured from 30 people walking in a straight line. For each subject, the dataset contains five sequences (corresponding to 5 walks in front of the Kinect camera) captured in three separate sessions during the same day. The

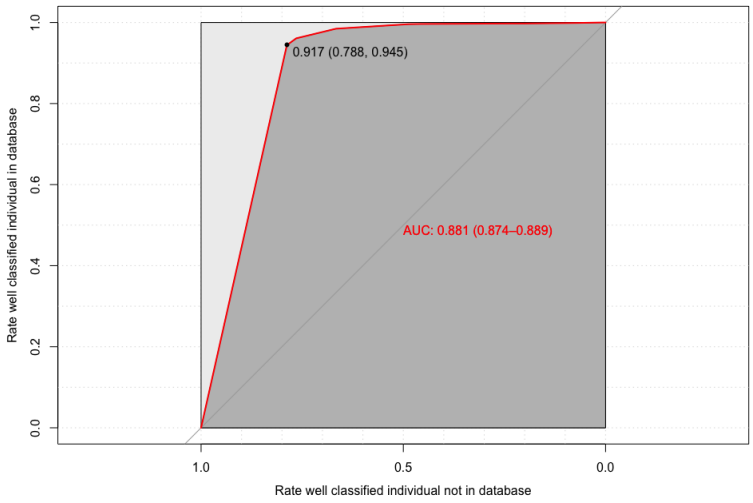


Figure 6.10: ROC curve: best threshold $\Theta = 0.917$, AUC= 0.881 (95% confidence interval $(0.874, 0.889)$).

,

authors extract from the dataset 16 dimensional vectors that encode the direction of every limb using two Euler angles. Then use several dissimilarity tests and achieve 93.29% identification rate. Some steps further are taken in [KTEF16] with a new method for fusing information from Riemannian and Euclidean features representation that achieves 95.67% accuracy on the same dataset. Moreover, the authors provide a new dataset for gait recognition captured from 30 people walking in a straight line and using the more recent Kinect v2, dataset available on demand. There are 10 sequences for every recorded person. As far as we know, that is the only dataset available captured with Kinect v2. The authors obtained 97.05% accuracy on that dataset.

A different approach to the problem of recognition of people is addressed by [CSM14]. Specifically, the authors address the problem of recognition from sequences with partial gait cycles, considered common at the airport security checkpoints. They use an own dataset with 60 distinct subjects and a hierarchical method for frontal gait recognition using Kinect depth and skeleton streams and achieve 66.67% accuracy.

An even more extensive dataset (90 people) of freestyle walks has been provided by [KNJ18], although it is not covered in our evaluation. Moreover, the authors propose a recognition method based on a KNN classifier with the Manhattan distance, applied at a frame level. The features used in this case are the relative joint positions with respect to a fixed joint, namely the spine base. They enforce the method with a new kind of features, the relative joint movement with respect to the same fixed joint, and achieve 96.8% accuracy.

The most extensive Kinect skeleton dataset that we found in the literature is described in [AdRMA15]. The authors provide a dataset with lateral recording of the subjects. Each sequence represents a round of the subject in front of a Kinect camera on a semi-circle path. The Kinect camera was placed at the center of the semi-circle, on a spinning dish and was rotated to keep the subject in the center of the view. Several recognition methods were tested on this dataset, most of them based on the nearest neighbor [CH67] (KNN method with K=1) with Manhattan distance, with variations in the type of features used. In particular, [AdRMA15] obtained 87.7% accuracy, [YDL+16] achieved 95.4% accuracy, and [KNJ18] reached 97% recognition rate.

Details of the features used by each related work can be found in subsection 6.1.2.

Ontologies and Kinect. There have also been some previous approaches to represent Kinect-related data using classical ontologies [DRWL+13] and fuzzy ones [DRPCLD14]. The authors even developed the so-called *Kinect ontology*. However, despite this generic name, their approach is strongly focused on a different application, recognition of human activity, and cannot be reused in our scenario. On the one hand, *Kinect ontology* was not designed to encode the information directly obtained from the sensors. On the other hand, its fuzzy extension does not provide a fuzzy representation of the relevant features for gait recognition that we discussed in Section 6.1.2.

Our work is a followup to [BDB17], which proposed the use of fuzzy ontologies for gait recognition. The main differences are an extended fuzzy ontology, a distributed architecture of the ontology files, a novel approach to learn the fuzzy datatypes (Section 3.1), a novel classification algorithm based on the similarity between individuals (Section 4.3) and a voting schema, the development of a new dataset with data from 91 volunteers, a detailed empirical evaluation of our classification algorithm, the use of linguistic summaries to improve the interpretability of the system (Section 6.6.3), and a solution to the problem of identifying new individuals.

6.2 Beer recommender system

Contributions

This section describes the system called GimmeHop, a knowledge-based and context-aware recommender system. It receives the user preferences for the items that s/he expects to receive (the values of some attributes) and provides a ranked list of beers, taking into account contextual information in order to recommend different

beers under different circumstances.

Most of the features of GimmeHop were already discussed in Section 5.2. In the following, Section 6.2.1 describes our Fuzzy Beer ontology, while Section 6.2.2 reports an evaluation of the quality of the linguistic labels and the recommendations, but also a evaluation about the performance of the system in terms of running time and data traffic.

6.2.1 Fuzzy ontology

Firstly, we proposed an distributed architecture to work with the knowledge mentioned in the Section 5.1.2. For this domain, two files are used, the schema ontology that contains the vocabulary about beers (definition of types and properties). Another file that populates the ontology scheme with beer instances and their attribute values. The distributed architecture is illustrated in Figure 6.11.

Our ontology is encoded in OWL 2 EL [W3C09b], one of the tractable profiles of OWL 2. The ontology schema has 411 axioms (215 logical axioms), 121 classes, 5 object properties, 14 data properties, 10 fuzzy datatypes, and 22 country individuals. There are two additional files populating the schema with individuals (we will give more details later on). The main features of schema ontology are:

Classes. Figure 6.12 (a) shows the main classes of the ontology. The most important ones form a hierarchy of beer types, grouped in 5 main styles: Ale, Lager, Sour, Wheat, and Specialty. These classes are abstract and cannot have instances, but have subclasses (that can be abstract or not). The depth of the subontology with the beer types is 5.

There are also some classes representing a brewery (Brewery), a location (Country), a currency (Currency) to specify the beer price, a won award (Award), and a hierarchy

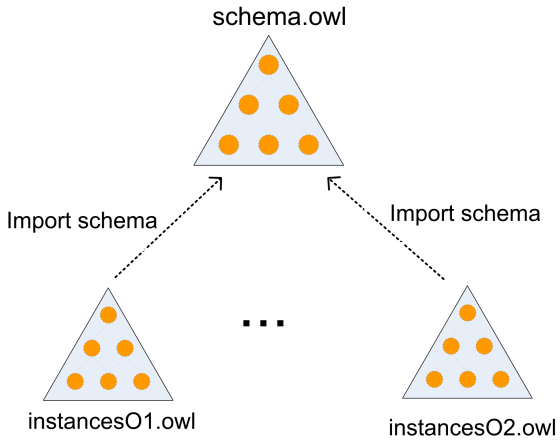
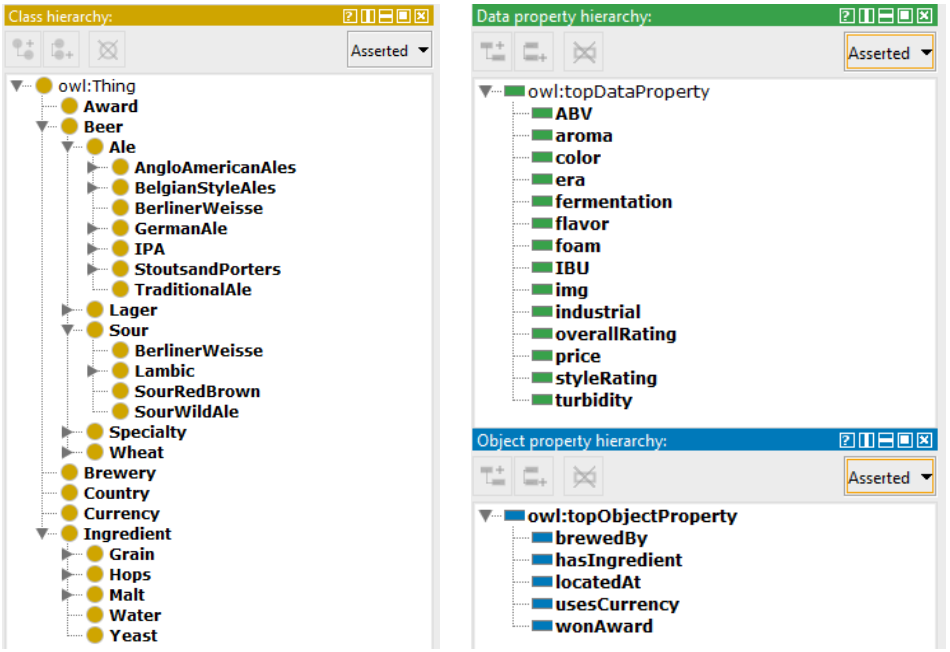


Figure 6.11: Ontology schema and two instance files.

of ingredients (Ingredient).

Our ontology imports *DBpedia* [LIJ+15] and uses its list of countries (e.g., dbpedia:Mexico)[4] and its list of currencies (e.g., dbpedia:Mexican_pesos).



a) b)

Figure 6.12: Ontology representation, (a) classes and (b) data and object properties.

Properties. The main properties of the ontology are depicted in Figure 6.12 (b). Our 5 object properties link instances of Beer or Brewery with instances of Country (locatedAt), and instances of Beer with instances of Award (wonAward), Brewery (brewedBy), Currency (usesCurrency), and Ingredient (hasIngredient). Note that we do not represent their inverse properties, as they are not allowed in OWL 2 EL [W3C09b]. Data properties make it possible to represent different beer attributes:

- ABV (Alcohol By Volume) is a standard measure of how much alcohol is contained in a given volume of an alcoholic beverage. It is measured in $[0, 100]$.
- IBU (International Bitterness Units) denotes the bitterness degrees, measured in $[0, 1000]$.
- color is a numerical value representing the color in Standard Reference Method (SRM) units, measured in $[0, 40]$.

[4]Note that some of them do not belong to the class dbpedia:Country, such as dbpedia:Scotland.

- turbidity is a numerical value representing how hazy a beer is in European Brewery Convention (EBC) units, measured approximately in $[0, 200]$.

- aroma, flavor, and foam represent some information using string values. These properties are not functional so that several values can be attached to a single beer. For example, we can have a data property assertion stating that it smells like bananas and another one stating that it tastes like clove.

- fermentation has the following possible values: top, bottom, any, wild, and aged.

- era indicates if the beer was brewed in a modern style, in a traditional style, or if belongs to a historical period and is no longer available.

- price is a numerical property representing the cost of the beer (recall that the object property usesCurrency indicates the semantics of the number).

- overallRating and styleRating are numerical values in $[0, 100]$ representing the percentile of the rating, compared to all the beers or to the beers of the same style, respectively, given by the community of users.

- industrial is a Boolean property (true indicates a industrial beer, false an artisan one).

- img is the path of an image file with a picture of the beer that could be displayed in the GUI.

In the future, the ontology could be extended with a hierarchy of classes representing aromas, flavors, and foam types; and replacing the data properties with object properties. However, because our aim is to manage fuzzy datatypes and there is no easy way to represent those attributes using fuzzy membership functions defined over a numerical scale, we have left it as future work.

Fuzzy datatypes. The ontology stores precise values using data property assertions (e.g., that the alcohol degree of Corona Extra is 4.5), but also includes 10 fuzzy datatypes, 5 associated to the alcohol and 5 associated to the bitterness. There are more data properties to which one could also associate fuzzy datatypes, such as price, color, or turbidity. However, our beer data did not include that information and hence they were not considered in the current version. Furthermore, we chose not to define fuzzy datatypes for the ratings because in a recommender system the user is interested in items with the best possible rating, as long as they satisfy his/her requirements.

To compute the linguistic values of the data properties of an ontology, we used Datil tool (see Section 3.2). Recall, Datil implements several clustering algorithms such as *k-means*, *fuzzy c-means*, and *mean shift*. Example of fuzzy membership functions built after the centroids (denoted by broken lines) is shown in Figure 6.16.

After our first experiments, we noticed that the results were counter-intuitive because of the existence of beers with very high alcohol values. Figure 6.13 shows the number of beers for each alcohol degree (with 1 digit precision). We can see that there are several beers with more than 20 degrees; one of them (Schorschbräu Schorschbock) with 57.7°. A consequence of having such values is that the centroids used to build the fuzzy membership functions are much higher than the expected values for a human expert. With Datil is possible to specify a minimal and a maximal threshold (denoted Θ_1 and Θ_2, respectively), so that lower and greater values, respectively, are ignored for the clustering algorithm. We will report later some experiments to select the thresholds Θ_1 and Θ_2, and the best clustering algorithm.

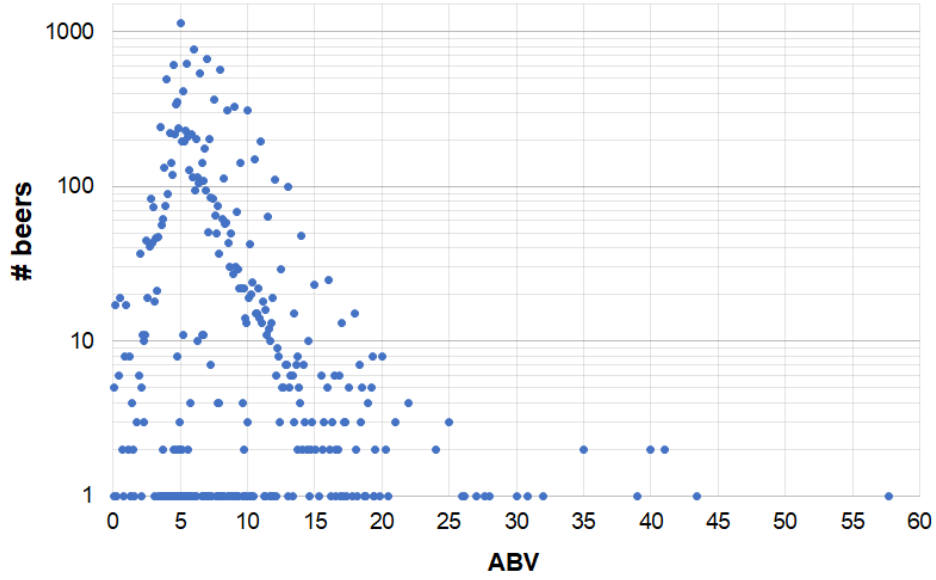


Figure 6.13: ABV of beers.

Class axioms. There are also some axioms imposing some restrictions on the classes. Firstly, there are disjointness axioms stating that two classes cannot share any instance. For example, Ale and Lager are disjoint. Note however that Ale, Sour, and Wheat, three of the five families of beer types, are not disjoint; indeed, BerlinerWeisse is a subclass of those three classes.

We can also find some necessary conditions of the beer types. For example, Lager restricts the value of the property fermentation to be low. Another typical restriction

involves the color, for instance, Schwarzbier restricts the value of the property color to be in $[17, 30]$ (SRM).

There are also a few General Concept Inclusion (GCI) axioms that make it possible to infer that if a beer is brewed by a brewery located in a country (e.g., México), that country should also be associated to the beer, e.g., $\left(\text{brewedBy some (locatedAt value dbpedia:Mexico)}\right)$ SubClassOf (locatedAt value dbpedia:Mexico)[5].

Individuals. A first file (denoted $O1$) populates the ontology with 15317 beer individuals and 4510 brewery individuals. In general, for each beer we know its beer type (membership to a class), its brewery and country (via 2 object property assertions), and the values of the data properties ABV, IBU, img, overallRating, and styleRating (via 5 data property assertions). Figure 6.14 illustrates a sample individual.

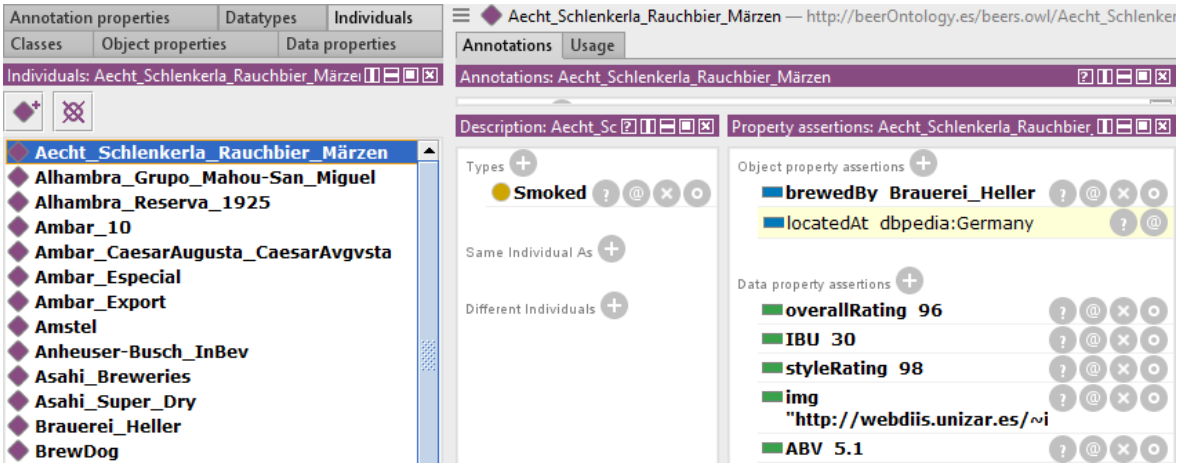


Figure 6.14: Example of a beer individual.

Table 6.8 shows some statistics about the individuals in ontology $O1$. In particular, it includes the number and the percentage of beers for which information about some features (alcohol, bitterness, country, and style rating) is available. While the alcohol degree is always known, this is not the case for other attributes (indeed, bitterness is unknown for 82% of the beers, and the country is unknown for 72% of the beers). Therefore, a beer recommender system needs to take the existence of missing data into account.

Since the number of beers in the ontology is too high, we selected a subset to be used in the evaluation of our system. In particular, we defined another file (denoted $O2$) as a subset of $O1$, with 30 beer individuals and 24 brewery individuals. These beers are likely to be rather popular, to make it easier finding human experts that could evaluate

[5]We use OWL 2 Manchester syntax for readability, but it does not actually provide any support for GCIs.

Data property	# Individuals	% Individuals
ABV	15317	100
IBU	2786	18
country	4365	28
styleRating	14378	94

Table 6.8: Statistics of available values for some features of the fuzzy ontology O1.

them. The first three columns on Table 6.9 detail the selected beers, the alcohol degrees, and the linguistic labels, computed using Datil (we will explain later the procedure to obtain them). The table shows the label with the highest membership degree; note that in the case of Mahou Clásica there are two maxima (the membership degrees to *Low* and *Neutral* are the same ones). The list of beers include many examples from Spain (and, in particular, from Zaragoza), because most of the experts that took part in the evaluation live there. Note also that there are no examples with very low or very high alcohol.

Beer	ABV	Datil label	# OK	% OK	# Valid reply	% Valid reply
Carling	4	Low	10	21.7	23	50
Guinness Draught	4.2	Low	2	4.3	43	93.5
Bud Light	4.2	Low	13	28.3	33	71.7
Pilsner Urquell	4.4	Low	11	23.9	32	69.6
Mort Subite Kriek	4.5	Low	4	8.7	24	52.2
Coronita	4.5	Low	18	39.1	46	100
Mahou Clásica	4.8	Low-Neutral	36	78.3	37	80.4
Quilmes Cristal	4.9	Neutral	18	39.1	33	71.7
Cruzcampo Premium Lager	5	Neutral	17	37	40	87
Amstel	5	Neutral	27	58.7	41	89.1
Asahi Super Dry	5	Neutral	8	17.4	25	54.3
Budweiser	5	Neutral	22	47.8	43	93.5
Franziskaner Hefe-Weissbier	5	Neutral	12	26.1	41	89.1
Heineken	5	Neutral	24	52.2	45	97.8
Ámbar CaesarAugusta	5.2	Neutral	9	19.6	29	63
Ámbar Especial	5.2	Neutral	24	52.2	38	82.6
Mahou 5 Estrellas	5.5	Neutral	26	56.5	39	84.8
BrewDog Punk IPA	6	Neutral	9	19.6	24	52.2
Alhambra 1925	6.4	Neutral	13	28.3	38	82.6
Hijos de Rivera 1906 Extra	6.5	Neutral	9	19.6	32	69.6
Leffe Blonde	6.6	Neutral	11	23.9	37	80.4
Ámbar Export	7	Neutral	9	19.6	33	71.7
Chimay Rouge	7	Neutral	9	19.6	27	58.7
Voll Damm	7.2	Neutral	2	4.3	38	82.6
Paulaner Salvator	7.9	Neutral	10	21.7	32	69.6
Pauwel Kwak	8.4	High	4	8.7	14	30.4
Delirium Tremens	8.5	High	12	26.1	27	58.7
Judas	8.5	High	16	34.8	33	71.7
Chimay Bleue	9	High	10	21.7	25	54.3
Ámbar 10	10	High	6	13	22	47.8

Table 6.9: List of beers in fuzzy ontology O2 and replies of our experts.

6.2.2 Evaluation

In this part we firstly report our evaluation of the quality of the linguistic labels, then an evaluation of the running time, and next the traffic data. Finally, we discuss the overall behavior of the system with the help of some sample queries.

Evaluation of the linguistic labels. As already mentioned, Datil offers several choices to compute the linguistic labels associated to the alcohol and to the bitterness. We will describe the evaluation of the quality of the results given by different clustering algorithms and parameters.

We invited some beer aficionados to evaluate the linguistic labels associated to the 30 beers in $O2$ and got 46 answers. We designed a webpage where each expert was asked to classify the alcohol of each beer using the following scale:

$$\{NoReply\ (0), VeryLow\ (1), Low\ (2), Neutral\ (3), High\ (4), VeryHigh\ (5)\}$$

Experts were specifically asked to select *NoReply* if they had never tried the beer or, more generally, did not feel qualified to answer properly (in the following, we will use the term valid answers to exclude no replies). They did not know the information about the numerical value of the ABV, they just answered according to their user experiences. Beers were presented sequentially, only one at a time. We restricted to the alcohol level, as we think that bitterness is much harder to evaluate, in particular if the answers are not given during a beer tasting. Then, we compared the answers given by the experts with the results given by Datil, selecting the best match.

Datil currently supports 3 clustering algorithms. For k-means and fuzzy c-means we tried with different number of clusters, namely 5 and 7. In mean shift the number of labels is not an input parameter; in all cases the value turned out to be 5. The value of Θ_1 was always 0; the values of Θ_2 were in $\{14, 15, 17, 20\}$. Figure 6.15 shows the number of beers per each group of alcohol degree for O_1 ontology; the biggest group is between 5°and 14°, with 9882 beers. Recall that Figure 6.13 shows the distribution of the ABV of beers.

For each clustering algorithm, we take into account two measures:

- The number of coincidences, i.e., the cases when both the expert and Datil gave the same classification.

- The distance between the classification given by the expert and Datil. For instance, if an expert classifies some beer as having *Low* alcohol and Datil chose *High*, the distance is $abs(2 - 4) = 2$.

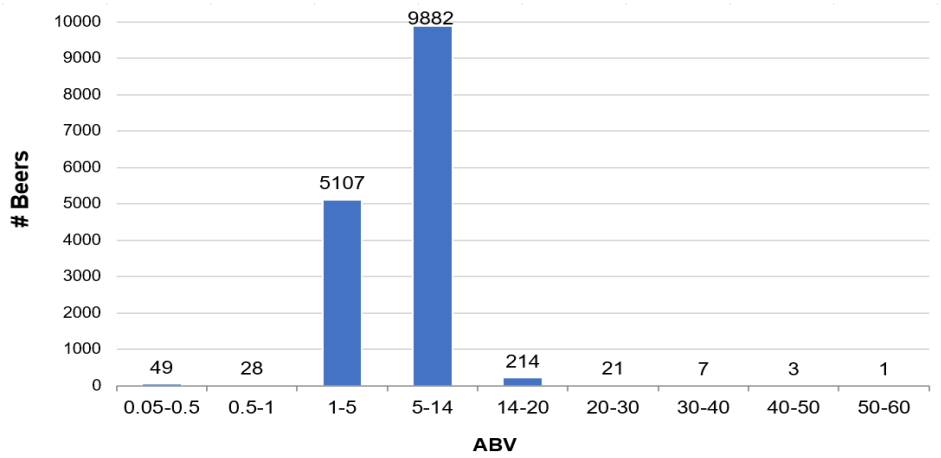


Figure 6.15: Alcohol level of beers.

Note that some answers are not directly comparable, as the expert scale has 5 linguistic labels and some clustering algorithms produced 7. To compute the distance in such cases, we merged the two lowest values and the two greatest ones (that is, we merged $VeryVeryLow$ and $VeryLow$, as well as $VeryVeryHigh$ and $VeryHigh$).

Table 6.10 shows the results: it includes the absolute number of coincidences (# OK), the percentage of coincidences out of the valid answers (% OK), the absolute sum of distances (Distance) and the average distance (Avg. distance, where the sum is divided by the number of evaluated beers and the number of valid answers). In all cases the best results are obtained by using mean shift with $\Theta_2 = 15$; the linguistic labels corresponding to that case are depicted in Figure 6.16(a). For the sake of completeness, Figure 6.16(b) shows the linguistic labels associated to the bitterness (IBU). The best average distance (0.779) could seem too high at first sight, but the use of fuzzy logic ensures that there is usually a non-zero membership degree to the precedent and the subsequent linguistic labels, so the set of linguistic labels behaves well in practice.

The four latter columns of Table 6.9 show, for each beer in O2, the number (absolute and percentage) of coincidences and the number (absolute and percentage) of valid answers, respectively, for the best clustering algorithm (mean shift with $\Theta_2 = 15$). We can see that there are beers not as popular as expected (such as Pauwel Kwak, with 30.4% of valid answers), and also popular beers with a small percentage of coincidences, such as Guinness (4.3%). Guinness is an example of counter-intuitive result: the majority of experts thought that it had a high ABV, which is not the case (4.2°).

Evaluation of the running time. Here, we evaluate the running time of GimmeHop, both in the local and in the remote modes. We also try to determine the maximal number of individuals that are acceptable from the perspective of user

Clustering	Θ_2	# Labels	# OK	% OK	Distance	Avg. distance
K–means	14	5	113	11.4	1598	1.606
	14	7	71	7.1	1888	1.897
	15	5	233	23.4	1138	1.144
	15	7	283	28.4	1009	1.014
	17	5	290	29.1	940	0.945
	17	7	88	8.8	1753	1.762
	20	5	110	11.1	1634	1.642
	20	7	65	6.5	1960	1.970
Fuzzy c–means	14	5	272	27.3	997	1.002
	14	7	177	17.8	1437	1.444
	15	5	285	28.6	995	1
	15	7	174	17.5	1456	1.463
	17	5	279	28	1032	1.037
	17	7	181	18.2	1432	1.439
	20	5	245	24.6	2416	2.428
	20	7	98	9.8	1830	1.839
Mean shift	14	5	286	28.7	936	0.941
	15	5	**401**	**40.3**	**775**	**0.779**
	17	5	386	38.8	779	0.783
	20	5	212	21.3	1181	1.187

Table 6.10: Results for different clustering algorithms and parameters.

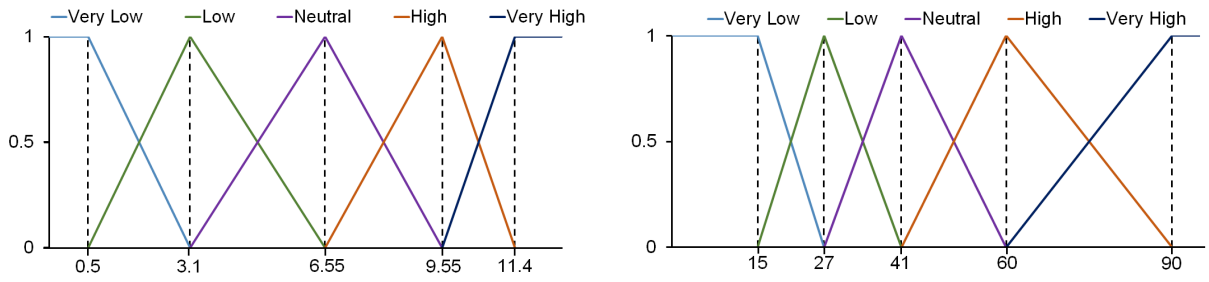


Figure 6.16: Linguistic labels (using Datil and mean-shift) for (a) ABV and (b) IBU.

experience.

We considered two mobile devices for these experiments: a tablet (denoted as A1) and a smartphone (denoted as A2). The tablet A1 is a Lenovo Yoga 2 10.1 (Android 5.0, Quad-core 1.86 GHz, Intel Atom Z3745, 2 GB RAM, released in 2014). The smartphone A2 is a ZTE Blade A610 (Android 6.0, Quad-Core 1.3 GHz ARM Cortex A-53, 2 GB RAM, released in 2016). We also used Amazon Web Services (AWS) and created an instance (denoted S1). S1 is a Ubuntu server 16.04 LST, amd64 xenial image, general purpose type t2.micro, 1 CPU and 1 GB RAM, located in the EU region (Paris). The versions of the semantic reasoners were Hermit 1.3.8 and TrOWL 1.5.

We considered two advanced searches ($Q1$ and $Q1'$), a basic search ($Q2$), and a similarity search ($Q3$). There are two advanced searches because the first one ($Q1$) might be significantly slower than the next ones (such as $Q1'$) . We also separated the loading time, as it is only necessary once. Note that this task could be run when the server starts and not when the client starts.

Firstly, Figure 6.17 shows the results on the server for both reasoners. Of course, remote reasoning is much faster than local reasoning. We can see that there are significant differences for both reasoners. TrOWL is much faster, although it does not support exact reasoning on OWL 2 (for more expressive languages than OWL 2 EL, reasoning is approximate). We can see that indeed the first advanced query ($Q1$) is slower than the next one ($Q1'$). Furthermore, the advanced query is not always more complex than $Q2$ and $Q3$. Using TrOWL it is possible to get the answer to $Q1$ about $O1$ in 1.5 seconds (not including the loading time), and the next one in 0.9 seconds. Of course, times are much faster on smaller ontologies, such as $O2$.

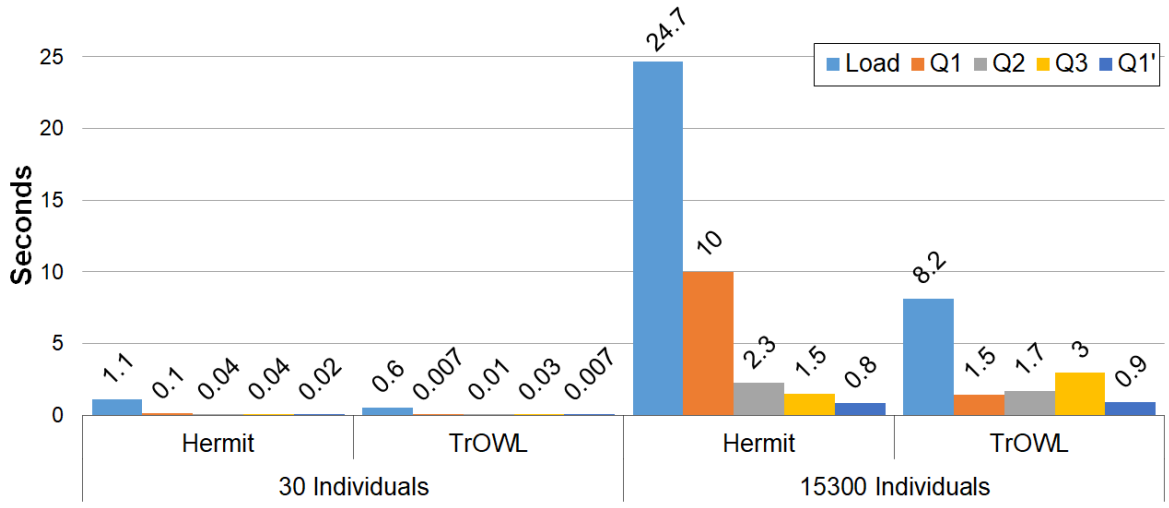


Figure 6.17: Running time of Load and all queries on S1.

We will show now the results on the Android devices. In this case, we additionally

tested further subontologies of $O1$ with different numbers of individuals, between 30 and 15300. Figure 6.18 and 6.19 show the results of $Q1$ and $Q1'$ on A1 and A2 using HermiT and TrOWL, respectively. Similarly, Figure 6.20 and 6.21 show the results for $Q2$ and $Q3$. Again, TrOWL was always faster than HermiT, and device $A2$ was always slower than $A1$ (the differences are much higher when using HermiT). There are some missing data because local reasoning was not possible with ontologies with 10000 individuals or more. As expected, $Q1$ was slower than $Q1'$, $Q2$, and $Q3$. When using TrOWL and the ontology with 250 individuals there was a strange outlier on both $A1$ and $A2$, as this ontology is a superset of $O2$ and a subset of the ontology with 500 individuals.

Figure 6.22 summarizes the result of the first advanced query (including the loading time) for the three devices, the two reasoners, and different ontology sizes. We could say that remote reasoning is feasible even with all the individuals, but local reasoning requires a smaller number. In such case, we must check that the latency of the first query is not much longer than the average time that mobile users are willing to wait. On $A2$ it seems feasible handling up to 2000 beer individuals; the first advanced query might take almost 16 seconds, but the next ones take less than 2 seconds. On $A1$ it seems feasible handling up to 3000 beer individuals.

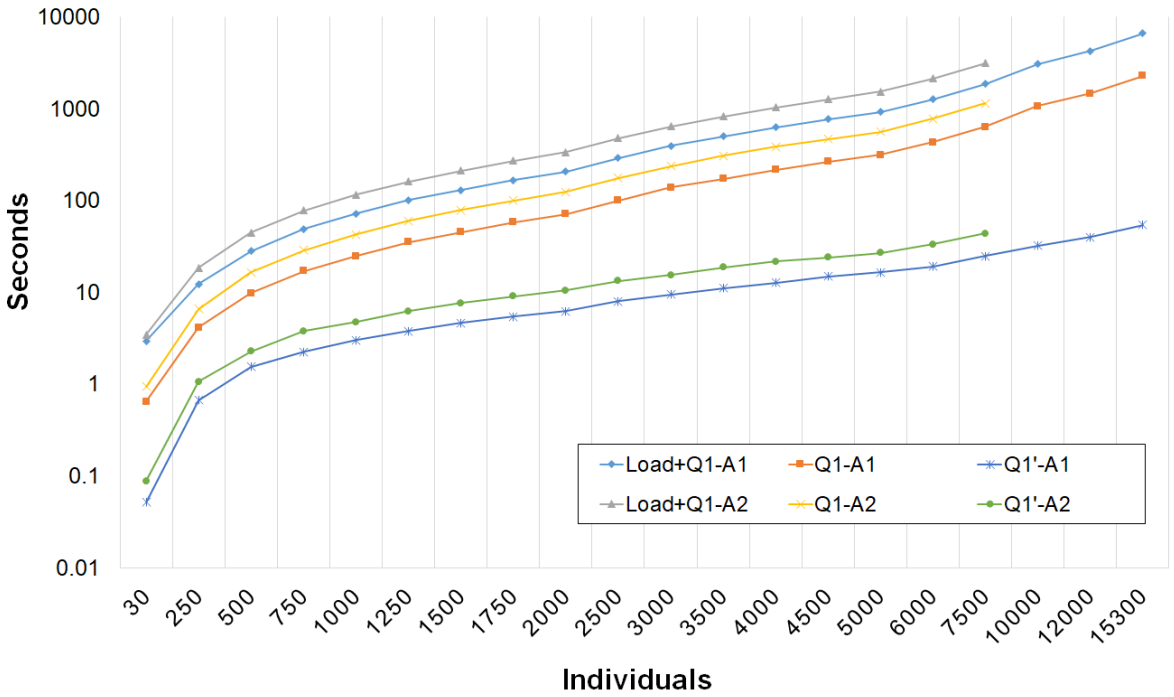


Figure 6.18: Running time of Load+Q1, Q1 and Q1' on HermiT.

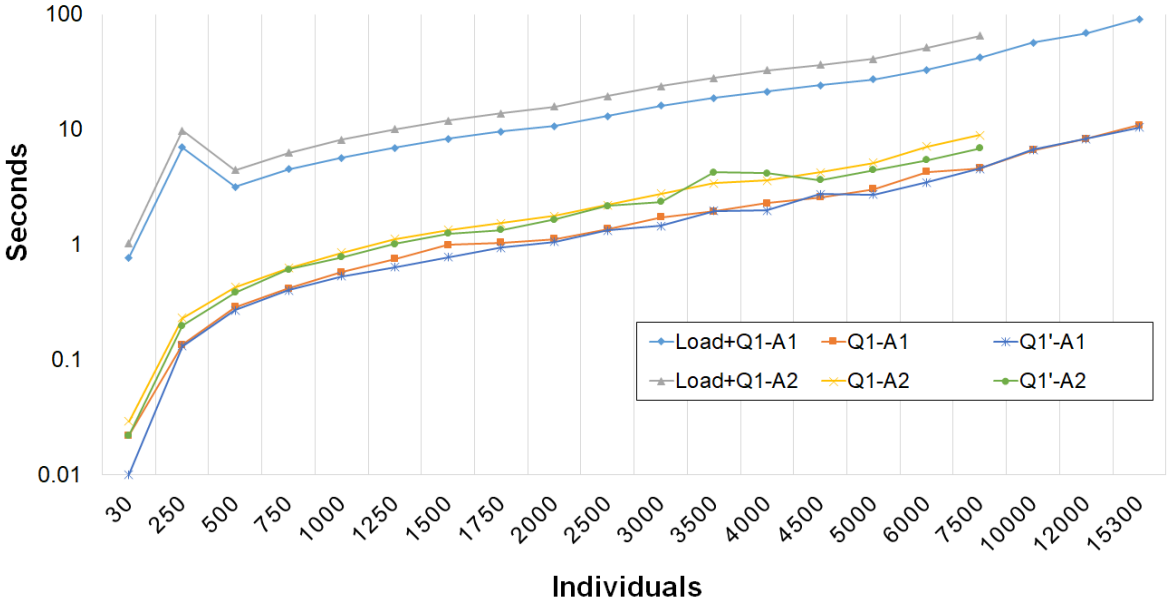


Figure 6.19: Running time of Load+Q1, Q1 and Q1' on TrOWL.

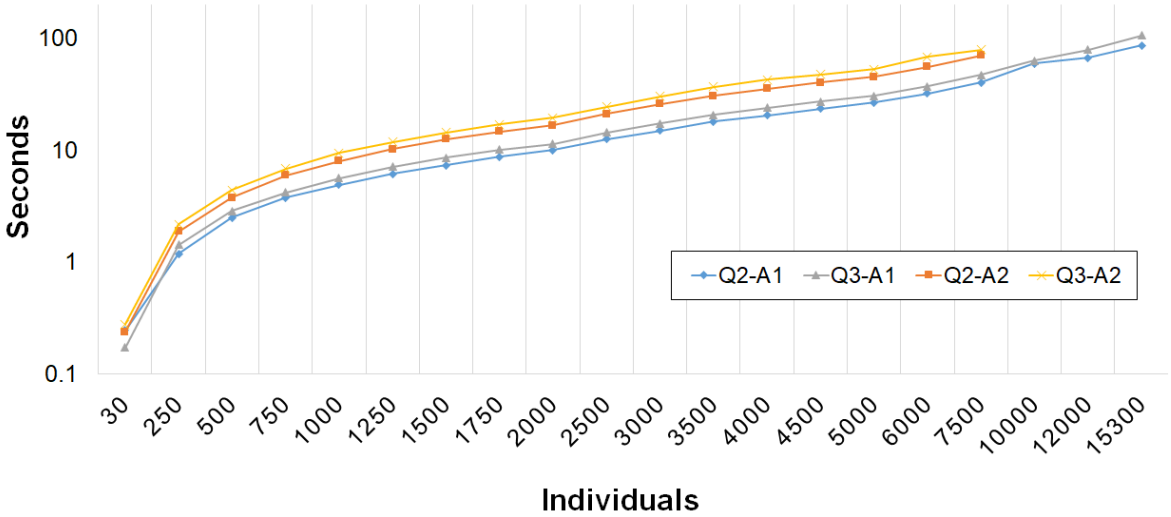


Figure 6.20: Running time of Q2 and Q3 on HermiT.

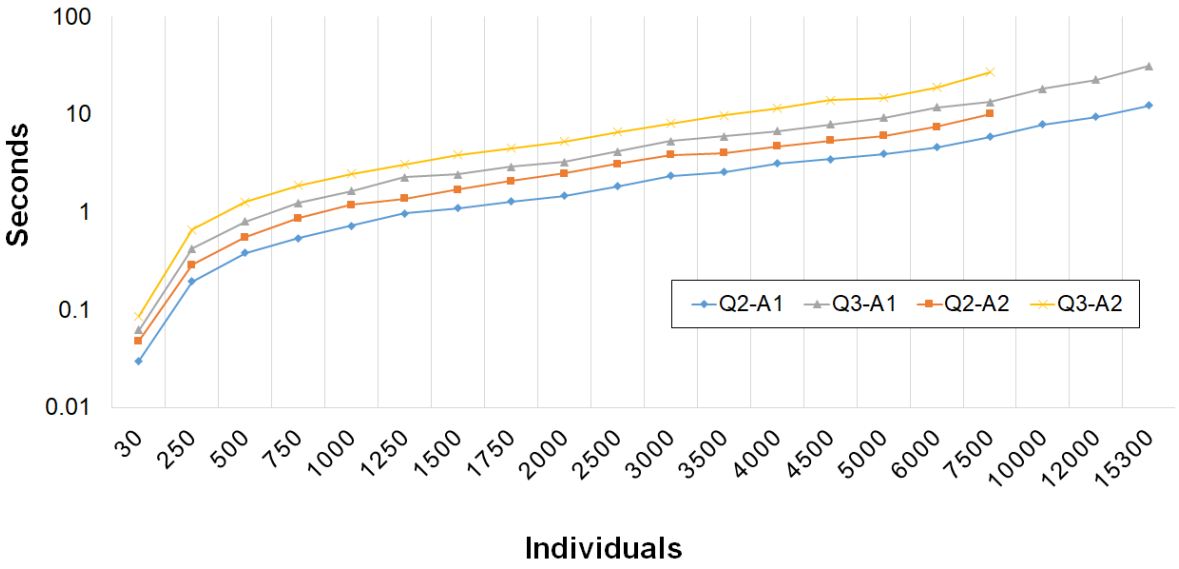


Figure 6.21: Running time of Q2 and Q3 on TrOWL.

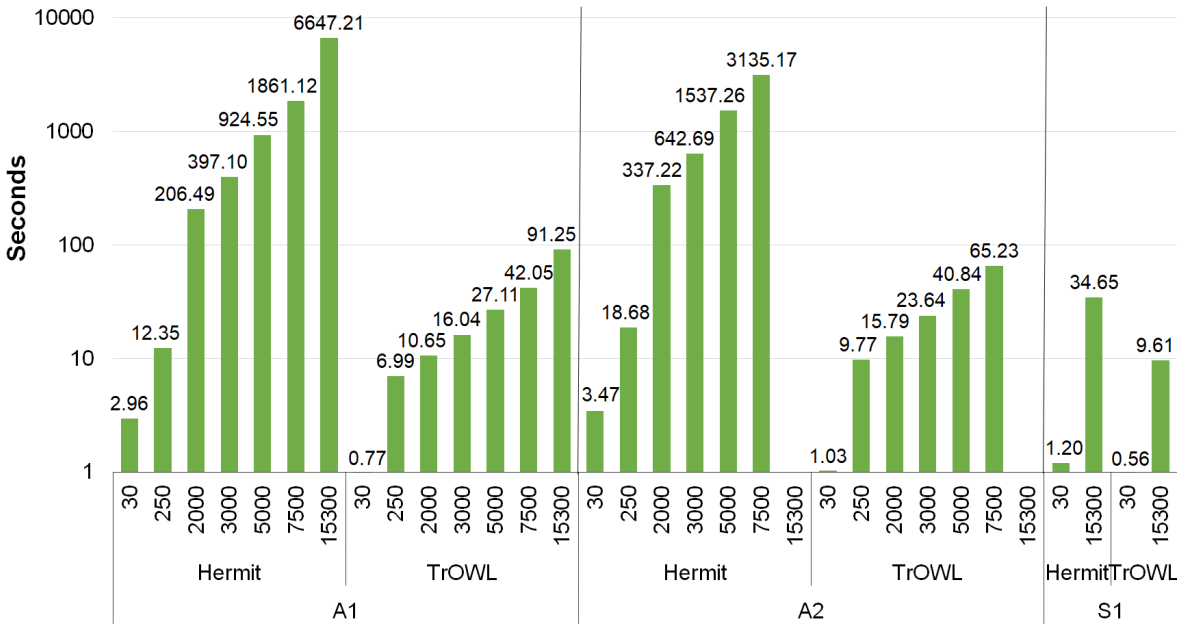


Figure 6.22: Running time of Load+Q1 on all devices and reasoners.

Evaluation of the data traffic. Users of apps requiring Internet connection are very often concerned of the data traffic, so we decided to investigate the cost of GimmeHop when using a remote reasoner.

In order to measure the data traffic of our application, we use the "Data Usage Monitoring" (DUM)[6] free app and the app-info utility of Android. The DUM makes it possible to analyze the data in a deeper way, as it provides not only mobile data but also Wi–Fi data, and makes it possible to obtain both sent and received data.

We considered the remote reasoning approach, and two ontologies with 150 ($O3$) and 15317 beers ($O1$). The reason to define $O3$ is that the size of $O2$ is appropriate for the evaluation of the linguistic labels, but it seems not enough to evaluate the data traffic.

We considered the following sequence of queries:

- Firstly, we considered an advanced query. We selected the most populated families and half of the times no other specification about the other attributes (ABU and IBU). This led to the worst case, when the number of results that the server sends back to the user is maximal.
- Then, we submitted a basic query, asking to retrieve a beer from its name. We also clicked on the result to display the beer information, including its image.
- Finally, we asked to retrieve similar beers to that one.

This sequence was repeated 12 times and we computed the average values. Table 6.11 presents the results of our evaluation in KB. The results show that the three methods used to measure the traffic coincide, and we think that the results are acceptable even for the very large ontology $O1$ (recall that the times include 3 queries).

# Individuals	Wi–Fi	DUM	App–info
150	56.33±6.11	54.85±6.27	54.83±6.19
15317	132.57±1.5	132.19±0.69	132.5 ±0.8

Table 6.11: Data traffic for a sequence of 3 queries (KB).

Recall that the server currently only sends the top k results back, with $k = 200$, and note that data traffic could be optimized by making k a configurable parameter and setting smaller values.

We also noticed that selecting the value of the attributes in the advanced search did not make an impact in our cases, although this is not always the case. In ontology $O1$,

[6] `http://play.google.com/store/apps/details?id=com.jsk.datausagemonitor`

the number of individuals was 200 even after restricting the attributes. In ontology $O2$, the number of individuals could be slightly different but without changes in the total data traffic.

When using local reasoning, retrieving the picture of a beer is the only data-consuming operation. We measured that the average traffic data is about 10.3 KB, although there is a big standard deviation because pictures can have very different file sizes.

Evaluation of the system and query examples. Table 6.12 shows 3 new examples of advanced queries. For each of them, the degree of satisfiability of each beer is shown in the two last columns (both when the user location is unknown and when the user is located in Belgium, respectively). In all three queries the user asks for an Ale beer, with a neutral alcohol and a neutral bitterness. The difference is that in the two first queries the user selects a preference in the most important attribute (alcohol and rating, respectively), so a weighted mean is used to combine the information. However, in the third query no attribute is selected as the most important, so an OWA operator is used.

Beer	ABV	IBU	Rating	Country	Partial degrees	Degree Country=?	Degree Country=BE
Query (WMEAN): ABV=Neutral, IBU=Neutral, Style=Ale, Important property=Alcohol							
Leffe Blonde	6.6	-	90	Belgium	[abv=0.98, rating=0.9]	0.95 (1st)	0.97 (1st)
Chimay Rouge	7	-	100	Belgium	[abv=0.85, rating=1]	0.91 (2nd)	0.95 (2nd)
Pauwel Kwak	8.4	-	90	Belgium	[abv=0.38, rating=0.9]	0.59 (4th)	0.77 (3rd)
Delirium Tremens	8.5	-	92	Belgium	[abv=0.35, rating=0.92]	0.57 (5th)	0.76 (4th)
Chimay Bleue	9	-	100	Belgium	[abv=0.18 , rating=1]	0.51 (7th)	0.71 (5th)
BrewDog Punk IPA	6	60	69	Scotland	[abv=0.84, rating=0.69, ibu=0]	0.67 (3rd)	0.67 (6th)
Judas	8.5	-	37	Belgium	[abv=0.35 , rating=0.37]	0.35 (8th)	0.6 (7th)
Guinness Draught	4.2	-	87	Ireland	[abv=0.31, rating=0.87]	0.53 (6th)	0.53 (8th)
Query (WMEAN): ABV=Neutral, IBU=Neutral, Style=Ale, Important property=Rating							
Chimay Rouge	7	-	100	Belgium	[abv=0.85, rating=1]	0.94 (1st)	0.97 (1st)
Leffe Blonde	6.6	-	90	Belgium	[abv=0.98, rating=0.9]	0.93 (2nd)	0.97 (2nd)
Pauwel Kwak	8.4	-	90	Belgium	[abv=0.38, rating=0.9]	0.69 (3rd)	0.83 (3rd)
Delirium Tremens	8.5	-	92	Belgium	[abv=0.35, rating=0.92]	0.69 (4th)	0.83 (4th)
Chimay Bleue	9	-	100	Belgium	[abv=0.18, rating=1]	0.67 (5th)	0.82 (5th)
Guinness Draught	4.2	-	87	Ireland	[abv=0.31, rating=0.87]	0.65 (6th)	0.65 (6th)
Judas	8.5	-	37	Belgium	[abv=0.35, rating=0.37]	0.36 (8th)	0.6 (7th)
BrewDog Punk IPA	6	60	69	Scotland	[abv=0.84, rating=0.69, ibu=0]	0.6 (7th)	0.6 (8th)
Query (OWA): ABV=Neutral, IBU=Neutral, Style=Ale,Important property=Indifferent							
Leffe Blonde	6.6	-	90	Belgium	[abv=0.98 rating=0.9]	0.93 (1st)	0.97 (1st)
Chimay Rouge	7	-	100	Belgium	[rating=1 abv=0.85]	0.91 (2nd)	0.95 (2nd)
Pauwel Kwak	8.4	-	90	Belgium	[rating=0.9 abv=0.38]	0.59 (3rd)	0.77 (3rd)
Delirium Tremens	8.5	-	92	Belgium	[rating=0.92 abv=0.35]	0.58 (4th)	0.76 (4th)
Chimay Bleue	9	-	100	Belgium	[rating=1 abv=0.18]	0.51 (7th)	0.71 (5th)
Judas	8.5	-	37	Belgium	[rating=0.37 abv=0.35]	0.36 (8th)	0.6 (6th)
Guinness Draught	4.2	-	87	Ireland	[rating=0.87 abv=0.31]	0.54 (5th)	0.54 (7th)
BrewDog Punk IPA	6	60	69	Scotland	[abv=0.84 rating=0.69 ibu=0]	0.52 (6th)	0.52 (8th)

Table 6.12: Results for 3 sample queries.

The first column of Table 6.12 includes the name of the beer. Columns 2–5 include the values of some features (ABV, IBU, style rating, and country); note that some values are missing. These values are the same ones for each query, but the order of the beers might be different. Column 6 includes the values to be aggregated, that is, the

membership degrees to the fuzzy membership functions defined over ABV and IBU, and the normalized rating.

Note that user preference indeed plays a role when ordering the beers. For instance, the best beer for the first query is Leffe Blonde, but the best beer for the second one is Chimay Rouge. Note that the user location also plays a role in the recommendation. For example, in the first query BrewDog Punk IPA drops from the third position to the sixth one when taking into account the user location.

We also tried three fuzzy quantifiers described using a right-shoulder, a linear, and a power function. It seems that the best results were obtained using the right-shoulder. For example, an effect of the power function is that the weight vector is always ordered in increasing order (if $q < 1$) or decreasing order (if $q > 1$).

We carefully checked this and other similar examples and concluded that the behavior of the system is reasonable when providing the recommendations. The final degrees might be too small for the user, but the system is effective at providing an ordering among the beers. In several cases where the user was not happy with the result, the reason was that s/he was wrong about the real ABV/type of the beer.

Related work

This section reviews some related work. Our aim is to highlight our contribution with the previous work on the domain (beers) and fuzzy semantic apps.

Beer ontologies and intelligent applications. There is a previous effort to build a Beer ontology[7]. However, the ontology only contains 19 beer types and 9 beers. Another limitation is that the only existing axioms are subclass/subproperty axioms. On the contrary, we impose some conditions on the beer types (such as necessary conditions or concepts disjointness), and include class assertions (representing beers, breweries, etc.), data property assertions representing attributes of each beer instance, and fuzzy datatype definitions allowing to deal with linguistic definitions of some attributes.

Another relevant work is the use of Artificial Intelligence to develop new beers or optimize existing ones [BPKP21]. In particular, the authors created $10,000$ new beer recipes using machine learning techniques. As a proof of concept, they crafted Deeper, the first beer built using an Artificial Intelligence recipe.

The automatic classification of beer styles has also been addressed by different authors [ACMM21, CCF17]. It is worth to stress that these approaches use fuzzy logic (fuzzy rules) but not ontologies.

[7] http://dbs.uni-leipzig.de/files/coma/sources/fd/beer.owl

There are many examples of the interest of the beer industry in Artificial Intelligence. To mention some examples:

- IntelligentX company use algorithms, machine learning, and customer preferences to adjust their beer recipes[8],
- IBM enhances the beer manufacturing line using data collected and analyzed by Watson IoT platform in the Sugar Creek Brewing project[9], and
- Carlsberg Research Laboratory works on a sensor platform using advanced analytics and intelligent cloud technology to get a better flavour and new fermentation organisms[10].

Instead, our approach offers a semantic recommender system using fuzzy ontologies, a novel minimalist reasoning algorithm, and a mobile application based on the user location that supports incomplete knowledge.

Semantic apps using fuzzy logic. MoveUp is a quite recent Android app that categorizes users according to the their activity [ST22]. The approach uses fuzzy logic and focuses on the mHealth domain. The app uses fuzzy IF-THEN rules to compute the profile from different input variables, where different types of user characteristics (physical, psychological, and social) are described using fuzzy sets. The app was evaluated on real scenarios during the COVID-19 pandemic [SST22]. Unlike our work, MoveUp does not use ontologies or semantic reasoning, and that the fuzzy membership functions are not automatically learned from real data.

We have extended the first version of Gimmehop recommender beer system described in [Ale17]. The author designed an initial beer ontology and populated all the beers. However, we consider a more complex ontology with more classes, roles, axioms and updated fuzzy datatypes for ABV. Also, we use a new architecture to support larger files. The initial work implemented a remote version supporting basic, similar and advanced queries, and a prototype for local reasoning using only HermiT reasoner. In contrast, we extended the advanced query implementing the minimalist algorithm (employing OWA and diverse fuzzy quantifiers when data are missing) and completely support both local and remote modes (with HermiT and TrOWL reasoners

[8] https://www.forbes.com/sites/bernardmarr/2019/02/01/how-artificial-intelligence-is-used-to-make-beer/?sh=3c0472d570cf

[9] https://www.ibm.com/blogs/think/2019/04/ai-and-iot-help-perfect-the-brew-at-sugar-creek-brewing-company/

[10] https://www.carlsberggroup.com/newsroom/carlsberg-research-laboratory-behind-beer-research-project-based-on-artificial-intelligence/

being supported). Another important contribution of the present work is the empirical evaluation of Gimmehop in both modes, offering an available local reasoning for fuzzy ontologies.

To the best of our knowledge, there is only one previous application of fuzzy ontologies working on mobile devices: a wine recommender system [MMWHVC16]. The authors represent wine attributes such as price, alcohol level, sugar, or acidity using fuzzy membership functions. Then, a Java application uses fuzzyDL reasoner [BS16a] to solve instance retrieval queries, where the output is a list of wines that satisfy some features combined using an OWA operator. This application is stored on a server and can be accessed from an Android app. However, there are several differences with our approach. The main one is that the authors do not use a semantic reasoner running on a mobile device but require it to be stored on an external server. On the contrary, we support both reasoning mechanisms: using a local reasoner or storing it on an external server. Furthermore, the authors require a concrete fuzzy ontology reasoner, while we can use any standard OWL 2 EL reasoner (of course, we need further computations to take care of the fuzzy part). We also take into account user preferences (by supporting weighted mean aggregation in addition to OWA) and manage the user context in a different way (by using fuzzy hedges). Last but not least, we address the problem of dealing with incomplete data by using qualified-guided OWA. It is worth to mention that the wine recommender system includes a procedure to reach a consensus between multiple users that could also be adopted in our app.

6.3 Blockchain smart contracts

Motivation

In recent years, there is a growing interest in the use of the *blockchain* paradigm in distributed transactional applications, including payments using cryptocurrencies, electronic voting, or managing medical histories [PMM+18]. While in traditional distributed transactional scenarios a trusted intermediary is needed, in the blockchain paradigm this is replaced by the use of a consensual distributed protocol. This protocol makes it possible to guarantee that the transactions, grouped in blocks, are stored in a verifiable and permanent way. Blockchain is a data structure composed by a linked list (or chain) of blocks using cryptographic tools, so that it is not possible to modify data already stored in the blockchain. In particular, each block has a hash value that depends both on the own contents of the block and on the hash of the predecessor block in the chain.

One of the most popular applications of the blockchain are cryptocurrencies.

In particular, Bitcoin[11] was the first blockchain. Another popular blockchain is *Ethereum* [Woo14][12]. Ethereum is based on a cryptocurrency called Ether (ETH), with a subunit Wei (1 ETH = 10^{-18} Wei). Ethereum includes networks with real money converted into Ethers, but also test networks (or testnets) with virtual Ethers, like *Rinkeby*[13].

A key feature of the blockchain paradigm are *smart contracts*. A smart contract SC is a piece of software that automatically processes the terms of a contract. For example, it can control cryptocurrencies (like ETH) or other valuable digital assets. SCs can be encoded in a procedural (imperative) or logical (declarative) language. They include a collection of rules (constraints) that are validated, in such a way that every part that executes the contract gets the same result. The SC can be agreed (in this case, typically, new transactions are added to the blockchain) or refused.

A use case where agreements are necessary can be found in the online shopping activity which has a huge demand on website and apps in 2019 and the first semester of 2020. For example, the current online marketplaces are a 56% of online sales and are estimated to be a 67% of global e-commerce sales by 2022. At the same time, shopping using mobile devices (m-commerce) is popular among consumers, and by 2021 the m-commerce is expected to be a 54% of total online sales[14].

Note that all the constraints in a smart contract are hard, so they must be fully satisfied. Instead, it could be interesting to replace some of them with soft constraints, so that they can be partially satisfied, and there is a partial agreement between the involved parts. For example, in an electronic commerce scenario, the seller and the customer could define their desired delivery time using a right-shoulder and a left-shoulder function, respectively. The longer the delivery time, the more the seller is satisfied, and the shorter the delivery time, the more the buyer is satisfied. Sometimes one cannot find a solution that completely satisfies both parts, but it is often possible to find a partial agreement, where the delivery time is acceptable for everybody.

Contributions

In this section, we show how to extend existing blockchain systems by using fuzzy ontologies. Firstly, this makes it possible to add knowledge into the process, taking profit of the advantages of ontologies, such as promoting reuse and interoperability or avoiding disambiguations. More importantly, this makes it possible to make smart contracts more flexible, including terms represented using fuzzy sets that can be

[11] `http://bitcoin.org`
[12] `http://www.ethereum.org`
[13] https://www.rinkeby.io
[14] `https://www.europarl.europa.eu/thinktank/en/home.html`

partially satisfied, leading to partial agreements among two or more involved parts (thanks to our matchmaking algorithm between individuals proposed in Section 4.4).

The remainder of this section is organized as follows. First, in Section 6.3.1 we describe the ontologies used to represent the knowledge. Then, Section 6.3.2 proposes an architecture to find partial agreements on an Ethereum blockchain for e-commerce scenarios.

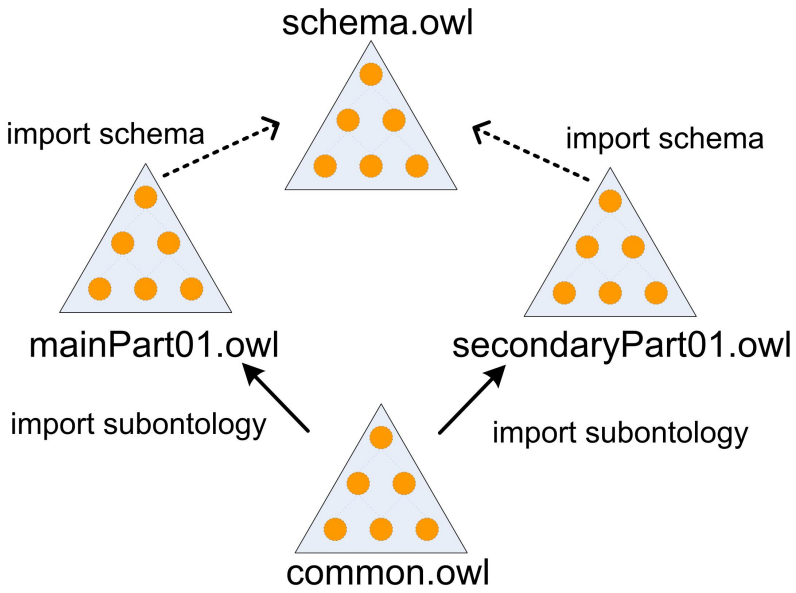


Figure 6.23: Ontology schema and instances files.

6.3.1 Ontologies

Our proposal is based on four types of fuzzy ontologies (using our distributed architecture in Section 5.1.2), as illustrated in Figure 6.23:

– *Schema fuzzy ontology.* It contains the vocabulary of the domain, such as classes, properties, or range definitions. For example, the price and the delivery time.

  An excerpt of the schema ontology is shown in Figure 6.24, where classes are denoted in yellow, object properties in blue, and data properties in green. The main classes are Contract, Transaction, Product, MainPart, and SecondaryPart. The hierarchy of class and properties is shown in Figure 6.25. It is important to mention that the data properties linked to a contract or a transaction are always present, but product attributes depend on the application.

– *Main part ontology.* It includes the personal definitions of the *main* part of the contract (e.g., the seller of a product). This ontology imports the schema ontology and populates it. For example, the seller part can define the car price.

The personal definitions are attributes or restrictions of a good/service and could be flexible or hard. Flexible restrictions are represented in the form $\exists T.\mathbf{d}$, where T is a data property and $\mathbf{d}$ is a fuzzy datatype. Hard restrictions could be defined using singleton crisp sets.

– *Secondary part ontology.* It is similar to the previous one, but includes the definitions of the *secondary* part of the contract. For example, the customer part can define the price of the car to buy.

– *Common ontology.* It includes only the personal definitions of each part (main and secondary) that are important for a transaction. The other ontologies are not imported as usual, but the relevant information is physically stored in the ontology to make it self-contained.

Actually, our fuzzy ontology model does not restrict to just having one main part and one secondary part. We require that there are at least two parts, but there can be zero or more main parts, and zero or more secondary parts.

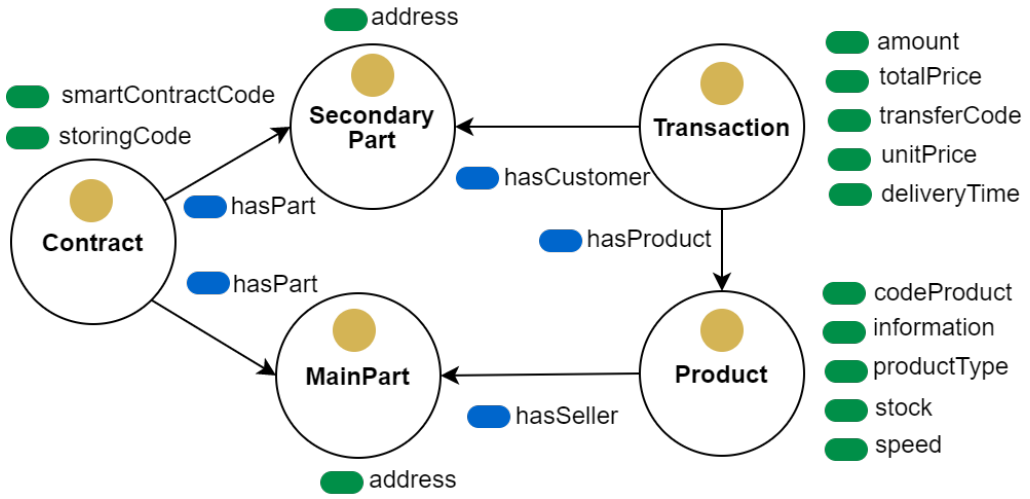


Figure 6.24: An excerpt of our ontology schema

6.3.2 Architecture

We focus on the specific case of smart contracts managing transactions where ether is transferred from a secondary part to a main part. Our smart contracts execute the terms of a contract: they firstly check if there is a (possibly partial) agreement between the involved parts, i.e., if all their constraints can be (possibly partially) satisfied. In that case they actually perform the transaction with the parameters that maximize the mutual satisfaction. We assume that both parts have already agreed on the codeProduct, e.g., the event for which a ticket is being sold is fixed.

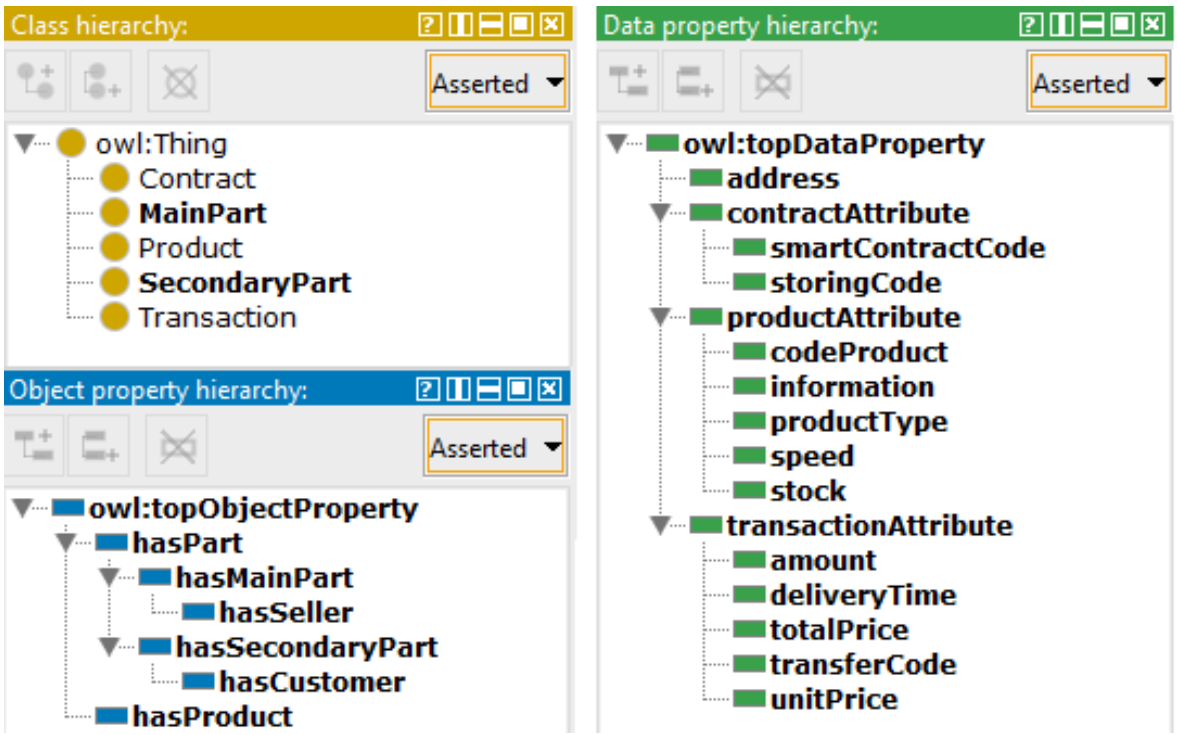


Figure 6.25: Classes and properties in the ontology schema.

The complete architecture is detailed in Figure 6.26. Let us now detail the steps of the process.

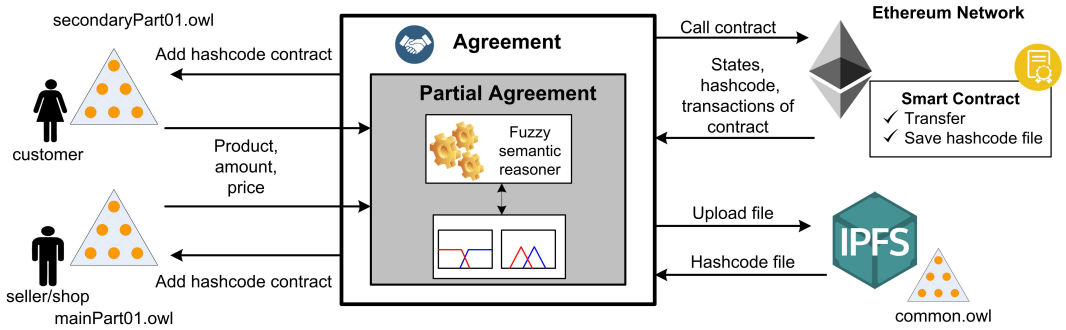


Figure 6.26: Architecture proposed.

1. The involved parts (typically, a main part and a secondary part, but recall that there could be n involved parts) submit their personal fuzzy ontologies, developed in Fuzzy OWL 2, including their definitions for a previously agreed transaction. For example, the desired delivery time or the expected price. Some information regarding the transaction is also needed, e.g., product id or number of units.

2. The system ensures that there are at least two parts and computes a self-contained common fuzzy ontology $\mathcal{O}$. To create local combinations of the restrictions, we use Algorithm 8 (Lines 6–13) extended to n individuals. To manipulate the input

ontologies, we use the OWL API, in Java. The common fuzzy ontology is encoded using fuzzyDL syntax (FDL format).

Example 37. *Figure 6.27 shows a common ontology $\mathcal{O}$ from Example 32. Line* 1 *defines Łukasiewicz fuzzy logic as the default semantics. Lines* 9- 13 *encode the fuzzy datatypes that will be used in the restrictions of each part. The definition of the concepts are in Lines* 15–18. *We used the fuzzyDL reasoner with its Java API [BS16a] to obtain the BSD (Łukasiewicz t-norm). For more details about the fuzzy matchmaking process between the two parts, please review Section 4.4.* □

```
1  (define-fuzzy-logic lukasiewicz )
2  (functional unitPrice )
3  (functional speed )
4  (functional deliveryTime )
5  (range unitPrice *real* 0 200 )
6  (range speed *real* 0 500 )
7  (range deliveryTime *real* 0 30 )
8
9  (define-fuzzy-concept CustomerPrice left-shoulder (0, 200, 165, 185) )
10 (define-fuzzy-concept SellerPrice right-shoulder (0, 200, 165, 169) )
11 (define-fuzzy-concept CustomerSpeed triangular (0, 500, 180, 240, 320) )
12 (define-fuzzy-concept CustomerDeliveryTime left-shoulder(0, 30, 10, 30) )
13 (define-fuzzy-concept SellerDeliveryTime right-shoulder (0, 30, 7, 14) )
14
15 (define-concept Main (and (some unitPrice SellerPrice ) (= speed 250 )
16     (some deliveryTime SellerDeliveryTime ) ) )
17 (define-concept Secondary (and (some unitPrice CustomerPrice ) (some speed CustomerSpeed )
18     (some deliveryTime CustomerDeliveryTime ) ) )
```

Figure 6.27: Example of common fuzzy ontology in fuzzyDL syntax.

Then, we use Algorithm 8 (Lines 15–19) extended to n individuals to find an optimal agreement between all instances of the classes MainPart and SecondaryPart. We use the fuzzyDL reasoner with its Java API [BS16a] to obtain the BSD of the combination using Łukasiewicz t-norm of the restrictions of each part. In particular, in Example 37, we compute $bsd(\mathcal{O}, \mathsf{MainPart} \sqcap_{\mathrm{L}} \mathsf{SecondaryPart})$. The algorithm is extended to retrieve not only the similarity degree but also a model of the fuzzy ontology, including the values of the data properties that lead to the Pareto optimal agreement

3. The system creates a smart contract with the values in the model of the partial agreement. It is encoded in Solidity (version 0.5.12). To create and compile it, we use the development environment Remix IDE[15]. We also use the Web3j[16] library to translate a Solidity binary file (with extension .sol) into Java.

[15]http://remix.ethereum.org
[16]https://docs.web3j.io

4. The system runs the smart contract. To do so, we install an Ethereum node and use the testnet Rinkeby, where we run the contract [Fue19]. The first time we create two accounts (for the two parts) and get some Ethers to simulate the transaction. To manage the transaction of Ethers between the accounts, we use the wallet MetaMask[17]. When the smart contract finishes, it emits an event (Eventheum) to backend services or clients to inform about the status of the execution.

   Rinkeby tesnet was installed in an Ethereum node. Here the contracts are executed. We implemented a service of events (Eventheum) to inform the status of execution to the clients.

5. If the smart contract does not run successfully (e.g., if the secondary does not have enough ether), the process finishes. Otherwise, the common fuzzy ontology is updated with a hashcode of the transaction payment (where ethers are transferred from the secondary to the main part). This could be needed, for example, to return items in the future.

6. The common ontology is uploaded to the IPFS network, which guarantees the security (persistence and immutability) and avoids storing large volumes of data in the blockchain.

7. The personal fuzzy ontologies are updated with the IPFS hash of the common ontology file and with the hash of the contract (notified by the contract using another Eventheum). This way, future access to them is possible.

Finally, the main features of our proposed architecture are the following ones:

- The proposal is independent of the domain. The ontology schema could be updated to the user needs.

- It supports partial agreements using fuzzy logic to compute a global satisfaction degree using fuzzy matchmaking. It offers a efficient and fair consensus.

- It uses blockchain technology avoiding intermediaries or referees. There is no centralization and it offers high security in the transactions and storing.

- It is possible to avoid the ambiguity of natural language because the involved parts use a formal language (Fuzzy OWL 2 ontologies) to represent the knowledge of an application domain, as well as to infer implicit knowledge or check for inconsistencies.

[17] https://metamask.io

Related work

Some authors have studied the use of logic-based languages in smart contracts. For example, Ugarte is one of the first researchers to envision the combination of Semantic Web technologies and blockchain systems [Uga17], using the term "semantic blockchain". He proposed three possible ways to semantify the blockchain: mapping Blockchain data to RDF, sharing RDF data on the Blockchain, and building semantic-ready Blockchains. Our approach combines features of the two first ways. The author also mentioned BLONDiE ontology to describe the blockchain structure, some technologies to link blockchains, and JSON-LD to encode smart contracts. We instead propose use a logic-based language supporting fuzzy ontology reasoning.

Governatori et al. compared the use of imperative and declarative languages, including a retractable logic (deontic defeasible logic) [GIM+18]. An inference engine is also used to check the correctness of a program. They verified if a smart contract is correct in terms of legal validity. However, we consider the family of logic-based languages, based on fuzzy logic and Semantic Web technologies.

Regarding smart contracts, D. McAdams develops a non-OWL ontology to describe smart contracts [McA07] based on states and transitions. Third and Domingue created a Linked Data index to query and retrieve data stored on the blockchain in disparate locations, to link data to other sources of information [TD17], and (with some limitations) to index smart contracts. Kim et al. used an ontology to describe the structure of smart contracts in the government domain [KLN18]. They also encoded some axioms of a non-OWL ontology (TOVE Traceabiliy Ontology) into smart contracts that could enforce traceability constraints [KL18]. Instead, our proposal is more general, supports a fuzzy extension of OWL, independent of the domain, and supports partial agreements. Choudhury et al. proposed a methodology to auto-generate smart contracts from ontologies (defining the domain-specific knowledge) and SWRL rules (defining the constraints) [CRS+18]. Instead, our smart contracts take into account the ontologies at running time, as solving a fuzzy ontology reasoning task is needed to check if there is a partial agreement.

Ruta el al. used Description Logics for the discovery and composition of services and resources in a blockchain based on the semantic distance between terms [RSI+17]. Instead, we propose to use standard fuzzy semantic reasoning services to compute a (possibly partial) agreement among the involved parts.

Fuentemilla developed an application combining ontologies and the Ethereum blockchain [Fue19]. The author used an ontology (only one) for shopping activity on the testnet Ethereum. He detailed the use of Rinkeby network, electronic wallets

and the Java library for programming in this scenario. As in our work, the author proposed to use fuzzyDL reasoner to evaluate a specific agreement but restricted only to two parts. Instead, we have extended this work in several ways. Firstly, our architecture is more general and includes distributed ontology files. Secondly, we propose a novel matchmaking algorithm to generate smart contracts, trigger events, and compute partial agreements. Thirdly, our approach supports more than two parts and a finite number of restrictions. We also study Pareto optimality of the solutions and propose the use of more general fuzzy operators.

6.4 Evaluation of the instance retrieval algorithm

Contributions

In this section we describe the evaluation of our novel reasoning algorithm to solve the instance retrieval problem described in Section 4.1. Our experiment consists in comparing the implementation of the novel algorithm with the previous algorithm implemented in fuzzyDL ontology reasoner. Because fuzzyDL did not previously implement an algorithm for concept realization, the evaluation of that algorithm is left as future work.

The structure of this section is the following. First, Section 6.4.1 describes the datasets and the experimental setup, and then Section 6.4.2 discusses the results.

6.4.1 Experimental setup

Ontology setup. The set of ontologies used to evaluate the instance retrieval algorithm is based on:

- *Absorption* dataset, developed in [BS16b]. It includes 51 ontologies: a fuzzy ontology (*Fuzzy Wine*) developed by humans, and fuzzy extensions of 50 crisp ontologies, randomly generated. For each of the original 50 crisp ontologies, there are several fuzzy versions with different semantics and percentage of fuzzy axioms. In this section, we will consider fuzzy ontologies of the form $l.66$, with a semantics given by Łukasiewicz fuzzy logic and 66% of fuzzy axioms.
- *Fuzzy Beer*, a fuzzy ontology with information about beers described in Section 6.2. Recall that has 15317 beer individuals and 10 fuzzy datatypes (5 of them for the alcohol level ABV).

For the experiments in this section, we firstly translated the ontologies into FDL format, the native syntax supported by fuzzyDL, using an existing parser [BS11]. For

Fuzzy beer, the parser discarded for each instance of the class Country an axiom that fuzzyDL was not able to support (an axiom to deduce the country associated to a beer given the brewery associated to a beer and the country associated to a brewery).

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 to several subsets of the Fuzzy Beer ontology, with different numbers of beers. In the following, we will use $Beer_n$ to denote the subset of Fuzzy Beer with n beer individuals. Note that the total number of individuals is actually higher than n, as there are also breweries and countries.

In Fuzzy Beer (and in its subsets Fuzzy $Beer_i$), $\mathcal{O}_{two\_or\_more}$ constrain 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).

Parameters of the experiments. Firstly, we solved 20 times $Beer_{500}$ 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 solving 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 6.13 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 to decrease the time needed to finish our experiments.

Measure	**Old**	**New**
Average (ms)	80126.0	2047.7
Standard deviation	3438.7	135.3
Coefficient of variation	4.3%	6.6%

Table 6.13: Coefficient of variation for the Fuzzy $Beer_{500}$ ontology.

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 Table 6.14. During our experiments, we set a timeout of 6 hours to solve the instance retrieval problem using the new algorithm (as the old seems to take even more time).

Implementation issues. 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 is shared by the next tests.

Equipment and tools. 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).

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	∃ 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

Table 6.14: List of queries.

6.4.2 Results and discussion

Table 6.15 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 (s)"), the running time in seconds of the novel algorithm (denoted "New(s)"), and some optional observations ("Comments").

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 $\mathcal{O}_{two\_or\_more}$ was empty and it was enough to solve a single MILP problem. There were only two exceptions: FuzzyWine.l.66, and lubm.l.66.

Ontology	#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	20402	20544	Inconsistent ontology
thesaurus.l.66	8	5	2	
Transportation.l.66	181	4	0.3	

Table 6.15: Running time (s) for the Absorption dataset.

Table 6.16 shows some information about the fuzzy ontologies in the Absorption dataset that could not be considered: 29 ontologies do not have any individual and 7 reached a timeout; in 4 cases the timeout is not surprising as there are more than 25000 individuals.

Table 6.17 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 $\mathcal{O}_{two\_or\_more}$ 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 $Beer_i^h$ because it is very similar to the time to solve the easier version $Beer_i^h$. We show the results for a query involving an atomic concept, but we also tried a complex concept (see Table 6.14) obtaining a similar trend.

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 6.28. 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 6.18. We can see that the old algorithm is faster than the new one, as with the Absorption dataset. Furthermore, we can

Ontology	#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	487944	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	25148	Timeout (many individuals)
FMA.l.66	94228	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	277804	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 6.16: Problems found in the Absorption dataset.

	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	20434	158	11184	866 objective dependent variables
5000	36128	263	18329	1093 objective dependent variables

Table 6.17: Running time (s) for subsets of the Fuzzy Beer ontology.

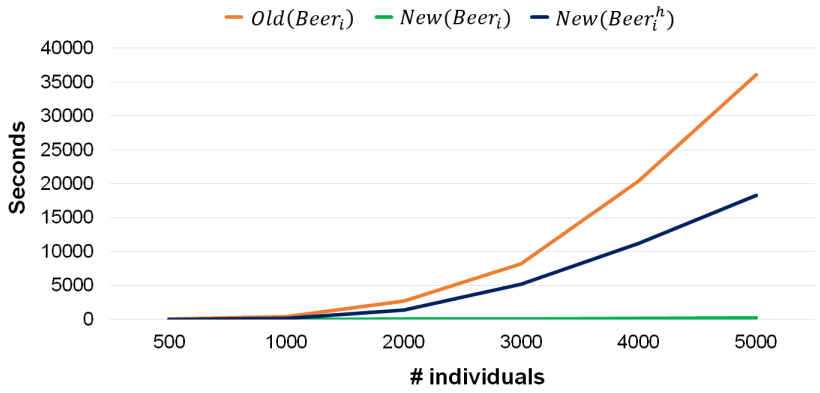


Figure 6.28: Running time for subsets of the Fuzzy Beer ontology.

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 $Beer_i$.

	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

Table 6.18: Running time (s) for subsets of the inconsistent Fuzzy Beer ontology.

6.5 Building Information Modeling

Motivation

Digitalization is a major innovation factor in the construction sector. The incorporation of new information management technologies is transforming how buildings are designed, planned and operated [SRI20]. A key element to achieve this vision is the Building Information Model (BIM), a digital representation of a building for integrated design, modeling, planning and operation during its whole lifecycle [AMCJ18], from inception to decommission. BIMs can help to optimize construction and maintenance costs, improve transparency and collaboration between different stakeholders, manage complex projects, and adapt to changing requirements quickly.

The BIM concept brings together several pieces of interconnected information, including a 3D geometric model of the building elements and a description of the materials used and their properties. To encode these data, the buildingSMART[18] organization proposed the Industry Foundation Classes (IFC), a neutral and open ISO standard for BIM data [CCP18]. The IFC specification defines a conceptual schema for BIM elements, encoded in the data modeling languages EXPRESS (ISO 10303-11) or XSD (XML Schema Definition), and file formats for specific building data, namely IFC-SPF (IFC STEP Physical Format) and ifcXML. Although these formats are light and easy to use, they lack the capabilities for sophisticated knowledge representation and reasoning offered by ontologies. Hence, there are several initiatives to evolve BIMs into semantic BIMs [PZL17], powered by Semantic Web technologies.

It has been shown that fuzzy ontologies can accomplish information retrieval tasks not available in current BIM systems; e.g., cross-domain information integration, flexible querying, and imprecise parametric modeling [GRBR$^+$15]. Unfortunately, semantic BIM tools and fuzzy inference engines suffer some limitations in terms of scalability, efficiency, and ease of use, which make them unsuitable for medium-scale models.

These problems can be addressed by using new fuzzy ontology reasoning algorithms. In particular, the flexible faceted instance retrieval problem is arguably the most common task in BIMs (and in many other domains). Our new Algorithm 5, which imposes some restrictions to ensure efficiency, seems very promising. Our research approach is aligned to recent BIM research initiatives [Eur19], which highlight the need for leveraging BIM data models and validating them on real use cases.

[18] `https://www.buildingsmart.org`

Contributions

In this section, we describe the implementation of the novel algorithm to solve the flexible faceted instance retrieval problem in a software prototype. We also evaluate the performance to answer some fuzzy queries over a real-world BIM, proving that the new algorithm can be useful to reason efficiently with real-world data.

This section is structured as follows. Firstly, we describe the implementation of the tool in Section 6.5.1. Next, we describe the dataset, taken from a real use case, in Section 6.5.2, and the results of an empirical evaluation in Section 6.5.3.

6.5.1 Implementation

The reasoning engine is the first point to be taken into consideration in the implementation of Algorithm 5. Line 8 requires solving the instance retrieval concept, Line 9 requires solving the classification problem, and Line 24 requires retrieving the values of data property, possibly not explicitly stored in the ontology. While the two former tasks are relatively well supported by a number of reasoners, this is not the case of last one. HermiT reasoner is one of the few exceptions, as it has indeed a method `getDataPropertyValues` to solve Line 24. TrOWL is a reasoner for the OWL 2 EL profile and by means of the OWL API it is possible to access to the values of the data properties. Pellet reasoner also needs a supports for OWL 2 DL and OWL 2 EL profile, but it needs a programmatic way to retrieve the values of the data properties.

There is another way to get the real values: using a SPARQL query. For example, Figure 6.29 illustrates the results of a simple query to obtain the overallHeight and overallWidth values of the instances of IfcWindow class from an RDF file obtained using IFC-to-RDF converter [THP19]. It is clear that if we need to infer knowledge or to classify the ontology, SPARQL is not appropriate.

We developed a prototype tool which is available online[19]. It implements Algorithm 5 and a graphical interface to submit queries. It is a Java (1.8) implementation using the OWL API to manage OWL 2 ontologies represented in Fuzzy OWL 2 language. The classical semantic reasoner used is TrOWL 3.4. To reduce the time to access the ontology, we stored the fuzzy concept assertions using a hash table and a NoSQL database (MongoDB 4.0.10). As a baseline, we also considered direct calls to the OWL API. A graphical user interface (for desktop computers) makes it possible to submit queries about building elements.

The general functionality of this software is shown next. The tool contains three tabs:

[19] `http://webdiis.unizar.es/~ihvdis/fuzzyBIMgui.html`

Figure 6.29: SPARQL query over an RDF file.

– The first one (see Figure 6.30) specifies the path of the ontology and the base URI (it corresponds to the IFC2X3 schema). By default, the converter software uses the base URI `http://linkedbuildingdata.net/schema/IFC2X3#`. Sometimes that URI could change, as it depends on the converter or the version schema. The fuzzy ontology file can have .owl or .ttl extensions. The user also needs to select the IFC element (a class) from the schema, such as IfcWindow.

  In this tab the user also needs to select the operator to combine the values and a fuzzy modifier. Possible operators include minimum (T-norm Min), maximum (T-conorm Max), weighted mean (WMEAN), and OWA. Figure 6.34 shows an example of OWA operator built using quantifier-guided aggregation. Possible modifiers are none, very, few, linear, and triangular. Figure 6.30 shows as an example the definition of very.

– The second tab shows all the data properties in the ontology and the user has the possibility to select some of them. Figure 6.31 shows an example where overallHeight and overallWidth properties are checked.

– The third tab allows to select or create the fuzzy datatypes for the chosen data properties (see Figure 6.32). One way is to select fuzzy datatypes already defined in the ontology file. It is also possible to create a new fuzzy datatype, using labels like VeryLow, Low, Neutral, High, and VeryHigh, and membership functions such as left-shoulder, triangular, trapezoidal, and right-shoulder.

Initially, the Run button is disabled until all necessary parameters are specified. When the "Run" button is clicked, a process is executed to solve the query. Eventually,

a dialog with a sorted list of instances is displayed, as shown in Figure 6.33.

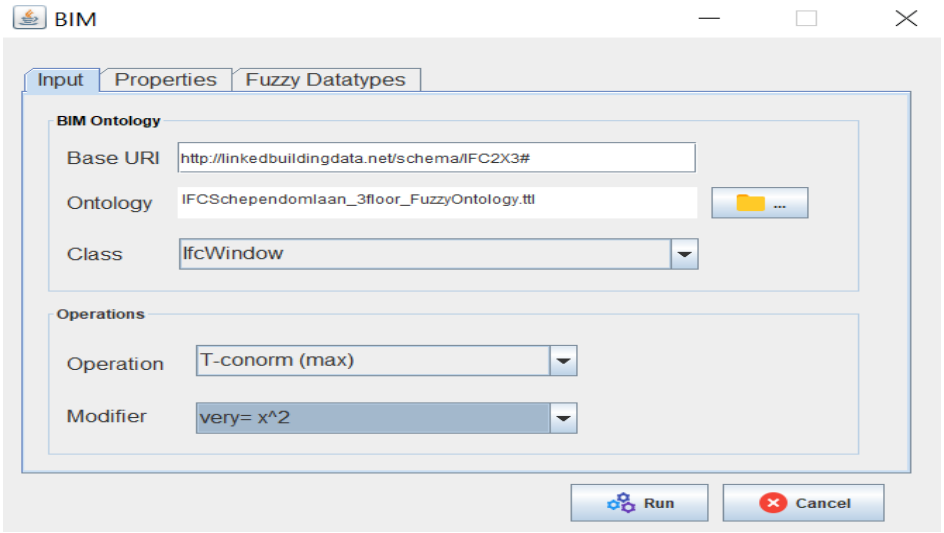


Figure 6.30: User interface: loading fuzzy BIM ontology, selection of a class, and fuzzy operators.

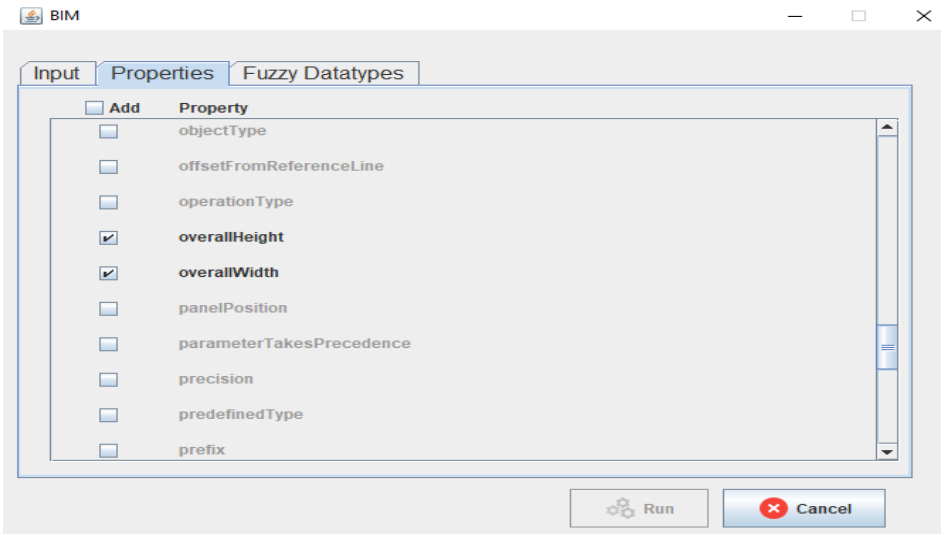


Figure 6.31: User interface: selection of data properties.

Example 38. *Assume we need to retrieve a set of windows with high width and very high height from the fuzzy ontology. So, we use the desktop tool and ask for a IFC building element called* IfcWindow*. We consider two data properties of a window, namely* overallWidth *and* overallHeight*. We define two fuzzy datatypes* HighOverallWidth*, using a triangular fuzzy function* $\mathbf{triangular}(900, 1200, 2000)$*, and* VeryHighOverallHeight*, using a right-shoulder fuzzy function* $\mathbf{right}(1700, 2500)$*. We choose the maximum t-conorm operator (auxDegree) and the fuzzy modifier* very*.*

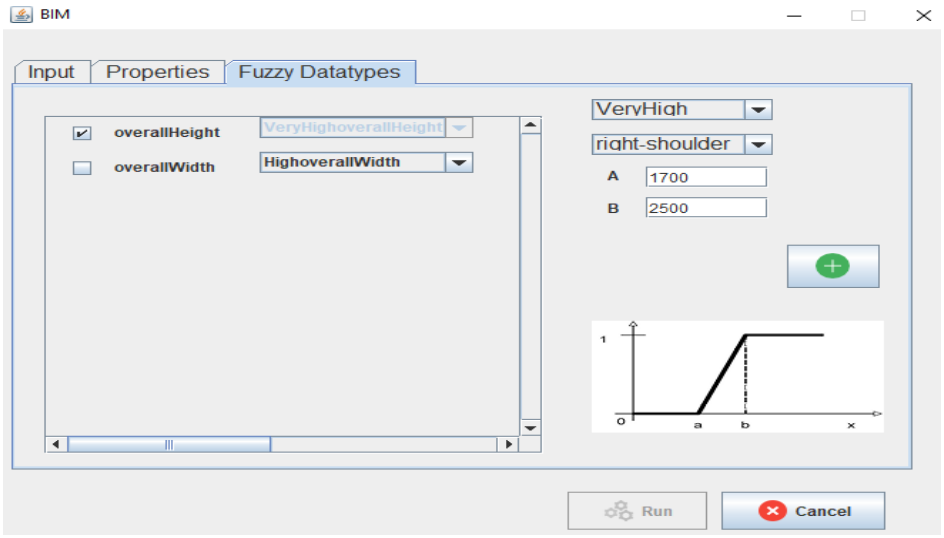


Figure 6.32: User interface: selection or creation of fuzzy datatypes.

Figure 6.33: User interface: final result.

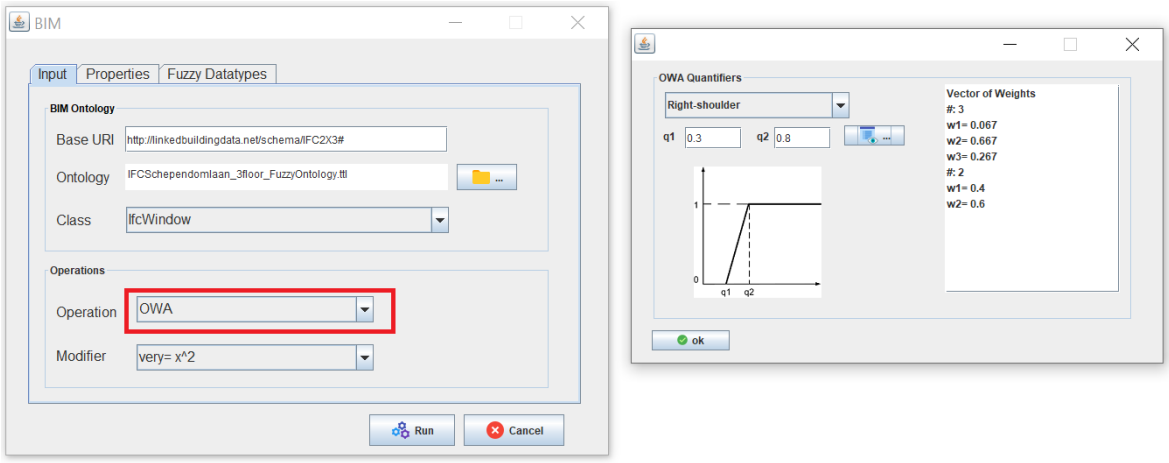


Figure 6.34: User interface: use of a quantifier to get the parameters of the OWA aggregation operator.

Table 6.19 shows the evaluation of 12 window instances their names are shortened for space limitations). β_i denotes the degree used in the fuzzy concept assertion, and was added randomly to each window in the fuzzy ontology. Figure 6.33 shows the result: a sorted list of windows (colored using the satisfaction degree of the query α_i). □

Window	overallWidth	overallHeight	tri	right	β_i	auxDegree	α_i
GUID_kMl	1430	2512	1	0.71	0.20	1	1
GUID_O8D	940	1760	0.13	0.75	0.70	0.70	0.48
GUID_7nV	940	1760	0.13	0.07	0.10	0.13	0.01
GUID_S27	940	1760	0.13	0.07	0.60	0.60	0.36
GUID_eYJ	1430	2512	0.71	1	1	1	1
GUID_Ryl	916	1760	0.05	0.07	0.50	0.50	0.25
GUID_wCu	940	1760	0.13	0.75	0.40	0.40	0.16
GUID_jtL	1430	2512	0.71	1	0.10	1	1
GUID_hhq	1430	2512	0.71	1	0.90	1	1
GUID_pct	916	1760	0.05	0.07	0.80	0.80	0.64
GUID_41F	940	1760	0.13	0.07	0.30	0.30	0.09
GUID_lLC	916	1760	0.05	0.07	0.20	0.20	0.04

Table 6.19: Set of individuals from IFCWindow.

6.5.2 Dataset

We evaluated our proposal using the Schependomlaan public BIM dataset[20]. This project was developed and built by Hendriks Bouw en Ontwikkeling[21] and comprises 10 apartments located in Nijmengen, Netherlands. The dataset contains a design

[20] https://github.com/openBIMstandards/DataSetSchependomlaan
[21] https://www.hendriksbouwenontwikkeling.nl/en

model in IFC, extract, suppliers, point clouds, schedules, and construction log files. Figure 6.35 shows the 3D model visualized on the academic version of Archicad 22[22].

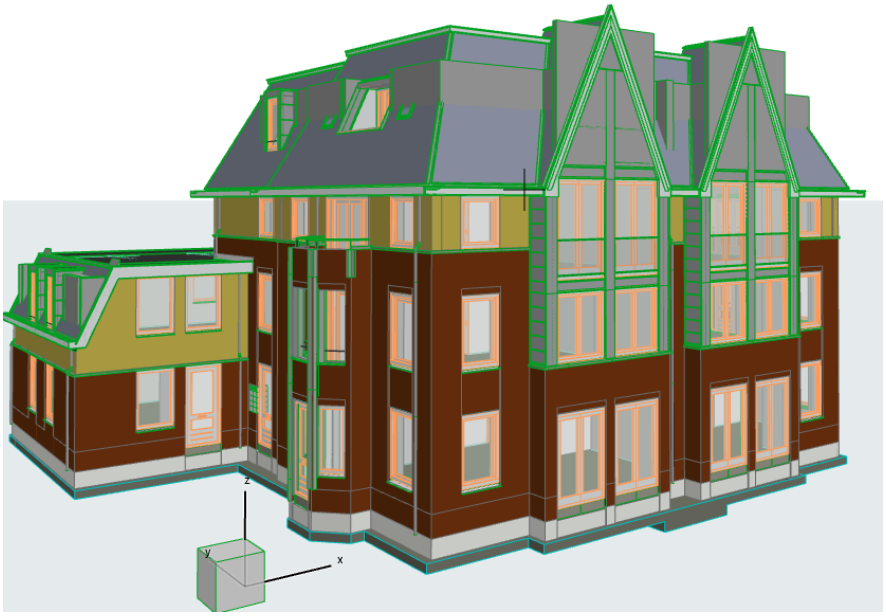


Figure 6.35: Use case: 3D representation.

For our purpose, we need to obtain a representation of our BIM model using an OWL 2 ontology, obtained from from the IFC model of the use case. The ontology that we need should consider classes, individuals, and relationships (data and object properties). For example, the class IfcDoor has the instance IfcDoor_01 with a data property overallHeight equals to 2282 mm and an object property representation linking it to the b179 instance.

Firstly, we need a conversion from IFC to RDF. After testing four IFC converters (IFC-to-RDF Version 1.0[23] [THP19], IFC2LD[24] [HT15], IFCtoRDF[25], and IFCtoLBD[26] [BOP+18]) we selected IFC-to-RDF. However using more sophisticated parsers could be possible.

Secondly, it involves using an OWL schema to categorize BIM elements. In this work, we used the ifcOWL ontology. Another option is the Building Typology Ontology (BOT)[27], or the BIM schemas used by other conversion tools.

In order to reduce both the file size and the reasoning time, we divided the use case in six submodules (the six stories of the original IFC building model) that

[22] https://www.graphisoft.es/archicad

[23] Not available online anymore. Latest version (1.5) is called Ifc2Rdf and is available at https://github.com/Web-of-Building-Data/Ifc2Rdf/tree/master/software

[24] https://github.com/Web-of-Building-Data/ifc2ld.git

[25] https://github.com/pipauwel/IFCtoRDF

[26] http://github.com/jyrkioraskari/IFCtoLBD

[27] http://www.student.dtu.dk/~mhoras/bot/index-en.html

correspond to foundation, ground floor, first floor, second floor, third floor, and roof. The fragmentation task was manually done with the help of the graphical environment Archicad. Next, we exported it to IFC format and then used the converter to obtain the ontology.

We focused on the third floor, because it has the smaller .ifc and .ttl files, and that makes reasoning more feasible without loss of generality. Table 6.20 shows some statistical data about the ontology representing the third floor.

In general, when dealing with real data, one needs to split the ontology into smaller subontologies. In this work, we did it manually. It would be possible to study methods to compute a split automatically given some restrictions. In particular, one could consider using a method to reduce the geometrical data (e.g., position and orientation of the building elements) which are not necessary unless one wants to reason with spatial semantics [DB14]. This makes it possible to reduce the size of the ontology while having a more efficient representation for some queries, e.g., those involving intersections of building elements.

Furthermore, we defined a *modified* version by making the following changes:

1. We removed the graphic elements that do not have property values that are needed for our queries. For example, walls or columns that do not have a height and a width. The priority was to have a high number of windows. The initial ontology has 12 windows, and 8 of them have values. The modified ontology was updated to have the 12 windows (we used the tool Measure of Archicad to obtain the missing sizes).

2. We modified in the schema file the range of the data properties overallHeight and overallWidth, to make it xsd:double.

3. We added some new classes representing specific styles defined in Archicad, and created some new instances of them (via concept assertions).

   – For the IfcWindow class we added 9 subclasses, namely BasicWindow (with 8 instances), DormersAndSkylights, EmptyWindowsOpenings, HistoricWindow, SingleDoubleHungWindow, SindingWindow, SpecialWindow (with 4 instances), StoreFronts, and TerraceDoors.

Tool	Classes	Data Properties	Object Properties	Individuals
IFC-to-RDF	1085	929	1502	10127

Table 6.20: Statistics of the conversion of the third floor.

– For IfcDoor class we added 8 subclasses, namely Bed, EmptyDoorOpenings, GarageDoor, HingedDoor (with 2 instances), RotatingDoor, SidingDoor, SidingFoldingDoor, and Table.

– For IfcWall class we added 5 subclasses: GenericWall, ExteriorWall, InteriorWall, PartitionalWall, and StructuralWall.

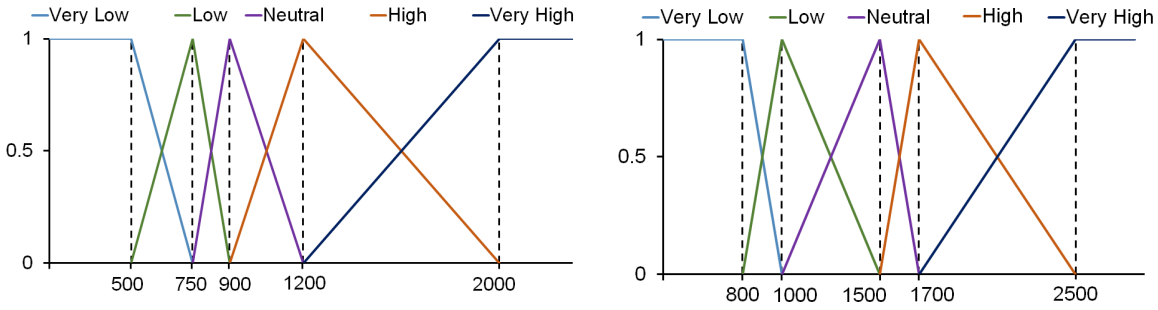
Finally, we fuzzified the ontology representing the third floor for testing our novel algorithm. We firstly defined a fuzzy ontology (called *Fuzzy1*) using the plugin Fuzzy OWL 2 for Protégé 4.3. In particular:

– We added 12 fuzzy concepts assertion, adding a degree of truth to some axioms at BasicWindow and SpecialWindow classes. The degree values in (0,1) were chosen in a random way.

– 10 fuzzy datatypes were created based on our experience about size windows [BS11]. The definition of the window labels is shown in Figure 6.36.

A common problem in fuzzy ontology development is how to obtain the linguistic labels, i.e., the concrete definitions of the fuzzy datatypes. We did it manually in this use case but it would be recommendable to use supporting tools, such as Datil (see Section 3.2) or Fudge (see Section 3.3).

We also created another version (*Fuzzy2*) by adding more individuals to the fuzzy ontology (in particular, 6498 individuals, with 1400 windows and 100 doors).



(a) (b)

Figure 6.36: Linguistic labels for a) overallWidth and b) overallHeight.

The fuzzy ontology and schema were saved using OWL/XML syntax. The ontology lost some valuable data (such as graphic placement and anonymous nodes) but this does not affect the result of our queries.

6.5.3 Results and discussion

Firstly, we evaluated the initialization time of our tool, which includes loading the ontology, computing the classification, and the initialization of a data structure with the degrees of truth. Secondly, we evaluated the proper query time, as well as the time to retrieve the values of the data properties. The evaluation was performed on a Intel Core i7-8550U 1.8 GHz, 16 GB RAM (7 GB were allocated for the JVM) laptop running Windows 7 64-bits.

Initialization time. Before describing the evaluation of the initialization time, it is worth to recall that it must be computed just once. Firstly, we tested three classical reasoners (Hermit 1.3.8, TrOWL 3.4, and Pellet 2.3.3) to measure the load and classification times for the original, modified, and fuzzy ontologies of the third floor. Table 6.21 shows the ontology, file size, reasoner used, time and number of named individuals. Time includes the time to load the ontology, to classify it by precomputing the class hierarchy and the class assertions, and to perform a consistency test. Note that HermiT run out of memory in all cases, after approximately 20 minutes. TrOWL also run out of memory for the original ontology, but the modified versions could be successfully processed. Pellet detected an inconsistency in all the ontologies (because of the datatypes).

We also evaluated the use of the auxiliary data structures to reduce the answering time (Lines 3–6). The results are shown in Table 6.22. For ontology Fuzzy1, OWL API method was slightly faster than the hash table, so it seems to be the best option to avoid the cost of maintaining the data structure. In particular, for such ontologies with a small number of fuzzy concept assertions, the database performs worse than the OWL API. For the ontology *Fuzzy2*, hash table is clearly faster than the other two methods.

Query time. Next, we evaluated the time to obtain the values from the data properties (height and width) of each individual (Lines 23–28). We used TrOWL

Ontology	File size (MB)	Reasoner	Time (s)	Individuals
Original	27.8	HermiT	OutOfMemoryError	10127
Modified	15.1	HermiT	OutOfMemoryError	5498
Fuzzy1	56.4	HermiT	OutOfMemoryError	5498
Original	27.8	TrOWL	OutOfMemoryError	10127
Modified	15.1	TrOWL	80.38	5498
Fuzzy1	56.4	TrOWL	83.28	5498

Table 6.21: Time (s) to load and classify the ontology.

Ontology	Hash table	Database	OWL API
Fuzzy1	0.07	1.09	0.01
Fuzzy2	0.32	2.04	6.52

Table 6.22: Time (s) to create the data structures.

reasoner and SPARQL for the modified and *Fuzzy1* versions. For the SPARQL queries we used the server Apache Jena Fuseki 3.14 and Jena Java API. It is worth to note, however, that a SPARQL query cannot be used in general to solve any query to the ontology. Table 6.23 shows the results (the average of 5 executions) for the fuzzy ontology. For the first query, the SPARQL query is solved faster than using the reasoner because it only needs to load the ontology, but the reasoner performs a more complex preprocessing including classification. However, for the next queries the reasoner is faster.

Then, we evaluated the full query time (Lines 8–34). Starting from the ontology *Fuzzy1* (with 5498 individuals where 12 are windows and 2 doors), we created a set of 6 queries, 5 of them about IfcWindow class and 1 about IfcDoor class. Queries were solved 5 times and we computed the average values. Table 6.24 summarizes the queries and the results. The first columns include the query ID and the parameters of the query: the class, the data property, the label (fuzzy datatype), the aggregation operator, and the modifier. The final columns include the query time (in seconds) when using a hash table, a NoSQL Database Mongo DB, or only calls to OWL API methods. As already discussed, hash table is slightly preferable.

We also repeated the same queries for the ontology *Fuzzy2*. Figure 6.37 shows the result of the query times. We can see that using the best data structure, query time is very fast (less than 0.62 s), making our algorithm acceptable for such models.

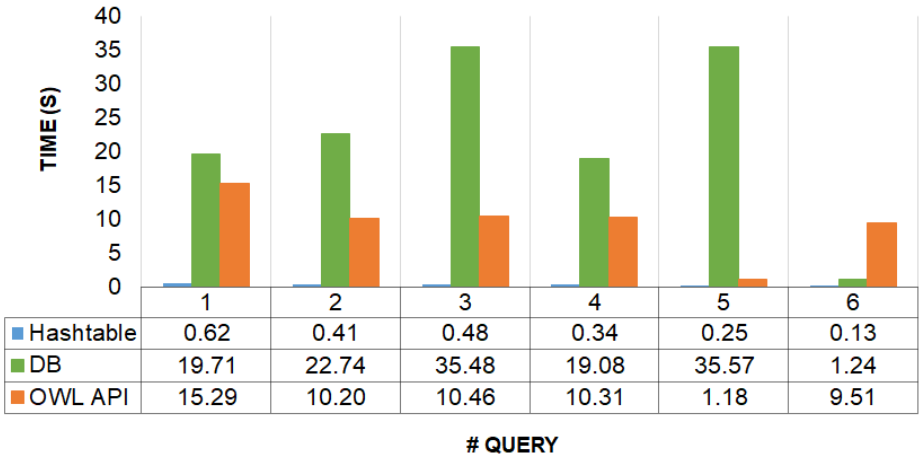
The previous query times assume that the system has already been initialized. Table 6.25 shows the total time for the first query. Likewise for the query time, OWL API performs similarly to the hash table version for *Fuzzy1*, but hash table performs clearly better for *Fuzzy2*.

Ontology	Reasoner	Loading + classification time (s)	Query time (s)
Modified	Jena	4.40	1.160
Modified	TrOWL	80.38	0.007
Fuzzy1	TrOWL	83.28	0.007

Table 6.23: Time (s) to get the data properties values in the IfcWindow class.

ID	Class	Property	Label	Operator	Modifier	Time (s)		
						Hash table	Database	OWL API
1	IfcWindow	overallWidth overallHeight	High VeryHigh	Maximum	very, x^2	0.14	0.18	0.18
2	IfcWindow	overallWidth overallHeight	Neutral High	Minimum	few, $\sqrt{x}$	0.11	0.12	0.11
3	IfcWindow	overallWidth overallHeight	Low Neutral	OWA	**lin**$(0.3)(x)$	0.06	0.10	0.06
4	IfcWindow	overallWidth overalHeight	Low Neutral	WMEAN	**tri**$(1000, 1500, 2000)(x)$	0.05	0.11	0.05
5	IfcWindow	overallWidth overallHeight	Neutral Low	Minimum	None	0.05	0.09	0.03
6	IfcDoor	overallWidth overallHeight	High High	Maximum	few, $\sqrt{x}$	0.10	0.09	0.08

Table 6.24: Queries and query time (s) for ontology *Fuzzy1*.



	1	2	3	4	5	6
Hashtable	0.62	0.41	0.48	0.34	0.25	0.13
DB	19.71	22.74	35.48	19.08	35.57	1.24
OWL API	15.29	10.20	10.46	10.31	1.18	9.51

Figure 6.37: Query time (s) for ontology *Fuzzy2*.

Ontology	Task	Time (s)		
		Hash table	DB	OWL API
Fuzzy1	Loading + Classification	83.28	83.28	83.28
	Data structure	0.07	1.09	0.01
	Query	0.08	0.11	0.1
	Total	83.43	84.48	83.39
Fuzzy2	Loading + Classification	112.63	112.63	112.63
	Data structure	0.32	2.04	6.52
	Query	0.37	22.30	9.49
	Total	113.32	136.97	128.64

Table 6.25: Total time (s) for the first query.

Verification. Our reasoning algorithm is correct, i.e., all retrieved instances satisfy the query. However, the solution is only complete if the fuzzy ontologies satisfies some restrictions.

Regarding the quality of the solutions, they depend on the quality of the linguistic labels. In this regard, it is worth mentioning that Datil's algorithm to learn fuzzy datatypes has been evaluated in the field of beer recommendation, showing that it provides similar results to a human expert (see Section 6.2.2).

Related work

In this section, we describe some previous approaches about querying and reasoning for semantic BIM.

The use of ontologies in the domains of architecture, engineering and construction (AEC) has notably increased over the last years, giving raise to the so-called semantic BIMs. Pauwels, Zhang and Lee stated that there are several motivations behind this interest [PZL17]: (1) facilitating interoperability and information exchange between heterogeneous tools, (2) linking cross-domain information to exploit synergies of related domains, and (3) equipping AEC data models with logic-based representation capacities. These authors concluded that are still many research gaps than remain unexplored, such as the combination of declarative and procedural techniques, and the automation of the data integration and retrieval procedures.

Recently, Mendes de Farias et al. explored the capabilities of rule-based reasoning in semantic BIMs [MdFRN18]. They proposed the concept of *view* to represent a minimal usable sub-graph of elements extracted from an IFC file modeling a whole facility. The view is materialized as a knowledge graph based on the ifcOWL ontology [PT16], created by applying logical rules in SWRL (the Semantic Web Rule Language), and queried in the same language. The Stardog[28] triplestore is used to solve SPARQL [HS13] queries on RDF data and SWRL inferences. We perform a similar process to translate the heavyweight IFC files into a simpler OWL model, but we rely instead on the creation of modules based on the physical features of the building via a graphical user interface. We also leverage this interface to facilitate the creation of fuzzy queries over the IFC entities, instead of directly using SPARQL—which can be difficult for non-expert users. Our algorithm also reduces the time required to solve the queries, which may take hours in their case.

Werbrouck et al. analyzed the limitations of IFC regarding modularity of BIM models and their support for query-solving [WSB+19]. Focusing on data represented

[28] `https://www.stardog.com/`

as RDF triples, they presented a comparative of the usability and the performance of SPARQL against GraphQL-LD [TVV18] and HyperGraphQL [Sem20], two query languages based on the REST API language GraphQL. The transformation between IFC and RDF was done with IFCtoLBD, which we also use in this work. The authors showed that BIM models can exploit standard Linked Data languages for pattern-based query and data federation, but their expressivity is low: mostly simple RDF property-value and type-of queries on BIM elements are addressed. Our proposal supports instead a richer fuzzy extension of OWL 2, and at the same time, allows using existing reasoning engines.

Another proposal is that of Fahad et al., who focused on formal verification of IFC models by means of a Linked Data consistency checker—namely, the Semantic BIM Reasoner (SBIM-Reasoner) [FBF18]. This issue was indeed mentioned in [WSB+19] (and also in [SMSW20], where the Shapes Constraint Language (SHACL) is suggested to address this issue). To that end, Fahad et al. developed a processing pipeline to extract geometry data from an IFC file, filter relevant information to reduce the model size, and create a resulting RDF graph. This model was managed with the Stardog triplestore via SPARQL queries and SWRL rules, in a similar way as in [MdFRN18]. In contrast, our proposal explores how fuzzy ontologies can be applied to define imprecise restrictions on data with the purpose of flexible querying. Fuzzy constraint satisfaction still remains as a future work.

To the best of our knowledge, the first approach to augment semantic BIMs with capabilities to manage imprecision and vagueness is proposed by Gómez-Romero et al. in [GRBR+15]. The authors used fuzzy ontologies and the fuzzy ontology reasoner *DeLorean* [BDGRS12] to propose solutions to several AEC tasks: cross-domain knowledge linking (e.g. with partial concept inclusions and graded relationships), imprecise BIM queries (e.g. by using linguistic labels and imprecise topological relations) and fuzzy parametric modeling (e.g. by means of fuzzy axioms and maximization of their degree of fulfillment). In this thesis, we further develop these ideas and focus on one unsolved issue: the efficiency and the scalability of the reasoning algorithms. To that aim, we present a new algorithm for instance retrieval in large BIM models, which is evaluated on a real-world BIM.

Abualdenien and Borrmann highlighted that vague, imprecise, and incomplete information is frequent in the AEC industry, and acknowledged that it should be somehow incorporated into the BIM methodology [AB20]. These authors focused on the visualization of uncertain aspects of the building design, and particularly, vagueness of geometrical properties. In contrast to our work, they did not use a formal framework for the representation of uncertainty and imprecision. Our approach, based

on Description Logics, allows us to guarantee the computational properties of the inference process and to use existing fuzzy and crisp reasoning engines.

6.6 Evaluation of Datil

Our main aim in this section is to perform an evaluation of fuzzy datatypes learning from numerical data properties using Datil software (described in Section 3.2). Our first experiment focuses on the learning time, and then we describe two applications to real use cases. The structure of this section is the following. First, Section 6.6.1 describes the evaluation of Datil for desktop and mobile devices. Next, Section 6.6.2 explains the integration of Datil with another systems for categorizing human life style. Finally, Section 6.6.3 describes the use of Datil to create linguistic summaries using gait data. An evaluation of the quality of the fuzzy datatypes can be found later (in Section 6.2.2), as part of the evaluation of GimmeHop system.

6.6.1 Running time on mobile devices

Contributions

In this part, we perform an empirical evaluation of Datil for desktop and mobile devices. Both Datil versions were described in Section 3.2 and Section 5.4.1 respectively. Our experiment consists of computing the learning time of the fuzzy datatypes for the Fuzzy beer ontology (described in Section 6.2.1). In particular, we generated 6 ontologies with a number of individuals ranging from 500 to 3000, and we learned fuzzy datatypes for two numerical data properties, namely the alcohol level (ABV) and bitterness (IBU).

The setup was defined using a configuration file. For each Datil version, we considered all the implemented clustering algorithms, namely k-means, fuzzy c-means and mean-shift. The number of clusters for k-means and fuzzy c-means was $k = 5$ for both data properties. The selected output was Fuzzy OWL 2 format and we used the ontology reasoner HermiT (in the mobile version, using a ported version of HermiT to Android). Experiments were performed on a laptop (Intel Core i7-8550U 1.8 GHz, 16 GB RAM, running Windows 7 64-bits), denoted PC, and a smartphone (ZTE Blade A610 running Android 6.0, Quad-Core 1.3 GHz ARM Cortex A-53, 2 GB RAM, released in 2016). Experiments were repeated 3 times for each ontology and device, and the shown learning time is the average of the 3 executions.

Figure 6.38 shows (in logarithmic scale) the time to learn the fuzzy datatypes on both devices. Although time is higher on the smartphone, it could be acceptable since learning can be done just once for each fuzzy ontology (i.e., less than 8 minutes for

1500 individuals). Clustering algorithm has an impact, and mean-shift can be 20% slower than k-means on the smartphone.

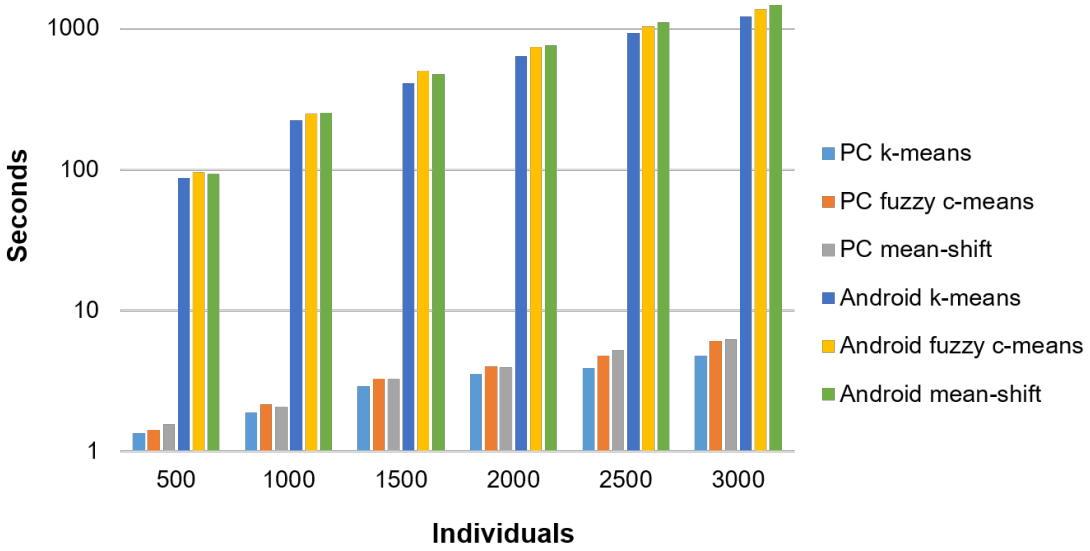


Figure 6.38: Running time on PC and smartphone.

Related work

A complete evaluation of the performance of DL semantic reasoners for Android operative system is performed in [BYBM15]. The authors started by describing the experience to port DL reasoners to Android devices (the platform supports OWL API) and some challenges. Then, they made an empirical evaluation of 9 ported reasoners (e.g., HermiT 1.3.8) and more than 300 non-fuzzy ontologies (from the ORE 2013 ontology set) on a smartphone and a tablet. This approach opened the door to generate more intelligent apps using semantic technologies and inference engines.

To the best of our knowledge, there is not any previous research that implements and evaluates techniques to learn fuzzy ontologies on mobile devices.

6.6.2 Lifestyle profile

Contributions

The aim in this section is to integrate two fuzzy ontology learning techniques and to validate them using real data. The first technique is proposed in Section 3.2 and implemented in Datil tool. It uses the values of data properties to build fuzzy membership functions (fuzzy datatypes). Then, we can use the output of Datil as an input for the second technique, which learns fuzzy General Concept Inclusions GCI using datatypes. Starting from the (possibly partial) membership of

individuals to classes, it is possible to automatically compute some partial inclusions between (possibly fuzzy) concepts. This learning strategy is implemented in the Fuzzy DL-Learner software. Interestingly, Fuzzy DL-Learner can use those fuzzy datatypes learned by Datil to learn fuzzy GCIs.

Lifestyle can be defined as a collection of routines and behaviors shaped by the social, economic, and environmental structure around a person. Healthy living is a hot topic in our current lives. The daily activity monitoring of our lifestyle could be used to improve it [Org99]. Currently, the daily activity monitoring can be done unobtrusively via sensors such as wearables, telecare technology, etc. Those sensors generate a large amount of heterogeneous data to be processed. A very important problem is how to categorize the lifestyle of humans in relation to the activities over time and space, for examples the sport time in the morning, habits, frequency of visited spaces, etc. Given a set of digital traces such as sleep and activity sensors, computational systems can serve to provide intuitive lifestyle categorizations. A model of lifestyle can be based on the matching of a predefined semantic template to the data.

In this section, we will describe a methodology to categorize the lifestyle of a group of people: starting from data obtained from sensors, we want to classify them in different profiles (such as MediterraneanWorker) using learning of fuzzy datatypes and fuzzy ontology axioms.

Firstly, we mention the tools used in our approach:

- Datil software, to learn fuzzy datatypes (see Section 3.2).
- Fuzzy DL-Learner[29] software, to learn fuzzy subclass axioms defining a rule for each profile.
- *fuzzyDL* reasoner, to solve the fuzzy instance retrieval task.
- *Lifestyles-KB*, a crisp ontology for wearable sensors[30]. Note that in this thesis we have renamed some data properties to provide more readable names.

To compute the categorization of a person into some lifestyle pattern, we propose to follow the following steps:

1. Build a crisp ontology $\mathcal{O}$ with the features of interest, using domain experts, e.g. data scientists, specialists in diet, specialists in monitoring cardiac disease patients, etc. At this point, experts should identify lifestyle patterns like MediterraneanWorker but without providing their definition. In our case, we use Lifestyles-KB.

[29] `http://www.umbertostraccia.it/cs/software/FuzzyDL-Learner`
[30] `http://github.com/NataliaDiaz/Ontologies`

Example 39. *Figure 6.39 shows a screenshot of the ontology Lifestyles-KB. Yellow circles represent classes (e.g.,* User*) and green circles represent data properties (e.g.,* endTime*). The ontology also includes axioms, e.g., stating that the range of the data property* endTime *is* xsd:decimal *and that* endTime *is functional.* □

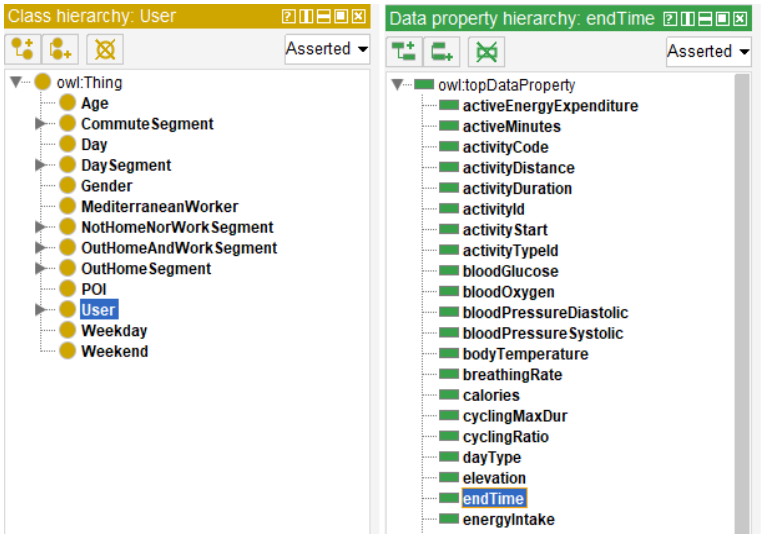


Figure 6.39: Fragment of Lifestyles-KB.

2. Create an ontology $\mathcal{O}_D$ by populating $\mathcal{O}$ with data property assertions obtained from sensors.

   Example 40. *Assume that the end time (corresponding to the data property* endTime*) of* user1 *is* 1089 *minutes (i.e.,* 18 *hours and* 15 *minutes), obtained at segment* `atWork`[31]*. This is represented using the OWL 2 axiom:*

   ```
   DataPropertyAssertion( endTimeAtWork user1 "1089"^^xsd:decimal)
   ```

   where the data property endTimeAtWork *is the result of restricting the data property* endTime *to the segment* `atWork`*.* □

3. Use Datil to learn some fuzzy datatypes from the data property assertions in $\mathcal{O}_D$, and add such fuzzy datatypes to both $\mathcal{O}$ and $\mathcal{O}_D$.

   Example 41. *A vector of centroids is computed for the property* endTimeAtWork *using the fuzzy k-means method. The fuzzy datatype* HighEndTimeAtWork *is learned as a triangular function* **tri**$(1039.74, 1100.97, 1246.59)$*. In fuzzyDL syntax, the definition is encoded as:*

[31] As we will discuss later, days will be divided into segments.

```
(define-fuzzy-concept HighEndTimeAtWork
  triangular(-10000, 10000, 1039.74, 1100.97, 1246.59) )
```

assuming a range $[-10000, 10000]$ *for the data property.* □

4. Ensure that $\mathcal{O}_D$ satisfies the functionality restrictions. That is, if there is an individual such that there is more than one data property assertion involving the same data property (or, if segments are used, the same data property and segment), all but one of them must be removed, or all of them can be replaced by an average of the values. This is necessary because fuzzy data properties are required to be functional. Note that this step was not mentioned in [HSDRB18].

5. Add to $\mathcal{O}_D$ the definition of some preliminary rules (concept equivalence axioms) with the help of an expert. Some of the concepts will have complex definitions, being defined in terms of the learned datatypes. For example, one can define the concept of MediterraneanWorker from the (late) starting and end times.

   Example 42. *Given* HighStartTimeAtWork *and* HighEndTimeAtWork *fuzzy datatypes (encoding the working hours for Mediterranean people using trapezoidal functions), the concept* MediterraneanWorker *can be defined from the (late) starting and ending times as:*

```
(define-concept MediterraneanWorker (g-and
  (some startTimeAtWork HighStartTimeAtWork )
  (some endTimeAtWork HighEndTimeAtWork)
) )
```

   □

6. Ask a fuzzy semantic reasoner (fuzzyDL) to retrieve all the instances of each of the defined concepts in $\mathcal{O}_D$ together with the degrees of membership.

   Example 43. *The fuzzyDL instance retrieval query*

```
(all-instances? MediterraneanWorker)
```

   returns person001 *with a membership degree* 0.84. □

7. Represent it as fuzzy concept assertions and add them to the fuzzy ontology $\mathcal{O}_D$.

   Example 44. *We would add to fuzzyDL the following axiom:*

```
(instance person001 MediterraneanWorker 0.84)
```

   □

8. Run a learning algorithm (Fuzzy DL-Learner) computing the final subclass axioms from the memberships to fuzzy classes in $\mathcal{O}_D$ and add them to $\mathcal{O}$. The learned axioms are complex definitions of lifestyle pattern concepts, similar to the preliminary rules but automatically derived from the real data.

Example 45. *A possible output of Fuzzy DL-Learner is:*

```
(define-primitive-concept MediterraneanWorker (g-and
  (some startTimeAtWork HighStartTimeAtWork )
  (some endTimeAtWork HighEndTimeAtWork)
  (some activityDurationAtWork HighActivityDurationAtWork)
) )
```

□

Validation. In order to populate the ontology, the only information that we have used are real data obtained from digital traces such as sleep and activity sensors and other wearable devices. In particular, we used 40 records of volunteers of middle age living in the Eindhoven area (The Netherlands). These data were provided by a private company (Philips Research) and are confidential (little details are thus given in this dissertation for privacy reasons). However, we would like to point out that this scenario is a typical case where we do not have data about the membership to classes but we do know the values of several data properties.

Each record contains data corresponding to one individual, obtained over different days. For each day, there are 14 day segments, such as *atWork* or *toHome*. We considered 68 data properties with numerical range (e.g., heartRate, calories, etc.). In the step 3 of our approach, we learned fuzzy datatypes. If we use k-means with $k = 5$ fuzzy datatypes, we end up trying to learn 4760 fuzzy datatypes, although for some combinations of data property and day segment there were no data. Finally, the result of the categorization was approved by human experts from the private company, but details about the final categorization are confidential.

Note that indeed the fuzzy datatypes add more knowledge, in the sense that we can make new inferences. For example, two individuals with slightly different heart rates at work, even if such values are different than the center of the triangular function (20.7), would be compatible with the fuzzy datatype LowHeartRateAtWork, possibly with different degrees of truth.

Related work

Automatic lifestyle profiling tries to categorize users according to their daily routine based lifestyles is an unexplored area. In a computational application the behaviors can

be represented by measurements from wearable sensors. The lifestyle of an individual can then be modeled by the statistics of the measurements conditioned by the elements of the surrounding structure. A model of a lifestyle can be discovered blindly from the data using clustering methods or it may be based on matching a predefined semantic template to the data. A blind method to model routines as linear combinations of eigen behaviors was proposed in [EP09]. The use of blind modeling is difficult because the discovered routines lack semantics that is needed to provide understandable feedback to the user. The lifestyle model proposed in [WHC+16] was based on matching a semantic template of workday and weekend routines to the wearable data. Semantic template models have also been used for mining personalized insights from wearable sensor data [IL11] to attach semantics to recurring ambulatory patterns [HG17]. Instead, we propose an approach based on fuzzy ontologies that supports imprecise definitions and uses two strategies to learn the elements of the fuzzy ontology from real data.

MoveUp app categorizes users in the mHealth domain according to the their activity using fuzzy logic [ST22]. Quoting the authors, their "fuzzy models developed are not devoted to discover the lifestyle of a person", like in our work, "but to derive a synthetic representation of real-time data of a person with declared, problematic lifestyle, to promote a behavior change". Another differences with respect to our work are that MoveUp does not use ontologies or semantic reasoning, and that the fuzzy membership functions are not automatically learned from real data.

Semantically meaningful and interpretable models to better understand the underlying statistics of individual lifestyle patterns of people is not a trivial task because of the variability of the individuals. Even if technology allows for a broad spectrum of sensors, it is not straightforward to choose the most appropriate data acquisition, data imputation and data fusion techniques [LKR07]. Attention should also be put into semantic definitions in order to achieve matching of lifestyle coaching programs, to target compatible profiles (i.e., accounting for the user's devices -and their metric units-, their diseases, time schedules, and hobbies). Common-sense representation can enhance data-driven processes and improve accuracy and precision of recognition in human activities [DRCC+14, DRCLD14, DRPCLD14]. Likewise, knowledge-driven human activity models can be upgraded through data-driven learning techniques [AAdIC15, AY09] improving accuracy and clustering techniques.

6.6.3 Fuzzy linguistic summaries

Contributions

In this section, we discuss another application of Datil to real data. The aim is to summarize the set of attribute data using a more human-friendly representation, namely linguistic terms described using fuzzy sets. This technique is an example of Explainable Artificial Intelligence (XAI), which "produces details or reasons to make its functioning clear or easy to understand" [BDRD+20].

In the domain of biometric systems, we developed a gait recognition system to identify people based on the way they walk (see Section 6.1). To test it, we generated a novel dataset with records of 91 volunteers walking a straight line, obtained using Kinect Microsoft version 2. In particular, each individual has 211 biometric features (with numerical range) associated, for example, the stepLength.

In order to describe the physical features of a human by means of linguistic labels, we used Datil tool to generate the fuzzy membership functions from dataset biometric features. For example, the linguistic label HighLengthStep was defined using a triangular fuzzy membership function. Furthermore, we can evaluate the membership degree of an individual with stepLength= 0.5467 as shown in Figure 6.40.

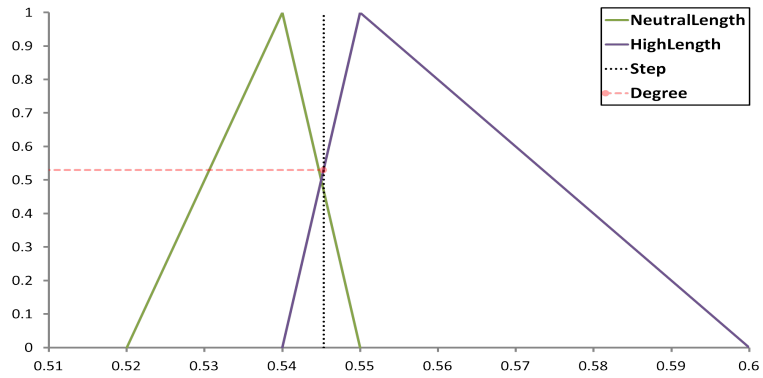


Figure 6.40: Membership degree to HighLengthStep of an individual with stepLength= 0.5467.

The main features of this approach are:

– It uses biometric features from a real use case and makes it possible to summarize them.

– It helps users to read and understand linguistic labels rather than single numbers to describe the features of a person. For instance, it provides explanations of specific properties as height or humerus size.

– Regarding the dataset, it permits to enrich the fuzzy ontologies with new fuzzy datatypes and use them in the description of a domain, e.g., in our gait recognition system.

– Regarding the gait recognition system, it improves system interpretability. For example, the recognition system can return the most similar individual and the summary can explain the biometric values using linguistic labels.

Related work

Yager introduced in 1989 the idea of building linguistic summaries to synthesize information [Yag89]. The most common scenario is building fuzzy linguistic summaries from numerical input data. For example, some of the applied techniques include clustering [SPL17], fuzzy decision trees [PS05] or hierarchical fuzzy partitioning [GC04]. For a good introduction to the field we refer the reader to [Yag21]; a more detailed overview of the state of the art can be found in [BMM12].

However, the combination of fuzzy linguistic summarization with fuzzy ontologies has not received a lot of attention. The only exception we are aware of is a fuzzy ontology-based approach to news summarization in Chinese language [LJH05]. To the best of our knowledge, ours is the first application of fuzzy linguistic summarization to biometric features (used in a gait recognition system).

6.7 Evaluation of Fudge aggregation

Contributions

In this section we describe the evaluation of the novel aggregation operator CONV–LRF (Eq. 1), supported in the *Fudge* software described in Section 3.3. In particular, we performed an evaluation comparing CONV–LRF with the existing similar operator CONV–RRF. Our objective is to give some insights about the similarities and differences of both operators.

Because these operators can be used to define LOWA–RRF and LOWA–LRF, our findings will affect these operators as well. However, because the difference between CONV and LOWA are whether weights are assigned to a specific expert or not, an evaluation of the LOWA operators does not seem necessary.

Experiment 1. To understand the differences between CONV–LRF and CONV–RRF, we computed the results of the aggregation of four values (i.e., the opinions of four experts) with different vectors of weights. Specifically, we

evaluated the combination of the vector $[\mathbf{d}_4, \mathbf{d}_3, \mathbf{d}_2, \mathbf{d}_1]$ with each possible combination of degrees $w_i = k \cdot 0.01, \sum_{i=1}^{4} w_i = 1, k \in \{1, 2, \ldots, 100\}, i \in \{1, 2, 3, 4\}$. Out of the 156849 possibilities, CONV–LRF and CONV–RRF give the same value in 114259 cases (72.85 %). In 39538 cases (25.2 %), CONV–LRF returns a higher value, whereas in 3052 cases (1.95 %) CONV–RRF returns a higher value. Therefore, CONV–LRF seems to have a slightly higher orness than CONV–RRF. When the weights associated to the higher value or to the smaller value were greater than 0.83, both approaches coincided. This was also the case when the other weights were greater than 0.74.

Experiment 2. This experiment is similar but considering a different ordering of the labels to be aggregated, i.e., $[\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4]$. Now, out of the 156849 possibilities, CONV–LRF and CONV–RRF give the same value in 114210 cases (72.82 %). In 39997 cases (25.50 %), CONV–RRF returns a higher value, whereas in 2642 cases (1.68 %) CONV–LRF returns a higher value. Therefore, it is CONV–RRF the one with a slightly higher orness. Analyzing the results of the first two experiments, when the labels are presented in a decreasing order, CONV–LRF returns higher values in general, but when the labels are presented in an increasing order, CONV–RRF returns higher values in general.

Experiment 3. This experiment generalizes Experiments 1 and 2 by considering all possible combinations of values to be aggregated, and different numbers of experts (from 2 to 5, always with non-zero weights). The number of total labels was fixed to 5, which is a very common option in practice.

Table 6.26 shows the results of the evaluation: the number of experts ("# Experts"), total number of cases ("# Cases"), percentage of cases where CONV–LRF gives a higher value ("% CONV–LRF"), percentage of cases where CONV–RRF gives a higher value ("% CONV–RRF"), and percentage of cases where both operators coincide ("% Coincidence").

It is clear that for 2 experts both operators always coincide. Indeed, it is the base case and both operators have the same definition. With a higher number of experts, both operators do not always coincide, and the percentage of coincidences is inversely proportional to the number of experts. It is also interesting to remark that both operators return a higher value in more or less the same number of cases, regardless of the number of experts.

To conclude this section, note that [HBGRS20] only includes Experiment 1, which suggests that CONV–LRF has a higher orness than CONV–RRF, but Experiment 3

# Experts	# Cases	% CONV–LRF	% CONV–RRF	% Coincidence
2	2500	0	0	100
3	631250	6.62	6.67	86.71
4	107312500	10.74	10.83	78.43
5	13816484375	13.45	13.52	73.03

Table 6.26: Results of the comparison between CONV–LRF and CONV–RRF.

shows that the ornesses are similar, with the orness of CONV–RRF being actually slightly higher.

6.8 Evaluation of the serializable and incremental fuzzyDL reasoner

Contributions

In this part, we describe our experimentation with the serializable and incremental version of fuzzyDL (explained in Section 5.3), the first reasoner of this type. We only compare fuzzyDL reasoner with JFact because it is the only serializable semantic reasoner (see Table 5.2).

The structure is the following. First, Section 6.8.1 describes the datasets and the experimental setup used in our evaluation. Then, Section 6.8.2 shows and discusses the results.

6.8.1 Experimental setup

Ontology setup. To perform our evaluation we used three datasets:

- *Fuzzy Beer* ontology, detailed in Section 6.2.
- *Absorption* dataset, detailed in Section 6.4.1. In this section, we will consider the original 50 crisp ontologies and Fuzzy Wine.
- *ORE 2013* dataset, developed in [GBJR+13]. We considered 36 large OWL 2 DL ontologies. ORE 2013 dataset contains 200 ontologies per profile (i.e., OWL 2 EL, OWL 2 RL, and OWL 2 DL) from the NCBO BioPortal[32], the Oxford Ontology Library[33], and the Manchester Ontology Repository[34]. Ontologies are classified according to their number of logical axioms as *small* (≤ 500), *medium* (between 500 and 4999), and *large* ontologies (≥ 5000).

[32] `http://bioportal.bioontology.org`
[33] `http://www.cs.ox.ac.uk/isg/ontologies`
[34] `http://rpc295.cs.man.ac.uk:8080/repository`

In total, we considered 88 ontologies but, as we will discuss later, there were errors while reading or managing some of them.

Parameters. Since we compare fuzzyDL with the crisp ontology reasoner JFact, in this section fuzzyDL assumes a semantics based on classical logic. When using JFact, ontologies are encoded in OWL 2 syntax, and when using fuzzyDL in FDL format.

Both reasoners use the Java serialization strategy. While fuzzyDL computes the preprocessing discussed in Section 5.3, JFact uses the method precomputeInferences to compute the following axioms: class assertions, class hierarchy, object property assertions, data property assertions, object property hierarchy, data property hierarchy, same individuals, different individuals, and disjoint classes.

During our experiments, we set a timeout of two hours to solve the required tasks. Experiments were repeated 5 times and we took the standard average of the computed values.

Equipment and tools. All experiments were performed on a laptop computer with Intel Core i7-8550U 1.8 GHz, 16 GB RAM under Windows 7 64-bits. The versions of the software were Java 1.8, JFact 4.0.4, OWL API 4.2.7, and Gurobi 8.1.0 build V8.1.0rc1 (Academic License); these are the versions of JFact and OWL API that were used in a comparison between JFact serialization and Fact++ persistence [BBMP17].

6.8.2 Results and discussion

Firstly, we discuss an evaluation of the serialized files (size and time) and then we discuss an evaluation of the reasoning time.

Evaluation of the serialized files: size and time. In this part of the evaluation, for each ontology in the datasets, we preprocess it, we serialize the reasoner (including the expanded fuzzy ontology) into a file, and we deserialize it.

The results of our experiments are shown in Table 6.27 for those ontologies that were successfully processed by both JFact anf fuzzyDL (datasets are separated using horizontal lines). For both reasoners we show three values:

- *SeriSize*: size in MB of the serialized reasoner (including the fuzzy ontology) after preprocessing the ontology.

- *SaveTime*: time in seconds needed to obtain a serialized version of the reasoner and to save it into a file.

– *LoadTime*: time in seconds needed to restore a version of the reasoner from a serialized file.

	JFact			**FuzzyDL**		
Ontology	SeriSize (MB)	SaveTime (s)	LoadTime (s)	SeriSize (MB)	SaveTime (s)	LoadTime (s)
Beer	120.89	65.96	91.88	18.95	7.46	11.47
amino-acid	0.22	0.25	0.30	0.04	0.03	0.04
cancer_my	0.31	0.32	0.40	0.04	0.04	0.05
chemical	0.16	0.22	0.27	0.02	0.03	0.02
EMAP.obo	26.41	8.67	12.72	2.85	1.34	2.14
FMA	1079.97	104.26	150.88	43.71	25.13	34.71
FuzzyWine	1.24	0.88	1.17	0.17	0.17	0.23
galen-ians-full-doctored	6.67	3.57	5.5	1.35	0.42	0.74
gene_ontology_edit.obo	6.67	3.54	5.19	4.97	2.59	3.83
goslim	0.30	0.18	0.27	0.05	0.06	0.06
lubm	8.66	4.76	6.86	3.83	3.02	3.90
matchmaking	0.27	0.30	0.37	0.03	0.03	0.03
pathway.obo	0.82	0.54	0.72	0.09	0.07	0.09
people.fd	0.25	0.53	0.65	0.04	0.04	0.05
pizza	0.38	0.40	0.45	0.05	0.08	0.12
po	0.86	0.56	0.74	0.06	0.08	0.08
SIGKDD-EKAW	0.34	0.33	0.42	0.03	0.03	0.04
so-xp.obo	2.64	1.39	2.01	0.31	0.17	0.24
spatial.obo	0.26	0.28	0.36	0.05	0.03	0.05
teleost_taxonomy.obo	33.58	19.37	27.90	4.84	2.63	3.81
worm_phenotype_xp.obo	2.90	1.38	2.20	0.46	0.18	0.30
teleost-taxonomy.1081	27.60	23.11	27.83	5.20	3.22	5.31

Table 6.27: Serialization of the ontologies in all datasets using *JFact* and *FuzzyDL*.

As we can see, fuzzyDL always computes smaller files. The differences are significant for (Fuzzy) Beer and for the Absorption dataset, but are quite impressive for ORE 2013. For example, while JFact requires 120.89 MB to serialize Fuzzy Beer ontology, fuzzyDL only uses 18.95 MB. It is worth to recall, however, that JFact does include some information that fuzzyDL does not (inferred class hierarchy).

Regarding the serialization and the deserialization times, we can see that deserialization is slightly slower than serialization. As both reasoners use the Java serialization strategy, and because fuzzyDL manages smaller files, it is not surprising that fuzzyDL is always faster. The differences can also be very important; for the example discussed in the previous paragraph, JFact requires 65.96 s to serialize and 91.88 s to restore, whereas fuzzyDL requires 7.46 s and 11.47 s, respectively.

22 ontologies (25%) were supported by both reasoners and in 66 ontologies (75%) at least one of two reasoners failed, 31 in the Absorption dataset (61 % of the dataset) and 35 in the ORE 2013 dataset (97 % of the dataset). Ontologies where at least one of the two reasoners failed are shown in Table 6.28 for Absorption dataset and Table 6.29 for ORE 2013 dataset.

JFact failed in 53 ontologies and fuzzyDL in 40. Focusing on fuzzyDL, we found the following problems:

– 21 timeouts,

– 12 ontologies included OWL 2 elements that are not currently supported by fuzzyDL, e.g., object property chains, cardinality restrictions, enumerations, or universal data property restrictions.

– 2 parsing errors when importing the ontology (for example, because of a non-ASCII character "á"), and

– 5 null pointer exceptions, requiring further investigation.

Ontology	**JFact**	**FuzzyDL**
AirSystem	Could not load imported ontology	Null pointer
atom-common	Could not load imported ontology	
biochemistry-complex	Could not load imported ontology	Timeout
chebi	Stack overflow	
chemistry-complex	Could not load imported ontology	Timeout
cton	Stack overflow	
earthrealm	Could not load imported ontology	
Economy	Illegal redeclarations of entities	
FBbt_XP		Timeout
fmaOwlDlComponent_1_4_0	Timeout	
GRO		Timeout
heart		Timeout
legal-action	Could not load imported ontology	Timeout
mosquito_insecticide_resistance.obo	Could not load imported ontology	Null pointer
mygrid-moby-service	Could not load imported ontology	
NCI	Stack overflow	
norm	Could not load imported ontology	Timeout
ontology		Timeout
organic-compound-complex	Could not load imported ontology	Timeout
periodic-table-complex	Could not load imported ontology	
photography		Timeout
PRO		Timeout
process	Could not load imported ontology	
propreo	Non simple role used as simple	
reaction	Could not load imported ontology	Timeout
relative-places	Could not load imported ontology	
subatomic-particle-complex	Could not load imported ontology	
time-modification	Could not load imported ontology	Timeout
thesaurus	Illegal redeclarations of entities	
Transportation	Unfound datatype	
yowl-complex	Could not load imported ontology	Timeout

Table 6.28: Ontologies from Absorption dataset with errors during the serialization.

Evaluation of the reasoning time. In this part of the evaluation, we focus on the reasoning time of the serialized incremental version of fuzzyDL. We do not compare fuzzyDL with JFact because it is not incremental, as discussed in Section 5.3. For each of the 48 ontologies where fuzzyDL did not fail, Table 6.30 shows several measures:

– *LoadTime*: time to load the ontology from a text file. This value is used to compare with the non-incremental version of the reasoner.

Ontology	JFact	FuzzyDL
00004		Timeout
00035	Out of memory	
00347	Stack overflow	Null pointer
00368	Stack overflow	
00371	Timeout	
00374	Timeout	
00386	Illegal redeclarations of entities	
00390	Illegal redeclarations of entities	
00398	Timeout	
00400	Illegal redeclarations of entities	
00462	Timeout	ObjectPropertyChain not supported
00463		ObjectPropertyChain not supported
00631		Timeout
00678		ObjectPropertyChain not supported
00680		ObjectPropertyChain not supported
00761	Stack overflow	Timeout
01c1c6df-2a38-46ce-a554-35f273f2ed1a_1245	Stack overflow	Timeout
16a2cd46-f29d-4d82-94e8-b4a225a5cb5a_bo	Out of memory	Timeout
290113a0-5a1b-4f85-a716-ced96a6499e9__links	Stack overflow	
3ebf89a1-6a06-4e1e-b03e-a8ddb0d6c335_rocess	Stack overflow	Null pointer
42d22996-95bc-4375-9552-f011acffebfb_cation		Parser error
5043d1d7-86d8-4e78-af21-f5eca30cab3f_tology	Inconsistent ontology	OneOf, Cardinality not supported
53f9f8e8-9463-4612-b082-45508fb37137_chains	Relevant vertex of type bad-tag	DataAllValuesFrom not supported
63153319-5b25-43ed-86b2-f0714c0acd7c_O-full	Inconsistent ontology	Cardinality not supported
645a3ff8-698d-4904-9cff-9bf134ff6a0c_tology	Literal time cannot be found	OneOf, Cardinality not supported
7e1c9977-4d4b-49ab-808f-1ca185be99c5_BCR	Inconsistent ontology	DataOneOf, Cardinality not supported
cbe2e729-0b10-4b02-80f5-e91b1c8bedf7_test2	Timeout	Timeout
cell-line-ontology.1245	Stack overflow	Timeout
d0e20d33-6bfa-4115-aba4-3a3f4ba8d586_mplied	Stack overflow	
d5c7f91d-b5eb-4af1-9293-d90e7ff63b1e_1070	Stack overflow	
e5c03a5b-05db-440a-ae23-4eadcaa1114f_oOntol		Parser error
ebf8d261-ed24-4f9f-8cc5-cd52bb1f5e1d_cton	Stack overflow	Null pointer
FMA-constitutionalPartForNS_ALCOI(D)	Stack overflow	OneOf, DataHasValue not supported
GALEN-Full-Union_ALCHOI(D)	NumberFormatException	OneOf not supported
GALEN-Heart_ALCHOI(D)	NumberFormatException	OneOf not supported

Table 6.29: Ontologies from ORE 2013 dataset with errors during the serialization.

Ontology	LoadTime (s)	CloneTime (s)	PreprocessTime (s)	SubTime (s)	SatTime (s)	EntTime (s)
Beer	0.95	0.040	11.91	0.02	2247.33	2250.22
amino-acid	0.02	0.0004	0.003	0.87	0.27	Timeout
atom-common	0.01	0.0004	0.001	0.02	0.008	Timeout
cancer_my	0.01	0.001	0.003	0.11	2.75	2.65
chebi	3.36	1.823	0.55	0.09	0.01	Timeout
chemical	0.01	0.0004	0.002	5.00	4.59	Timeout
cton	0.17	0.005	0.10	0.03	0.007	Timeout
earthrealm	0.05	0.003	0.02	0.39	0.36	0.34
Economy	0.01	0.003	0.02	0.01	0.07	0.06
EMAP.obo	0.11	0.005	0.05	0.02	0.006	Timeout
FuzzyWine	0.09	0.001	0.02	0.02	284.51	272.19
fmaOwlDlComponent_1_4_0	0.44	0.007	0.09	0.07	Timeout	Timeout
FMA	0.96	0.167	0.33	0.12	0.029	Timeout
galen-ians-full-doctored	0.08	0.007	0.04	Timeout	0.008	Timeout
gene_ontology_edit.obo	0.21	0.012	0.09	0.02	0.009	Timeout
goslim	0.01	0.001	0.001	0.004	0.02	0.02
lubm	0.38	0.007	2.31	0.003	495.16	516.7
matchmaking	0.01	0.0004	0.001	0.03	0.002	Timeout
mygrid-moby-service	0.04	0.0006	0.01	0.02	0.002	Timeout
NCI	0.25	0.011	0.13	0.04	0.02	Timeout
pathway.obo	0.02	0.0004	0.006	0.01	0.002	Timeout
people.fd	0.01	0.001	0.002	0.01	0.85	0.85
periodic-table-complex	0.01	0.0002	0.004	0.07	0.03	Timeout
pizza	0.04	0.000	0.01	163.51	0.01	0.01
po	0.01	0.001	0.003	0.81	0.84	0.81
process	0.07	0.003	0.02	0.44	0.37	0.40
propreo	0.03	0.001	0.01	Timeout	394.95	Timeout
relative-places	0.01	0.0002	0.001	0.10	0.06	Timeout
SIGKDD-EKAW	0.02	0.0004	0.002	0.41	0.002	Timeout
so-xp.obo	0.04	0.001	0.02	0.02	0.002	Timeout
spatial.obo	0.02	0.0002	0.006	0.06	0.002	Timeout
subatomic-particle-complex	0.02	0.0004	0.003	0.10	0.06	Timeout
teleost_taxonomy.obo	0.17	0.017	0.08	0.03	0.01	Timeout
thesaurus	0.67	0.031	0.37	0.40	0.19	0.20
Transportation	0.01	0.001	0.004	0.01	0.03	0.03
worm_phenotype_xp.obo	0.05	0.002	0.01	0.01	0.003	Timeout
00035	0.13	0.003	0.11	Timeout	0.01	Timeout
00368	0.40	0.216	0.15	0.02	0.01	305.19
00371	0.48	0.167	0.30	Timeout	0.01	346.26
00374	0.66	0.166	0.43	0.03	0.01	388.31
00386	0.50	0.192	0.30	Timeout	0.01	343.11
00390	0.41	0.170	0.29	Timeout	0.01	304.40
00398	0.46	0.165	0.28	Timeout	0.01	346.48
00400	0.43	0.163	0.36	Timeout	0.01	421.10
290113a0-5a1b-4f85-a716-ced96a6499e9_links	0.22	0.013	0.10	0.01	Timeout	5.15
d0e20d33-6bfa-4115-aba4-3a3f4ba8d586_mplied	0.21	0.007	0.16	0.02	0.01	Timeout
d5c7f91d-b5eb-4af1-9293-d90e7ff63b1e_1070	0.30	0.011	0.14	0.06	0.01	Timeout
teleost-taxonomy.1081	0.26	0.021	0.13	0.05	0.02	Timeout

Table 6.30: Evaluation of different parts of the reasoning in fuzzyDL.

– *CloneTime*: time to obtain a copy of an object representing the reasoner. This value is used to compare with the time needed to load the state of a reasoner from its serialization.

– *PreprocessTime*: time to preprocess the ontology, computing all the inferences that can be shared when solving any query.

– *SubTime*: time to solve a concept subsumption query, assuming that the ontology has already been preprocessed. This query considers two randomly selected named concepts in the ontology.

– *SatTime*: time to solve a concept satisfiability query, assuming that the ontology has already been preprocessed. This query considers a randomly selected named concept in the ontology.

– *EntTime*: time to solve an entailment query, assuming that the ontology has already been preprocessed. In particular, we consider the entailment of a concept assertion axiom involving an individual and a named concept in the ontology, both randomly selected.

Based on these values, Table 6.31 shows the following times[35]:

– *TimeR* (restore): time to prepare incremental and serializable reasoning, i.e., time to deserialize the reasoner.

– *TimeO* (old): time to solve a query without incremental reasoning. This includes the time to load the ontology from a text file, the time to preprocess it, and the time to solve the query.

– *TimeD* (download): time to obtain a remote serialized file and to prepare incremental and serializable reasoning. This includes the time to download the file, and the time to deserialize the reasoner. The time to download the file is estimated by dividing the size file (shown in Table 6.27) by the data transfer speed. The data transfer speed depends on the technology (e.g., WiFi, mobile broadband, etc.) and is typically rather variable; we have assumed 17.6 Mbps, as it was the global (after analyzing 87 countries) average mobile connection in 2019 [Boy19].

[35]Note that, while TimeR and TimeD are independent of the query type, TimeO and TimeQ have to be considered for each query type (concept subsumption, concept satisfiability, and entailment of a concept assertion).

Ontology	TimeR (s)	TimeD (s)	Subsumption		Satisfiability		Entailment	
			TimeO (s)	TimeQ (s)	TimeO (s)	TimeQ (s)	TimeO (s)	TimeQ (s)
Beer	11.47	20.08	12.89	0.06	2260.19	2247.37	2263.09	2250.26
amino-acid	0.04	0.06	0.89	0.87	0.28	0.27	Timeout	Timeout
atom-common	0.02	0.02	0.03	0.02	0.02	0.01	Timeout	Timeout
cancer_my	0.05	0.07	0.12	0.11	2.76	2.76	2.66	2.65
chebi	157.46	223.07	4.00	1.91	3.92	1.83	Timeout	Timeout
chemical	0.02	0.03	5.01	5.00	4.60	4.59	Timeout	Timeout
cton	2.15	3.51	0.31	0.04	0.28	0.01	Timeout	Timeout
earthrealm	0.80	1.11	0.45	0.39	0.43	0.36	0.40	0.34
EMAP.obo	2.14	3.44	0.18	0.02	0.17	0.01	Timeout	Timeout
Economy	0.47	0.72	0.03	0.01	0.09	0.07	0.09	0.07
FuzzyWine	0.23	0.30	0.13	0.02	284.62	284.52	272.3	272.19
fmaOwlDlComponent_1_4_0	2.26	3.45	0.60	0.08	Timeout	Timeout	Timeout	Timeout
FMA	34.71	54.58	1.41	0.29	1.31	0.20	Timeout	Timeout
galen-ians-full-doctored	0.74	1.36	Timeout	Timeout	0.13	0.01	Timeout	Timeout
gene_ontology_edit.obo	3.83	6.09	0.33	0.04	0.31	0.02	Timeout	Timeout
goslim	0.06	0.09	0.01	0.01	0.03	0.02	0.03	0.02
lubm	3.90	5.64	2.69	0.01	497.85	495.17	519.39	516.71
matchmaking	0.03	0.05	0.04	0.03	0.02	0.003	Timeout	Timeout
mygrid-moby-service	0.12	0.19	0.08	0.02	0.05	0.003	Timeout	Timeout
NCI	4.36	7.73	0.43	0.05	0.40	0.03	Timeout	Timeout
pathway.obo	0.09	0.13	0.03	0.01	0.03	0.003	Timeout	Timeout
people.fd	0.05	0.07	0.02	0.02	0.85	0.85	0.85	0.85
periodic-table-complex	0.05	0.07	0.08	0.07	0.05	0.03	Timeout	Timeout
pizza	0.12	0.15	163.56	163.51	0.06	0.01	0.06	0.01
po	0.08	0.11	0.82	0.81	0.85	0.84	0.82	0.81
process	0.98	1.30	0.53	0.45	0.46	0.37	0.49	0.40
propreo	0.10	0.16	Timeout	Timeout	394.99	394.95	Timeout	Timeout
relative-places	0.02	0.03	0.11	0.10	0.08	0.06	Timeout	Timeout
SIGKDD-EKAW	0.04	0.05	0.42	0.41	0.02	0.003	Timeout	Timeout
so-xp.obo	0.24	0.38	0.08	0.02	0.06	0.002	Timeout	Timeout
spatial.obo	0.05	0.07	0.08	0.06	0.02	0.003	Timeout	Timeout
subatomic-particle-complex	0.05	0.07	0.12	0.10	0.08	0.06	Timeout	Timeout
teleost_taxonomy.obo	3.81	6.01	0.28	0.04	0.27	0.02	Timeout	Timeout
thesaurus	11.47	21.04	1.44	0.43	1.23	0.22	1.24	0.23
Transportation	0.21	0.28	0.02	0.01	0.05	0.03	0.04	0.03
worm_phenotype_xp.obo	0.30	0.51	0.07	0.01	0.06	0.01	Timeout	Timeout
00035	2.09	3.98	Timeout	Timeout	0.25	0.01	Timeout	Timeout
00368	21.36	33.84	0.57	0.24	0.55	0.22	305.73	305.40
00371	20.87	33.39	Timeout	Timeout	0.78	0.18	347.03	346.42
00374	28.93	41.45	1.12	0.20	1.10	0.17	389.40	388.47
00386	27.02	39.56	Timeout	Timeout	0.80	0.20	343.90	343.30
00390	20.7	32.53	Timeout	Timeout	0.71	0.18	305.10	304.57
00398	21.06	33.25	Timeout	Timeout	0.75	0.17	347.22	346.65
00400	22.59	35.61	Timeout	Timeout	0.81	0.17	421.89	421.26
290113a0-5a1b-4f85-a716-ced96a6499e9__links	3.99	6.33	0.35	0.04	0.33	0.02	Timeout	Timeout
d0e20d33-6bfa-4115-aba4-3a3f4ba8d586_mplied	3.06	4.95	0.39	0.03	0.38	0.01	Timeout	Timeout
d5c7f91d-b5eb-4af1-9293-d90e7ff63b1e_1070	6.10	7.99	0.51	0.07	0.46	0.02	Timeout	Timeout
teleost-taxonomy.1081	5.46	7.82	0.45	0.07	0.41	0.04	Timeout	Timeout

Table 6.31: Reasoning times of the classical version and the serializable version of fuzzyDL.

– *TimeQ* (query): time to solve the queries using incremental and serializable reasoning: time to clone the restored version of the reasoner plus time to solve the query.

The main finding of our experiments is that being an incremental reasoner clearly decreases the reasoning time: to answer the first query, the reasoning time is the same in both non-incremental and incremental versions (because both need to expand the ontology and solve the query), but for the next queries, it is possible to save the time to compute the ontology and get all the inferences (PreprocessTime). Instead, one needs to compute a local copy of the reasoner (CloneTime), which is much faster. We can indeed see that TimeQ is always smaller than TimeO, for all query types (see Table 6.31). The decrease in the reasoning time is modest for easy ontologies, but can be quite significant for relatively complex ones. For example, in Fuzzy Beer, the incremental version requires 0.06 s instead of 12.89 s to solve the Subsumption query. However, there are also ontologies where PreprocessTime is much smaller than the query time (e.g., subsumption in Pizza in Table 6.27), so in this case the decrease of the reasoning time in the incremental version is very small.

The time to restore the ontology (TimeR) can be significant, even higher than TimeO, but the advantage is that only needs to be computed once for a given ontology. If we also need to download the serialized version of the reasoner (TimeD), the time slightly increases but because it is only done once, the decrease in the reasoning time makes it worth.

Chapter 7

Conclusions and future work

In this chapter, we summarize the main conclusions of this thesis, dedicated to the advance management of fuzzy semantic knowledge. Conclusions are grouped in several categories: contributions to fuzzy ontology learning (Section 7.1), fuzzy ontology reasoning (Section 7.2), support for mobile devices (Section 7.3), and development of real-world applications (Section 7.4). Finally, we discuss the main directions to follow in future work (Section 7.5).

7.1 Fuzzy ontology learning

In this section, we summarize the main conclusions to the learning of elements of fuzzy ontologies, which demonstrates that objective **O1** has been met.

- We proposed a strategy to learn local fuzzy datatypes, specific of one individual. Using the values of numerical data properties, we build a triangular fuzzy membership function per each data property using the average and the standard deviation. This approach has been validated on a gait recognition application, where the imprecision of the device used to capture the data can be managed by combining the values corresponding to different video frames.

- We proposed a strategy to learn global fuzzy datatypes from the values of numerical data properties. After running a centroid-based clustering algorithm, it uses the centroids as the parameters of some fuzzy membership functions that partition the domain. This approach was implemented in the Datil system, which supports different input and output formats, three well-known clustering algorithms, and can extend crisp ontologies to the fuzzy case but also enrich existing fuzzy ontologies. The tool automatically computes readable names for the discovered fuzzy datatypes. We have also discussed how fuzzy datatype

learning has been applied to three real-world applications: semantic lifestyle profiling, beer recommendation, and gait recognition.

  - In lifestyle profiling, we characterized the lifestyle of people given their digital footprints: we used numerical data obtained from different sensors to build fuzzy datatypes using Datil, so that the definitions of categories use linguistic terms that are easily interpretable by human users.
  - In gait recognition, Datil was used to compute linguistic summaries of the features of the individuals in the knowledge base. This can be used to provide explanations of the decisions of the system, promoting Explainable Artificial Intelligence.
  - In the beer domain, Datil computed the definition of fuzzy dataypes representing, for example, a low alcohol degree, or a very high bitterness. A group of experts evaluated the quality of the linguistic labels computed by Datil, showing that it is similar to the definitions given by humans. Furthermore, we were able to identify the best learning strategy: using mean-shift clustering algorithm.

- We solved the problem of building a single fuzzy datatype from multiple fuzzy datatype definitions by using linguistic aggregation operators. Our proposal supports Fuzzy OWL 2 fuzzy datatypes and is implemented in the Fudge system. While aggregation operators have been previously used in fuzzy ontologies to combine numerical values in $[0, 1]$, we focus on the aggregation of fuzzy membership functions. Our approach can use new and existing aggregation operators. In particular, we have proposed two new operators, namely a left recursive form of the convex combination (CONV–LRF) and of the linguistic OWA (LOWA–LRF). An empirical analysis shows that the percentage of coincidences between both operators is inversely proportional to the number of experts, and that both operators have a similar orness degree.

  Regarding existing aggregation strategies, we have studied CONV–RRF, standard linguistic OWA (LOWA–RRF), weighted mean (WMEAN), and fuzzy OWA (FOWA). We discussed the most relevant characteristics of all these aggregation strategies, e.g. the possibility to assign a weight to a specific expert, and the possibility to obtain as an output a value which was not provided by any expert. Furthermore, we showed how to obtain the weights from fuzzy quantifiers or an orness value in problems with incomplete data —in which not all experts provide a definition for every fuzzy datatype.

7.2 Reasoning

This section summarizes the main conclusions to fuzzy ontology reasoning, proving that objective **O2** has been met.

- We proposed two specific algorithms to solve 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 can be implemented in a family of algorithms to reason with fuzzy DLs combining a tableaux algorithm and an optimization problem, and 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.

  Our instance retrieval algorithm 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.

- We identified a novel reasoning task, flexible faceted instance retrieval, that extends the traditional fuzzy instance retrieval to narrow down the query results by imposing some conditions, described using fuzzy datatypes, on the values of some data properties. We also identified some cases that are common in practice (possible restrictions in the elements of the ontology that are actually fuzzy) and simplify the solution of the problem. In particular, we proposed two minimalist reasoning algorithms whose main idea is to reduce the problem to classical crisp inference, which can be solved by any classical semantic reasoner, and then perform some additional computation managing the fuzzy part of the ontologies. These algorithms have proved their validity on a beer recommender system and in querying Building Information Modeling files.

– We provided a novel approach to compute the fuzzy similarity between individuals in a fuzzy ontology. Compared to the only existing similar work, our approach is able to compute local similarities (i.e., between the values of a data property) when the values are represented using fuzzy datatypes instead of numerical values, and it is based on the defuzzification of an intersection of fuzzy sets. Furthermore, our global similarity (combining local similarities) is very general as it supports any aggregation operator. We also discussed the case where two individuals do not have the same properties. This approach was successfully evaluated in a gait recognition application.

– We provided a novel approach to compute a fuzzy matchmaking between individuals in a fuzzy ontology. We formulated our problem as a fuzzy ontology reasoning task, computing the best satisfiability degree of a combination of fuzzy concepts representing the constraints of each of the parts. We extended previous approaches by considering more than two involved parts and by allowing more general functions. We also showed that strict weighted sum and strict t-norms lead to Pareto-optimal solutions. Using strict weighted sum, we can control too unfair agreements by weighting the importance of the involved parts. We also showed that all parts should include existential restrictions involving the same properties to deal correctly with the Open World Assumption.

7.3 Mobile devices

In this section, we summarize the main conclusions to the support for fuzzy ontologies on mobile devices, demonstrating that objective **O3** has been met.

– In order to increment the plethora of tools supporting fuzzy ontologies on mobile devices, we developed two Android versions of Datil and Fudge. To the best of our knowledge, these are the first apps to learn fuzzy datatypes on mobile devices. Our evaluation of Datil shows that although the time required to finish the learning process is higher on a smartphone, it is acceptable. Furthermore, we observed that the difference with respect to a laptop computer depends on the choice of the clustering algorithm used in the learning.

– To enlarge the number of real-world fuzzy ontology-based apps, to promote local and remote reasoning on mobile devices, and to illustrate the usefulness of having distributed fuzzy ontology files, we developed GimmeHop, a beer recommender system for Android devices. It is a proof of concept showing that fuzzy logic, fuzzy ontologies, semantic reasoners, and both local and remote reasoning can

be combined in mobile applications. Our experiments about the data traffic and running time show that remote reasoning is feasible and cheap (in terms of both data traffic and time). Local reasoning is only feasible if we limit the number of individuals. The tested devices were able to support 2000–3000 beers, which in our opinion would be enough in practice for most bars or stores interested in recommending beers to their users.

– To promote hybrid reasoning on mobile devices, we developed a new version of the fuzzy ontology reasoner fuzzyDL to make it the first semantic reasoner that is both serializable and incremental. fuzzyDL can expand a fuzzy ontology with some inferences that can be reused when answering different queries. It is possible to serialize the Java object that represents the reasoner and save it into a file (serializable) and reuse those inferences without restarting from scratch (incremental). These features are particularly interesting for mobile devices, but can be used on any device.

  Our experiments show that fuzzyDL computes smaller serialized files than JFact, the only other semantic reasoner that is serializable. fuzzyDL is also faster at both serializing and deserializing. While being incremental is helpful at decreasing the reasoning time, being also serializable slightly increases the cost of the first query because it is necessary to restore the serialized version of the file. We have also estimated the cost of downloading the file from a remote server before restoring the reasoner and it seems acceptable. Therefore, the idea of reusing from a mobile device a fuzzy ontology that was previously expanded in a different place (e.g., in a fast dedicated server) seems promising.

7.4 Real-world applications

This section summarizes the main conclusions to the development of fuzzy ontology-based real-world applications, demonstrating that objective **O4** has been met.

Profiling. We developed a system to define categories to classify the lifestyle profile of a person. To do so, we considered a large volume of heterogeneous data obtained from wearable sensors (used by a group of 40 volunteers) and described a methodology to learn subclass fuzzy ontology axioms to define categories of people. This methodology combines two fuzzy ontology learning techniques, the algorithm to learn fuzzy datatypes implemented in Datil, and the algorithm to learn subclass axioms implemented in Fuzzy DL-Learner. Both the data and the

results of the evaluation are owned by a private company and are confidential, so only a few details are given here.

Recommendation. We developed GimmeHop, a beer recommender system for Android mobile devices, using fuzzy ontologies and semantic reasoners. The application domain, beers, is a hot topic which is receiving a notable attention in the last years. In fact, two local companies were interested in the results of our project. Using a fuzzy ontology with more than 15000 beers, GimmeHop can be used to retrieve beers that satisfy some desired features, but also to retrieve similar beers to a given one. GimmeHop is able to deal with user context (in particular the location) by using fuzzy hedges, with user preferences by using weighted mean aggregation, and with incomplete data by using quantifier-guided OWA to provide weighting vectors with different sizes. We performed an extensive evaluation of several features of the system, namely the data traffic, the running time, the quality of the recommendations, and the quality of the linguistic labels.

Gait recognition. We described a gait recognition system based on Microsoft Kinect and fuzzy ontologies. This has several possible applications, including security and medicine. Our system is rather quick at computing the recognition, making it suitable for real-world scenarios. The main characteristic of our system is the use of fuzzy ontologies, which has several benefits: they are more robust against small changes in the values of the biometrical measures across different steps, allow more detailed (by assigning a degree to the classification of a person) and interpretable results (thanks to linguistic summaries), and all the benefits of classical ontologies, such as providing automated reasoning (e.g., to check that there are no inconsistencies), more readable datasets, or making maintenance and data integration across applications easier. We have proposed an architecture based on a schema ontology and several instantiated fuzzy ontologies including biometric features of individuals. Our approach makes it possible to represent both Microsoft Kinect V1 and V2 data and Nuitrack data, and all of the biometric features reported in the literature, plus some new ones (211 in total). We have also built a new dataset with 91 individuals (called the Zar2 dataset), discussed how to build the fuzzy ontology (enumerating several preprocessing techniques to improve the quality of the data), and showed how to use fuzzy logic to obtain linguistic descriptions of the biometric features.

Such fuzzy ontologies were used to train a novel gait recognition algorithm from Kinect data. The algorithm is based on step segmentation of the sequences,

use of fuzzy logic to deal with the imprecision of the sensor, and a voting scheme to aggregate the values obtained for each step. Our system provides an answer in a short time after the data are loaded in memory. The results of our experiments show that our new method outperforms existing algorithms in the case of individuals walking a straight line. We have also evaluated different defuzzification operators, with some differences in the accuracy depending on the dataset. Additionally, we evaluated our algorithm for lateral recordings using a moving camera. In this case, our algorithm usually performs worse and only the average defuzzification is comparable. We have also approached the problem of recognizing new individuals not included in our knowledge base, showing promising results.

Blockchain. We proposed a novel procedure to integrate fuzzy ontologies in blockchain systems. This way, users can represent flexible restrictions using fuzzy sets, and it is possible to develop smart contracts where there is a partial agreement among the involved parts. For example, this is useful for commercial transactions where some parameters are not strict but flexible, such as the price or the delivery time. Another advantage is that it is possible to avoid the ambiguity of natural languages, as well as to infer implicit knowledge or check for inconsistencies, because the involved parts use a formal language to represent the knowledge of an application domain. In particular, we proposed an architecture based on a common ontology schema, a personal fuzzy ontology for each of the involved parts, and a common ontology including the agreed values of the smart contract. Our approach has been implemented in the Ethereum network, using fuzzyDL reasoner to obtain the partial agreements, and IPFS P2P network to store the common ontology. To compute partial agreements, we used a novel strategy to compute a Pareto optimal fuzzy matchmaking between individuals.

Construction. We proposed a novel strategy to reason with larger Building Information Modeling files, closer to those used in real-world applications, based on flexible-faceted instance retrieval, and developed a prototype desktop application. We considered a real BIM model as a case of study. The model was converted from IFC to OWL (RDF syntax) using an existing tool. We showed that such a big model could not be supported by two classical reasoners. Hence, the final ontology was fragmented; for operativeness, we restricted the tests and evaluations to just the third floor of the dataset. Also, with this submodule we built a fuzzy ontology updated with new property assertions, concept assertions and fuzzy datatypes.

We evaluated the performance of our proposal by measuring the times of retrieval of the data property values of each individual. We conclude that TrOWL reasoner give us satisfactory results. We also found that the query times can be reduced when using additional data structures (an extra hash table). Another finding is that our tool requires a considerable initial time to classify the ontology, but following queries require less time.

7.5 Future work

To conclude this thesis, let us mention some possible directions for our future research.

Fuzzy ontology learning

- We plan to evaluate Datil on more real-world domains, as this practical experience will surely provide more ideas to extend our tool. It would also be interesting to evaluate Datil taking into account some objective measures, such as the partition coefficient (PC), the partition entropy (PE), or the Chen index [ESAA+18]. We could also implement more sophisticated clustering algorithms such as DBSCAN, which is more robust as it manages the noise [EKSX96, SSE+17], or incremental algorithms [BKT20] to support a dynamic update of the definitions.

- The modular design of Fudge makes it very easy to incorporate more fuzzy operators. Therefore, we plan to add more general linguistic aggregation operators, e.g., some not verifying internality or assuming a vector of non-numerical weights [Xu12]. More general linguistic operators include the linguistic weighted OWA [Tor97] or the fuzzy triangular ordered weighted arithmetic operators (based on a t-norm and a t-conorm) [SG18]. It would be interesting to support different quantifiers, or alternative rankings between fuzzy membership functions. Last but not least, it would be interesting to study other scenarios in which the experts are allowed to use already existing linguistic terms, possibly different to the labels used by other experts. To this aim, we plan to leverage existing work on defining reasoning-preserving mappings between local linguistic terms [ACEG+94].

- As some users have requested, we plan to wrap our implementations of Datil and Fudge as Protégé plug-in in order to better integrate the tool into the fuzzy ontology development process. This way, it would be possible to directly create a fuzzy ontology, learn some fuzzy datatypes, and query the aggregated ontology from Protégé. Right now, these steps can be carried out, but an intermediate step

involving Datil or Fudge is needed. The plug-ins may also be good for gathering data from user usage to carry out a more extensive experimental validation.

- Since datatype learning is a complementary technique to other approaches for fuzzy ontology learning, our implementations of Datil and Fudge could be used to extend Fuzzy DL-Learner, a system learning fuzzy subclass axioms.
- We would also like to study how to learn the definitions for non-numerical data properties, such as dates.

Fuzzy ontology reasoning

- Regarding the flexible faceted instance retrieval, it would be interesting to develop similar algorithms for other similar sets of restrictions on the fuzzy ontology.
- Regarding the instance retrieval algorithm, we could evaluate it 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 $\mathcal{O}_{two\_or\_more}$, solving all of them independently, or using a hybrid approach).
- We would also like to implement (by extending fuzzyDL) and evaluate the realization algorithm.
- Developing more specific algorithms for other reasoning tasks, such as the classification, would also be interesting.
- The similarity between individuals could be evaluated on more real applications, to have a better understanding of the impact of alternative fuzzy operators.
- Regarding the fuzzy matchmaking algorithm, a fuzzy similarity degree between non-numerical constraints would be useful.

Mobile devices

- The main task is developing a version of fuzzyDL working on mobile devices. Because it is implemented in Java, it seems easier to develop an Android version. So far, the only problem is that it uses a third-party library (Gurobi) for which currently there is no Android version. A possibility could be to replace Gurobi with another library solving mathematical optimization problems (in particular, Mixed Integer Linear Programming problems) completely developed in the fragment of Java compatible with Android. With this Android version,

one could investigate whether in mobile devices with limited resources the time to expand a fuzzy ontology, which is expected to be higher, will be higher than the deserialization time more often than in our evaluation. Another obvious idea is the development of a new version of fuzzyDL supporting the OWL API, so that it is possible to manage volatile information as proposed in [BBMP17]. Fortunately, to make the communication between the metareasoner and fuzzyDL possible, it might be possible to implement only a very small fragment of the OWL API. Last but not least, fuzzyDL parser to load OWL 2 ontologies could be improved (we found some bugs in ontologies encoded in the fragment of OWL 2 that fuzzyDL supports) and fuzzyDL preprocessing could be extended (e.g., with class classification [Str13]) in order to reduce the query time.

– We could also evaluate the performance of the Android version of Fudge, comparing the performance on mobile devices and desktop computers.

– Finally, the evaluation of the performance of Datil on mobile devices could be improved by considering more ontologies and more heterogeneous mobile devices.

Applications

– Regarding the identification of categories to profile people, we would like to consider other real-world domains using open data, so the evaluation is more reproducible. Furthermore, given the importance of sensor data, we could think of developing a mobile version of the system using Datil and without any remote computing.

– Regarding GimmeHop, the main direction for the future work is to take user feedback into account for future recommendations. So far, we take into account a global rating defined by the community, but it would also be interesting to take into account the user personal rating. Another possible extension is providing social recommendations, i.e., recommending products that a user with similar profiles liked. This would require a characterization of the user profile and letting the aggregation operator take into account the similarity between user profiles. As already discussed, it would also be desirable to build more complex representations of the possible values for some attributes of a beer, such as flavor, aroma, or foam. Furthermore, the individuals of the ontology could include values for the attributes price, color, or turbidity. In that case, we could use Datil to compute their associated linguistic labels and to include them in the recommendation process as well. Our definition of *context* could also be

extended to consider other factors, such as the temperature (e.g., Berliner Weiße is particularly appropriate for summer) or food to combine with (e.g., Guinness is a good choice to combine with a chocolate cake). Moreover, we could introduce fuzziness at other levels. For instance, we could assume that a beer belongs to a type with some degree, or that a beer type is a subtype of another one with some degree. In this case, data acquisition seems particularly challenging.

– Regarding our gait recognition system, there are many ideas for future work. Firstly, other Artificial Intelligence techniques could be applied to this problem to learn the biometric features. In particular, we think that neural networks and deep learning might be interesting: a preliminary step in this direction is described in [DALV+18]. Secondly, fuzzy logic theory provides a plethora of different operators that could be applied. For example, we could apply alternative defuzzification operators to compute the similarity between two features, or aggregation operators to compute the similarity between two steps. It seems particularly promising the use of alternative operators to compute the similarity between two sequences. When the number of steps is very high, as it happened in the dataset And1, a t-norm seems too strong, and other aggregation operators, such as a weighted mean or OWA, might be more suitable. Thirdly, it could be interesting to investigate the optimal number of sequences for each individual. If it is possible to keep the accuracy of the system, the smaller the training size, the faster the classification. Fourthly, we noticed when building our dataset that reflective clothing affects the quality of the recordings. We would like to understand the limits of the sensor by studying whether different footwear (e.g., high heels vs. flat shoes) or clothing (e.g. wide clothes vs. tight clothes) in the training and test data has an impact on the classification. We expect some measures to keep having similar values (such as the length of the bones), but the global effect is still unknown. Fifthly, human gait is not exactly symmetrical. So far, although our fuzzy ontology stores the moving leg of a step, we do not take it into account to compute the similarity between steps. We think that the two most similar steps should correspond to the same leg, but we have not checked that empirically. This could allow to reduce the number of comparisons by half, as only steps of the same leg should be considered. Last but not least, more sophisticated procedures to identify new individuals involving the degree of similarity between the steps might be interesting.

– Regarding our blockchain system, we need to evaluate our proposed architecture on a real scenario with multiple users and transactions in real time. The

evaluation could include the running time, the performance of the Eventheum events of the system, as well as a security and vulnerability analysis. In this regards, we could do a security analysis of smart contracts as suggested in [NSA+19]. Hopefully, the use of logical languages can also help to improve the security of blockchain systems.

– Regarding the prototype to query BIM models, we could improve the size of the fragments that can be supported, as the whole ontology is not currently supported by the classical reasoners and could not be evaluated. For example, a possible strategy would be using a preprocessing step to filter the ontology (or fragments) and reduce the sizes guided by some specific classes (used as an initial signature to be preserved). Another future work could be improving the implementation to automatically split the ontology into subontologies, or using more sophisticated parsers to translate the BIM model into OWL. Finally, it would be interesting to test a set of use cases with a high-level digital representation of a real building.

Conclusiones y trabajo futuro

En este capítulo, resumimos las principales conclusiones de esta tesis, dedicada a la gestión avanzada del conocimiento semántico difuso. Las conclusiones se agrupan en varias categorías: contribuciones al aprendizaje de ontologías difusas, razonamiento con ontologías difusas, soporte para dispositivos móviles y desarrollo de aplicaciones del mundo real. Finalmente, discutimos las principales direcciones a seguir en el trabajo futuro.

Aprendizaje de ontologías difusas

En esta sección, resumimos las principales conclusiones del aprendizaje de elementos de ontologías difusas, lo que demuestra que se ha cumplido el objetivo **O1**.

- Propusimos una estrategia para aprender tipos de datos difusos locales, específicos de un individuo. Usando los valores numéricos de las propiedades de datos, construimos una función de pertenencia triangular para cada propiedad de los datos usando el promedio y la desviación estándar. Este enfoque se ha validado en una aplicación de reconocimiento de la marcha, donde la imprecisión del dispositivo utilizado para capturar los datos se puede gestionar combinando los valores correspondientes a diferentes fotogramas de vídeo.

- Propusimos una estrategia para aprender tipos de datos difusos globales a partir de los valores numéricos de las propiedades de datos. Tras ejecutar un algoritmo de agrupamiento basado en centroides, utiliza los centroides como parámetros de algunas funciones de pertenencia difusas que dividen el dominio. Este enfoque se implementó en el sistema Datil, que admite diferentes formatos de entrada y salida, tres algoritmos de agrupamiento bien conocidos y puede extender ontologías clásicas al caso difuso, pero también enriquecer las ontologías difusas existentes. La herramienta calcula automáticamente nombres legibles para los tipos de datos difusos descubiertos. También hemos discutido cómo el aprendizaje de tipos de datos difusos se ha aplicado a tres aplicaciones del mundo real: perfiles

semánticos de estilo de vida, recomendación de cerveza y reconocimiento de la forma de andar.

  - En el perfil de estilo de vida, caracterizamos el estilo de vida de las personas a partir de sus rastros digitales: usamos datos numéricos obtenidos de diferentes sensores para construir tipos de datos difusos usando Datil, de modo que las definiciones de categorías usan términos lingüísticos que son fácilmente interpretables por usuarios humanos.
  - En el reconocimiento de la marcha, se utilizó Datil para calcular resúmenes lingüísticos de las características de los individuos en la base de conocimiento. Esto se puede utilizar para proporcionar explicaciones de las decisiones del sistema, promoviendo la Inteligencia Artificial Explicable.
  - En el dominio de la cerveza, Datil calculó la definición de tipos de datos difusos que representan, por ejemplo, un bajo grado de alcohol o un amargor muy alto. Un grupo de expertos evaluó la calidad de las etiquetas lingüísticas calculadas por Datil, demostrando que es similar a las definiciones dadas por humanos. Además, pudimos identificar la mejor estrategia de aprendizaje: usar el algoritmo de agrupamiento mean-shift.

- Resolvimos el problema de crear un solo tipo de dato difuso a partir de múltiples definiciones de tipos de datos difusos mediante el uso de operadores de agregación lingüística. Nuestra propuesta admite los tipos de datos difusos del lenguaje Fuzzy OWL 2 y está implementada en el sistema Fudge. Si bien los operadores de agregación se han utilizado previamente en ontologías difusas para combinar valores numéricos en $[0, 1]$, nos enfocamos en la agregación de funciones de pertenencia difusas. Nuestro enfoque puede utilizar operadores de agregación nuevos y existentes. En particular, hemos propuesto dos nuevos operadores, una forma recursiva por la izquierda de la combinación convexa (CONV–LRF) y del OWA lingüístico (LOWA–LRF). Un análisis empírico muestra que el porcentaje de coincidencias entre ambos operadores es inversamente proporcional al número de expertos, y que ambos operadores tienen un grado de *orness* similar. En cuanto a las estrategias de agregación existentes, hemos estudiado CONV–RRF, OWA lingüístico estándar (LOWA–RRF), media ponderada (WMEAN) y OWA difusa (FOWA). Discutimos las características más relevantes de todas estas estrategias de agregación, por ejemplo, la posibilidad de asignar un peso a un experto específico y la posibilidad de obtener como resultado un valor que no fue proporcionado por ningún experto. Además, mostramos cómo obtener los pesos

de cuantificadores difusos o un valor *orness* en problemas con datos incompletos, en los que no todos los expertos brindan una definición para cada tipo de datos difuso.

Razonamiento con ontologías difusas

Esta sección resume las principales conclusiones del razonamiento de la ontología difusa, demostrando que se ha cumplido el objetivo **O2** .

– Propusimos dos algoritmos específicos para resolver los problemas de recuperación de instancias y realización. Hasta donde sabemos, este es el primer trabajo que no repite una tarea de razonamiento (mejor grado de implicación lógica) para todos los individuos o conceptos de la ontología. Nuestro enfoque se puede implementar en una familia de algoritmos para razonar con DLs difusas que combinan un algoritmo tableaux y un problema de optimización, y se basa en fusionar los problemas de optimización en tres problemas de optimización de acuerdo con la cantidad de variables a optimizar: cero, una, o más de una. La clave es que los dos primeros problemas se pueden resolver solo una vez. Además, nuestra experiencia muestra que, en la práctica, este último problema a menudo está vacío; es decir, la recuperación de instancias a menudo conduce a problemas de optimización independientes que se pueden fusionar para resolverlos como un solo problema.

  Nuestro algoritmo de recuperación de instancias ha sido implementado en el razonador de ontologías difusas fuzzyDL y realizamos una evaluación empírica con varias ontologías difusas, algunas de ellas con un número importante de individuos. Nuestros experimentos confirman que nuestro nuevo algoritmo para calcular la recuperación de instancias supera a la implementación anterior en todos los casos que involucran ontologías consistentes, y la reducción del tiempo de razonamiento es más importante a medida que crece el número de individuos en la ontología. Además, en casi todos los casos, fue suficiente con resolver un solo problema de optimización. Sin embargo, en ontologías inconsistentes, el algoritmo básico encuentra la inconsistencia más rápido.

– Identificamos una tarea de razonamiento novedosa, la recuperación de instancias flexible y facetada, que amplía la recuperación de instancias difusas tradicional para reducir los resultados de la consulta al imponer algunas condiciones, descritas mediante tipos de datos difusos, en los valores de algunas propiedades de datos. También identificamos algunos casos que son comunes en la práctica

(posibles restricciones en los elementos de la ontología que son realmente difusos) y simplifican la solución del problema. En particular, propusimos dos algoritmos de razonamiento minimalistas cuya idea principal es reducir el problema a una inferencia crisp clásica, que puede ser resuelta por cualquier razonador semántico clásico, y luego realizar algunos cálculos adicionales manejando la parte difusa de las ontologías. Estos algoritmos han demostrado su validez en un sistema de recomendación de cerveza y en la consulta de ficheros de modelado de información de construcción.

– Proporcionamos un enfoque novedoso para calcular la similitud difusa entre individuos en una ontología difusa. En comparación con el único trabajo similar existente, nuestro enfoque es capaz de calcular similitudes locales (entre los valores de una propiedad de datos) cuando los valores se representan utilizando tipos de datos difusos en lugar de valores numéricos, y se basa en la defuzzificación de una intersección de conjuntos difusos. Además, nuestra similitud global (combinando similitudes locales) es muy general ya que admite cualquier operador de agregación. También discutimos el caso donde dos individuos no tienen las mismas propiedades. Este enfoque se evaluó con éxito en una aplicación de reconocimiento de la marcha.

– Proporcionamos un enfoque novedoso para calcular un emparejamiento difuso entre individuos en una ontología difusa. Formulamos nuestro problema como una tarea de razonamiento de ontología difusa, calculando el mejor grado de satisfacibilidad de una combinación de conceptos difusos que representan las restricciones de cada una de las partes. Ampliamos los enfoques anteriores considerando más de dos partes involucradas y permitiendo funciones más generales. También demostramos que la suma ponderada estricta y las t-normas estrictas conducen a soluciones que verifican la óptimalidad de Pareto. Usando una suma ponderada estricta, podemos controlar los acuerdos demasiado injustos al ponderar la importancia de las partes involucradas. También mostramos que todas las partes deben incluir restricctiones existenciales sobre las mismas propiedades para manejar correctamente la hipótesis de mundo abierto.

Soporte a dispositivos móviles

En esta sección, resumimos las principales conclusiones del soporte de ontologías difusas en dispositivos móviles, demostrando que se ha cumplido el objetivo **O3**.

– Para incrementar la plétora de herramientas que admiten ontologías difusas en

dispositivos móviles, desarrollamos dos versiones de Android de Datil y Fudge. Hasta donde sabemos, estas son las primeras aplicaciones que aprenden tipos de datos difusos en dispositivos móviles. Nuestra evaluación de Datil muestra que aunque el tiempo requerido para terminar el proceso de aprendizaje es mayor en un teléfono inteligente, es aceptable. Además, observamos que la diferencia con respecto a una computadora portátil depende de la elección del algoritmo de agrupamiento utilizado en el aprendizaje.

– Para aumentar la cantidad de aplicaciones basadas en ontologías difusas del mundo real, promover el razonamiento local y remoto en dispositivos móviles e ilustrar la utilidad de tener ontologías difusas en ficheros distribuidos, desarrollamos GimmeHop, un sistema de recomendación de cerveza para dispositivos Android. Es una prueba de concepto que muestra que la lógica difusa, las ontologías difusas, los razonadores semánticos y el razonamiento tanto local como remoto se pueden combinar en aplicaciones móviles. Nuestros experimentos sobre el tráfico de datos y el tiempo de ejecución muestran que el razonamiento remoto es factible y eficiente (tanto en términos de tráfico de datos como de tiempo). El razonamiento local sólo es factible si limitamos el número de individuos. Los dispositivos probados fueron capaces de soportar 2000–3000 cervezas, lo que en nuestra opinión sería suficiente en la práctica para la mayoría de bares o tiendas interesadas en recomendar cervezas a sus usuarios.

– Para promover el razonamiento híbrido en dispositivos móviles, desarrollamos una nueva versión del razonador de ontología difusa fuzzyDL para convertirlo en el primer razonador semántico serializable e incremental. fuzzyDL puede expandir una ontología difusa con algunas inferencias que se pueden reutilizar al responder diferentes consultas. Es posible serializar el objeto Java que representa el razonador y guardarlo en un archivo (serializable) y reutilizar esas inferencias sin reiniciar desde cero (incremental). Estas características son particularmente interesantes para dispositivos móviles, pero se pueden usar en cualquier dispositivo.

  Nuestros experimentos muestran que fuzzyDL calcula archivos serializados más pequeños que JFact, el otro razonador semántico que también es serializable. Además, fuzzyDL es más rápido tanto en la serialización como en la deserialización. Si bien ser incremental es útil para disminuir el tiempo de razonamiento, ser también serializable aumenta ligeramente el costo de la primera consulta porque es necesario restaurar la versión serializada del archivo. También hemos estimado el costo de descargar el archivo desde un servidor remoto antes

de restaurar el razonador y parece aceptable. Por lo tanto, la idea de reutilizar desde un dispositivo móvil una ontología difusa que se expandió previamente en un lugar diferente (por ejemplo, en un servidor dedicado) parece prometedora.

Aplicaciones reales

Esta sección resume las principales conclusiones del desarrollo de aplicaciones reales basadas en ontologías difusas, demostrando que se ha cumplido el objetivo **O4**.

Perfiles. Desarrollamos un sistema de definición de categorías para clasificar el perfil de estilo de vida de una persona. Para hacerlo, consideramos un gran volumen de datos heterogéneos obtenidos de sensores *wearable* (utilizados por un grupo de voluntarios de 40) y describimos una metodología para aprender axiomas de ontología difusa de subclases para definir categorías de personas. Esta metodología combina dos técnicas de aprendizaje de ontologías difusas, el algoritmo para aprender tipos de datos difusos implementado en Datil y el algoritmo para aprender axiomas de subclases implementado en Fuzzy DL-Learner. Tanto los datos como los resultados de la evaluación son propiedad de una empresa privada y son confidenciales, por lo que aquí solo se brindan algunos detalles.

Recomendación. Desarrollamos GimmeHop, un sistema de recomendación de cerveza para dispositivos móviles Android, utilizando ontologías difusas y razonadores semánticos. El dominio de aplicación, cervezas, es un tema importante que está recibiendo una atención notable en los últimos años. De hecho, dos empresas locales se interesaron por los resultados de nuestro proyecto. Usando una ontología difusa con más de 15000 cervezas, GimmeHop puede usarse para recuperar cervezas que satisfacen algunas características deseadas, pero también para recuperar cervezas similares a una dada. GimmeHop es capaz de tratar con el contexto del usuario (en particular, la ubicación) mediante el uso de modificadores difusos, con las preferencias del usuario mediante el uso de la media ponderada como agregación y con datos incompletos utilizando cuantificadores lingüísticos para calcular vectores de pesos con diferentes tamaños. Realizamos una evaluación exhaustiva de varias características del sistema: el tráfico de datos, el tiempo de ejecución, la calidad de las recomendaciones y la calidad de las etiquetas lingüísticas.

Reconocimiento de la marcha. Describimos un sistema de reconocimiento de la marcha basado en Microsoft Kinect y ontologías difusas. Esto tiene varias

aplicaciones posibles, como la seguridad y la medicina. Nuestro sistema es bastante rápido para calcular el reconocimiento, lo que lo hace adecuado para escenarios del mundo real. La principal característica de nuestro sistema es el uso de ontologías difusas, que tiene varios beneficios: son más robustas frente a pequeños cambios en los valores de las medidas biométricas a lo largo de diferentes pasos, permiten obtener información más detallada (asignando un grado a la clasificación de una persona) y resultados interpretables (gracias a los resúmenes lingüísticos), y todos los beneficios de las ontologías clásicas, como proporcionar razonamiento automatizado (por ejemplo, para verificar que no haya inconsistencias), conjuntos de datos más legibles o facilitar el mantenimiento y la integración de datos entre aplicaciones. Propusimos una arquitectura basada en una ontología de esquema y varias ontologías difusas instanciadas que incluyen características biométricas de individuos. Nuestro enfoque hace posible representar tanto los datos de Microsoft Kinect V1 y V2 como los datos de Nuitrack, y todas las características biométricas reportadas en la literatura, además de algunas nuevas (211 en total). También construimos un nuevo conjunto de datos con 91 individuos (llamado conjunto de datos Zar2), discutimos cómo construir la ontología difusa (enumerando varias técnicas de preprocesamiento para mejorar la calidad de los datos) y mostramos cómo usar la lógica difusa para obtener descripciones lingüísticas de las características biométricas.

Estas ontologías difusas se utilizaron para entrenar un nuevo algoritmo de reconocimiento de la marcha a partir de los datos de Kinect. El algoritmo se basa en la segmentación por pasos de las secuencias, el uso de lógica difusa para lidiar con la imprecisión del sensor y un esquema de votación para agregar los valores obtenidos para cada paso. Nuestro sistema proporciona una respuesta en poco tiempo después de que los datos se cargan en la memoria. Los resultados de nuestros experimentos muestran que nuestro nuevo método supera a los algoritmos existentes en el caso de personas que caminan en línea recta. También hemos evaluado diferentes operadores de defuzzificación, con algunas diferencias en la precisión según el conjunto de datos. Además, evaluamos nuestro algoritmo para grabaciones laterales utilizando una cámara en movimiento. En este caso, nuestro algoritmo suele funcionar peor y solo un método de defuzzificación (el promedio) es comparable. También hemos abordado el problema del reconocimiento de nuevos individuos no incluidos en nuestra base de conocimiento, mostrando resultados prometedores.

Blockchain. Propusimos un procedimiento novedoso para integrar ontologías difusas en sistemas blockchain. De esta manera, los usuarios pueden representar restricciones flexibles utilizando conjuntos difusos y es posible desarrollar contratos inteligentes donde existe un acuerdo parcial entre las partes involucradas. Por ejemplo, esto es útil para transacciones comerciales donde algunos parámetros no son estrictos sino flexibles, como el precio o el tiempo de entrega. Otra ventaja es que es posible evitar la ambigüedad de los lenguajes naturales, así como inferir conocimientos implícitos o verificar inconsistencias, ya que las partes involucradas utilizan un lenguaje formal para representar el conocimiento de un dominio de aplicación. En particular, propusimos una arquitectura basada en un esquema de ontología común, una ontología difusa personal para cada una de las partes involucradas y una ontología común que incluye los valores acordados del contrato inteligente. Nuestro enfoque se ha implementado en la red Ethereum, utilizando el razonador fuzzyDL para obtener los acuerdos parciales y la red IPFS P2P para almacenar la ontología común. Para calcular los acuerdos parciales, utilizamos una estrategia novedosa para calcular un emparejamiento difuso entre individuos verificando la optimalidad de Pareto.

Construcción. Propusimos una estrategia novedosa para razonar con ficheros de modelado de información de construcción (BIM) más grandes, más cercanos a los que se usan en aplicaciones del mundo real, basada en la recuperación de instancias flexible y facetada, y desarrollamos un prototipo. Consideramos un modelo BIM real como caso de estudio, que convertimos de IFC a OWL (sintaxis RDF) usando una herramienta existente. Mostramos que dos razonadores clásicos no podían soportar un modelo tan grande, por lo que la ontología final quedó fragmentada; por operatividad, restringimos las pruebas y evaluaciones solo al tercer piso del conjunto de datos. Además, con este submódulo construimos una ontología difusa actualizada con nuevos asertos de propiedades, asertos de conceptos y tipos de datos difusos.

Evaluamos el desempeño de nuestra propuesta midiendo los tiempos de recuperación de los valores de las propiedades de datos de cada individuo. Concluimos que el razonador TrOWL da resultados satisfactorios. También descubrimos que los tiempos de consulta se pueden reducir cuando se usan estructuras de datos adicionales (una tabla hash). Otro hallazgo es que nuestra herramienta requiere un tiempo inicial considerable para clasificar la ontología, pero las siguientes consultas requieren menos tiempo.

Trabajo futuro

Para concluir esta tesis, mencionemos algunas posibles direcciones para nuestra futura investigación.

Aprendizaje de ontologías difusas

- Planeamos evaluar Datil en más dominios del mundo real, ya que esta experiencia práctica seguramente proporcionará más ideas para ampliar nuestra herramienta. También sería interesante evaluar Datil teniendo en cuenta algunas medidas objetivas, como el coeficiente de partición (PC), la entropía de partición (PE) o el índice de Chen [ESAA+18]. También podríamos implementar algoritmos de agrupamiento más sofisticados como DBSCAN, que es más sólido ya que gestiona el ruido [EKSX96, SSE+17], o algoritmos incrementales [BKT20] para admitir una actualización dinámica de las definiciones.

- El diseño modular de Fudge hace que sea muy fácil incorporar más operadores difusos. Por lo tanto, planeamos agregar operadores de agregación lingüística más generales, por ejemplo, algunos que no verifican la internalidad o asumen un vector de pesos no numéricos [Xu12]. Entre los operadores lingüísticos más generales se hallan OWA ponderado lingüístico [Tor97] o los operadores aritméticos ponderados triangulares (basados en una t-norma y una t-conorma) [SG18]. Sería interesante soportar diferentes cuantificadores, o relaciones de orden alternativas entre funciones de pertenencia difusas. Por último, pero no menos importante, sería interesante estudiar otros escenarios en los que se permita a los expertos utilizar términos lingüísticos ya existentes, posiblemente diferentes a las etiquetas utilizadas por otros expertos. Con este objetivo, planeamos aprovechar el trabajo existente sobre la definición de asignaciones que preservan el razonamiento entre los términos lingüísticos locales [ACEG+94].

- A petición de algunos usuarios, planeamos codificar nuestras implementaciones de Datil y Fudge como plug-ins de Protégé para integrar mejor la herramienta en el proceso de desarrollo de ontologías difusas. De esta forma, sería posible crear directamente una ontología difusa, aprender algunos tipos de datos difusos y consultar la ontología desde Protégé. En este momento, estos pasos se pueden llevar a cabo, pero se necesita un paso intermedio que involucre a Datil o Fudge. Los plug-ins también puede ser bueno para recopilar datos del uso del usuario para llevar a cabo una validación experimental más extensa.

– Dado que el aprendizaje de tipos de datos es una técnica complementaria a otros enfoques para el aprendizaje de ontologías difusas, nuestras implementaciones de Datil y Fudge podrían usarse para extender Fuzzy DL-Learner, un sistema que aprende axiomas de subclases difusas.

– También nos gustaría estudiar cómo aprender las definiciones de las propiedades de datos no numéricos, como las fechas.

Razonamiento con ontologías difusas

– Con respecto a la recuperación de instancias flexible y facetada, sería interesante desarrollar algoritmos similares para otros conjuntos similares de restricciones en la ontología difusa.

– En cuanto al algoritmo de recuperación de instancias, podríamos evaluarlo con más ontologías difusas, ya sean reales o artificiales, con más variables dependientes. En tales casos, nos gustaría estudiar la mejor estrategia para resolver los problemas de optimización (es decir, fusionar todos los problemas en $\mathcal{O}_{two\_or\_more}$, resolverlos todos de forma independiente o usar un enfoque híbrido).

– También nos gustaría implementar (extendiendo fuzzyDL) y evaluar el algoritmo de realización.

– También sería interesante desarrollar algoritmos más específicos para otras tareas de razonamiento, como la clasificación.

– La similitud entre los individuos podría evaluarse en aplicaciones más reales, para tener una mejor comprensión del impacto de los operadores difusos alternativos.

– Con respecto al algoritmo de emparejamiento difuso, sería útil un grado de similitud difuso entre restricciones no numéricas.

Soporte a dispositivos móviles

– La tarea principal es desarrollar una versión de fuzzyDL que funcione en dispositivos móviles. Debido a que está implementado en Java, parece más fácil desarrollar una versión de Android. Hasta ahora, el único problema es que utiliza una librería de terceros (Gurobi) para la que actualmente no existe una versión de Android. Una posibilidad podría ser reemplazar Gurobi con otra librería que resuelva problemas de optimización matemática (en particular, problemas de programación lineal entera mixta) completamente desarrollada en

el fragmento de Java compatible con Android. Con esta versión de Android se podría investigar si en dispositivos móviles con recursos limitados el tiempo de expansión de una ontología difusa será mayor que el tiempo de deserialización con más frecuencia que en nuestra evaluación. Otra idea obvia es el desarrollo de una nueva versión de fuzzyDL que admita la OWL API, de modo que sea posible manejar información volátil como se propone en [BBMP17]. Afortunadamente, para hacer posible la comunicación entre el meta–razonador y fuzzyDL, podría ser posible implementar solo un fragmento muy pequeño de la OWL API. Por último, pero no menos importante, se podría mejorar el analizador fuzzyDL para cargar ontologías OWL 2 (encontramos algunos errores en las ontologías codificadas en el fragmento de OWL 2 que admite fuzzyDL) y se podría ampliar el preprocesamiento de fuzzyDL (por ejemplo, con clasificación de clases [Str13]) para reducir el tiempo de consulta.

- También evaluamos el rendimiento de la versión Android de Fudge, comparando el rendimiento en dispositivos móviles y computadoras de escritorio.

- Finalmente, la evaluación del desempeño de Datil en dispositivos móviles podría mejorarse considerando más ontologías y dispositivos móviles más heterogéneos.

Aplicaciones reales

- En cuanto a la identificación de categorías para el perfil de personas, nos gustaría considerar otros dominios del mundo real usando datos abiertos, para que la evaluación sea más reproducible. Además, dada la importancia de los datos de los sensores, podríamos pensar en desarrollar una versión móvil del sistema usando Datil y sin ninguna computación remota.

- Con respecto a GimmeHop, la dirección principal para el trabajo futuro es tener en cuenta los comentarios de los usuarios para futuras recomendaciones. Hasta ahora, tenemos en cuenta una calificación global definida por la comunidad, pero sería interesante tener en cuenta también la calificación personal del usuario. Otra posible extensión es brindar recomendaciones sociales, es decir, recomendar productos que le gusten a un usuario con perfiles similares. Esto requeriría una caracterización del perfil de usuario y dejar que el operador de agregación tenga en cuenta la similitud entre los perfiles de usuario. Como ya se discutió, también sería deseable construir representaciones más complejas de los posibles valores de algunos atributos de una cerveza, como sabor, aroma o espuma. Además, los individuos de la ontología podrían incluir valores para los atributos precio,

color o turbiedad. En ese caso, podríamos usar Datil para calcular sus etiquetas lingüísticas asociadas e incluirlas también en el proceso de recomendación. Nuestra definición de contexto también podría extenderse para considerar otros factores, como la temperatura (por ejemplo, Berliner Weiße es particularmente apropiada para el verano) o comida para maridar (por ejemplo, Guinness es una buena opción para combinar con una tarta de chocolate). Además, podríamos introducir ambigüedad a otros niveles. Por ejemplo, podríamos suponer que una cerveza pertenece a un tipo con algún grado, o que un tipo de cerveza es un subtipo de otro con algún grado. En este caso, la adquisición de datos parece particularmente desafiante.

– Con respecto a nuestro sistema de reconocimiento de la marcha, hay muchas ideas para trabajos futuros. En primer lugar, se podrían aplicar otras técnicas de Inteligencia Artificial a este problema para aprender las características biométricas. En particular, creemos que las redes neuronales y el aprendizaje profundo pueden ser interesantes: un paso preliminar en esta dirección se describe en [DALV+18]. En segundo lugar, la lógica difusa proporciona una plétora de diferentes operadores que podrían aplicarse. Por ejemplo, podríamos aplicar operadores de defuzzificación alternativos para calcular la similitud entre dos características u operadores de agregación para calcular la similitud entre dos pasos. Parece particularmente prometedor el uso de operadores alternativos para calcular la similitud entre dos secuencias. Cuando el número de pasos es muy alto, como sucedió en el conjunto de datos And1, una t-norma parece demasiado estricta y otros operadores de agregación, como una media ponderada u OWA, podrían ser más adecuados. En tercer lugar, podría ser interesante investigar el número óptimo de secuencias para cada individuo. Si es posible mantener la precisión del sistema, cuanto menor sea el tamaño del entrenamiento, más rápida será la clasificación. En cuarto lugar, notamos al construir nuestro conjunto de datos que la ropa reflectante afecta la calidad de las grabaciones. Nos gustaría comprender los límites del sensor al estudiar si el calzado diferente (ejemplo, tacones altos frente a zapatos planos) o ropa (ejemplo, ropa ancha frente a ropa ajustada) en los datos de entrenamiento y prueba tiene un impacto en la clasificación. Esperamos que algunas medidas sigan teniendo valores similares (como la longitud de los huesos), pero aún se desconoce el efecto global. En quinto lugar, la marcha humana no es exactamente simétrica. Hasta ahora, aunque nuestra ontología difusa almacena el pie de un paso, no lo tomamos en cuenta para calcular la similitud entre pasos. Pensamos que los dos pasos más

similares deberían corresponder al mismo pie, pero no lo hemos comprobado empíricamente. Esto podría permitir reducir el número de comparaciones a la mitad, ya que solo se deberían considerar los pasos del mismo pie. Por último, pero no menos importante, podrían ser interesantes procedimientos más sofisticados para identificar nuevos individuos que involucren el grado de similitud entre los pasos.

- Con respecto a nuestro sistema blockchain, necesitamos evaluar nuestra arquitectura propuesta en un escenario real con múltiples usuarios y transacciones en tiempo real. La evaluación podría incluir el tiempo de ejecución, el rendimiento de los eventos Eventheum del sistema, así como un análisis de seguridad y vulnerabilidad. En este sentido, podríamos hacer un análisis de seguridad de los contratos inteligentes como se sugiere en [NSA+19]. Esperemos que el uso de lenguajes lógicos también puede ayudar a mejorar la seguridad de los sistemas de cadena de bloques.

- En cuanto al prototipo para consultar modelos BIM, podríamos mejorar el tamaño de los fragmentos que se pueden soportar, ya que toda la ontología no está actualmente soportada por los razonadores clásicos y no se puede evaluar. Por ejemplo, una posible estrategia sería usar un preprocesamiento para filtrar la ontología (o fragmentos) y reducir los tamaños guiados por algunas clases específicas (usadas como signatura inicial a preservar). Otro trabajo futuro podría ser mejorar la implementación para dividir automáticamente la ontología en subontologías, o usar analizadores más sofisticados para traducir el modelo BIM a OWL. Finalmente, sería interesante probar un conjunto de casos de uso con una representación digital de alto nivel de un edificio real.

Bibliography

[AAdIC15] Gorka Azkune, Aitor Almeida, Diego López de Ipiña, and Liming Chen. Extending knowledge-driven activity models through data-driven learning techniques. *Expert Systems with Applications*, 42(6):3115–3128, 2015.

[AAJS14] Mohammed Ahmed, Naseer Al-Jawad, and Azhin T Sabir. Gait recognition based on Kinect sensor. In *Proceedings of SPIE Photonics Europe 2014*, volume 9139, page 91390B. SPIE, 2014.

[AB20] Jimmy Abualdenien and Andre Borrmann. Vagueness visualization in building models across different design stages. *Advanced Engineering Informatics*, 45:101107, 2020.

[ABB+13] Teresa Alsinet, David Barroso, Ramón Béjar, Félix Bou, Marco Cerami, and Francesc 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, 2013.

[ACEG+94] Jaume Agustí-Cullell, Francesc Esteva, Pere García, Lluís Godo, Ramon López de Mántaras, and Carles Sierra. Logical multi-valued logics in modular expert systems. *Journal of Experimental and Theoretical Artificial Intelligence*, 6(3):303–321, 1994.

[ACMM21] Jose María Alonso, Ciro Castiello, Luis Magdalena, and Corrado Mencar. *Explainable Fuzzy Systems: Paving the Way from Interpretable Fuzzy Systems to Explainable AI Systems*, volume 970 of *Studies in Computational Intelligence*, chapter Design and Validation of an Explainable Fuzzy Beer Style Classifier, pages 169–217. Springer, 2021.

[AD07] Muhammad Abulaish and Lipika Dey. A fuzzy ontology generation framework for handling uncertainties and nonuniformity in domain knowledge description. In *Proceedings of the 2007 International Conference on Computing: Theory and Applications (ICCTA 2007)*, pages 287–293. IEEE, 2007.

[ADG16] Eva Armengol, Pilar Dellunde, and Àngel García-Cerdaña. On similarity in fuzzy description logics. *Fuzzy Sets and Systems*, 292:49–74, 2016.

[ADP06] Rafal A. Angryk, Jacob Dolan, and Frederick E. Petry. Development of ontologies by the lowest common abstraction of terms using fuzzy hypernym chains. In *Soft Computing in Ontologies and Semantic Web*, volume 204 of *Studies in Fuzziness and Soft Computing*, pages 123–148. Springer, 2006.

[AdRMA15] Virginia O. Andersson and de Ricardo M. Araújo. Person identification using anthropometric and gait data from Kinect sensor. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015)*, pages 425–431. AAAI Press, 2015.

[AESB+21] Ebtsam Adel, Shaker El-Sappagh, Sherif Barakat, Jong-Wan Hu, and Mohammed Elmogy. An extended semantic interoperability model for distributed electronic health record based on fuzzy ontology semantics. *Electronics*, 10:1733, 2021.

[AIK+18] Farman Ali, S. M. Riazul Islam, Daehan Kwak, Pervez Khan, Niamat Ullah, Sangjo Yoo, and Kyung Sup Kwak. Type-2 fuzzy ontology-aided recommendation systems for iot-based healthcare. *Computer Communications*, 119:138–155, 2018.

[AK09] Safdar Ali and Stephan Kiefer. μ-OR – A micro OWL DL reasoner for ambient intelligent devices. In *Proceedings of the 4th International Conference on Grid and Pervasive Computing (GPC 2009)*, volume 5529 of *Lecture Notes in Computer Science*, pages 305–316. Springer, 2009.

[AKK15] Farman Ali, Eun Kyoung Kim, and Yong-Gi Kim. Type-2 fuzzy ontology-based opinion mining and information extraction: A proposal to automate the hotel reservation system. *Applied Intelligence*, 42(3):481–500, 2015.

[AKR+17] Farman Ali, Pervez Khan, Kashif Riaz, Daehan Kwak, Tamer AbuHmed, Daeyoung Park, and Kyung Sup Kwak. A fuzzy ontology and svm-based web content classification system. *IEEE Access*, 5:25781–25797, 2017.

[Ale17] Fernando Alegre. Desarrollo de un sistema recomendador de cervezas basado en ontologías y lógica difusa. Undergraduate thesis project, University of Zaragoza, 2017.

[AMCJ18] Borrmann Andre, König Markus, Koch Christian, and Beetz Jakob. Building information modeling: Why? what? how? In *Building Information Modeling: Technology Foundations and Industry Practice*, pages 1–24. Springer International Publishing, 2018.

[ARCGHJR13] Ana Armas-Romero, Bernardo Cuenca-Grau, Ian Horrocks, and Ernesto Jiménez-Ruiz. MORe: a modular owl reasoner for ontology classification. In *Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013)*, volume 1015, pages 61–67. CEUR Workshop Proceedings, 2013.

[AS21] Esther Aguado and Ricardo Sanz. *Using Ontologies in Autonomous Robots Engineering*, volume 219 of *Studies in Computational Intelligence*. IntechOpen, 2021.

[AWKA12] Panos Alexopoulos, Manolis Wallace, Konstantinos Kafentzis, and Dimitris Askounis. IKARUS-Onto: A methodology to develop fuzzy ontologies from crisp ones. *Knowledge and Information Systems*, 32:667–695, 2012.

[AY09] Louis Atallah and Guang-Zhong Yang. The use of pervasive sensing for behaviour profiling — A survey. *Pervasive and Mobile Computing*, 5(5):447–464, 2009.

[AZ21] Houda Akremi and Sami Zghal. DOF: a generic approach of domain ontology fuzzification. *Frontiers of Computer Science*, 15:153322, 2021.

[BBG07] Afef Bahri, Rafik Bouazi, and Faiez Gargouri. Dealing with similarity relations in fuzzy ontologies. In *Proceedings of the 16th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007)*, pages 1–6. IEEE, 2007.

[BBIM14] Carlos Bobed, Fernando Bobillo, Sergio Ilarri, and Eduardo Mena. Answering continuous description logic queries: Managing static and volatile knowledge in ontologies. *International Journal on Semantic Web and Information Systems*, 10(3):1–44, 2014.

[BBMP17] Carlos Bobed, Fernando Bobillo, Eduardo Mena, and Jeff Z. Pan. On serializable incremental semantic reasoners. In *Proceedings of the 9th International Conference on Knowledge Capture (K-CAP 2017)*, pages 187–190. ACM, 2017.

[BCE+15] Fernando Bobillo, Marco Cerami, Francesc Esteva, Àngel García-Cerdaña, Rafael Peñaloza, and Umberto Straccia. Fuzzy description logics. In Petr Cintula, Christian Fermüller, and Carles Noguera, editors, *Handbook of Mathematical Fuzzy Logic Volume III*, volume 58 of *Studies in Logic, Mathematical Logic and Foundations*, chapter XVI, pages 1105–1181. College Publications, 2015.

[BCFGR12] Fernando Bobillo, Miguel Calvo-Flores, and Juan Gómez-Romero. DeLorean: A reasoner for fuzzy OWL 2. *Expert Systems with Applications*, 39:258–272, 2012.

[BCP17] Stefan Borgwardt, Marco Cerami, and Rafael Peñaloza. The complexity of fuzzy $\mathcal{EL}$ under the Ł ukasiewicz t-norm. *International Journal of Approximate Reasoning*, 91:179–201, 2017.

[BDB17] Fernando Bobillo, Lacramioara Dranca, and Jorge Bernad. A fuzzy ontology-based system for gait recognition using Kinect sensor. In *Proceedings of the 11th International Conference on Scalable Uncertainty Management (SUM 2017)*, volume 10564 of *Lecture Notes in Computer Science*, pages 397–404. Springer, 2017.

[BDGR09] Fernando Bobillo, Miguel Delgado, and Juan Gómez-Romero. Crisp representations and reasoning for fuzzy ontologies. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 17(4):501–530, 2009.

[BDGRS12] Fernando Bobillo, Miguel Delgado, Juan Gómez-Romero, and Umberto Straccia. Joining Gödel and Zadeh fuzzy logics in fuzzy description logics. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 20(04):475–508, 2012.

[BDRD+20] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[Bez81] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, 1981.

[BG17] Franz Baader and Oliver Fernandez Gil. Decidability and complexity of threshold description logics induced by concept similarity measures. In Ahmed Seffah, Birgit Penzenstadler, Carina Alves, and Xin Peng, editors, *Proceedings of the Symposium on Applied Computing (SAC 2017)*, pages 983–988. ACM, 2017.

[BHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284:34–43, 2001.

[BHLS17] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017.

[BHP05] Nikolaos V Boulgouris, Dimitrios Hatzinakos, and Konstantinos N Plataniotis. Gait recognition: a challenging signal processing technology for biometric identification. *IEEE signal processing magazine*, 22(6):78–90, 2005.

[BHS07] Franz Baaderl, Ian Horrocks, and Ulrike Sattlerl. Description logics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation*, pages 135–179. Elsevier, 2007.

[BKT20] Adil M. Bagirov, Napsu Karmitsa, and Sona Taheri. *Partitional Clustering via Nonsmooth Optimization*, chapter Incremental Clustering Algorithms, pages 185–200. Unsupervised and Semi-Supervised Learning. Springer, 2020.

[BLS06] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. CEL - A polynomial-time reasoner for life science ontologies. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning*

(IJCAR 2006), volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer, 2006.

[BMM12] Bernadette Bouchon-Meunier and Gilles Moyse. Fuzzy linguistic summaries: Where are we, where can we go? In *Proceedings of the 2012 IEEE Conference on Computational Intelligence for Financial Engineering and Economics (CIFEr 2012)*, pages 317–324. IEEE, 2012.

[BMSL22] Alexandre Bento, Lionel Médini, Kamal Singh, and Frédérique Laforest. Do Arduinos dream of efficient reasoners? In *Proceedings of the 19th Extended Semantic Web Conference (ESWC 2022)*, 2022.

[Bob08] Fernando Bobillo. *Managing Vagueness in Ontologies*. PhD thesis, University of Granada, Spain, 2008.

[Bob16] Fernando Bobillo. The role of crisp elements in fuzzy ontologies: The case of fuzzy OWL 2 EL. *IEEE Transactions on Fuzzy Systems*, 24:1193–1209, 2016.

[BOP+18] Mathias Bonduel, Jyrki Oraskari, Pieter Pauwels, Maarten Vergauwen, and Ralf Klein. The IFC to linked building data converter: current status. In *Proceedings of the 6th Linked Data in Architecture and Construction Workshop*, volume 2159 of *CEUR Workshop Proceedings*, pages 34–43. CEUR-WS.org, 2018.

[Bor97] Willem Nico Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, University of Twente, 1997.

[Boy19] Peter Boyland. The state of mobile network experience - Benchmarking mobile on the eve of the 5G revolution. `http://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2019-05/the_state_of_mobile_experience_may_2019_0.pdf`, 2019. Visited on January 2020.

[BPKP21] Marc Bravin, Daniel Pfäffli, Kevin Kuhn, and Marc Pouly. Towards crafting beer with Artificial Intelligence. In *Proceedings of the 8th Swiss Conference on Data Science (SDS 2021)*, pages 54–55, 2021.

[BS09] Fernando Bobillo and Umberto Straccia. Fuzzy description logics with general t-norms and datatypes. *Fuzzy Sets Systems*, 160(23):3382–3402, 2009.

[BS11] Fernando Bobillo and Umberto Straccia. Fuzzy ontology representation using OWL 2. *International Journal of Approximate Reasoning*, 52(7):1073–1094, 2011.

[BS12] Fernando Bobillo and Umberto Straccia. Generalized fuzzy rough description logics. *Information Sciences*, 189:43–62, 2012.

[BS13] Fernando Bobillo and Umberto Straccia. Aggregation operators for fuzzy ontologies. *Applied Soft Computing*, 13(9):3816–3830, 2013.

[BS14] Fernando Bobillo and Umberto Straccia. A MILP-based decision procedure for the (fuzzy) description logic $\mathcal{ALCB}$. In *Proceedings of the 27th International Workshop on Description Logics (DL 2014)*, volume 1193, pages 378–390. CEUR Workshop Proceedings, 2014.

[BS15] Fernando Bobillo and Umberto Straccia. On partitioning-based optimisations in expressive fuzzy description logics. In *Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2015.

[BS16a] Fernando Bobillo and Umberto Straccia. The fuzzy ontology reasoner fuzzyDL. *Knowledge-Based Systems*, 95:12–34, 2016.

[BS16b] Fernando Bobillo and Umberto Straccia. Optimising fuzzy description logic reasoners with general concept inclusions absorption. *Fuzzy Sets and Systems*, 292:98–129, 2016.

[BS17] Fernando Bobillo and Umberto Straccia. Generalizing type-2 fuzzy ontologies and type-2 fuzzy description logics. *International Journal of Approximate Reasoning*, 87:40–66, 2017.

[BS18] Fernando Bobillo and Umberto Straccia. Reasoning within fuzzy OWL 2 EL revisited. *Fuzzy Sets and Systems*, 351:1–40, 2018.

[BYBM15] Carlos Bobed, Roberto Yus, Fernando Bobillo, and Eduardo Mena. Semantic reasoning on mobile devices: Do Androids dream of efficient reasoners? *Journal of Web Semantics*, 35(4):167–183, 2015.

[CBM12] Christer Carlsson, Matteo Brunelli, and József Mezei. Decision making with a fuzzy ontology. *Soft Computing*, 16(7):1143–1152, 2012.

[CC03] Shi-Jay Chen and Shyi-Ming Chen. A new method for handling multicriteria fuzzy decision-making problems using FN-IOWA operators. *Cybernetics and Systems*, 34(2):109–137, 2003.

[CC16] David Chandran and Keeley Crockett. Fuzzy ontologies in semantic similarity measures. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4942–4949. IEEE, 2016.

[CCF17] Giovanna Castellano, Ciro Castiello, and Anna Maria Fanelli. The FISDeT software: Application to beer style classification. In *Proceedings of the 26th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2017)*, pages 1–6, 2017.

[CCP18] European Commission, Joint Research Centre, and Martin Poljanšek. *Building Information Modelling (BIM) standardization*. Publications Office, 2018. `https://data.europa.eu/doi/10.2760/36471`.

[CDB86] Robert Cannon, Jitendra Dave, and James C. Bezdek. Efficient implementation of the fuzzy c-means clustering algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8:248 – 255, 1986.

[CGHM+08] Bernardo Cuenca-Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics*, 6(4):309–322, 2008.

[CH67] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[Che95] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[CLDW21] Zehong Cao, Chin-Teng Lin, Yong Deng, and Gerhard-Wilhelm Weber. Guest editorial: Fuzzy systems toward human-explainable artificial intelligence and their applications. *IEEE Transactions on Fuzzy Systems*, 29(12):3577–3578, 2021.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions Pattern Analysis Machine Intelligence*, 24(5):603–619, 2002.

[CRS+18] Olivia Choudhury, Nolan Rudolph, Issa Sylla, Noor Fairoza, and Amar Das. Auto-generation of smart contracts from domain-specific ontologies and semantic rules. In *Proceedings of the 2018 IEEE Cybermatics for Cyber-enabled Hyperworld (Cybermatics 2018)*, pages 963–970. IEEE, 2018.

[CS21] Franco Alberto Cardillo and Umberto Straccia. Fuzzy OWL-Boost: Learning fuzzy concept inclusions via real-valued boosting. *Fuzzy Sets and Systems*, 2021.

[CSM14] Pratik Chattopadhyay, Shamik Sural, and Jayanta Mukherjee. Frontal gait recognition from incomplete sequences using RGB-D camera. *IEEE Transactions Information Forensics and Security*, 9(11):1843–1856, 2014.

[CWBM21] Roberto Confalonieri, Tillman Weyde, Tarek R. Besold, and Fermín Moscoso del Prado Martín. Using ontologies to enhance human understandability of global post-hoc explanations of black-box models. *Artificial Intelligence*, 296:103471, 2021.

[CYZW09] Wei Chen, Qing Yang, Li Zhu, and Bin Wen. Research on automatic fuzzy ontology generation from fuzzy context. In *Proceedings of the 2nd International Conference on Intelligent Computation Technology and Automation (ICICTA 2009)*, pages 764–767, 2009.

[DALV+18] Lacramioara Dranca, Álvaro Lozano, Rubén Vigara, Jorge Bernad, Ignacio Huitzil, and Fernando Bobillo. Técnicas de Inteligencia Artificial aplicadas al reconocimiento a través de la marcha. In *Proceedings of the VI Congreso Nacional de I+D en Defensa y Seguridad (DESEi+d 2018)*, 2018.

[Dav12] Andrew Davison. *Kinect Open Source Programming Secrets: Hacking the Kinect with OpenNI, NITE, and Java*. McGraw-Hill, 2012.

[DB14] Simon Daum and André Borrmann. Processing of topological BIM queries using boundary representation based methods. *Advanced Engineering Informatics*, 28(4):272–286, 2014.

[Dey01] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5:4–7, 2001.

[DGRMPP05] Miguel Delgado, Juan Gómez-Romero, Pedro Javier Magaña, and Ramón Pérez-Pérez. A flexible architecture for distributed knowledge based systems with nomadic access through handheld devices. *Expert Systems with Applications*, 29(4):965–975, 2005.

[DHHVM98] Miguel Delgado, Francisco Herrera, Enrique Herrera-Viedma, and Luis Martínez. Combining numerical and linguistic information in group decision making. *Journal of Information Sciences*, 107:177–194, 1998.

[DLG+18] Lacramioara Dranca, Urko López-de-Abetxuko, Alfredo Goñi, Arantza Illarramendi, Irene Navalpotro-Gómez, Manuel Delgado-Alvarado, and María Cruz Rodríguez-Oroz. Using Kinect to classify Parkinson's disease stages related to severity of gait impairment. *BMC Bioinformatics*, 19:471, 2018.

[DMS15] Tommaso Di Noia, Marina Mongiello, and Umberto Straccia. Fuzzy description logics for component selection in software design. In *Software Engineering and Formal Methods: SEFM 2015 Collocated Workshops, Revised Selected Papers*, volume 9509 of *Lecture Notes in Computer Science*, pages 228–239. Springer, 2015.

[DRCC+14] Natalia Díaz-Rodríguez, Olmo León Cadahía, Manuel P. Cuéllar, Johan Lilius, and Miguel Delgado. Handling real-world context awareness, uncertainty and vagueness in real-time human activity tracking and recognition with a fuzzy ontology-based hybrid method. *Sensors*, 14(10):18131–18171, 2014.

[DRCLD14] Natalia Díaz-Rodríguez, Manuel P. Cuéllar, Johan Lilius, and Miguel Delgado. A survey on ontologies for human behavior recognition. *ACM Computing Surveys*, 46(4):1–33, 2014.

[DRPCLD14] Natalia Díaz-Rodríguez, Manuel Pegalajar-Cuéllar, Johan Lilius, and Miguel Delgado. A fuzzy ontology for semantic modelling and recognition of human behaviour. *Knowledge-Based Systems*, 66(1):46–60, 2014.

[DRWL+13] Natalia Díaz-Rodríguez, Robin Wikström, Johan Lilius, Manuel Pegalajar-Cuéllar, and Miguel Delgado. Understanding movement and interaction: An ontology for Kinect-based 3D depth sensors. In *Proceedings of the 7th International Conference on Ubiquitous*

Computing and Ambient Intelligence (UCAmI 2013), volume 8276 of *Lecture Notes in Computer Science*, pages 254–261. Springer, 2013.

[DVV93] Miguel Delgado, José Luis Verdegay, and María Amparo Vila. On aggregation operations of linguistic labels. *International Journal of Intelligent Systems*, 8(3):351–370, 1993.

[DW87] W. M. Dong and F. S. Wong. Fuzzy weighted averages and implementation of the extension principle. *Fuzzy Sets and Systems*, 21(2):183–199, 1987.

[EHK+14] Markus Eich, Ronny Hartanto, Sebastian Kasperski, Sankaranarayanan Natarajan, and Johannes Wollenberg. Towards coordinated multirobot missions for lunar sample collection in an unknown environment. *Journal of Field Robotics*, 31(1):35–74, 2014.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD 1996)*, pages 226—231. AAAI Press, 1996.

[EP09] Nathan Eagle and Alex S. Pentland. Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, 2009.

[ESAA+18] Shaker El-Sappagh, José M. Alonso, Farman Ali, Amjad Ali, Jun-Hyeog Jang, and Kyung-Sup Kwak. An ontology-based interpretable fuzzy decision support system for diabetes diagnosis. *IEEE Access*, 6:37371–37394, 2018.

[Eur19] European Construction Sector Observatory. Building Information Modelling in the EU construction sector. Technical report, European Commission, 2019. `https://ec.europa.eu/docsroom/documents/34518`.

[FBF18] Muhammad Fahad, N. Bus, and B. Fies. Semantic bim reasoner for the verification of ifc models. In *eWork and eBusiness in Architecture, Engineering and Construction*, pages 361–368. CRC Press, 2018.

[FH51] Evelyn Fix and J.L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. Technical Report 4, USAF

School of Aviation Medicine, Randolph Field, Texas, 1951. https://apps.dtic.mil/dtic/tr/fulltext/u2/a800276.pdf.

[Fue19] Álvaro Fuentemilla. Desarrollo de smart contracts en una blockchain basados en información semántica. Undergraduate thesis project, University of Zaragoza, 2019.

[GBJR+13] Rafael S. Gonçalves, Samantha Bail, Ernesto Jiménez-Ruiz, Nicolas Matentzoglu, Bijan Parsia, Birte Glimm, and Yevgeny Kazakov. OWL Reasoner Evaluation (ORE) workshop 2013 results: Short report. In *Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013)*, volume 1015, pages 1–18. CEUR Workshop Proceedings, 2013.

[GC04] Serge Guillaume and Brigitte Charnomordic. Generating an interpretable family of fuzzy partitions from data. *IEEE Transactions on Fuzzy Systems*, 12(3):324–335, 2004.

[GG95] Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In *Towards very Large Knowledge bases: Knowledge Building and Knowledge sharing*, pages 25–32. IOS Press, 1995.

[GH07] Stephan Grimm and Pascal Hitzler. Semantic matchmaking of web resources with local closed-world reasoning. *International Journal of Electronic Commerce*, 12(2):89–126, 2007.

[GHM+14] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. HermiT: An OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.

[GI13] Andrey V. Grigorev and Alexander G. Ivashko. TReasoner: System description. In *Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013)*, volume 1015, pages 26–31. CEUR Workshop Proceedings, 2013.

[GIM+18] Guido Governatori, Florian Idelberger, Zoran Milosevic, Regis Riveret, Giovanni Sartor, and Xiwei Xu. On legal contracts, imperative and declarative smart contracts, and blockchain systems. *Artificial Intelligence and Law*, 26(4):377–409, 2018.

[GLGS07] Huamao Gu, Hexin Lv, Ji Gao, and Jinqin Shi. Towards a general fuzzy ontology and its construction. In *Proceedings of the International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2007)*, Advances in Intelligent Systems Research, 2007.

[GR08] Juan Gómez-Romero. *Knowledge Mobilization: Architectures, Models and Applications*. PhD thesis, University of Granada, Spain, 2008.

[GRBR+15] Juan Gómez-Romero, Fernando Bobillo, María Ros, Miguel Molina-Solana, María Dolores Ruiz, and María José Martín-Bautista. A fuzzy extension of the semantic Building Information Model. *Automation in Construction*, 57:202–212, 2015.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[GZ22] Abolfazl Ghorbani and Kamran Zamanifar. Type-2 fuzzy ontology-based semantic knowledge for indoor air quality assessment. *Applied Soft Computing*, page 108658, 2022.

[H́98] Petr Hájek. *The Metamathematics of Fuzzy Logic*. Kluwer, 1998.

[Hab07] Hashim 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.

[HB11] Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011.

[HBB20] Ignacio Huitzil, Jorge Bernad, and Fernando Bobillo. Algorithms for instance retrieval and realization in fuzzy ontologies. *Mathematics*, 8(2):154:1–16, 2020.

[HBGRS20] Ignacio Huitzil, Fernando Bobillo, Juan Gómez-Romero, and Umberto Straccia. Fudge: Fuzzy ontology building with consensuated fuzzy datatypes. *Fuzzy Sets and Systems*, 401:91–112, 2020.

[HC76] Harry M. Hersh and Alfonso Caramazza. A fuzzy set approach to modifiers and vagueness in natural language. *Journal of Experimental Psychology: General*, 105:254–276, 1976.

[HDG+06] Matthew Horridge, Nick Drummond, John Goodwin, Alan L. Rector, Robert Stevens, and Hai Wang. The Manchester OWL Syntax. In *Proceedings of the Workshop on OWL: Experiences and Directions (OWLED 2006)*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[Hel13] Kim Hellbom. Mobile web apps as an alternative to native mobile apps: The future of mobile web apps on the competitive marketplace. Arcada University of Applied Sciences (Swedish: Yrkeshögskolan Arcada), Degree thesis, 2013.

[HG17] Aki Härmä and Koen Groot. Automatic characterization of ambulatory patterns of utilitarian and leisure trips. In *Proceedings of the 25th European Signal Processing Conference (EUSIPCO 2017)*, pages 1897–1901, 2017.

[HHMW12] Volker Haarslev, Kay Hidde, Ralf Möller, and Michael Wessel. The RacerPro knowledge representation and reasoning system. *Semantic Web Journal*, 3(3), 2012.

[HHVV96] Francisco Herrera, Enrique Herrera-Viedma, and José Luis Verdegay. Direct approach processes in group decision making using linguistic OWA operators. *Fuzzy Sets and Systems*, 79(2):175–190, 1996.

[HPS07] Volker Haarslev, Hsueh-Ieng Pai, and Nematollaah 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.

[HS13] Steve Harris and Andy Seaborne. SPARQL 1.1 query language. `http://www.w3.org/TR/sparql11-query`, 2013.

[HSDRB18] Ignacio Huitzil, Umberto Straccia, Natalia Díaz-Rodríguez, and Fernando Bobillo. 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), Part II*, volume 854 of *Communications in Computer and Information Science*, pages 100–112. Springer, 2018.

[HT15] Nam V. Hoang and Seppo Törmä. Implementation and experiments with an ifc-to-linked data converter. In *Proceedings of the 32nd CIB W78 Conference*, pages 285–294, 2015.

[HWDC09] Maja Hadzic, Pornpit Wongthongtham, Tharam Dillon, and Elizabeth Chang. *Ontology-Based Multi-Agent Systems*, volume 219 of *Studies in Computational Intelligence*. Kluwer Academic Publishers, 2009.

[IL11] Josue Iglesias and Jens Lehmann. Towards integrating fuzzy logic capabilities into an ontology-based inductive logic programming framework. *International Conference on Intelligent Systems Design and Applications, ISDA*, pages 1323–1328, 2011.

[JSJ21] Shivani Jain, K.R. Seeja, and Rajni Jindal. A fuzzy ontology framework in information retrieval using semantic query expansion. *International Journal of Information Management Data Insights*, 1(1):100009, 2021.

[JWZS15] Shuming Jiang, Yufei Wang, Yuanyuan Zhang, and Jiande Sun. Real time gait recognition system based on Kinect skeleton feature. In *Proceedings of the ACCV 2014 Workshop on Human Gait and Action Analysis in the Wild: Challenges and Applications*, volume 9008 of *Lecture Notes in Computer Science*, pages 46–57. Springer, 2015.

[KA07] Stasinos Konstantopoulos and Georgios Apostolikas. Fuzzy-DL reasoning over unknown fuzzy degrees. In *Proceedings of the 3rd International Workshop on Semantic Web and Web Semantics (SWWS 2007), Part II*, volume 4806 of *Lecture Notes in Computer Science*, pages 1312–1318. Springer, 2007.

[Kaz09] Yevgeny Kazakov. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In *Proceedings of the 21st International Joint Conference on Artificial intelligence (IJCAI 2009)*, pages 2040–2045, 2009.

[KDS+08] Michal Koziuk, Jaroslaw Domaszewicz, Radoslaw Olgierd Schoeneich, Marcin Jablonowski, and Piotr Boetzel. Mobile context-addressable messaging with DL-Lite domain model. In *Proceedings of the 3rd European Conference on Smart Sensing and Context (EuroSSC 2008)*, volume 5279 of *Lecture Notes in Computer Science*, pages 168–181. Springer, 2008.

[KKS14] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. The incredible ELK. *Journal of Automated Reasoning*, 53:1–61, 2014.

[KL18] Henry M. Kim and Marek Laskowski. Toward an ontology-driven blockchain design for supply-chain provenance. *Intelligent Systems in Accounting, Finance and Management*, 25(1):18–27, 2018.

[KLN18] Henry M. Kim, Marek Laskowski, and Ning Nan. A first step in the co-evolution of blockchain and ontologies: Towards engineering an ontology of governance at the blockchain protocol level. `https://arxiv.org/abs/1801.02027`, 2018.

[KMP00] Erich-Peter Klement, Radko Mesiar, and Endre Pap. *Triangular Norms*, volume 8 of *Trends in Logic*. Springer, 2000.

[KNJ18] Nirattaya Khamsemanan, Cholwich Nattee, and Nitchan Jianwattanapaisarn. Human identification from freestyle walks using posture-based gait feature. *IEEE Transactions on Information Forensics and Security*, 13(1):119–128, 2018.

[KPHL10] Taehun Kim, Insuk Park, Soon J. Hyun, and Dongman Lee. MiRE4OWL: Mobile rule engine for OWL. In *Proceedings of the 2nd IEEE International Workshop on Middleware Engineering (ME 2010)*, pages 317–322. IEEE, 2010.

[KTEF16] Dimitris Kastaniotis, Ilias Theodorakopoulos, George Economou, and Spiros Fotopoulos. Gait based recognition via fusing information from euclidean and riemannian manifolds. *Pattern Recognition Letters*, 84:245 – 251, 2016.

[KTT+15] Dimitris Kastaniotis, Ilias Theodorakopoulos, Christos Theoharatos, George Economou, and Spiros Fotopoulos. A framework for gait-based recognition using Kinect. *Pattern Recognition Letters*, 68(Part 2):327–335, 2015.

[KV14] Avinash J. Kamble and T. Venkatesh. Some results on fuzzy numbers. *Annals of Pure and Applied Mathematics*, 7(2):90–97, 2014.

[KY95] George J. Klir and Bo Yuan. *Fuzzy sets and fuzzy logic - theory and applications*. Prentice Hall, 1995.

[LAHS05] Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research*, 23:667–726, 2005.

[LB10] Michael J. Lawley and Cyrill Bousquet. Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In *Proceedings of the Australasian Ontology Workshop 2010 (AOW 2010)*, pages 45–50, 2010.

[LIJ+15] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[LJH05] Chang-Shing Lee, Zhi-Wei Jian, and Lin-Kai Huang. A fuzzy ontology and its application to news summarization. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 35(5):859–880, 2005.

[LK99] Werner V. Leekwijck and Etienne E. Kerre. Defuzzification: criteria and classification. *Fuzzy Sets System*, 108(2):159–178, 1999.

[LKR07] Niels Landwehr, Kristian Kersting, and Luc De Raedt. Integrating naïve Bayes and FOIL. *Journal of Machine Learning Research*, 8:481–507, 2007.

[LL04] Puyin Liu and Hongxing Li. *Fuzzy Neural Network Theory and Application*. World Scientific, 2004.

[LLNE16] Mounaim Latif, Younes Lakhrissi, El Habib Nfaoui, and Najia Es-Sbai. Cross platform approach for mobile application development:A survey. In *Proceedings of the 2016 International Conference on Information Technology for Organizations Development (IT4OD)*, pages 1–5, 2016.

[Llo82] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LM18] Francesca A. Lisi and Corrado Mencar. A granular computing method for OWL ontologies. *Fundamenta Informaticae*, 159(1-2):147–174, 2018.

[LMR16] Xin Li, José-Fernán Martínez, and Gregorio Rubio. A new fuzzy ontology development methodology (FODM) proposal. *IEEE Access*, 4:7111–7124, 2016.

[LS08] Thomas Lukasiewicz and Umberto Straccia. Managing uncertainty and vagueness in description logics for the Semantic Web. *Journal of Web Semantics*, 6(4):291–308, 2008.

[LS13] Francesca A. Lisi and Umberto Straccia. A logic-based computational method for the automated induction of fuzzy ontology axioms. *Fundamenta Informaticae*, 124(4):503–519, 2013.

[LS15] Francesca A. Lisi and Umberto Straccia. Learning in description logics with fuzzy concrete domains. *Fundamenta Informaticae*, 140(3-4):373–391, 2015.

[LWH10] Chang-Shing 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.

[MBHG17] Tim A. Majchrzak, Andreas Biørn-Hansen, and Tor-Morten Grønli. Comprehensive analysis of innovative cross-platform app development frameworks. In *Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS 2017)*, pages 6162–6171, 2017.

[McA07] Darryl McAdams. An ontology for smart contracts, 2007.

[MCvdHST12] Carmen Martínez-Cruz, Albert van der Heide, Daniel Sánchez, and Gracian Triviño. An approximation to the computational theory of perceptions using ontologies. *Expert Systems with Applications*, 39(10):9494–9503, 2012.

[MdFRN18] Tarcisio Mendes de Farias, Ana Roxin, and Christophe Nicolle. A rule-based methodology to extract building model views. *Automation in Construction*, 92:214–229, 2018.

[Men12] Julian Mendez. jcel: A modular rule-based reasoner. In *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE 2012)*, volume 858 of *CEUR Workshop Proceedings*, pages 1–6, 2012.

[MHK12] Boris Motik, Ian Horrocks, and Su Myeon Kim. Delta-reasoner: A Semantic Web reasoner for an intelligent mobile platform. In

Proceedings of the 21st World Wide Web Conference (WWW 2012), Companion Volume, pages 63–72, 2012.

[MHLN06] Felix Müller, Michael Hanselmann, Thorsten Liebig, and Olaf Noppens. A tableaux-based mobile DL reasoner - An experience report. In *Proceedings of the 19th International Workshop on Description Logics (DL 2006)*, volume 189 of *CEUR Workshop Proceedings*, pages 1–2, 2006.

[MIGL17] Björn Müller, Winfried Ilg, Martin A Giese, and Nicolas Ludolph. Validation of enhanced Kinect sensor based motion capturing for gait assessment. *PloS one*, 12(4):e0175813, 2017.

[MKC+17] Juan Antonio Morente-Molinera, Gang Kou, Rubén González Crespo, Juan M. Corchado, and Enrique Herrera-Viedma. Solving multi-criteria group decision making problems under environments with a high number of alternatives using fuzzy ontologies and multi-granular linguistic modelling methods. *Knowledge-Based Systems*, 137:54–64, 2017.

[MKP+19] Juan Antonio Morente-Molinera, Gang Kou, C. Pang, Francisco Javier Cabrerizo, and Enrique Herrera-Viedma. An automatic procedure to create fuzzy ontologies from users' opinions using sentiment analysis procedures and multi-granular fuzzy linguistic modelling methods. *Information Sciences*, 476:222–238, 2019.

[MLY08] Zongmin M. Ma, Yan-Hui Lv, and Li Yan. A fuzzy ontology generation framework from fuzzy relational database. *International Journal on Semantic Web and Information Systems*, 4(3):1–15, 2008.

[MMWHVC16] Juan Antonio Morente-Molinera, Robin Wikström, Enrique Herrera-Viedma, and Christer Carlsson. A linguistic mobile decision support system based on fuzzy ontology to facilitate knowledge mobilization. *Decision Support Systems*, 81:66–75, 2016.

[MPUH15] Juan Antonio Morente-Molinera, Ignacio J. Pérez, M. Raquel Ureña, and Enrique Herrera-Viedma. Building and managing fuzzy ontologies with heterogeneous linguistic information. *Knowledge-Based Systems*, 88:154–164, 2015.

[MPUH16] Juan Antonio Morente-Molinera, Ignacio J. Pérez, M. Raquel Ureña, and Enrique Herrera-Viedma. Creating knowledge databases for storing and sharing people knowledge automatically using group decision making and fuzzy ontologies. *Information Sciences*, 328:418–434, 2016.

[MSS+12] Theofilos P. Mailis, Giorgos Stoilos, Nikos Simou, Giorgos B. Stamou, and Stefanos D. Kollias. Tractable reasoning with vague knowledge using fuzzy $\mathcal{EL}^{++}$. *Journal of Intelligent Information Systems*, 39(2):399–440, 2012.

[Mus15] Mark A. Musen. The Protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015.

[MZYC11] Zong Min Ma, Fu Zhang, Li Yan, and Jingwei Cheng. Representing and reasoning on fuzzy UML models: A description logic approach. *Expert Systems with Applications*, 38(3):2536–2549, 2011.

[NM01] Natalya T. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001. `http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html`.

[NMNS19] Tommaso Di Noia, Marina Mongiello, Francesco Nocera, and Umberto Straccia. A fuzzy ontology-based approach for tool-supported decision making in architectural design. *Knowledge and Information Systems*, 58(1):83–112, 2019.

[NSA+19] Nishara Nizamuddin, Khaled Salah, Muhammad Ajmal Azad, Junaid Arshad, and Muhammad Habib Ur Rehman. Decentralized document version control using Ethereum blockchain and IPFS. *Computers & Electrical Engineering*, 76:183–197, 2019.

[OBT04] Aris M. Ouksel, Yair M. Babad, and Thomas Tesch. Matchmaking software agents in B2B markets. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS 2004)*, pages 1–9, 2004.

[OEA21] Olaide N. Oyelade, Absalom E. Ezugwu, and Sunday A. Adewuyi. Enhancing reasoning through reduction of vagueness using fuzzy OWL–2 for representation of breast cancer ontologies. *Neural Computing and Applications*, pages 1–26, 2021.

[Org99] World Health Organization. Healthy living : what is a healthy lifestyle? Technical report, Regional Office for Europe, Copenhagen : WHO Regional Office for Europe, 1999. `https://apps.who.int/iris/handle/10665/108180`.

[OVB19] Mehdi Ousmer, Jean Vanderdonckt, and Sabin Buraga. An ontology for reasoning on body-based gestures. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 1–6, New York, NY, USA, 2019. Association for Computing Machinery.

[PAG06] Jorge Pérez, Marcelo Arenas, and Claudio Gutiérrez. The semantics and complexity of SPARQL. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, volume 4273 of *Lecture Notes in Computer Science*, pages 30–43. Springer, 2006.

[PB96] Frederick E. Petry and Patrick Bosc. *Fuzzy Databases: Principles and Applications*. Kluwer Academic Publishers, 1996.

[Ped97] Witold Pedrycz. *Fuzzy Evolutionary Computation*. Kluwer Academic Publishers, 1997.

[PKWLP12] Johannes Preis, Moritz Kessel, Martin Werner, and Claudia Linnhoff-Popien. Gait recognition with Kinect. In *Proceedings of the 1st International Workshop on Kinect in Pervasive Computing*, pages P1–P4, 2012.

[PMM+18] Deepak Puthal, Nisha Malik, Saraju P. Mohanty, Elias Kougianos, and Gautam Das. Everything you wanted to know about the blockchain: Its promise, components, processes, and problems. *IEEE Consumer Electronics Magazine*, 7(4):6–14, 2018.

[PS05] Witold Pedrycz and Zenon A. Sosnowski. C-fuzzy decision trees. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):498–511, 2005.

[PSS+08] Jeff Z. Pan, Giorgos Stamou, Giorgos Stoilos, Edward Thomas, and Stuart Taylor. Scalable querying service over fuzzy ontologies. In *Proceedings of the 17th International Conference on World Wide Web (WWW 2008)*, pages 575–584, 2008.

[PT16] Pieter Pauwels and Walter Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016.

[PWM+13] Ignacio J. Pérez, Robin Wikström, József Mezei, Christer Carlsson, and Enrique Herrera-Viedma. A new consensus model for group decision making using fuzzy ontology. *Soft Computing*, 17(9):1617–1627, 2013.

[PWS+21] Evan W. Patton, William Van Woensel, Oshani Seneviratne, Giuseppe Loseto, Floriano Scioscia, and Lalana Kagal. The Punya platform: Building mobile research apps with linked data and semantic features. In *Proceedings of the 20th International Semantic Web Conference (ISWC 2021)*, volume 12922 of *Lecture Notes in Computer Science*, pages 563–579. Springer, 2021.

[PYJF14] Primal Pappachan, Roberto Yus, Anupam Joshi, and Tim Finin. Rafiki: A semantic and collaborative approach to community health-care in underserved areas. In *Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2014)*, pages 322–331. IEEE, 2014.

[PZL17] Pieter Pauwels, Sijie Zhang, and Yong-Cheol Lee. Semantic Web technologies in AEC industry: A literature overview. *Automation in Construction*, 73:145–165, 2017.

[QHF06] Thanh Tho Quan, Siu Cheung Hui, and Alvis Cheuk M. Fong. Automatic fuzzy ontology generation for semantic help-desk support. *IEEE Transactions on Industrial Informatics*, 2(3):155–164, 2006.

[QHFC06] Thanh Tho Quan, Siu Cheung Hui, Alvis Cheuk M. Fong, and Tru Hoang Cao. Automatic fuzzy ontology generation for Semantic Web. *IEEE Transactions on Knowledge and Data Engineering*, 18(6):842–856, 2006.

[RFIB] Ishak Riali, Messaouda Fareh, Mohamed Chakib Ibnaissa, and Mounir Bellil. A semantic-based approach for hepatitis C virus prediction and diagnosis using a fuzzy ontology and a fuzzy Bayesian network. *Journal of Intelligent & Fuzzy Systems*, pages 1–15.

[Rod13] James A. Rodger. A fuzzy linguistic ontology payoff method for aerospace real options valuation. *Expert Systems with Applications*, 40(8), 2013.

[Ros10] Timothy J. Ross. *Fuzzy Logic with Engineering Applications, 3rd edition*. John Wiley & Sons, Ltd, 2010.

[RSB+08] Azzurra Ragone, Umberto Straccia, Fernando Bobillo, Tommaso Di Noia, and Eugenio Di Sciascio. Fuzzy bilateral matchmaking in e-Marketplaces. In *Proceedings of the 12th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Part III (KES 2008)*, volume 5179 of *Lecture Notes in Computer Science*, pages 293–301. Springer, 2008.

[RSB+22] Michele Ruta, Floriano Scioscia, Ivano Bilenchi, Filippo Gramegna, Giuseppe Loseto, Saverio Ieva, and Agnese Pinto. A multiplatform reasoning engine for the semantic web of everything. *Journal of Web Semantics*, page 100709, 2022.

[RSdS+14] Michele Ruta, Floriano Scioscia, Maria di Summa, Saverio Ieva, Eugenio Di Sciascio, and Marco Sacco. Semantic matchmaking for Kinect-based posture and gesture recognition. *International Journal of Semantic Computing*, 8(4):491–514, 2014.

[RSG+19] Michele Ruta, Floriano Scioscia, Filippo Gramegna, Ivano Bilenchi, and Eugenio Di Sciascio. Mini-ME Swift: the first OWl reasoner for iOS. In *Proceedings of the 16th Extended Semantic Web Conference (ESWC 2019)*, Lecture Notes in Computer Science, pages 298–313. Springer, 2019.

[RSI+17] Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, and Eugenio Di Sciascio. Semantic blockchain to improve scalability in the internet of things. *Open Journal of Internet of Things*, 3(1):46–61, 2017.

[RSP+19] Michele Ruta, Floriano Scioscia, Agnese Pinto, Filippo Gramegna, Saverio Ieva, Giuseppe Loseto, and Eugenio Di Sciascio. CoAP-based collaborative sensor networks in the semantic web of things. *Journal of Ambient Intelligence and Humanized Computing*, 10(7):2545–2562, 2019.

[RSS10] Michele Ruta, Floriano Scioscia, and Eugenio Di Sciascio. Mobile semantic-based matchmaking: A fuzzy DL approach. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010), Part I*, volume 6088 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2010.

[SBF98] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: principles and methods. *Data and knowledge engineering*, 25(1):161–198, 1998.

[SDY22] Mojtaba Shokohinia, Abbas Dideban, and Farzin Yaghmaee. A method for improving reasoning and realization problem solving in descriptive logic- based and ontology-based reasoners. *Malaysian Journal of Computer Science*, 35:37–55, 2022.

[Sem20] Semantic Integration Ltd. HyperGraphQL. `https://www.hypergraphql.org`, 2020.

[SG18] Ulrich Florian Simo and Henri Gwét. Fuzzy triangular aggregation operators. *International Journal of Mathematics and Mathematical Sciences*, 2018, Article ID 9209524, 13 pages, 2018.

[SK05] Alex Sinner and Thomas Kleemann. KRHyper - In your pocket. In *Proceedings of the 20th International Conference on Automated Deduction (CADE-20)*, volume 3632 of *Lecture Notes in Computer Science*, pages 452–458. Springer, 2005.

[SKG09] Luke Steller, Shonali Krishnaswamy, and Mohamed Medhat Gaber. Enabling scalable semantic reasoning for mobile services. *International Journal on Semantic Web and Information Systems*, 5(2):91–116, 2009.

[SLG14] Andreas Steigmiller, Thorsten Liebig, and Birte Glimm. Konclude: System description. *Journal of Web Semantics*, 27-28:78–85, 2014.

[SM15] Umberto Straccia and Matteo Mucci. PFOIL-DL: Learning (fuzzy) $\mathcal{EL}$ concept descriptions from crisp OWL data using a probabilistic ensemble estimation. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC 2015)*, pages 345—-352. ACM, 2015.

[SMSS10] Nikos Simou, Theofilos P. Mailis, Giorgos Stoilos, and Giorgos B. Stamou. Optimization techniques for fuzzy description logics. In *Proceedings of the 23rd International Workshop on Description Logics (DL 2010)*, volume 573 of *CEUR Workshop Proceedings*, pages 244–254. CEUR-WS.org, 2010.

[SMSW20] Ranjith K. Soman, Miguel Molina-Solana, and Jennifer Whyte. Linked-Data based Constraint-Checking (LDCC) to support look-ahead planning in construction. *Automation in Construction*, 120:103369, 2020.

[Smu12] Pavel Smutný. Mobile development tools and cross-platform solutions. In *Proceedings of the 13th International Carpathian Control Conference (ICCC 2012)*, pages 653–656, 2012.

[SOGP16] Antonio A. Sánchez-Ruiz, Santiago Ontañón, Pedro A. González-Calero, and Enric Plaza. Measuring similarity of individuals in description logics over the refinement space of conjunctive queries. *Journal of Intelligent Information Systems*, 47(3):447–467, 2016.

[SPC+07] Evren Sirin, Bijan Parsia, Bernardo Cuenca-Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, 2007.

[SPL17] Grégory Smit, Olivier Pivert, and Marie-Jeanne Lesot. Vocabulary elicitation for informative descriptions of classes. In *Proceedings of the 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS 2017)*, pages 1–8, 2017.

[SPS09] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer, 2009.

[SRES+21] Nora Shoaip, Amira Rezk, Shaker El-Sappagh, Louai Alarabi, Sherif Barakat, and Mohammed M. Elmogy. A comprehensive fuzzy ontology-based decision support system for Alzheimer's disease diagnosis. *IEEE Access*, 9:31350–31372, 2021.

[SRI20] Anil Sawhney, Mike Riley, and Javie Irizarry. *Construction 4.0: An Innovation Platform for the Built Environment*. Routledge, 2020.

[SRL+14] Floriano Scioscia, Michele Ruta, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. A mobile matchmaker for the ubiquitous semantic web. *International Journal on Semantic Web and Information Systems*, 10(4):77–100, 2014.

[SS11] Christian Seitz and René Schönfelder. Rule-based OWL reasoning for specific embedded devices. In *Proceedings of the 10th International Semantic Web Conference (ISWC 2011), Part II*, volume 7032 of *Lecture Notes in Computer Science*, pages 237–252. Springer, 2011.

[SSE+17] Erich Schubert, Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems*, 42(3):1–21, 2017.

[SSG08] Luciano Stefanini, Laerte Sorini, and Maria Letizia Guerra. Fuzzy numbers and fuzzy arithmetic. In *Handbook of Granular Computing*. Wiley, 2008.

[SSSK06] Giorgos Stoilos, Nikolaos Simou, Giorgos Stamou, and Stefanos Kollias. Uncertainty and the Semantic Web. *IEEE Intelligent Systems*, 21(5):84–87, 2006.

[SST22] Fabio Sartori, Marco Savi, and Jacopo Talpini. Tailoring mHealth apps on users to support behavior change interventions: Conceptual and computational considerations. *Applied Sciences*, 12(8), 2022.

[ST22] Fabio Sartori and Lidia Lucrezia Tonelli. Fuzzy personalization of mobile apps: A case study from mhealth domain. In *Proceedings of the 29th International Conference on Information Systems Development (ISD 2011)*, volume 55 of *Lecture Notes in Information Systems and Organisation*, pages 91–108. Springer, 2022.

[Str05] Umberto Straccia. Description logics with fuzzy concrete domains. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*, pages 1–9. AUAI Press, 2005.

[Str13] Umberto Straccia. *Foundations of Fuzzy Logic and Semantic Web Languages*. CRC Studies in Informatics Series. Chapman & Hall, 2013.

[SVM02] Dragan Z. Saletic, Dusan M. Velasevic, and Nikos E. Mastorakis. Analysis of basic defuzzification techniques. In N. Mastorakis and V.Mladenov, editors, *Recent Advances in Computers, Computing and Communications*, pages 247–252. WSEAS Press, 2002.

[SVS15] Giorgos Stoilos, Tassos Venetis, and Giorgos Stamou. A fuzzy extension to the OWL 2 RL ontology language. *The Computer Journal*, 58(11):2956–2971, 2015.

[SYR13] Archana Singh, Avantika Yadav, and Ajay Rana. K-means with three different distance metrics. *International Journal of Computer Applications*, 67(10):13–17, 2013.

[TD17] Allan Third and John Domingue. Linked data indexing of distributed ledgers. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW 2017 Companion)*, pages 1431–1436, 2017.

[TDKM14] Dorothea Tsatsou, Stamatia Dasiopoulou, Ioannis Kompatsiaris, and Vasileios Mezaris. LiFR: A lightweight fuzzy DL reasoner. In *Proceedings of the 11th Extended Semantic Web Conference (ESWC 2014), Posters and Demo sessions*, volume 8798 of *Lecture Notes in Computer Science*, pages 263–267. Springer, 2014.

[TH06] Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: system description. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, pages 292–297, 2006.

[TH10] Chia-Chi Teng and Richard Helps. Mobile application development: Essential new directions for it. In *Proceedings of the 7th International Conference on Information Technology: New Generations (ITNG 2010)*, pages 471–475, 2010.

[THP19] Seppo Törmä, Nam V. Hoang, and Pieter Pauwels. IFC-to-RDF conversion tool. http://www.rymreport.com/pre/result/opening-bim-to-the-web-ifc-to-rdf-conversion-software, 2019.

[TKO15] Wei Tai, John Keeney, and Declan O'Sullivan. Resource-constrained reasoning using a reasoner composition approach. *Semantic Web*, 6:35–59, 2015.

[Tor97] Vicenç Torra. The weighted OWA operator. *International Journal of Intelligent Systems*, 12(2):153–166, 1997.

[TPR10] Edward Thomas, Jeff Pan, and Yuan Ren. TrOWL: tractable OWL 2 reasoning infrastructure. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010), Part II*, volume 6089 of *Lecture Notes in Computer Science*, pages 431–435. Springer, 2010.

[Tur99] Berwin A. Turlach. Bandwidth selection in kernel density estimation: A review. *CORE and Institut de Statistique*, pages 1–33, 1999.

[TVV18] Ruben Taelman, Miel Vander Sande, and Ruben Verborgh. GraphQL-LD: Linked Data querying with GraphQL. In *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks*, volume 2180 of *CEUR Workshop Proceedings*, pages 1–4. CEUR-WS.org, 2018.

[TY05] Luigi Troiano and Ronald R. Yager. Recursive and iterative OWA operators. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 13(6):579–600, 2005.

[Uga17] Héctor E. Ugarte R. A more pragmatic web 3.0: Linked blockchain data. Technical report, Rheinische Friedrich-Wilhelms-Universität Bonn, 2017. http://doi.org/10.13140/RG.2.2.10304.12807/1.

[Voj07] Peter Vojtáš. $\mathcal{EL}$ description logic with aggregation of user preference concepts. In Marie Duží, Hannu Jaakkola, Yasushi Kiyoki, and Hannu Kangassalo, editors, *Information modelling and Knowledge Bases*, volume XVIII, pages 154–165. IOS Press, 2007.

[W3C04a] W3C. RDF Primer. http://www.w3.org/TR/rdf-primer, 2004.

[W3C04b] W3C OWL Working Group. OWL Web Ontology Language overview. https://www.w3.org/TR/owl-features/, 2004.

[W3C04c] W3C OWL Working Group. OWL Web Ontology Language semantics and abstract syntax. https://www.w3.org/TR/owl-semantics/, 2004.

[W3C09a] W3C. OWL 2: Web Ontology Language Manchester syntax. http://www.w3.org/TR/owl2-manchester-syntax, 2009.

[W3C09b] W3C. OWL 2 Web Ontology Language profiles: OWL 2 EL. http://www.w3.org/TR/2009/REC-owl2-profiles-20091027/#OWL_2_EL, 2009.

[W3C12a] W3C OWL Working Group. OWL 2 Overview. http://www.w3.org/TR/owl2-overview/, 2012.

[W3C12b] W3C OWL Working Group. OWL 2 Web Ontology Language profiles. https://www.w3.org/TR/owl2-profiles/, 2012.

[W3C14] W3C. RDF 1.1 Turtle. http://www.w3.org/TR/turtle, 2014.

[WA18] William Van Woensel and Syed Sibte Raza Abidi. Optimizing semantic reasoning on memory-constrained platforms using the RETE algorithm. In *Proceedings of the 15th Extended Semantic Web Conference (ESWC 2018)*, volume 10843 of *Lecture Notes in Computer Science*, pages 682–696. Springer, 2018.

[WHC+16] Arlette Wissen, Aki Härmä, Illapha G. Cuba, Dietwig Lowet, and Rim Helaoui. Optimization of automated health programs by simulating user behaviors and program effects. In *Proceedings of the Conference on Measuring Behavior (CMB 2016)*, pages 1–5, 2016.

[WMY09] Hailong Wang, Zongmin M. Ma, and Junfu 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, 2009.

[Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, https://github.com/ethereum/yellowpaper, 2014.

[WSB+19] Jeroen Werbrouck, Madhumitha Senthilvel, Jakob Beetz, Pierre Bourreau, and Léon Van Berlo. Semantic query languages for

knowledge-based web services in a construction context. In *Proceedings of the 26th International Workshop on Intelligent Computing in Engineering (EG-ICE 2019)*, volume 2394 of *CEUR Workshop Proceedings*, pages 1–13. CEUR-WS.org, 2019.

[WY01] Dwi H. Widyantoro and John Yen. Using fuzzy ontology for query refinement in a personalized abstract search engine. In *Proceedings of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference (IFSA-NAFIPS 2001)*, volume 4, pages 610–615, 2001.

[XS08] Liu Xinwang and Han Shilian. Orness and parameterized RIM quantifier aggregation with OWA operators: A summary. *International Journal of Approximate Reasoning*, 48:77–97, 2008.

[XT15] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165–193, 2015.

[Xu08] Zeshui Xu. Linguistic Aggregation Operators: An Overview. In *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models*, volume 220 of *Studies in Fuzziness and Soft Computing*, pages 163–181. Springer, 2008.

[Xu12] Zeshui Xu. *Linguistic Decision Making: Theory and Methods*. Springer, 2012.

[Yag88] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.

[Yag89] Ronald R. Yager. On linguistic summaries of data. In *Proceedings of IJCAI Workshop on Knowledge Discovery in Databases*, pages 379–389, 1989.

[Yag91] Ronald R. Yager. Connectives and quantifiers in fuzzy sets. *Fuzzy Sets and Systems*, 40(1):39–75, 1991.

[Yag96] Ronald R. Yager. Quantifier guided aggregation using OWA operators. *International Journal of Intelligent Systems*, 11(1):49–73, 1996.

[Yag21] Ronald R. Yager. An introduction to linguistic summaries. In *Fuzzy Approaches for Soft Computing and Approximate Reasoning:*

Theories and Applications, volume 394 of *Studies in Fuzziness and Soft Computing*, pages 151–162. Springer, 2021.

[YDL+16] Ke Yang, Yong Dou, Shaohe Lv, Fei Zhang, and Qi Lv. Relative distance features for gait recognition with Kinect. *Journal of Visual Communication and Image Representation*, 39:209–217, 2016.

[YDNG18] Aleksandr Yurievich Yurin, Nikita Olegovich Dorodnykh, Olga A. Nikolaychuk, and Maksim Andreevich Grishenko. Designing rule-based expert systems with the aid of the model-driven development approach. *Expert Systems*, 35(5), 2018.

[YMII14] Roberto Yus, Eduardo Mena, Sergio Ilarri, and Arantza Illarramendi. SHERLOCK: Semantic management of location-based services in wireless environments. *Pervasive and Mobile Computing*, 15:87–99, 2014.

[YP15] Roberto Yus and Primal Pappachan. Are apps going semantic? A systematic review of semantic mobile applications. In *Proceedings of the 1st International Workshop on Mobile Deployment of Semantic Technologies (MoDeST 2015)*, volume 1506 of *CEUR Workshop Proceedings*, pages 2–13. CEUR-WS.org, 2015.

[YPD+14] Roberto Yus, Primal Pappachan, Prajit Kumar Das, Eduardo Mena, Anupam Joshi, and Tim Finin. FaceBlock: Privacy-aware pictures for Google Glass. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys 2014)*, pages 366–366, 2014.

[YRbJb09] Wang Ying, Zhang Ru-bo, and Lai Ji-bao. Measuring concept similarity between fuzzy ontologies. In *Fuzzy Information and Engineering Volume 2*, volume 62 of *Advances in Intelligent and Soft Computing*, pages 163–171. Springer, 2009.

[Zad65] Lotfi A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[Zad75] Lotfi A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning I. *Information Sciences*, 8:199–249, 1975.

[ZMY13] Fu Zhang, Zong Min Ma, and Li Yan. Construction of fuzzy ontologies from fuzzy XML models. *Knowledge-Based Systems*, 42:20–39, 2013.

[ZMYC13] Fu Zhang, Zong Min Ma, Li Yan, and Jingwei Cheng. Construction of fuzzy OWL ontologies from fuzzy EER models: A semantics-preserving approach. *Fuzzy Sets and Systems*, 229:1–32, 2013.

[ZMYW12] Fu Zhang, Zong Min Ma, Li Yan, and Yu Wang. A description logic approach for representing and reasoning on fuzzy object-oriented database models. *Fuzzy Sets and Systems*, 186(1):1–25, 2012.

Appendix A

Publications

The results of this thesis have been published in 5 journal articles and 6 peer-reviewed conference/workshops papers (4 of them in international conferences, 1 in an international workshop, and 1 in a national conference).

Journal papers

For each journal paper, we show some relevant characteristics: impact factor and quartile (according to Journal Citation Reports), the subject category to which it belongs to, and the number of citations (according to Google Scholar).

Journal article 1. I. Huitzil, L. Dranca, J. Bernad, F. Bobillo. "Gait Recognition Using Fuzzy Ontologies and Kinect Sensor Data". International Journal of Approximate Reasoning 113:354-371, August 2019, doi:10.1016/j.ijar.2019.07.012.

- Impact Factor 2019: 2.678 (Q2)
- Category: Computer Science, Artificial Intelligence
- Total cites (Accessed on 15/06/2022): 17 (13 external)

Journal article 2. I. Huitzil, J. Bernad, F. Bobillo, "Algorithms for Instance Retrieval and Realization in Fuzzy Ontologies". Mathematics 8(2), 154:1-16, January 2020, doi:10.3390/math8020154.

- Impact Factor 2020: 2.258 (Q1)
- Category: Mathematics
- Total cites (Accessed on 15/06/2022): 4 (2 external)

Journal article 3. I. Huitzil, F. Alegre, F. Bobillo. "GimmeHop: A Recommender System for Mobile Devices using Ontology Reasoners and Fuzzy Logic". Fuzzy Sets and Systems 401:55-77, December 2020, doi:10.1016/j.fss.2019.12.001.

- Impact Factor 2020: 3.343 (Q1)
- Category: Computer Science, Theory and Methods
- Total cites (Accessed on 15/06/2022): 11 (6 external)

Journal article 4. I. Huitzil, F. Bobillo, J. Gómez-Romero, U. Straccia. "Fudge: Fuzzy Ontology Building with Consensuated Fuzzy Datatypes". Fuzzy Sets and Systems 401:91-112, December 2020, doi:10.1016/j.fss.2020.04.001.

- Impact Factor 2020: 3.343 (Q1)
- Category: Computer Science, Theory and Methods
- Total cites (Accessed on 15/06/2022): 6 (3 external)

Journal article 5. I. Huitzil, M. Molina-Solana, J.Gómez-Romero, F. Bobillo, "Minimalistic fuzzy ontology reasoning: An application to Building Information Modeling", Applied Soft Computing 103:107158, February 2021, doi:10.1016/j.asoc.2021.107158.

- Impact Factor 2020: 6.725 (Q1)
- Category: Computer Science, Artificial Intelligence
- Total cites (Accessed on 15/06/2022): 8 (7 external)

Peer-reviewed conference papers

For each conference, we show some relevant characteristics: scope (international or national), CORE ranking, GGS ranking, and the number of citations (according to Google Scholar).

Conference paper 1. N. Díaz-Rodríguez, A. Härmä, R. Helaoui, **I. Huitzil**, F. Bobillo, U. Straccia, "Couch Potato or Gym Addict? Semantic Lifestyle Profiling with Wearables and Knowledge Graphs", Proceedings of the 6th Workshop on Automated Knowledge Base Construction (AKBC 2017), December 2017.

- Scope: International
- Total cites (Accessed on 15/06/2022): 5 (4 external)

Conference paper 2. I. Huitzil, U. Straccia, N. Díaz-Rodríguez, F. Bobillo, "Datil: Learning Fuzzy Ontology Datatypes", Proceedings of the 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2018), Part II, Springer, volume 854, pp. 100-112, Communications in Computer and Information Science, June 2018, doi:10.1007/978-3-319-91476-3_9.

- Scope: International
- CORE 2020: C
- Total cites (Accessed on 15/06/2022): 14 (5 external)

Conference paper 3. I. Huitzil, A. Fuentemilla, F. Bobillo, "I Can Get Some Satisfaction: Fuzzy Ontologies for Partial Agreements in Blockchain Smart Contracts", Proceedings of the 29th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2020), IEEE Press, ISBN 978-1-7281-6932-3, July 2020, doi:10.1109/FUZZ48607.2020.9177732.

- Scope: International
- CORE 2020: B
- GGS 2020: 3
- Total cites (Accessed on 15/06/2022): 1

Conference paper 4. I. Huitzil, U. Straccia, C. Bobed, E. Mena, F. Bobillo, "The Serializable and Incremental Semantic Reasoner fuzzyDL", Proceedings of the 29th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2020), IEEE Press, ISBN 978-1-7281-6932-3, July 2020, doi:10.1109/FUZZ48607.2020.9177835.

- Scope: International
- CORE 2020: B
- GGS 2020: 3
- Total cites (Accessed on 15/06/2022): 2 (1 external)

Conference paper 5. I. Huitzil, "Advanced Management of Fuzzy Semantic Information", 1st Doctoral Consortium at the European Conference on Artificial Intelligence DC-ECAI 2020, (Finalist in the DC-ECAI 2020 Best Presentation Award), August 2020.

- Scope: International

Conference paper 6. I. Huitzil, F. Alegre and F. Bobillo, "CAEPIA-APP Competition: GimmeHop", Actas de la XIX Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA 20-21), pp. 989-992, September 2021.

- Scope: National