

POLITECNICO DI MILANO
Dipartimento di Elettronica e Informazione
Ph.D. Program in Information Technology - XXV Cycle



DESCRIPTION LOGICS AND
SEMANTIC QUERY LANGUAGES IN
ROBOTIC APPLICATIONS

Doctoral dissertation of:
Nicola Vitucci

Advisor:
Prof. Giuseppina Gini

Tutor:
Prof. Andrea Bonarini

Doctoral program supervisor:
Prof. Carlo Fiorini

XXV cycle

*Random actions may inspire
new thinking attitudes*

*We shall not cease from exploration,
and the end of all our exploring
will be to arrive where we started
and know the place for the first time.
(T. S. Eliot)*

*Take wrong turns. Talk to strangers.
Open unmarked doors. And if you see
a group of people in a field, go find out
what they're doing. Do things without
always knowing how they'll turn out.
(xkcd.com)*

Contents

Summary	VII
Acknowledgements	IX
1 Introduction	1
1.1 Objectives	2
1.2 Contributions	4
1.3 Structure of the thesis	4
2 Description logics and design of knowledge bases	7
2.1 State of the art	7
2.1.1 The OWL family	8
2.1.2 OWL 2 Profiles	11
2.1.3 Fuzzy and probabilistic extensions	12
2.2 Using knowledge representation technologies	14
2.2.1 Why to use semantic languages?	14
2.2.2 The use of DLs in robotic-related fields	15
2.3 Design issues	16
2.3.1 Representation	16
2.3.2 The role of reasoning	24
2.3.3 Expressivity of the logic	28
2.3.4 Knowledge base management	32
2.3.5 Queries	33
2.4 Summary	34
3 Description logics for object representation	35
3.1 State of the art	36
3.1.1 Recognition-by-components	37
3.1.2 Shape description and modelling	37
3.1.3 Use of knowledge	40
3.2 Issues in part decomposition	42

3.2.1	Conceptual issues	42
3.2.2	Practical issues	42
3.3	Implementation	44
3.3.1	Datasets	46
3.3.2	Knowledge base	50
3.4	Experiments	51
3.4.1	Examples of queries	51
3.4.2	Evaluation	58
3.5	Discussion	58
3.5.1	Future work	62
4	Description logics and grasping	63
4.1	State of the art	63
4.2	Representation issues	65
4.3	Implementation	67
4.3.1	Dataset	67
4.3.2	The grasping ontologies	67
4.3.3	Use grasp features to describe objects	70
4.3.4	Use object features to retrieve grasps	80
4.4	Experiments	81
4.5	Discussion	90
4.5.1	Future work	91
5	Description logics and cognitive architectures	93
5.1	State of the art	93
5.1.1	The POETICON++ project	96
5.2	Issues in semantic memory design	97
5.3	Implementation	98
5.3.1	PRAXICON overview	98
5.3.2	Requirements	100
5.4	Experiments	102
5.5	Discussion	109
5.5.1	Future work	110
6	Conclusions and future work	111
6.1	Comparison with KnowRob	113
6.2	Future work	114
	Bibliography	117

List of Figures

2.1	Examples of membership functions.	13
2.2	Interpretation issues on structural definitions.	19
2.3	Examples of necessary, sufficient and equivalence conditions.	22
2.4	Example hierarchy.	26
2.5	Seaside image with segmentation.	28
2.6	Example of property chain.	31
3.1	Biederman’s geons.	38
3.2	Marr’s structural representation.	39
3.3	Objects with their topological graphs.	45
3.4	Shapes from the AIM@SHAPE dataset.	47
3.5	Example of an object with all the bounding boxes.	48
3.6	Example of an object with all the bounding ellipsoids.	49
3.7	Query 1: Objects having at least n parts.	53
3.8	Query 2: Oblong parts.	53
3.9	Query 3: Objects having one part connected to a big part.	54
3.10	Query 4: Objects with parts connected to at least other 6 parts.	54
3.11	Objects having the same topological graph.	55
3.12	Query 5a: Objects with the same topology.	56
3.13	Query 5b: Objects with the same topology and different geometric constraints.	57
3.14	Query 6: Class retrieval using number of matching examples.	59
4.1	Untransformed and transformed trajectories of the finger tips.	69
4.2	Example of the two grasping ontologies.	71
4.3	Distances between finger tips for the Large Diameter grasp.	73
4.4	Representative grasp measures.	77
4.5	Example of generated membership functions.	78
4.6	Membership functions related to the object shape.	79
4.7	Objects used in the experiments.	83

5.1	Example of PRAXICON relations and their arrangement. . .	100
5.2	Object models.	107
5.3	Graphical representation of the results of the queries.	108

List of Tables

2.1	Basic logics, their extensions and some common aliases. . . .	9
3.1	Results for objects having n parts with varying n	60
3.2	Results of the example queries.	60
4.1	Objects used in the grasping experiments with their sizes. . .	68
4.2	Measures for each grasp.	73
4.3	Errors between estimated and real measures.	75
4.4	Chosen representative grasps.	77
4.5	Grasp names abbreviations.	83
4.6	Airplane.	84
4.7	Bearing.	85
4.8	Chair.	86
4.9	Table.	87
4.10	Cup.	88
4.11	Vase.	89
5.1	Objects real sizes.	107

Summary

The use of semantic technologies in robotics has seen a growing trend in the last decade, partly because of the popularity and the support provided by well-grounded research in this field, partly because of a partial return of robotics to its AI roots. Each application in the robotic domain requires a peculiar representation of knowledge, specific reasoning services and suitable storing and querying facilities depending on the task at hand. In many cases a logic-based representation can be advantageous and provide more flexibility, while in other cases it can be either less efficient or less scalable with respect to other state-of-the-art approaches; therefore, a thorough analysis of the requirements of a task and of the advantages such representations can offer has to be performed prior to each application.

The purpose of this thesis is twofold. From a robotics point of view its aim is to discuss how semantic technologies, especially description logics and semantic query languages, can be used for robotic-related tasks: we will discuss applications within different fields such as shape modelling and retrieval, computer vision, manipulation, linguistic and cognitive modelling along with some examples in order to highlight the current issues and the possible challenges. From a knowledge representation point of view, on the other hand, this thesis provides an analysis of several less common domains, focusing then on the balance between the expressivity needs of a representation and its scalability requirements, which all together drive the choice of the tools to be used.

Acknowledgements

Like every long project, this work would have been hardly possible without the support of many people I would like to thank here. First of all, of course, a big “Thank you” goes to my advisor, Prof. Giuseppina Gini, for she gave me all the freedom and much good advice I needed for my work. Another “Thank you” goes to Mario, as somehow he always found the time to think and discuss about my ideas. I also want to thank my reviewers, Prof. Stefano Caselli and Dr. Moritz Tenorth, for the time they found to read the thesis and writing their precious comments. As a part of the thesis focuses on my experience abroad, I want to thank the CAS-CVAP lab in Stockholm for hosting me (thanks Danica, Jeannette, Marianna, Andrzej, Lazaros, Magnus, Alessandro and Marin) and the CSRI group in Athens for the many interesting discussions and the very inspiring atmosphere I could be a part of (so thanks Katerina, Giorgos, Panagiotis, Eirini, Argyro and Dimitris).

A PhD cannot be a lonely journey, thus a friend and colleague such as Flavio is exactly what you need for making it a performance more than a work: attending conferences could have never been so “interesting” (and possibly risky for the Ministry of Foreign Affairs) without him. I think I should thank him a lot and say he is a great researcher, but this would inflate his ego too much. I also want to spend some words about all my AIRLab friends for sharing offices, beers, houses and whatever was available at the moment, so: thanks Martino, Simone, Fabio, Luigi, Davide and Giulio for the good time and the “light” lunches spent together.

They say there is no place like home (or localhost, for that matters), thus a mention to Alessandro (the Source of Countless Ideas), Gerardo (the Guru), Michele (the Master of Being on Time), Ettore (the Wise) and Giampaolo (the Pragmatist) is absolutely necessary as, when I needed to discuss something or just to rant about it all, they were all there to listen to me and to suggest me the dos and don’ts. Thanks guys.

My life in Milan would never have been so interesting without my “good

old friends” Alessandro, Sante, Massimo, Marco, Graziano, Graziella, Anna, Elena, Clara: the countless dinners and evenings out make me feel indebted with all these people for the hours of fun had together. Without you, I’d probably be a sad man right now. Thanks also to my “good old classmates” Danilo (and Desirèe), Fabio, Iacopo, Beppe (and Emanuela), Emiliano (and Kristina) for the get-togethers to remember the “good old times”, and thanks to Marta for the “ranting together” evenings.

I have no problems in admitting that I would not have finished my PhD without the support of people like Marco, Maddalena, Mauro and Maria Rosa: their bets on my future (either on success or failure) and all the candies I robbed from them (not to mention the long conversations) made everything easier and, especially, gave me a lot of good material for when I’ll be writing my biography. And by the way, talking about concrete help, Nadia saved so many times my... er... “day” that she deserves a mention on her own.

A big loving “Thank you” goes obviously to my family, my parents and my brother, and I guess this does not need any further explanations. As for Raminta, I guess nothing would say more than *Žalioji*. And *Labai ačiū*, probably.

Last, but not least, thanks to the taxpayers for having supported this PhD.

My hope is that, in the years to come, I’ll be able to return all the people I mentioned at least something of what they gave me.

Chapter 1

Introduction

Research in robotics has moved throughout several phases in its history: the *sense-think-act* paradigm, where inputs from the sensors are used to generate a plan which is then transformed into action, has been challenged by a *behaviourist* approach, where the control system responds dynamically to the changes in the environment by means of separate activities arranged in layers. The main difference between these two points of view is in the use of *representations*: while in the latter “the world is its best model”, the former, being rooted in artificial intelligence, needs symbols, concepts and reasoning abilities for building rich world models.

The interest in the behaviourist approach has been revived by recent theories on the importance of *embodiment* and *neuropsychological development* in robotics, so that sensing and acting are considered as tightly coupled and the focus is shifted on the robot body and its perception, introspection and proprioception; the study of this perspective is supported by the availability of research platforms such as the iCub, which provide researchers some common grounds for developing and testing such theories. On the other hand, approaches based on *cognitive modelling* focus more on the different aspects of human intelligence such as the use of language, learning skills, generalization abilities and interaction with people; for this reason, the disciplines involved in research on cognitive robotics are many and diverse.

We are now in a phase where AI-based approaches are being reintegrated, albeit with different claims than in the past: rather than relying on purely symbolic approaches as the only models for intelligence, the current perspective is now to adopt symbols as a support for integration of different representations and as a way to code (and make readily available to robots) what is called *high-level knowledge*, which is the basis of human communication and interaction. In order to serve to this purpose a symbolic system has

to interact with different sources of data, from the low level (such as data coming from sensors) to the high level (such as linguistic representations, plans, categories and so on); therefore, a symbolic representation should be able to integrate such different channels of information and possibly act as an interface among them.

Description logics [Baader et al., 2003] define a logic-based framework for knowledge representation and reasoning which has received much attention in the last decade, the reason being their decidability property which makes them more attractive with respect to traditional AI approaches (i.e. theorem proving based on first order logic). As description logics provide a unified formal approach to semantics, they have been used in many fields related to robotics for different kinds of applications along with *semantic query languages*, which constitute the semantic counterpart of query languages for databases.

1.1 Objectives

In this thesis we will discuss the issues related to the use of description logics and semantic query languages in the robotic domain, analyzing use cases from different yet related fields along with possible issues and examples of solutions. We will focus in particular on the following aspects:

- Representation: as the knowledge base design is the first step to build a logic-based framework, what do classes, instances and relations represent in the specific domain and how to structure them accordingly?
- Reasoning tasks: as semantic languages are commonly used because of the reasoning facilities they offer, what kind of reasoning tasks are expected to be performed within the specific domain?
- Expressivity: as there exists different reasoning engines and semantic storage solutions which are optimized on specific logics, is there a minimum expressivity needed for the representation and reasoning tasks to use?
- Knowledge base management: based on the expressivity of logic, the reasoning tasks and the expected volume of data, what is the best solution for storing the knowledge base and how new data is added?
- Queries: what is the expected type of query to execute on the knowledge base and what are the languages to be used?

- Extensions: are any extensions to the standard description logics needed for the specific application?

By focusing on such knowledge representation issues we aim to have a better understanding of what does *semantics* mean in robotics and of the “level of semanticity” we can expect to be useful within robotic tasks; additionally, we analyze how different levels of information can be integrated by means of semantics. More specifically, the fields we have explored are the following:

- *object representation*, where the main problem is on whether a representation of an object based on part decomposition can be effectively used and how it should be designed using a logic-based approach to make it usable and scalable;
- *grasping*, where the main problem is on how to decide the possible grasps to be performed on an object, given a geometric representation of both the shape of a hand and the shape of the object itself in terms of a fuzzy extension of description logics;
- *cognitive architectures*, where the main problem is on how to build an *embodied ontology* to integrate different representations (from low- to high-level) for concepts, using language as a guiding criterion to structure the knowledge.

Our aim is therefore to discuss strengths and weaknesses of a logic-based representation on different levels: in each chapter it will be shown what is the amount of information provided respectively by humans (e.g. in defining names for concepts, grouping criteria for parts and so on) and by raw data. The long-term goal of this research is to obtain a suitable conceptual model for integrating all the available sources of information, thus semantics is here used as a means for achieving interoperability and for making the acquisition of new information more structured and autonomous.

Language has a great deal of importance for “grounding” knowledge; as we will see later, grounding perceptions directly to symbols via a “translation” does not provide a sufficient account of the neural and cognitive processes underlying the low-level representation of entities, while language provides some organizational principles to guide the acquisition, organization and comparison of symbols. Knowledge representation formalisms such as DLs are based on language because they are used for providing common vocabularies in order to describe specific domains; thus, the integration of human-understandable concepts makes resources for robots accessible by

humans; in fact, as a byproduct of our analysis of low-level information, we built several resources which can be useful for humans as well.

1.2 Contributions

The main contributions of this thesis are:

- a review of the application of semantic technologies to the robotic domain (knowledge acquisition, object modelling, grasping, vision, language processing and understanding, cognitive modelling);
- the identification and discussion of several aspects in the design of a knowledge base to use in robotic-related tasks;
- the realization of a knowledge base of objects, where knowledge is not only related to linguistic metadata but also to lower level characteristics of the objects themselves (such as geometric features and part decomposition);
- the definition and evaluation of a scalable logical framework for building graph queries using objects' topology and geometry;
- examples of integration of different sources of information, providing a test bench for state-of-the-art semantic resources (description logics and their extensions, reasoners, query languages and engines, semantic storage systems etc.);
- a discussion on the balance between the level of expressivity offered by description logics and the scalability needed by application making use of big quantities of data.

This thesis work brought to the publication of some papers [Vitucci et al., 2010a, Vitucci et al., 2010b, Vitucci, 2011, Vitucci et al., 2012].

1.3 Structure of the thesis

The thesis is organized as follows: **Chapter 2** introduces description logics, their state of the art along with some extensions and the issues related to the design of a knowledge base with a focus on robotic-related domains; **Chapter 3** deals more specifically with the problem of representing objects using a logical formalism, highlighting conceptual and practical issues which depend on the specific tasks the representation will be used for and providing an extended example of formalization and retrieval; in **Chapter 4** some

extensions to description logics will be presented along with a possible application to the manipulation domain, where a logic representation of objects and grasps might be needed; in **Chapter 5** some results of the application of description logics to cognitive robotics will be shown, implementing a knowledge base on top of an ongoing research on embodied linguistics and its applications to robotics; finally, in **Chapter 6** the overall results will be discussed and evaluated and possible future works will be proposed.

Chapter 2

Description logics and design of knowledge bases

Description logics (DLs) are a family of formal languages based on logics used for knowledge representation. As they provide an expressive logical formalism for structuring knowledge, DLs are mostly used for two tasks: to achieve *interoperability* between different representations by providing common vocabularies and terminologies, as the Semantic Web aims to do, and to perform *inference*, the process of extending existing knowledge about the world by automatic reasoning on formal concepts. Since their introduction in the 1980s as an alternative to frames and semantic network, DLs have seen a huge research effort to make them expressive while retaining the properties which make them computationally viable, with the aim to make them effectively usable for a variety of tasks.

The purpose of this chapter is to provide an overview of this formalism, the main issues related to its use in the modelling of a general domain and the possible applications to real-world domains related to robotics. We will highlight some aspects which have to be carefully evaluated while building a knowledge base, discussing each of them with examples from robotic-related fields and drawing some conclusions and suggestions for modelling similar tasks.

2.1 State of the art

The term *description logics* is used to denote a range of different logics, each of them providing several constructs and a certain level of expressivity for formalizing knowledge. Description logics are more expressive than propositional logic but less expressive than first order logic (FOL): this means that

FOL provides more “richness” or “flexibility” in expressing knowledge on a domain, but this comes to the expense of an important property called *decidability*.

A logical expression is said to be *satisfiable* if there exists an assignment of truth values to its variables which makes the expression true; for instance, the expression *John is tall and thin* is satisfiable because, if John is both tall and thin, it is true. The problem of deciding whether an expression is satisfiable is *decidable* in propositional logics, which means that, given an expression in a propositional logic, it is always possible to say whether it is satisfiable or not; this is not the case in FOL, where the problem is not decidable. DLs make use of subsets of FOL in order to gain expressivity with respect to propositional logics while retaining the decidability property.

DLs are used for building *ontologies* and *knowledge bases*, which can be described as collections of statements regarding *concepts* (also called *classes*), *roles* (also known as *properties*) and *individuals* (denoted also as *instances* or *objects*) organized in a *terminological box* (or *TBox*, containing axioms describing concepts and their relations such as $\text{Man} \sqsubseteq \text{Person}$) and an *assertional box* (or *ABox*, containing axioms describing individuals and their relations with concepts and other individuals such as $\text{Man}(\text{John})$).

One of the most important reasons to use DLs is that they can be used to infer new knowledge through a process called *reasoning* by the means of a *semantic reasoner*. More expressive DLs require more complex reasoning algorithms; this is the reason why there exist different DL families, which can become more or less complex depending on the constructs that are added or removed. Table 2.1 shows the basic DL families along with their extensions and some aliases.

2.1.1 The OWL family

Statements expressed in a DL can be represented as *triples* (data entities having the format *subject-predicate-object*) according to the RDF¹ data model and stored in relational databases or *triple stores*. RDFS² and the family of OWL³ languages are built on top of RDF and provide the additional expressivity needed for building complex ontologies. In particular, the current standards OWL 2 DL³, OWL 1 DL⁴ and OWL-Lite⁴ are aliases respectively for the logics $\mathcal{SROIQ}(\mathcal{D})$, $\mathcal{SHOIN}(\mathcal{D})$ and $\mathcal{SHIF}(\mathcal{D})$. The last language has been introduced separately for building simple taxonomies

¹<http://www.w3.org/TR/rdf-primer/>

²<http://www.w3.org/TR/rdf-schema/>

³<http://www.w3.org/TR/owl2-overview/>

⁴<http://www.w3.org/TR/owl-features/>

Logic	Characteristics
\mathcal{AL}	atomic concept negation ($\neg C$, where C is an atomic concept), concept intersection ($C \sqcap D$), universal restrictions ($\forall R.C$) and limited existential quantification ($\exists R.T$)
\mathcal{FL}	concept intersection, universal restrictions, limited existential quantification and role restriction ($R _C$, roles having C as a filler)
\mathcal{EL}	concept intersection and full existential quantification ($\exists R.C$)
Symbol	Meaning
\mathcal{E}	full existential quantification
\mathcal{U}	concept union ($C \sqcup D$)
\mathcal{C}	complex concept negation ($\neg D$, where D is a generic concept); includes \mathcal{U} and \mathcal{E}
\mathcal{H}	role hierarchy ($R \sqsubseteq S$, where R and S are roles)
\mathcal{R}	inverse roles, intersection and union of roles etc., reflexivity and irreflexivity, role disjointness; includes \mathcal{H}
\mathcal{O}	restrictions where the restriction class is an individual (such as $\text{RedObject} \equiv \text{hasColor}\{red\}$) and nominals (enumerated classes such as $\text{LowNumber} \equiv \{one, two, three\}$ where one , two and $three$ are individuals)
\mathcal{I}	inverse properties ($S \equiv R^-$)
\mathcal{F}	functional properties (if R is functional, an individual cannot have more than one relation R with other individuals)
\mathcal{N}	cardinality restrictions ($C \equiv \geq nR$, $C \equiv \leq nR$, $C \equiv = nR$ with $n \geq 0$); includes \mathcal{F}
\mathcal{Q}	qualified cardinality restrictions (e.g. $C \equiv \geq nR.D$, $C \equiv \leq nR.D$, $C \equiv = nR.D$ with $n \geq 0$); includes \mathcal{N}
(\mathcal{D})	the use of datatype properties, data values or data types (e.g. strings, numbers etc.)
Alias	Explanation
\mathcal{S}	alias for \mathcal{ALC}^+ , i.e. an abbreviation for \mathcal{ALC} (or equivalently for \mathcal{ALUE}) with transitive roles
\mathcal{FL}^-	sub-language of \mathcal{FL} without role restriction, also equivalent to \mathcal{AL} without atomic negation
\mathcal{FL}_o	sub-language of \mathcal{FL}^- without limited existential quantification
\mathcal{EL}^{++}	alias for \mathcal{ELRO}

Table 2.1: Basic logics, their extensions and some common aliases ([Baader et al., 2003]).

which would not need the complexity of OWL DL, but it has not received significant attention in that the limited advantage it provides on complexity does not justify the lack of most of the useful constructs which can be found in OWL 1. A fourth language called OWL Full has been created to relax several constraints posed by OWL DL, but differently from the other three it is not decidable in that it mixes freely OWL and RDF(S) without enforcing any distinction between classes, properties and instances or putting any constraints on the axioms (e.g. in OWL DL cardinality restrictions cannot be used on transitive properties, and individual equality can only be stated between named individuals). OWL 1 has been replaced by OWL 2, which is now the reference language for building ontologies.

RDF triples can be *serialized* using XML syntax: for example the axiom *Men are persons*, represented as $\text{Man} \sqsubseteq \text{Person}$ in DL and as the triple $\langle \text{Man} \text{ subClassOf } \text{Person} \rangle$, in RDF/XML becomes:

```
<owl:Class rdf:ID="Man">
<rdfs:subClassOf rdf:resource="#Person" />
</owl:Class>
```

while the axiom *John is a man* becomes:

```
<owl:NamedIndividual rdf:about="John">
<rdf:type rdf:resource="#Man" />
</owl:NamedIndividual>
```

Other serialization formats such as Turtle⁵ (the most common), N3⁶ and N-triples⁷ are available.

The use of RDF makes it possible to represent the relations among resources as a graph, so that they can be queried using languages such as SPARQL⁸, where the queries are expressed as problems of graph matching. For instance, a query like *Find all the individuals who are men* will look like the following:

```
SELECT ?i
WHERE {?i rdf:type myont:Man}
```

⁵<http://www.w3.org/TeamSubmission/turtle/>

⁶<http://www.w3.org/TeamSubmission/n3/>

⁷<http://www.w3.org/2001/sw/RDFCore/ntriples/>

⁸<http://www.w3.org/TR/rdf-sparql-query>

where `rdf` and `myont` are *namespaces*, the “places” in which the description of the resources `Type` and `Man` can be found.

Although it is possible to use SPARQL with OWL ontologies, a reasoner has to be used first for the engine to be able to perform queries on the inferred model. Furthermore, using SPARQL for queries on the TBox is not trivial.

2.1.2 OWL 2 Profiles

Sometimes it is not necessary to make use of the full OWL 2, either because only a few constructs in addition to RDF(S) are needed or because the application needs to be optimized for TBox or ABox reasoning. For instance, a reason why to use OWL 2 RL instead of RDF(S) is that, in addition to RDF(S) constructs, several OWL 2 constructs (namely intersection, union, existential and universal quantification, inverse properties) are available, albeit with some limitations; if such constructs are necessary for the application, a system capable not only to store the RDF(S) triples but also to reason on this logic has to be chosen. This is the reason why, as we will see later, the level of expressivity of a DL or of any of its fragments is very important to be assessed having in mind the final application, because it restricts the choice of reasoning engines (and possibly of storage systems) that can have good performances or that can be used altogether.

In order to optimize the performances for specific applications, some fragments of OWL 2 called *profiles* have been created selecting different subsets of constructs:

- OWL 2 EL is based on \mathcal{EL}^{++} [Baader et al., 2005, Baader et al., 2008] and it is mostly used with large TBoxes and small (or empty) ABoxes, as in the case of biomedical ontologies;
- OWL 2 QL is based on DL-Lite [Calvanese et al., 2007] and it is mostly used for scalable query answering in large ABoxes with simple TBoxes;
- OWL 2 RL is based on Description Logic Programs (DLP) [Grosz et al., 2003] and provides a kind of “compromise” between the expressivity of OWL 2 DL and the scalability required by query answering tasks in large ABoxes.

The three fragments differ in the constructs that can be used within a subsumption axiom of the kind $C \sqsubseteq D$, where C and D are said to belong respectively to the *left side* and to the *right side* of the subsumption; the peculiarities can be summarized as follows (see [Krötzsch, 2012] for a more detailed discussion):

- Intersection: always allowed but on the left side in OWL 2 QL;
- Union: never allowed but on the left side in OWL 2 RL;
- Negation: allowed only on the right side in OWL 2 RL/QL;
- Inverses: allowed in OWL 2 RL/QL but not in OWL 2 EL;
- Existential quantifiers: allowed completely in OWL 2 EL, with restrictions on the left side in OWL 2 QL, only on the left side in OWL 2 RL;
- Universal quantifiers: allowed in OWL 2 RL (on the right side) but not in OWL 2 EL/QL.

When a construct is allowed only on one side, no equivalence axioms can be built using it because an equivalence is a shortcut for a pair of subsumption axioms, i.e. $C \equiv D$ is a synonym for the couple of axioms $\langle C \sqsubseteq D, D \sqsubseteq C \rangle$; this means that the equivalence axioms that can be written in OWL 2 QL can only make use of the limited existential quantification (e.g. $C \equiv \exists R.T$), while in OWL 2 RL they can only make use of concept intersection (e.g. $C \equiv D \sqcap E$). To summarize,

“OWL EL has all features of OWL QL other than inverse properties. Adding them makes all standard reasoning tasks EXPTIME-HARD.” ([Krötzsch, 2012, p. 64])

and

“All profiles also support datatypes and property hierarchies. OWL EL and OWL RL further support equality, keys, nominals, property chains, and Self (EL only).” ([Krötzsch, 2012, p. 67])

2.1.3 Fuzzy and probabilistic extensions

In order to deal with the limitations affecting the use of description logics in real-world domains, and especially for dealing with vagueness and uncertainty, several extensions have been proposed and developed such as *probabilistic description logics* (see [Klinov, 2008]) and *fuzzy description logics* (see [Lukasiewicz and Straccia, 2008, Bobillo and Straccia, 2008, Stoilos et al., 2008]). While the former are based on the probability theory the latter are based on fuzzy logics, which are many-valued logics (where truth values can be more than the usual *true* and *false*) that usually make use of *linguistic variables* to express assertions and to build rules without

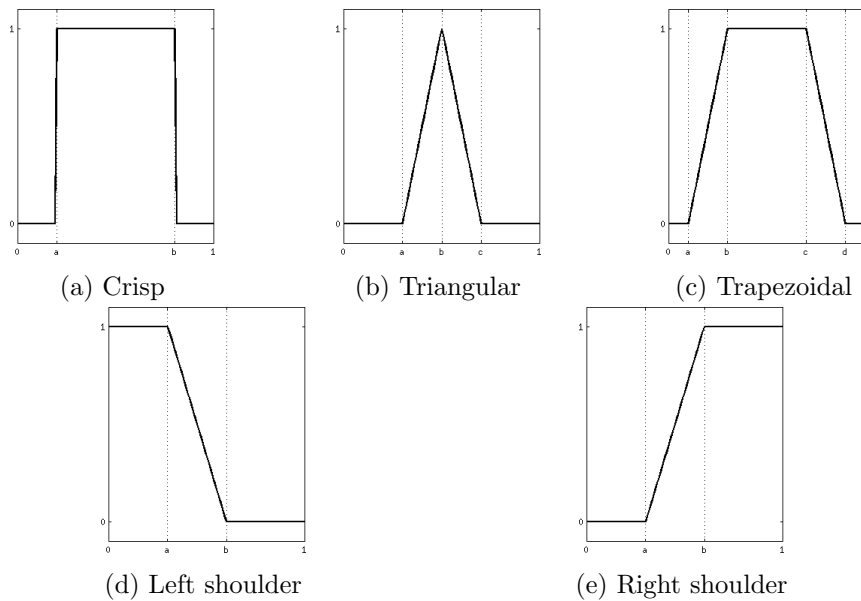


Figure 2.1: Examples of membership functions used for defining concrete fuzzy concepts.

using numbers, e.g. *If temperature is very cold, then stop fan*. Numeric values are associated to variables such as “very cold” using *membership functions*, which describe the degree of membership of a numeric value to the sets denoted by all the variables (see Fig. 2.1).

It is important to point out that “vague” in this context is used not in the sense of “unknown”, but rather in the sense of “flexible”: a fuzzy formalization of knowledge is considered more “human-like” as it can make use of concepts such as **High** and **Low** instead of exact numerical quantities, **very** or **little** as truth-value modifiers and so on; furthermore, an important characteristic of fuzzy DLs is that an assertion does not need to be “either true or false”: depending on “how much” its components are true an assertion can be “at least/at most *this much* true”, where “this much” is a real value between 0 and 1 (this is the concept of *best lower/upper truth-value bound*).

The state-of-the-art fuzzy reasoners available are fuzzyDL⁹ and FiRE¹⁰. fuzzyDL reasons on $f - SHIF$, a fuzzy extension of the $SHIF$ logic, and provides several operators from the fuzzy set theory (such as t -norm, t -conorm, negation and implication), constructs to define fuzzy sets having an explicit membership function (called *concrete fuzzy concepts*, see Fig.

⁹<http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html>

¹⁰<http://www.image.ece.ntua.gr/~nsimou/FiRE>

2.1), *features* (functional datatype attributes with a concrete range) and weighted sums such as in the following example:

```
(implies
  (and C
    (w-sum (0.3 (some rel A)) (0.7 (some rel2 B))))
  D)
```

which is a sufficient condition axiom (of the form $A \sqcap B \sqsubseteq C$) to be interpreted as follows: if an object is an instance of a class C and is related to objects belonging respectively to the classes A and B , giving some preference to the latter relation, then it is considered to be an instance of class D (with a certain degree of truth). A concrete example of the usage of weighted sums can be the axiom *A patch is considered a “clouded sky” if it depicts both a portion of sky and some clouds, giving more importance to its “skyness”*:

$$\text{CloudedSky} \equiv \text{Cloud}_{0.3} \sqcap \text{Sky}_{0.7}$$

FiRE, on the other hand, reasons on f -*SHIN*; while its underlying logic is more expressive, it lacks several of the fuzzy constructs provided by fuzzyDL such as fuzzy assertions in TBox.

There exists also a fuzzy extension for RDF called URDF¹¹ (not to be confused with the URDF language for robotics mentioned in Sec. 2.2.2).

2.2 Using knowledge representation technologies

2.2.1 Why to use semantic languages?

Generally speaking, the advantages of using semantic languages such as RDF and OWL, along with powerful query languages such as SPARQL and SPARQL-DL, are several:

- not only data instances but also the data *model* can be queried, which means that the relationships among data can be discovered and extended by means of suitable queries;
- several knowledge bases can be queried with a single simple query, which can be seen as a graph instead of a series of joins;

¹¹<http://urdf.mpi-inf.mpg.de>

- it is possible to make use of structured and semi-structured data;
- it is not necessary to explicitly encode all the domain knowledge, for a well-structured knowledge base can infer new facts with the use of a semantic reasoner.

In order to make use of such advantages, a knowledge base has to be “well built”: if it misses any axioms or some of the axioms are wrong, the inference process will give useless, unexpected, wrong or no results at all; on the other hand, if it is well designed, it can produce some interesting facts or be used to check that all the constraints are met and data are consistent.

The design of a knowledge base using DLs requires a solid understanding of their roots in logics; in fact, misunderstandings and misconceptions of the logical foundations can cause errors in the formalization or give false expectations on the results. As the DL paradigm is different from the paradigm of databases, some attention has to be paid in the use of query languages as well.

2.2.2 The use of DLs in robotic-related fields

Semantic technologies have seen an uprise in robotic-related fields such as hardware description, navigation, object description, localization and recognition, manipulation, planning and so on. Some recent efforts include:

- KnowRob [Tenorth and Beetz, 2009, Tenorth and Beetz, 2012], a framework including common concepts related to robots and knowledge processing facilities, which will be discussed more in detail later;
- RoboEarth [Waibel et al., 2011], a database where information about objects, actions and environments are gathered and can be exchanged among robots with the aim of building a “WWW for robots”;
- the OpenRobot Commonsense Ontology (ORO) [Lemaignan et al., 2010], which is based on OpenCyc [Lenat, 1995] and aims at formalizing some common sense knowledge useful for robotic applications;
- SRDL (Semantic Robot Description Language [Kunze et al., 2011]), a semantic version of URDF (Unified Robot Description Format, used in ROS¹²) for describing the structure of a robot;

¹²<http://www.ros.org/wiki/urdf>

- OUR-K [Lim et al., 2011], a unified knowledge base for robotic tasks with the aim of bridging high-level and low-level data in a unique framework.

The use of such technologies has spawned several discussion boards and panels, such as the Autonomous Robots Ontology Subgroup of IEEE RAS¹³, in which standards and objectives for semantic languages and knowledge representation for robotics applications are discussed.

Several works deal with other aspects of robotics, such as urban search and rescue [Schlenoff and Messina, 2005] and planning [Hartanto and Hertzberg, 2009]; there exists also a whole body of work in computer vision and graphics applications, from low-level data description to high-level scene interpretation and object representation; such applications will be discussed in the next section and in Chap. 3.

2.3 Design issues

In this section we will discuss the requirements and the problems inherent to the use of semantic technologies with a focus on robotic-related fields. We will highlight some criteria which will guide the design process for the examples discussed in the next chapters.

2.3.1 Representation

The first aspects to evaluate when building a knowledge base are *what* to represent and *how*. As we already mentioned, the core concepts in knowledge representation are classes (or concepts), instances (or individuals) and properties (or relations); to avoid any misunderstanding, it is useful to remember that a class is a set (in mathematical sense) of instances, which might be related among each other through binary properties; thus, every operation involving classes is applied to sets of objects, while operations on instances are applied to single elements.

Classes or instances?

The use of classes and individuals is a common matter of debate while designing a knowledge base: down to which level of detail should the model be described by classes? From which level do the instances “begin”? How should classes and instances be represented?

¹³<http://aro.svn.sourceforge.net>

It is of course different to say that *All the tables have four legs* and *The table in my kitchen has four legs*, because the former is a feature of (or constraint on) all the tables within the domain, while the latter is a feature of a specific table. Extending this example, we might have two different representations of a table:

1. $\text{Table} \equiv \#1 \text{ hasTop.}(\text{RectangularShape} \sqcup \text{RoundShape}) \sqcap$
 $\#4 \text{ hasLeg.}(\text{LongObject} \sqcap \text{SquareBase})$
2. $\text{hasTop}(\text{table}, \text{top}), \text{SquareShape}(\text{top}), \text{hasLeg}(\text{table}, \text{leg1}),$
 $\text{hasLeg}(\text{table}, \text{leg2}), \text{hasLeg}(\text{table}, \text{leg3}), \text{hasLeg}(\text{table}, \text{leg4}),$
 $\text{LongObject}(\text{leg1}), \text{SquareBase}(\text{leg1}), \text{LongObject}(\text{leg2}), \text{SquareBase}(\text{leg2}),$
 $\text{LongObject}(\text{leg3}), \text{SquareBase}(\text{leg3}), \text{LongObject}(\text{leg4}), \text{SquareBase}(\text{leg4})$

The first form provides a *model* for an object to be a table, thus saying that a table has to have a top, which can be either rectangular or round¹⁴, while the second form gives *facts* that are known about a specific table, thus stating that a certain object called `table` is known to have a square top, but there is nothing to prevent it to have a round or triangular top (or even to have more than one top), if there are no restrictions on the class to which it belongs.

The second form can be interpreted not only as a description of a specific table (e.g. the table with number #1 in a restaurant): the instance `table` might also be representing a *generic* table having such attributes. Although in this case it would not be possible to perform any inference on unknown objects to decide whether they are tables or not (as `table` is not a class), another instance such as `circularTable` would represent a different *kind* of table with different attributes which do not need to (or cannot) be known beforehand.

Terminological axioms, when they do not merely define a hierarchy of concepts, can be used as consistency checks: if tables have to have *exactly* four legs and a specific instance of a table is said to have five, the knowledge base will become inconsistent; this means that, provided that the remaining knowledge in the model is correct, the information on the affected instance is not valid and has to be removed. Additionally, TBox axioms can be used to infer some “general” knowledge related to whole classes of objects: for instance, if the class of stable objects is described as the class of objects having at least two legs, tables as described before will be inferred to be

¹⁴We can assume that there are no other possibilities and that the two shapes are different, i.e. the two classes are disjoint.

stable objects, and this will hold for all the instances of the class (it will be *intrinsic* to the definition of the class).

On the other hand, instances can be used like entries in a database, each one having its own attributes: ABox queries can be executed to retrieve from the knowledge base objects having the attributes of interest. For instance, if we are looking for objects having four long square-based legs and a square top, the instance `table` will be retrieved; we will discuss in more detail this kind of queries in 2.3.2.

A common choice in robotic applications is to use concepts to represent classes of objects (such as the class `Cup` as the set of all the cups) and individuals to model single objects (e.g. `cup1` is a specific cup on the desk, having its own size, weight and so on).

Structural descriptions

Complex concepts like *A table has four legs and a surface* and *A table is either made of wood or of plastic* can be formed using intersection, union and other available constructs. It is anyway important to note that concepts can only have a tree-like representation in OWL, which means that axioms such as *A table has four legs parallel to each other* cannot be expressed, because an axiom like `Table \equiv =4 hasPart.(Leg \sqcap =3 parallelTo.Leg)` would have other models different from the expected one (Fig. 2.2).

More in general, a TBox formalization becomes difficult to be written correctly if it has to include parts (both direct and indirect), mereotopology axioms, constraints making use of reflexive and transitive roles and so on (see [Keet and Artale, 2008, Varzi, 2007]), the main problems being related to the complexity of reasoning on the needed logic and, mostly, in the logical correctness; in the robotic domain this can be a concern when attempting to model places, objects or kinematic chains as classes because, as it is shown in Fig. 2.2, the results can be different than expected. In these cases structural information about objects is not only difficult to formalize, but it also requires a highly expressive logic; thus, different formalisms (such as SWRL/DL-safe rules¹⁵ and description graphs [Motik et al., 2008]) or different representations (e.g. in terms of instances) are needed. For more examples and a detailed discussion on the problem of formalizing structures in OWL, see also [Motik et al., 2008, Hastings et al., 2010].

¹⁵<http://www.w3.org/Submission/SWRL/>

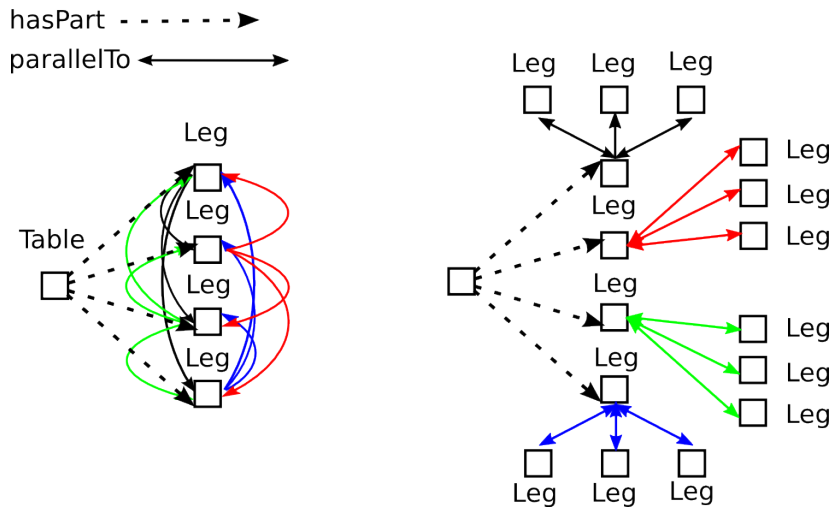


Figure 2.2: Interpretation issues on structural definitions. Two possible models for $\text{Table} \equiv \text{hasPart}(\text{Leg} \sqcap \text{parallelTo}(\text{Leg}))$ are represented.

Level of detail

Another problem often arising in the formalization of knowledge is the level of detail down (or up) to which information can be represented. As we have mentioned, there are several works in the computer vision research field in which ontologies and DLs are used for object recognition, classification, description and so on. In [Falomir et al., 2011, Straccia, 2009, Hudelot et al., 2008, Bloch, 2006] a formal approach (possibly extended with the use of fuzzy sets) is used for building higher level concepts directly from low-level data. Research on qualitative spatial reasoning has been carried in many works by Bennett [Bennett, 2011, Mallenby and Bennett, 2007, Bennett, 2002] and Cohn [Bennett et al., 2000, Cohn et al., 2006, Cohn et al., 1993]. In these fields, the main problem is to find the maximum level of detail such that knowledge can be grounded directly and conveniently on input information such as visual data.

For instance, if a classifier can recognize only edges or surfaces, higher level knowledge should be built on top of this information; if, on the contrary, it is able to provide higher level labels, it is important to know down to what level of detail they can be used to discriminate objects. Another example comes from object recognition tasks, where an object classifier can provide different levels of classification for single objects (labels can be like “box”, “cereal box”, “SpecificBrandTM cereal box”, “SpecificBrandTM open cereal box” and so on, from the least to the most specific); depending both on

how specific a label can be and on what kind of task has to be performed, the domain knowledge should be formalized accordingly. In the example, if the classifier can distinguish cereal boxes having different brands, it might be useful to have a class for each brand, all of them being subclasses of a `CerealBox` class (and, on an upper level, of a `Box` class); on the other hand, if the classifier can only discriminate boxes from bottles, it might be unnecessary to create more subclasses of such classes as they would be built on an arbitrary basis.

Red, `hasColor.Red` or `hasColor.{red}`?

There are several alternatives for expressing that an entity has a certain property, for example to state that an object is red. The three mentioned forms are different in terms of modelling and expressivity, so the choice among them has to be performed at the beginning because of the constraints they might pose on the reasoner and on the complexity of the representation.

From a modelling perspective, the difference among these formalizations has been explained in the OntoClean methodology [Guarino and Welty, 2004]: while a class `Red` of red objects can be convenient when “there are a large number of entities that need to be partitioned according to the value of some attribute” [Guarino and Welty, 2004, p. 18], it might be more convenient “to model attributions with a simple attribute, like color, and a value, such as red” [Guarino and Welty, 2004, *ibid.*].

From an expressivity point of view, the first form requires the use of a class `Red` to which red-colored objects belong, making their being red a kind of “intrinsic” characteristic; the second form requires a property `hasColor` having objects as its domain and a color as its range, thus in this case an object would have a property linking it to another object which is the color itself (in this case belonging to the subclass `Red`, possibly including a variety of different instances of the red color) and requiring the full existential quantification construct; the third form would link the object to a *specific* instance `red` (possibly belonging to the `Color` class) entitled to represent the red color itself, requiring the construct \mathcal{O} for enumerated classes. The choice here is bound to the way a color is assigned by a classifier (e.g. as a discrete quantity called “red” or as a hexadecimal value).

Necessary and sufficient conditions

An axiom like *Tables have four legs* can be interpreted in three different ways:

$\text{Table} \sqsubseteq =4 \text{ hasPart.Leg}$
 $=4 \text{ hasPart.Leg} \sqsubseteq \text{Table}$
 $\text{Table} \equiv =4 \text{ hasPart.Leg}$

The first form states a *necessary condition* for an object to be a table, that is to have four parts each of them being a leg; in other words, a table is an object with four legs, but not all the objects having four legs have to be tables. The second type of axiom, less frequently used, treats the presence of four legs as a *sufficient condition* for an object to be considered a table, meaning that all the objects with four legs will be considered tables, but a table does not have to be described only by the fact that it has four legs (it might have to have a top part, for example). The third form, instead, is a *necessary and sufficient condition* or *equivalence*, stating that tables and objects with four legs are the same thing, which means that an object which is a table will be inferred by the reasoner to have four legs and the other way round.

When building the terminology of an ontology, it is important to understand the difference among this three formalizations (see Fig. 2.3) as it plays a major role when the reasoner has to be used as a classifier, for example in an object or scene recognition task. The following example is taken from [Simou et al., 2007], where the fragment of the TBox is used for scene recognition:

$\text{CloudedSky} \equiv \text{Cloud} \sqcap \text{Sky}$
 $\text{Sand} \equiv \exists \text{below-of.Sea}$
 $\text{Sky} \equiv \exists \text{above-of.Sea}$
 $\text{Leg} \equiv \text{Natural-Person} \sqcap (\exists \text{below-of.Body})$
 $\text{Head} \equiv \text{Natural-Person} \sqcap (\exists \text{above-of.Body})$

Here a leg is defined as something related to a person and located in the lower part of a body: this means that not only whatever is belonging to a person and is found in the lower part of a body is a leg, but also that a leg always belongs to a person and is located in the lower part of a body. This definition lets the reasoner infer that, in presence of two instances respectively belonging to the class **Natural-Person** and **Body** related through a *below-of* property, the first one is also an instance of the class **Leg**; this also implies that, if we add a similar definition (e.g. *A foot is located in the lower part of a body*) we will obtain that the foot is equivalent to a leg.¹⁶

¹⁶Obviously a foot can be described as something found in the lower part of a leg instead of a body, but this is for example purposes.

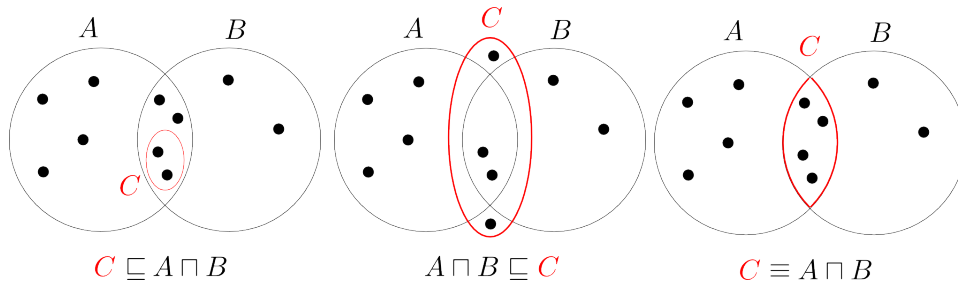


Figure 2.3: Examples of necessary, sufficient and equivalence conditions.

More generally, the use of equivalence axioms describing objects such as *A knife is something which is sharp and made of metal* makes the recognition of an object possible if evidence of its features is collected; on the other side, such axioms restrict the number of objects which can be described without obtaining unwanted equivalences. It is possible to loosen the equivalence axiom by writing it as a sufficient condition:

$$\text{Natural-Person} \sqcap (\exists \text{below-of.Body}) \sqsubseteq \text{Leg}$$

The drawback of such approach is that it needs an anonymous class on the left side, which prevents the possibility to build a taxonomy and is not always allowed (e.g. it is not supported in OWL 2 QL) because it is a source of complexity for a reasoner.

Inheritance and typicality

An axiom stating that a class has a certain attribute (e.g. *Birds have feathers and fly*) will be inherited by all the subclasses of the affected class, and clearly it will hold for all the instances of that class; this is not always desirable, as it may happen that several subclasses are considered as belonging to their parent class with some exceptions (the classical example is given by penguins, considered as “birds that do not fly”). This situation may arise upon the update of a knowledge base, for example when concepts are created automatically from other sources (e.g. WordNet) by collecting a number of identifying features, or when the creation of an ontology is driven by “common sense”: from this point of view, for example, it is “common sense” to consider a penguin as a bird or a dolphin as a fish. In any case, it would be desirable to keep the ability to perform concept classification even in presence of such cases.

From a robotic-related point of view, this can happen when not all the features identifying a concept (an object, a scene etc.) are known beforehand, so one assumes that a definition holds as long as counterexamples can

be found; for example, if a class `Ball` has been defined as “the class of objects having a `Spherical` shape” along with two subclasses `Basketball` (objects having an `Orange` color) and `SoccerBall` (objects having a `BlackAndWhite` color), then a `RugbyBall` is added as a subclass of `Ball` having an `Elongated` shape, the knowledge base will become inconsistent¹⁷. A similar example can be made with knives: if they are described as having a sharp blade (i.e. `Knife` \sqsubseteq \exists `hasBlade.Sharp`) and all the types of knives are put in a taxonomy having `Knife` as the uppermost class, a problem might arise when inserting in the ontology a butter knife, whose blade is not sharp but dull.

Inheritance, in the way it is implemented in description logics, does not allow exceptions; the standard solution to this problem is to loosen the definition of problematic classes by hand, creating subclasses respectively having and not having the conflicting attribute (e.g. dividing the birds in flying and non-flying birds, and describing birds as just having feathers); other approaches can be the use of a system for adding exceptions via justification (see [Kolovski et al., 2006]) or the use of a different logic (e.g. [Baader and Hollunder, 1995]), which would require ad-hoc reasoners to be practically usable.

An alternative can be found in a representation centered on instances rather than on classes: as instances are related to each other through relations (e.g. *A certain bird (or type of bird) can fly*), for every instance it is possible to state which features are present. In this case the inheritance relation can be used as a “hint” more than a strict hierarchical definition, so it can be implemented via an object property such as `isA` or `hasFatherConcept`; this representation also has the advantage to allow the use of a minimum, maximum or exact number of “transitivity steps” by using *property paths* as specified in SPARQL 1.1¹⁸. Anyway, one has always to be aware of the Open World Assumption (see 2.3.2): if a certain type of bird (represented as an instance) is not explicitly said to have the capability to fly, it is not implied that it does not fly; thus, this consequence can only be derived if it is assumed that the knowledge base already contains all the needed knowledge (so in a sense it is “closed”).

Use of DL extensions

The main problems affecting DL extensions are related to their being very recent, so that they have not yet become a standard nor they have been

¹⁷To be precise, it becomes inconsistent if the classes `Spherical` and `Elongated` are declared disjoint and the property `hasShape` is functional.

¹⁸<http://www.w3.org/TR/sparql11-property-paths>

extensively studied from an implementation point of view if not for research case studies; this makes such technologies difficult to be used in real applications, for example when massive quantities of data (or TBox axioms) are used and scalability is needed. On the other hand, “traditional” approaches to description logics (and to Semantic Web languages in general such as RDF) have a long implementation history and several user-level and industrial-strength products are available.

Deriving concepts from real-world data requires anyway a careful evaluation; in fact, a completely fuzzy formulation of an ontology, including the use of fuzzy quantities for describing quantities, can cause problems when integrating different sources of information: for example, a degree of truth 0.7 for a concept like “the productivity of a researcher” can have a different meaning in two different ontologies depending on how it is calculated [Schockaert et al., 2011].

In [Vitucci et al., 2010b] we have explored the use of f -*SHIN* for part representation of images, but because of some technical limitations in the reasoner the scalability of such approach is limited. In Chap. 4 we will be discussing another application of fuzzy description logics to a smaller domain, where scalability is not a primary concern.

2.3.2 The role of reasoning

The works making use of ontologies usually state that the rationale behind the choice of a knowledge representation language is in the possibility of using a semantic reasoner *to infer some knowledge*; while this is a very broad claim, it is important to understand *what kind* of inference one expects it will be performed:

- is the ontology used as a vocabulary, so that inference is mostly related to taxonomic information (e.g. *All the cups are containers, which in turn are objects, which ...*)?
- is the ontology used to represent processes, so that inference is mostly used for dealing with sequences of events (e.g. *For grasping an object, the hand should be open first* or *To set up a table, plates have to be put on the table top before cutlery*)?
- is the ontology used to represent structures, so that inference is mostly used to find relations among parts (e.g. *A hand has five fingers* or *If a finger tip is attached to a finger and the finger is part of a hand, the finger tip is part of the hand too*)?

- is the ontology used for classification, so that inference is mostly used for retrieving an object’s class from its features (e.g. *If an object is spherical and orange, then it is a basketball* or *If in the scene there is sea and sand, it is a seaside image*)?
- is the ontology used for anything else?

Depending on the application and on the expected kind of results, different reasoning services might be needed. Furthermore, care should be taken to the reasoning services offered when selecting a semantic reasoner: for example, although both RacerPro [Haarslev and Möller, 2003] and FaCT [Horrocks, 1998] support *SHIQ* expressivity¹⁹, FaCT does not support the use of datatype properties; this means that, if the application requires to reason on concrete domains such as floating-point numbers, FaCT cannot be used. Another example is related to the two fuzzy reasoners mentioned in 2.1.3: although FiRE supports a more expressive logic than fuzzyDL, the latter provides more expressivity in the TBox axioms such as fuzzy concept assertions, fuzzy numbers and weighted sums of concepts.

As we have seen, for recognition purposes usually instances are used to represent real objects while concepts are used to represent classes of objects; thus, the most commonly needed services are ABox services. This is also due to the fact that the classification of concepts per se is not usually needed in robotics, while in other domains such as in medicine and chemistry this is actually the purpose of the use of ontologies altogether.

Classification and realization

Two of the main tasks performed by a semantic reasoner are *classification*, in which a hierarchy of concepts is built using TBox axioms, and *realization*, which consists in finding the most specific class each individual belongs to. The classification service basically builds a hierarchy of sets, deriving for every set its including and included sets; the realization service instead derives all the sets an individual is an element of. The classification task has to be always performed before a realization task can be completed as well. As an example of the use of such tasks in the robotic domain, we can describe several objects from the kitchen domain such as spoons, knives, forks, cups, mugs, etc. as arranged in the hierarchy shown in Fig 2.4. Specific objects such as *Jack’s cup* and *spoon with Mickey Mouse decoration* can be added as instances belonging respectively to the classes Cup and

¹⁹FaCT++, the new version of FaCT, now supports *SRQIQ(D)*[Tsarkov and Horrocks, 2006].

```

Utensil
  Container
    Cup
    Mug
  PieceOfCutlery
    Spoon
    Fork
    Knife

```

Figure 2.4: Example hierarchy.

Spoon; the reasoner, after the classification process, will infer that these two objects belong respectively to the **Utensil** and **Container** classes as well. The taxonomy can be extended with the definition of “exotic cups”, which are cups having some drawings of animals on them:

$$\text{ExoticCup} \equiv \text{Cup} \sqcap \exists \text{hasDrawing}.\text{AnimalDrawing}$$

Now the reasoner will infer that the class **ExoticCup** is a subclass of **Cup**, because all exotic cups are first of all cups (classification step); also, if there exists an instance such as **lionDrawing** of a drawing depicting animals, related to Jack’s cup through an axiom like **hasDrawing(Jack’sCup, lionDrawing)**, the reasoner will infer that the most specific class describing Jack’s cup is now **ExoticCup** (realization step). For this reason, if classes are described in terms of the features they have to have, the realization service can be used to find all the classes an unknown instance belongs to depending on its features.

As it can be seen from the example, in such applications it might not be very interesting to execute queries on the taxonomy, while it might be more interesting to ask to which class(es) an instance can be assigned or whether an instance can be verified to belong to a certain class.

Open World and Unique Name Assumptions

It is common, while using databases, to assume that “any statement that is not known to be true is false”; in other words a database is considered *complete*, meaning that all the relationships between entities are listed in its tables, therefore information which is not present in the database counts as false information. As an example, if a table in a database lists the books bought by a customer, it can be assumed that the customer has not

bought any other books. Such assumption is called Closed World Assumption (CWA) and it is the opposite of the Open World Assumption (OWA), which on the contrary states that the truth value of a statement does not depend on the currently available knowledge. In other words, according to the OWA, if a fact is not present in a knowledge base at a given time it is not implied that it is not true; it *might* be true or false, but this information is not known (yet). In the previous example, the list of books bought by a customer would state some facts about the customer and such books, but nothing else is implied; in particular, it might be that the customer has bought other books but the information has not been added to the knowledge base yet.

The Unique Name Assumption (UNA) instead states that “different names always refer to different entities in the world”; in other words, if we have two individuals called `ball1` and `sphericalObjectA`, as their names are different they will be referring to different entities as well. These two assumptions together provide a way to make inferences on an intrinsically distributed model of knowledge such as the Web, where information is likely to change, to be underspecified or to be redundant, by not making any assumption on what is not known.

Description logics usually do not make use of neither the CWA nor of the UNA, so one has to be careful when formalizing knowledge using this formalism. Taking the example from [Dasiopoulou and Kompatsiaris, 2010], we can have a context like the following (see Fig. 2.5): an image depicts a seaside scene (i.e. the image instance has been declared to belong to the class `SeasideImage`) with two regions and an axiom in the knowledge base states that *Seaside images contain at least one region depicting sea*. Such axiom might have different interpretations within the mentioned image, so the sea can be depicted in `region1`, in `region2`, in both the regions or in another region (not found by the image classifier); this happens because from its definition it follows that it *has* to have at least a region depicting sea, but there is no further information regarding, for example, its position. An interpretation task has therefore to take into account that a description (of an image, an object, a scene etc.) may not contain all the needed information so that interpretation can be performed by logical inference alone: an assumption similar to CWA is most probably needed. In some cases *closure axioms* can be useful for providing some level of closure to the world: for example, an axiom stating that *Seaside images contain a sea region and a sand region* do not say anything about other regions, so that a seaside image might have other kinds of regions; a closure axiom can add a limitation on the kinds of region, so the previous axiom would become *Seaside images contain a sea*

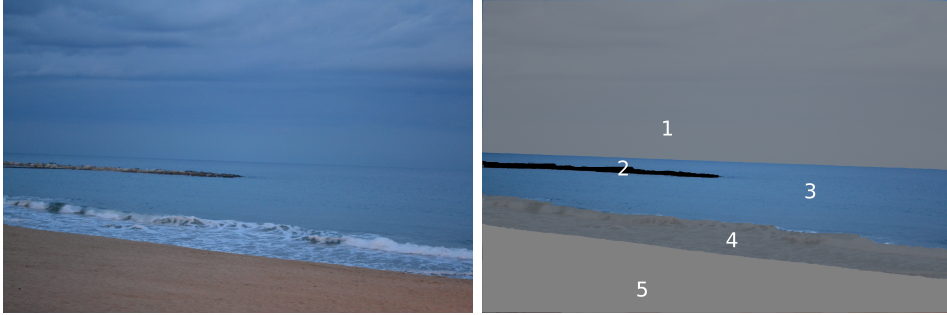


Figure 2.5: Seaside image with segmentation.

region and a sand region and only these kinds of region.

Another important issue arising from the use of the OWA comes with the use of qualified cardinality restrictions: if the image analysis module has been successful in identifying four table legs which are part of another object, it is possible to infer that such object is a table if there is an axiom stating that *Tables must have at least four legs*; if, on the contrary, the axioms states that *Tables must have at most (i.e. not more than) four legs*, or *exactly four legs*, the reasoner will not be able to infer the class of the object. In fact, the reasoner cannot conclude that the considered object *cannot have* more than four legs; in other words, the fact that there is no evidence that the object has more legs does not mean it cannot have more (so maybe another leg is not visible or has not been recognized as a leg by the image classifier). We have been dealing with such problem in [Vitucci et al., 2010b], where we used the $f - \mathcal{SHIN}$ logic to represent and recognize complex object such as forks, which are made of several parts; besides other technical limitations, the main problem is that, if the reasoner has to be used for object recognition, objects cannot be described using strict cardinality restrictions.

2.3.3 Expressivity of the logic

As we have mentioned, an important step in the process of designing a knowledge base is to evaluate the expressivity requirements; expressivity is a source of complexity for a reasoner, so several optimizations might be available and advantageous for the task to be performed at the cost of sacrificing some expressivity in favor of computational efficiency. As an example, in biomedical applications the main use of DLs is to infer terminological information from huge TBoxes where the axioms mostly make use of full existential quantification and intersection; as these

axioms are contained in the OWL 2 EL profile, ad-hoc reasoners such as Snorocket [Lawley and Bousquet, 2010] can classify such ontologies in a very little time when compared to other “standard” reasoners such as Pellet [Sirin et al., 2007] and HermiT [Shearer et al., 2008] (see [Dentler et al., 2011] for a comparison), because the worst-case computational complexity is PTIME (polynomial time) for \mathcal{EL}^{++} while being N2EXPTIME (nondeterministic double-exponential time) for OWL 2. In some cases, some inference tasks can be performed by a query engine instead of a reasoner (e.g. in the case of inverse or transitive relations, as it will be shown later), thus lifting some constraints on the description logic family to use.

Expressivity needs

Is it reasonable to try and find a minimum expressivity for a logic to be used in the robotic domain? Although it is not possible to answer to this question in general as it depends largely on the target application, it is possible to make several considerations on the use of each construct:

- the minimum logic should provide at least concept intersection (for building new concepts combining preexisting concepts) and full existential quantification (for describing concepts using restriction on features and properties and for assigning domain and range to the properties), so that axioms such as *A basketball is a ball and has an orange color* can be expressed as:

$$\text{Basketball} \equiv \text{Ball} \sqcap \exists \text{hasColor.Orange}$$

- universal restrictions are useful for providing closure axioms on the properties of an object (see 2.3.2) such as *A basketball is a ball, has an orange color and cannot have any other color than orange* can be expressed as:

$$\text{Basketball} \equiv \text{Ball} \sqcap \exists \text{hasColor.Orange} \sqcap \forall \text{hasColor.Orange}$$

- union (and, more generally, complex concept negation and concept disjointness) can be useful for expressing alternatives like *A basketball has either an orange or a red color* as:

$$\text{Basketball} \equiv \text{Ball} \sqcap \exists \text{hasColor.}(\text{Orange} \sqcup \text{Red})$$

but it adds some complexity to the reasoning, thus some OWL profiles such as OWL 2 EL disallow its use;

- a role hierarchy (i.e. a hierarchy of property with more specific properties being “children” of more general properties) is generally

useful for creating more specific roles while retaining information on more general relations; for example, it lets it possible to formalize the concepts of parts and wholes it is useful to distinguish a “generic” *part* of an object from a *proper part*, being the latter a subrole of the former with different properties²⁰, as:

$$\text{partOf_directly} \sqsubseteq \text{partOf}$$

- inverse properties can be useful to make the model clearer and intuitive by explicitly stating the “backward” relations (e.g. it can be useful to know that the inverse relation of `hasSuccessor` is `hasPredecessor`, while it might be less useful to know that the inverse of `hasColor` is `isColorOf`);
- more generally, complex role inclusion axioms (making use of role hierarchy, role chains, role intersection and so on) can be useful when n -ary relations are to be used, as the common practice is to reify such relations and then adding property chains; an example can be the axiom *A basketball has a color with a percentage of surface covered*, in which the relation among the ball, the color and the percentage of the surface covered has to be expressed as an individual (e.g. `coverage1` belonging to the class `Coverage`) related to the color and the percentage through the properties `color` and `percentage`, then the relation between the ball and these two features has to be expressed by the property chains:

$$\text{hasCoverage} \circ \text{color} \sqsubseteq \text{hasColor}$$

$$\text{hasCoverage} \circ \text{percentage} \sqsubseteq \text{hasPercentage}$$

where `hasCoverage` relates the ball to the reified relation and the properties `hasColor` and `hasPercentage` relate the ball directly with the color and its percentage (see Fig. 2.6); furthermore, role disjointness can help in distinguishing features of an object (e.g. the color of an object is different from its shape) and other cases (asymmetric, reflexive and irreflexive properties) are less common but might be useful or even necessary to draw more inferences;

- functional properties are useful for describing unique properties for an object (e.g. its shape and its length, as an object cannot have two different shapes or lengths at the same time);

²⁰This issue is considered so important that the W3C has published some best practices on the topic (see for example <http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/simple-part-whole-relations-v1.3.html>).

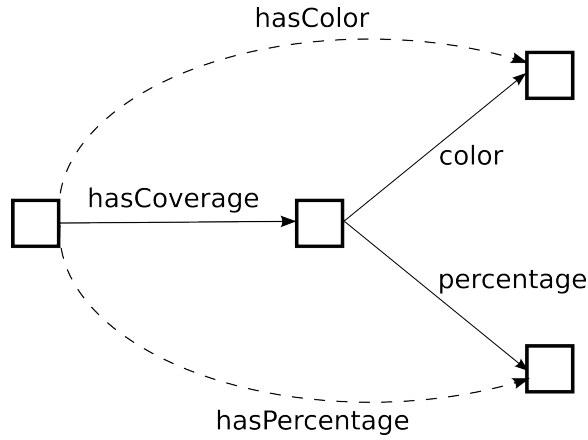


Figure 2.6: Example of property chain. The properties `hasColor` and `hasPercentage` are obtained through the chains `hasCoverage ◦ color ⊑ hasColor` and `hasCoverage ◦ percentage ⊑ hasPercentage`.

- cardinality restrictions are mostly useful when are qualified: in the case of robot parts, for example, it can be known that a robot has to have n links (this would be enough to distinguish robots having just a different *number* of links), but it would be more useful to know what *kind* each of such links has to be, so that for example in axioms such as *A hand has exactly five links which are fingers* can be formalized as:

$$\text{Hand} \sqsubseteq = 5 \text{ hasLink.Finger}$$

- concrete datatypes are important when using real numeric data on which to base the reasoning, for example size, weight, date and so on, but they are not always supported by reasoners; they can be used for axioms such as *A basketball has a circumference between 75 and 76 cm*, which can be formalized as:

$$\text{Basketball} \sqsubseteq \text{Ball} \sqcap \exists \text{hasCirc. } \geq 75.0, \leq 76.0$$

We can therefore conclude that the whole OWL 2 is not (always) needed, but rather the minimum suitable logic can be the $\mathcal{ALH}(\mathcal{D})$ logic; a good compromise between expressivity and suitability for big quantities of data would be the $\mathcal{ALCHI}(\mathcal{D})$ logic.

Domain constraints

Besides the possibilities offered by expressive logics, constraints and peculiar characteristics of the domain to model have to be taken into account:

- as robotic tasks make intensive use of data, in a realistic and scalable

application more importance should be given to operations on data rather than to reasoning on the model, which is not supposed to change frequently;

- besides building of “simple” taxonomies, almost no deductions are needed on TBox as the aim of the formalization is not to discover new concepts but rather to infer new relations among existing instances;
- logical inference alone is not enough for recognition, so either extensions using fuzzy logic or probability are to be used or additional steps have to be performed;
- building complex concepts with many constraints is either expensive (if it has to be done by hand) or less reliable (if it is performed automatically using other sources of information, especially just raw data – see for example [Vitucci et al., 2010b]).

Thus, for tasks within the robotic domain there is not a clear advantage in using highly expressive logics to model information related to changing entities such as (classes of) objects. For some applications RDF(S) can be sufficient, as it provides the basic expressivity for building taxonomies of concepts and relations and efficient storage solutions; if a fragment of OWL 2 is needed, however, solutions providing both storage and efficient reasoning can be adopted. In fact, although it is a “low-level” language, SPARQL provides several facilities (e.g. the mentioned property paths) to “move” some reasoning tasks to the query engine: for instance, it is not necessary to explicitly model a property as transitive to derive transitively related elements via query; the other advantage of using such a technique is that the number of “hops” between two entities can be decided a priori, so the inference can be limited. The same holds for inverse properties and property chains, as the same results can be obtained by modelling them in a suitable way within a SPARQL query.

2.3.4 Knowledge base management

The size of the ontology, depending on the task to perform, may play a role in the design decisions. Big ontologies with “simple” axioms, such as big taxonomies, can be reasoned upon with an in-memory reasoner such as HermiT or Pellet, while smaller ontologies making use of very expressive axioms may require hours for classification on a standard computer.

Generally speaking, anyway, ontologies making use of an expressive logic (and mostly reasoning on the TBox) need a reasoner tailored on their ex-

pressivity needs; in fact, as discussed in Sec. 2.3.3, different reasoners can give very different performances depending on the structure of an ontology. In the case of intensive ABox reasoning, as in the case in which an ontology is to be used as a knowledge base with a big amount of factual knowledge and a relatively simple schema, it is generally better to make use of a semantic repository or a database interface, thus sacrificing some of the potential expressivity. Some repositories such as OWLIM²¹ have been developed together with rule-based semantic reasoners to bridge the expressivity of a description logic with the flexibility and scalability of a triple store; for a comparison of state-of-the-art RDF stores see [Haslhofer et al., 2011].

Another key aspect to consider is the way instances are used: if the ontology is used to store information about a high number of objects for later retrieval, it can be preferable to use a persistent storage such as a database or a triple store; on the contrary, if the ontology is used for realization and instance checking so that instances are mostly created only temporarily, the schema can be stored as a single file in a suitable format such as RDF/XML.

2.3.5 Queries

The choice of a reasoner, a family of logics and a storage system influences the type of queries which can be executed on a knowledge base: the types of TBox queries which is possible to perform depend on the reasoner, while the availability of ABox queries depends on the query engine implementation. SPARQL-DL [Sirin and Parsia, 2007] aims to provide a unified framework for both the kinds of queries, but its future is still uncertain as it has not become a standard (yet).

Obviously queries depend on the type of information one needs to retrieve: finding the super- or subclasses of a certain class has a different meaning from finding the most specific concepts for an instance, finding the relations among several instances and so on. In the cases where a reasoner is geared towards TBox reasoning, ABox reasoning might lack efficiency; furthermore, not all the types of TBox queries can be performed with every reasoner. What is important to note is that query languages such as SPARQL, after the reasoning phase (which has to be performed by an external reasoner), can be used to match *currently known* data and even to “close the world”, for example by implementing negation by failure. From this point of view, assuming that a knowledge base contains all the relevant knowledge, it is possible to ask whether an object has *exactly* four legs as this would imply “as far as it is known now” (see Chap. 3 for such applications

²¹<http://owlim.ontotext.com>

of semantic query languages).

2.4 Summary

In this chapter we have reviewed the issues characterizing the design of a knowledge base, focusing our attention to robotic applications. We have discussed the choice between classes and individuals to describe the different aspects of a domain, the importance of the reasoning tasks one expects to be able to perform on the knowledge base, the expressivity of the logic one might need to use, the types of query which might be needed and some assumptions (such as the OWA and the UNA) which characterize description logics, along with some extension (such as fuzzy DLs) proposed to overcome some challenges introduced by the use of knowledge representation in real domains such as multimedia information, image understanding and scene description.

Our purpose in the following chapters is to describe some guidelines to choose a suitable knowledge representation strategy, then to show some examples of application where the aspects which have been highlighted in this chapter will be discussed.

Chapter 3

Description logics for object representation

Dealing with objects is one of the most important robotic applications; hence, the study of object detection, localization and recognition has a long tradition in the history of robotics. Several approaches have been proposed to make these tasks as robust and autonomous as possible, each of them proposing a peculiar representation of objects depending on the needed features.

Although the vast majority of the techniques are based on low-level information and make use of statistical descriptors, the part decomposition of an object for recognition and modelling purposes has regained some attention in the last decades, the reasons lying mostly in the advances in the fields of computer graphics, graph theory and artificial intelligence, and in the availability of cheaper scanners and devices for acquisition of 3D information. This “new wave” has brought to the cross-fertilization of different research fields including description logics.

The model of part decomposition has also been (and is still) considered a viable theory of human vision and object recognition, so it has been studied by the cognitive science and psychology community as well; as it will be discussed, anyway, this approach has several engineering issues which make it difficult to be used for this purpose. Nevertheless, the description of an object in terms of its composing parts has other advantages when such representation is used by humans.

Our focus in this chapter is to show the main issues related to a part-based object representation and to discuss the cases in which this approach might offer some advantages, along with some qualitative and quantitative evaluations.

3.1 State of the art

An early work on shape decomposition is presented in [Rom and Medioni, 1994], where Straight Homogeneous Generalized Cylinders (SHGCs) and Planar Right Constant Generalized Cylinders (PRCGCs) are used for describing 3D shapes; in [Svensson and di Baja, 2002] the distance transform is used for decomposition while in [Lien et al., 2006] the shape is decomposed during the process of skeletonization. In [Sukumar et al., 2006] significant parts are found through the use of a 3D feature called curvature variation measure (CVM), while in [Marras et al., 2012] and [Anguelov et al., 2004] particular attention is paid to the joints which characterize articulated objects (often more difficult to describe because of the high variability in the shape and position of their parts).

In [Horikoshi and Suzuki, 1993] and [Zhang et al., 2003] sparse range data are fitted to shapes with the use of superquadrics, while in [Ning et al., 2010] a part decomposition algorithm is applied directly on point clouds rather than on meshes. The algorithm presented in [Mozos et al., 2011] works on point clouds as well by using a different decomposition algorithm, but point clouds are actually obtained by 3D models downloaded from the Web.

To the best of our knowledge, the most comprehensive work comparing quantitatively the performance of shape decomposition algorithms is presented in [Chen et al., 2009]; the dataset is publicly available¹ and has been used in this work. Another benchmark on mesh segmentation algorithms can be found in [Shamir, 2008].

Part decomposition of images has recently regained some attention as well due to some shifts in the image description paradigm (see for example [Farhadi et al., 2009] and the works presented in the Workshop on Parts and Attributes at ECCV 2010²), where the question on whether a part-decomposition-centered representation of 2D images is actually needed (and, in case, up to which level of “semanticity”). Furthermore, works such as [Falomir et al., 2011] and [Bilodeau and Bergevin, 2003], focused respectively on qualitative and fuzzy reasoning, exploit the availability of expressive logical formalisms to provide a fine-grained part-based representation of 2D images.

Finally, several works in robotics make use of 3D part decomposition for detection [Gächter et al., 2008], recognition [Mozos et al., 2011] or manipulation purposes [Aleotti and Caselli, 2012].

¹<http://segeval.cs.princeton.edu>

²<http://rogerioferis.com/PartsAndAttributes>

3.1.1 Recognition-by-components

One of the most important tasks for an autonomous robot is *object recognition* whose aim is, given an object, to assign it to a known category; the problem of how humans recognize objects, anyway, is still largely unsolved. Part decomposition has played an important role in the theories of human vision: the seminal works by Biederman [Biederman, 1987] and Marr [Marr and Nishihara, 1978], for example, provide a theory which makes respectively use of the so-called *geons*, tridimensional shapes which are considered the units of vision (see Fig. 3.1), and of a structural representation of a shape (see Fig. 3.2). This theory has been thoroughly analyzed in works such as [Dickinson et al., 1997] and for a long time it has been considered useful and computationally viable; in [Edelman, 1997] and [Hummel, 2000], however, a critical analysis of the main models of recognition highlighted some problems in the RBC theory such as:

- the lack of metric information, in that geons can provide *shapes* but not *measures*;
- the computational difficulty in the process of recovering parts from a real scene, in that complex fitting models should be applied to all the perceived objects;
- the instability of a description in terms of parts, in that several representations of the same object are possible: for instance, the level of detail up to which an object like a car should be decomposed – and the composing parts – are difficult to define unambiguously.

3.1.2 Shape description and modelling

Besides the limitations on the definition of “parts” highlighted in the previous section, the part decomposition approach found several applications in shape description and modelling. An example of such applications is given by the ShapeAnnotator software³ presented in [Attene et al., 2007], where the parts characterizing an object can be found semi-automatically (i.e. the user has to choose a mesh segmentation algorithm and provide it some parameters such as the expected number of parts to find) and then annotated using a suitable ontology (e.g. the ontology of human bodies, in which specific concepts such as *Torso*, *Hand*, *Leg* and relations such as *attached-to* or *length* are provided). In this case, as it will be discussed later, a semantic description of such parts seems appropriate.

³<http://shapeannotator.sourceforge.net>

CROSS SECTION







Geon	Edge	Symmetry	Size	Axis
	Straight S Curved C	Rot & Ref ++ Ref + Asymm -	Constant ++ Expanded - Exp & Cont --	Straight + Curved -
	S	++	++	+
	C	++	++	+
	S	+	-	+
	S	++	+	-
	C	++	-	+
	S	+	+	+

Figure 3.1: Biederman's geons [Biederman, 1987].

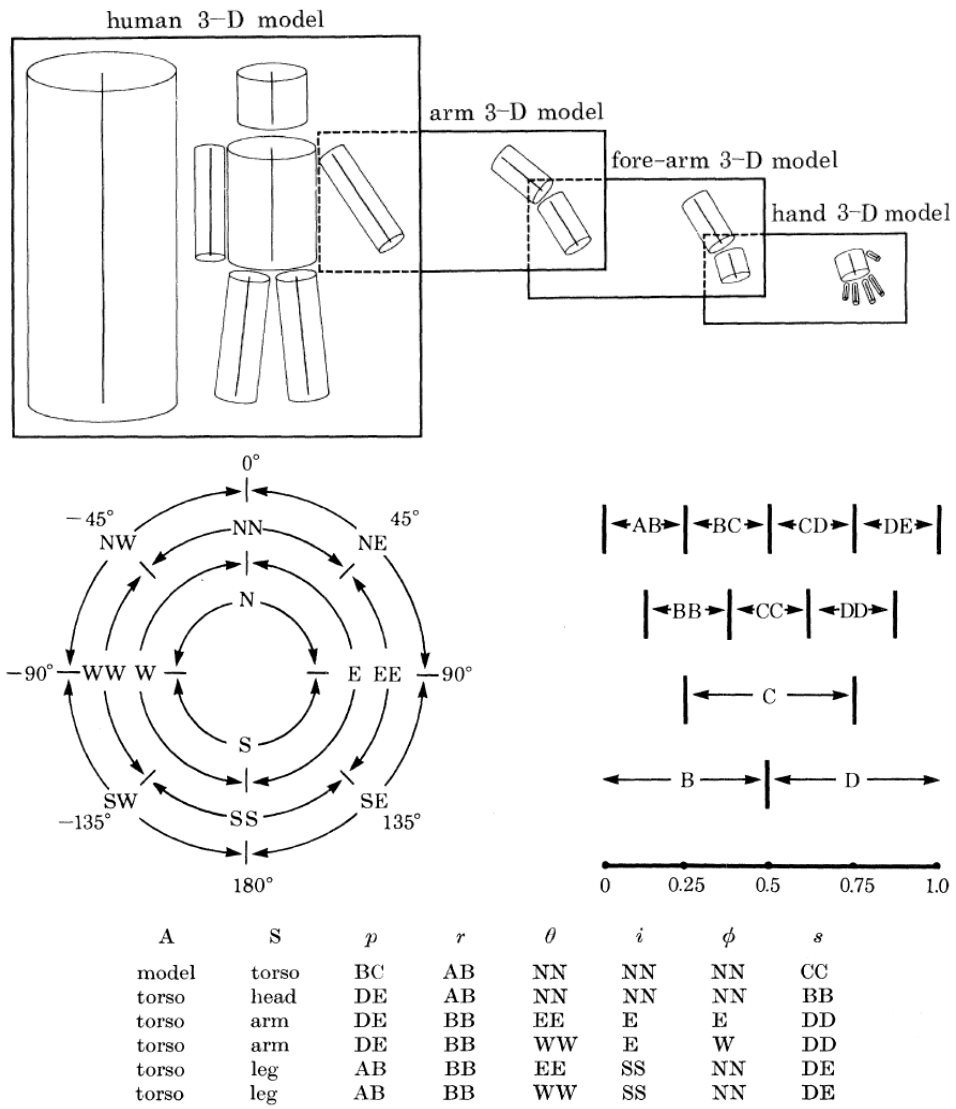


Figure 3.2: Marr's structural representation [Marr and Nishihara, 1978].

Several projects such as AIM@SHAPE⁴, FocusK3D⁵ and the Princeton 3D Model Search Engine⁶ have been dealing with the problem of 3D models description; the first two are more concerned with the shape decomposition and the semantic description of the parts, while the aim of the third one is to evaluate the performance of different approaches to shape description for retrieval purposes. The AIM@SHAPE project released a dataset containing *watertight meshes* (this is often a strict requirement for shape decomposition algorithms) which has been used in [Chen et al., 2009] for comparing mesh segmentation algorithms; other datasets such as Semantic3D⁷ have been obtained from real-world measurements via scanners or 3D cameras.

Graph-based representations have gained some popularity as they are invariant to scale and pose (so do not require any normalization), can be queried for partial matches and are suitable for semantic descriptions; the nodes of a graph can represent functional parts, shape primitives or faces, while the edges can represent adjacency relations (among functional parts or shapes) or boundary lines. Graphs can also be obtained directly from shape skeletons or via a quotient function (Reeb graphs), so they represent the parts of an object and their connections. Several works using graph-based representations can be found in [Natali et al., 2011], [Aleotti and Caselli, 2012], [Tung and Schmitt, 2004] and [Attene and Biasotti, 2011], and more in general in [Marini et al., 2007] structural descriptors and their performances are discussed.

An important application of shape description can be found in *content-based retrieval*, that is the retrieval of 3D models based on their shape features rather than on tags and external descriptions. A survey of content-based 3D shape retrieval models, has been conducted in [Tangelder and Veltkamp, 2007], and in [Wagan et al., 2009] part decomposition is used for shape retrieval based on decomposition similarity.

3.1.3 Use of knowledge

The use of knowledge together with 3D objects has been reintroduced recently. With the improvements in knowledge management systems and the availability of more powerful and expressive tools driven by the research on knowledge representation, the inclusion of semantic technologies as an integration framework is gaining more and more attention.

⁴<http://www.aimatshape.net>

⁵<http://www.focusk3d.eu>

⁶<http://shape.cs.princeton.edu>

⁷<http://ias.in.tum.de/software/semantic-3d>

Several works such as [Attene et al., 2007, Catalano et al., 2011, Vasilakis et al., 2010, Attene et al., 2009] and the cited ShapeAnnotator software, for example, explore the use of ontologies and description logics in the domain of 3D object modelling and description for a variety of purposes: specific domain ontologies are linked to the semi-automatic segmentation of 3D shapes, integrating information related to the object characteristics with (meta)information related to the acquisition devices used, the target application and so on; the specific segmentation algorithms to use is not a crucial choice, because such tools have been built for human users and the automatic segmentation of a shape is meant only as a guidance. The applications for this approach are mostly in the fields of 3D models lifecycle description and maintenance and virtual molecule modelling.

Other works such as [Horrocks and Graves, 2008] and [Graves, 2008] provide some insights on the use of OWL for engineering and design purposes, while [Dartigues et al., 2007] explores the use of feature ontologies related to computer-aided design (CAD) and computer-aided process planning (CAPP), where semantic knowledge is used for checking the constraints on the product design. A description of object affordances along with component parts for robotic applications is mentioned in [Varadarajan and Vincze, 2011], although the formalization does not strictly use semantic technologies but rather attempts to give some semantics to a database representation.

In the field of shape retrieval external knowledge is used to support “high level” queries, namely on attributes and parts: such knowledge bases should be able to answer to queries like *Find the objects having a big head and two legs* and so on. Studies like [Zhang et al., 2012] and [Kalogerakis et al., 2010] aim to efficiently produce and transfer semantic labels on parts from one model to another, thus making it possible to obtain a large database of tagged models not having to resort to manual segmentation; the specific segmentation method is not important as well. In [Kassimi and beqqali, 2012] several semantic descriptors are identified to be used within an ontology and for SPARQL queries, although no examples of queries are shown; in [van Kaick et al., 2011] the focus is on the problem of part correspondence, using knowledge to help the matching process.

3.2 Issues in part decomposition

3.2.1 Conceptual issues

As we have seen in the previous sections, the problems that have to be dealt with when formalizing an object in terms of its composing parts are:

- What is a part?
- Does it make sense to define “semantic” or “functional” parts?
- Are they useful?
- How are they linked to the “low level”?

Some of these questions have been addressed for 2D images by Bernt Schiele in an ECCV Workshop⁸. More in general, some critiques have been brought on the use of “semantic features” in general, especially on whether they can actually be learned by classical algorithms in an unambiguous way (see for example [Farhadi et al., 2009]). As we discuss in Chap. 5, the identification of concepts and discriminative yet expressive semantic features is still an active research topic in the fields of cognitive science, computer vision and pattern recognition; in fact, it is difficult to predict what kind of features humans would consider more important: if language is to be used as a reference, features need to be linguistically expressive in order for communication to happen. In this case, vagueness (related to the given description) and uncertainty (related to the experienced perception) play an important role: a human would describe an object as “red”, “round” or “big”, or would say that a table has two, three or four legs depending on how many of them he is able to see.

3.2.2 Practical issues

Besides conceptual issues, we can divide the practical issues regarding a part decomposition framework in two categories, namely problems related to the mesh decomposition process itself and problems related to the use of knowledge representation formalisms in real-world domains.

Some of the issues related to description logics and image interpretation have been discussed in [Dasiopoulou and Kompatsiaris, 2010]; although the domain is different, the remarks hold also for 3D object representation:

⁸<http://rogerioferis.com/PartsAndAttributes/pages/material/SchielePnA2010.pdf>

- ambiguity due to incomplete and/or conflicting assertions: if classifiers give as a result that an object belongs to two different classes and such classes are disjoint, the KB will become inconsistent – thus this information cannot be used;
- imprecision due to degrees of uncertainty or truth: when dealing with real-world data it is quite unlikely for a classifier to state that an object *definitely* belongs to a certain class, while on the contrary this information has to be interpreted both as a probabilistic information (it is not sure whether the object belongs to the class or not) and as a fuzzy information (the object cannot be considered as a “full” instance of a class, but rather it has a degree of membership to such class);
- semantics of computational perception “per se”: as we have already discussed there are several ways in which concepts can be built from numerical data, depending on the modelling choices and the features to consider (there are many possibilities to link low and high levels, and different agents have different perceptions);
- the use of an open/closed domain model (or OWA/CWA in the DL domain): as we have seen in Chapter 2, the use of OWA or CWA makes reasoning different and makes certain queries impossible to be expressed or to give the expected results.

These considerations, together with results from the use of description logics for scene interpretation [Neumann and Möller, 2008], bring to the light the intrinsic difficulty of an interpretation task (solely) as a logical inference:

“Unless all regions correspond to distinct objects (or parts) and all classifications are accurate, the explicitly asserted data comprise an incomplete, partial only view of the actual image content” [Dasiopoulou and Kompatsiaris, 2010, p. 3].

Even in this case, we would still have some degree of ambiguity due to incomplete or contradictory information provided by different persons, exactly as in the case of part decomposition. It is then clear that purely deductive reasoning is not enough, and some advantage might be taken by using rules and by handling vagueness by means of fuzzy and probabilistic extensions; in this case, other problems related to the availability of a suitable framework and its scalability might rise.

Within data-intensive scenarios, scalability has to be taken into account as a serious issue:

“A second issue, which has been raised in the past few years and is relevant in the context of 3D semantic media, is the impossibility of description logics to find partial solutions to a query since description-logic-based inference tries to find all the solutions of a search. Going to the Web scale where the amount of data is now massive, this approach does not work anymore: it is preferable to retrieve partial data in a few seconds instead of all data (or even failure) in days” [Catalano et al., 2010, p. 79].

Several works focus on such aspects, in particular on the need of partial and approximate queries for increasing responsiveness sacrificing some exactness [Tran et al., 2011] and on the possibility to formulate preference queries making use of similarity and concepts such as “usually” [Wang and Pan, 2007].

3.3 Implementation

Although the part decomposition strategy seems neither scalable nor reliable to be used in a completely automatic fashion, it is anyway possible to evaluate its performances within several domains. The work presented in this chapter focuses on the following aspects:

- semantic description of objects obtained from decomposed models (see Fig. 3.3);
- full and partial instance retrieval, query optimization;
- approximate classification.

The aim of our work is to find out whether and how it is possible to represent a 3D object in terms of its parts by using description logics and to evaluate this approach within retrieval and classification tasks. The choice of semantic languages should thus look quite clear as they offer a natural way to express structural knowledge; taking into account the limitations in the structural description of object with class restrictions, we propose here a representation and retrieval framework based on instances and structural queries. The byproduct of this analysis is the realization of a knowledge base containing information on shape decomposition and geometry that is easily extendable with other sources of information (e.g. semantic labelling of parts with labels such as *Leg*, *Head* and so on, inclusion of information on the real scale of objects and their typical usage etc.).

The aspects we evaluate are:

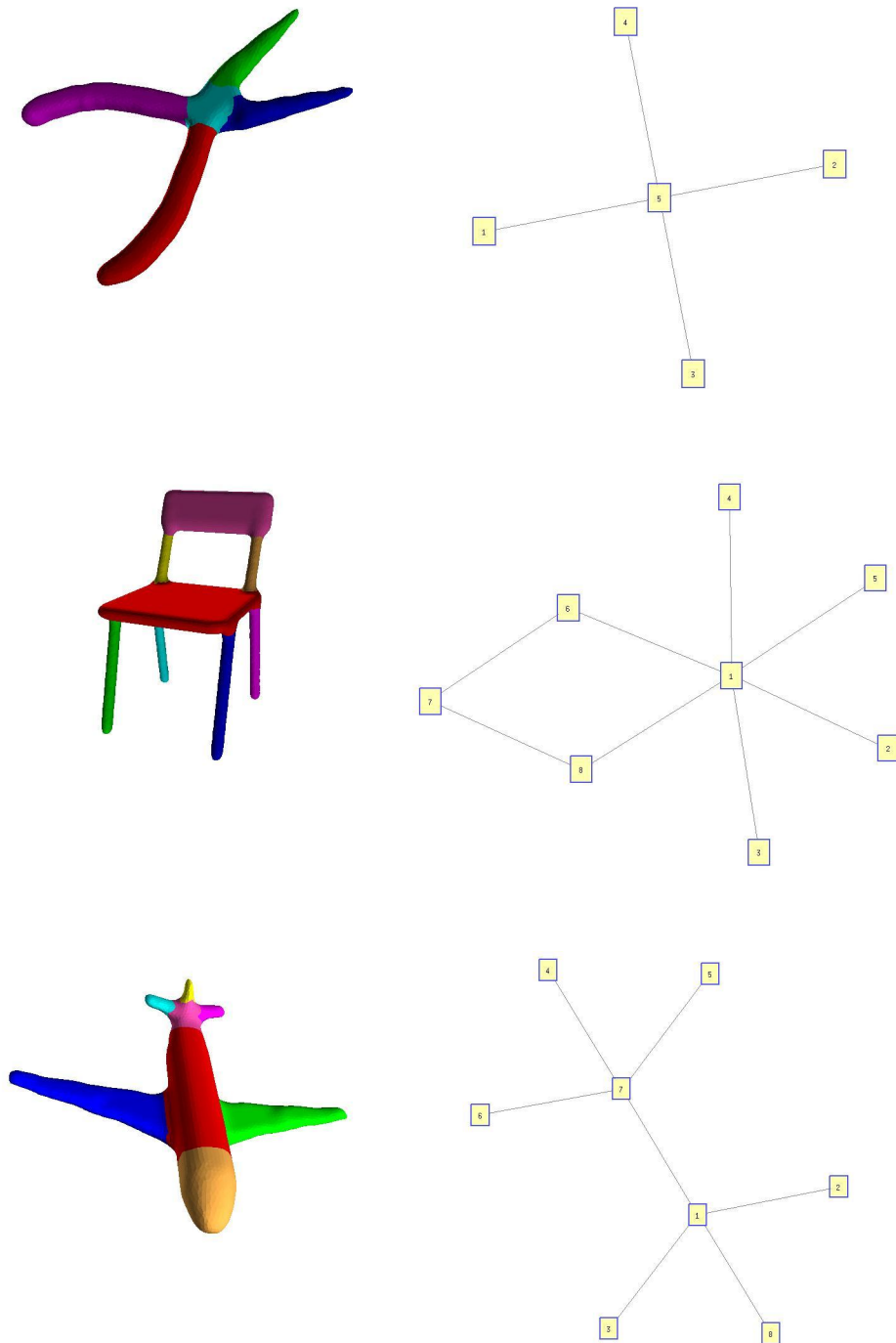


Figure 3.3: Objects with their topological graphs.

- the level of expressivity it is possible to achieve while describing an object by its composing parts and their characteristics;
- the scalability of the approach, i.e. the performances that can be achieved when the knowledge base grows;
- the possibility of writing “simple” yet useful queries;
- the discriminativity of the representation, related for example to the possible ambiguity of an only-topological representation.

3.3.1 Datasets

For our experiments we used the dataset presented in [Chen et al., 2009] and shown in Fig. 3.4. The dataset contains 380 objects arranged in 19 categories, so for example in the category *Human* there are shapes representing a human figure in different positions, while in the category *Vase* several kinds of vases (even very different from each other geometrically and topologically) are grouped together. Each object has been segmented both by humans and by the shape decomposition algorithms.

The result of the comparison among part decomposition algorithms which has been conducted in [Chen et al., 2009] is that none of the algorithms offers a generally better performance with respect to the others; although it would be possible to select the “right” segmentation method for each class using such results, we consider this to be irrelevant as it would be anyway limited to the objects present in the database; furthermore, manual segmentation provides the ground truth for part decomposition tasks, therefore we decided to build our ontology from human segmentations. This is not a restricting assumption, anyway, as crowdsourced labelling has already been used (see e.g. the LabelMe project⁹) and the lack of labelled datasets in this domain is one of the reasons why such methods have not been studied extensively yet.

As several human segmentations are available for each object, as a criterion for the choice of the number of parts to represent a single instance we take the mode of the number of segmentations (i.e. the most often performed segmentation). For each part composing the object we calculate:

- the volume of the convex hull for the whole object;
- the volume and size of the axis-aligned minimum bounding box (AABB) for the whole object and each of its parts (see Fig. 3.5);

⁹<http://labelme.csail.mit.edu>

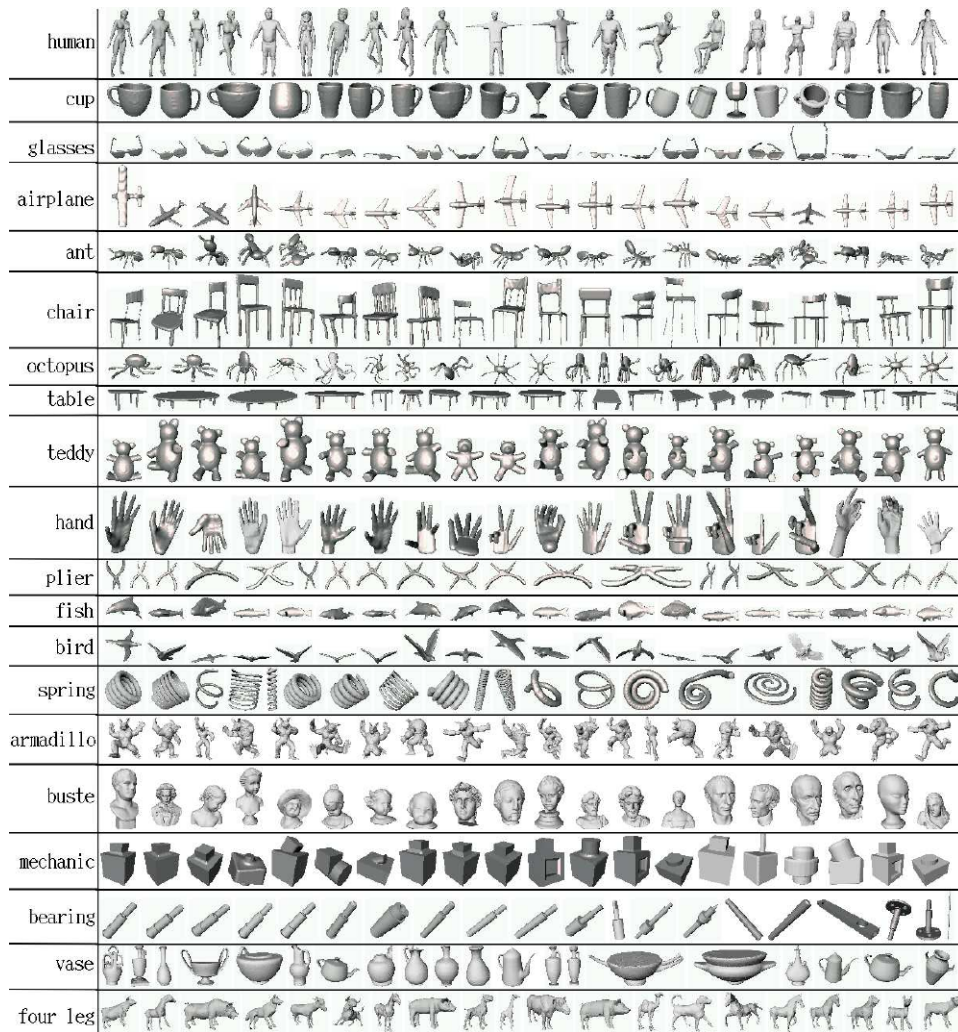


Figure 3.4: Shapes from the AIM@SHAPE dataset (see [Giorgi et al., 2007] for more information).

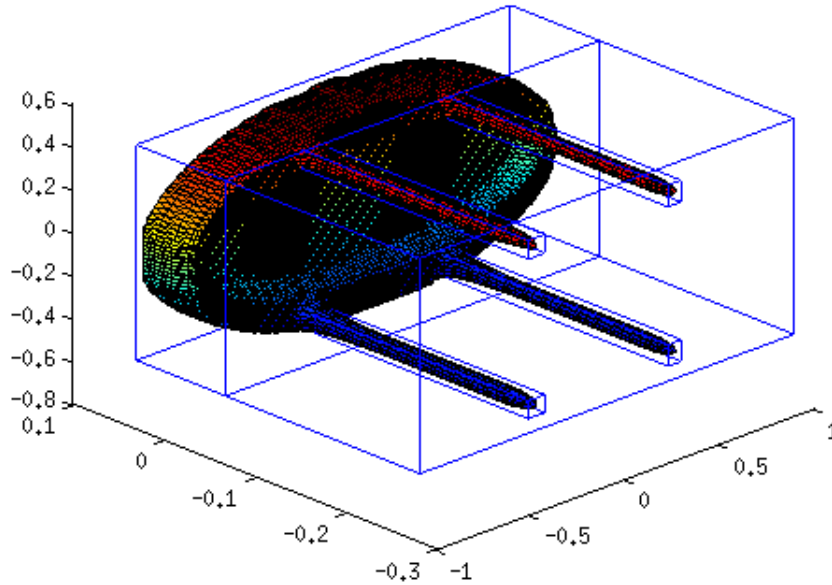


Figure 3.5: Example of an object with all the bounding boxes.

- the volume and size of the minimum volume enclosing ellipsoid (MVEE) for the whole object and for each of its parts (see Fig. 3.6);
- the eccentricities for each ellipsoid;
- the distance between each pair of bounding boxes and ellipsoids;
- the angle between each pair of ellipsoids as the angle between their main axes;
- the relative volume, length, width and height for each of the parts with respect to the whole object, using both the bounding boxes and the enclosing ellipsoids.

The advantage in using MVEEs is that they usually fit objects better than bounding boxes (especially in the case of curved objects) and do not need to be aligned to the axes, as they provide directions for their axis which can be compared to calculate their degree of parallelness; the drawback of this approach is that ellipsoids require computationally intensive calculations and tend to overestimate the dimensions. Anyway, the ontology can easily be extended to include other criteria and measures.

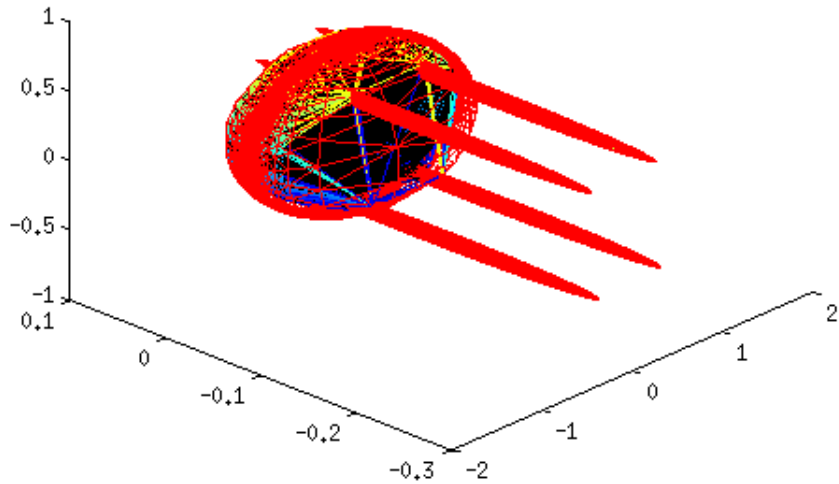


Figure 3.6: Example of an object with all the bounding ellipsoids.

Similarly to [Goldfeder et al., 2009], we added information about the real scale to several objects; such information was retrieved via the use of e-commerce websites where the characteristics of different kinds of the same object were specified (for example, different sizes of tools such as wrenches and knives have been obtained by general¹⁰ or specialized¹¹ websites). All the calculations are performed in Matlab, using the Ellipsoidal Toolbox [Kurzhanziy and Varaiya, 2006] for ellipsoid-related operations such as the computation of MVEEs.

In our ontologies we described ten objects per class in order to have a representative sample of each class without having too many instances per class. Each object class is represented as a concept in the DL notation (e.g. `Human`), while each segmentation is represented as an instance such as `human1` belonging to this class; the parts related to the selected segmentation are represented as instances such as `p1-1` belonging to the class `Part` and related to the segmentation through a `isPartOf` relation. As it can be noted, there are no “high level” labels for the parts such as `Leg`, `Finger` and so on: the reason is that for obtaining such knowledge a step of annotation

¹⁰www.snapon.com

¹¹www.knifecenter.com

has to be performed before, while geometric and structural knowledge can be derived automatically from the shape analysis step. By representing the part segmentation of an object as a set of ABox axioms, the ontology does not suffer of the limitations mentioned in Chap. 2; on the other hand, recognition via instance check and realization cannot be performed, so we perform recognition on a similarity basis via SPARQL queries.

In addition to the models taken from the dataset, we include also the models taken from the Princeton Shape Benchmark¹². The dataset contains 1,814 models taken from various sources along with some metadata (e.g. their format, bounding box, principle axes etc.) and different levels of classification to describe the classes and their members. Although there is no information on part decomposition, we added more objects not only for they contribute to the size of the ontology but also because some information (e.g. on bounding boxes) can be used anyway. For each of these objects we calculated the MVEE as well.

3.3.2 Knowledge base

We decided to implement two different ontologies, derived from the same data but using two different representations. As the data related to the features obtained from the images are numerical, there are several possibilities for representing them in an ontology:

1. creating numeric datatypes and associating them to the instances the measures are related to (e.g. to say that the object `ball1` has a volume of 3.5, the datatype `hasVolumeMeasure` with range `double` can be created and associated to `ball1`);
2. creating annotations associated to the instances: in the example before, `hasVolume` would be an annotation associated to the instance of `ball1` rather than a datatype;
3. discretizing the measures using instances such as `smallVolume` or `bigVolume` and associating them to an individual via an object property such as `hasVolume`.

The first solution, which is the more natural, has been used for the first ontology; the problem affecting such solution is that relations such as the angle between two objects cannot have an associated value: in this case it is necessary to *reify* such relations, that is to convert them into instances

¹²The dataset can be downloaded from <http://shape.cs.princeton.edu/benchmark>

associated to their elements (via object properties) and to their degree (as a datatype property). In the example above, the axiom *Objects 1 and 2 form an angle of 75°* would be expressed as an instance `angle1` of class `Relation` (and possibly of its subclass `Angle`) related to `obj1` and `obj2` via an object property `hasElem` and having a datatype property `hasValue` whose value is 75.

The second solution, which is adopted by extensions of description logics for “custom” reasoning (e.g. for using fuzzy operators), can be used to assign a value to properties too; the problem is that reasoners do not reason over annotations, and SPARQL queries over annotations are anyway more complex.

The third solution has been used for the second ontology; although it simplifies the ontology in that numeric values and relation reification are no longer needed, the main problem is the method used for discretization in that it can be defined in different ways; furthermore, the discretization is “crisp”, so values differing by a relatively small amount can be assigned to two different classes. In this case a fuzzy implementation would be useful, but as our aim is to obtain scalable solution we will adopt a standard language.

In the discretized version these numerical quantities are converted to labels using these criteria:

- the volume, length, width and height percentages are given a label: `VeryLow` (0%-20%), `Low` (20%-40%), `Medium` (40%-60%), `High` (60%-80%), `VeryHigh` (80%-100%);
- the distances are assigned the property `connectedTo` if their bounding boxes intersect;
- the distances are assigned the property `intersecting` if their ellipsoids intersect (e.g. their distance is 0);
- the angles are given a label depending on the scalar product between the main axes of their enclosing ellipsoids: `notParallel` (0-0.2), `slightlyParallel` (0.2-0.4), `fairlyParallel` (0.4-0.6), `highlyParallel` (0.6-0.8), `definitelyParallel` (0.8-1.0).

3.4 Experiments

3.4.1 Examples of queries

In the definition of the queries we have adopted a “weak” form of CWA, in that we assume that the knowledge for each segmentation example is

complete; thus, if an example of segmentation of an airplane has six parts, it is assumed that it cannot have more. This lets us use cardinality restrictions as SPARQL filters as in Fig. 3.7.

Another advantage of using SPARQL is that it makes it easy to formulate queries to support some kind of inexact graph matching by increasing or decreasing the level of detail of the queries themselves; for example it is easy to create a query making use only of the topology of an object, then extend it with further constraints on the geometry of the parts or on their relative position (see query example n. 5). In the following we will show some use cases along with examples of queries on both the ontologies; in the queries, the PREFIX part is always the following:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

For the first ontology the additional prefix is:

```
PREFIX : <http://www.semanticweb.org/.../ObjectOntology2#>
```

while for the second one it is:

```
PREFIX : <http://www.semanticweb.org/.../ObjectOntology_discr#>
```

An example of SPARQL query will be shown for each kind of query, showing also the difference between the numerical and discretized formulation (not in all the cases for space reasons).

Query 1: Objects having at least n parts This is a generic query (Fig. 3.7) which is to be mostly used as a subquery to limit the results of a generic structural query. In fact, when formulating a query to find objects with a matching topological structure, it is more convenient to look first among objects having the same number of parts: in this way the query becomes a problem of graph isomorphism matching, a subclass of the more complex subgraph isomorphism problem. The performance of the execution of this query as n grows is reported in Tab. 3.1.

```

PREFIX [...]
SELECT DISTINCT ?q (COUNT(DISTINCT ?r) AS ?count)
WHERE {?q :hasPart ?r}
GROUP BY ?q
HAVING (?count >= n)

```

Figure 3.7: Query 1: Objects having at least n parts.

<pre> PREFIX [...] SELECT DISTINCT ?q WHERE { ?q rdf:type :MSB_Part . ?q :hasRelativeBE_Length ?l . ?q :hasRelativeBE_Width ?w . ?q :hasRelativeBE_Height ?h . FILTER (?l >= 0.6 && ?l < 0.8 && ?w >= 0.2 && ?w < 0.4 && ?h >= 0.2 && ?h < 0.4) } </pre>	<pre> PREFIX [...] SELECT DISTINCT ?q WHERE { ?q rdf:type :HighLength . ?q rdf:type :LowWidth . ?q rdf:type :LowHeight . } </pre>
--	---

Figure 3.8: Query 2: Oblong parts.

Query 2: Oblong parts This query can be formulated in different ways depending on one’s concept of “oblong” (an example is shown in Fig. 3.8, where an object is considered oblong if it has a high relative length and low relative width and height). Queries on the geometric aspect of objects or, more specifically, of their composing parts are useful in contexts like manipulation, in which objects to be grasped with the available hand might need a specific shape.

Query 3: Objects having one part connected to a big part This kind of query (Fig. 3.9), where topological constraints are used together with geometric constraints, is useful not only for discriminating similar objects but also for manipulation tasks in which an object is made of a part which cannot be grasped, so that the robot has to find an alternative graspable part. A “big part” here means a part having a high relative volume with respect to the volume of the whole object.

```

PREFIX [...]
SELECT DISTINCT ?q
WHERE {
  ?q :hasPart ?p1 .
  ?q :hasPart ?p2 .
  ?p1 :isElemOf ?r .
  ?p2 :isElemOf ?r .
  ?r rdf:type :MSB_Rel .
  ?r :hasEllDist ?c .
  ?p2 :hasRelativeBE_Volume ?v .
  FILTER (?p1 != ?p2 &&
    ?c = 0 && ?v >= 0.8)
}

```

```

PREFIX [...]
SELECT DISTINCT ?q
WHERE {
  ?q :hasPart ?p1 .
  ?p1 :connectedTo ?p2 .
  ?p2 rdf:type :VeryHighVolume
}

```

Figure 3.9: Query 3: Objects having one part connected to a big part.

```

PREFIX [...]
SELECT DISTINCT ?obj ?q (COUNT(DISTINCT ?r) AS ?count)
WHERE {
  ?obj :hasPart ?q .
  ?q :connectedTo ?r
}
GROUP BY ?obj ?q
HAVING (?count >= 6)

```

Figure 3.10: Query 4: Objects with parts connected to at least other 6 parts.

Query 4: Objects with parts connected to at least other n parts

This kind of query (Fig. 3.10), where only topological constraints appear, can be used for retrieving objects having a specific structure.

Query 5: Objects with the same topology In this example, the first query (Fig. 3.12) is used for retrieving objects having a specific topological structure; as several objects of different classes are found (an example of two objects having the same specified topology is shown in Fig. 3.11), the query is extended (Fig. 3.13) to include geometric constraints and limit the search further.

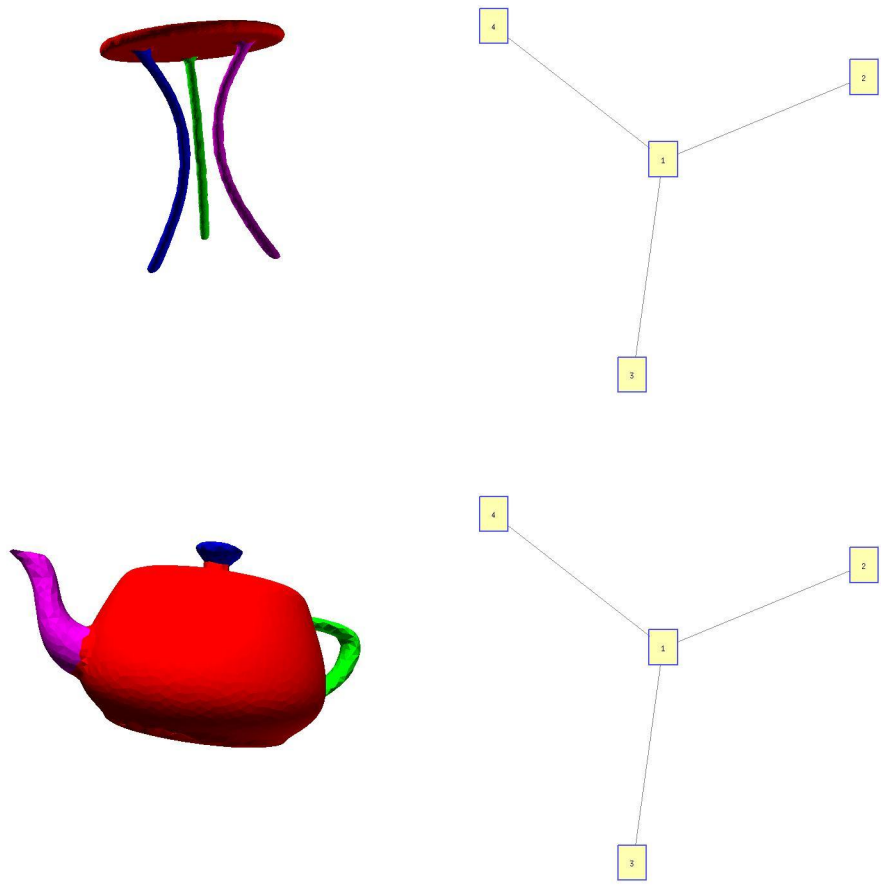


Figure 3.11: Objects having the same topological graph.

```

PREFIX [...]
SELECT DISTINCT ?q
WHERE {
  {
    SELECT DISTINCT ?q (COUNT(DISTINCT ?r) AS ?count)
    WHERE {?q :hasPart ?r}
    GROUP BY ?q
    HAVING (?count = 4)
  }

  ?q :hasPart ?p1 .
  ?q :hasPart ?p2 .
  ?q :hasPart ?p3 .
  ?q :hasPart ?p4 .
  ?p1 :isElemOf ?r1 .
  ?p2 :isElemOf ?r1 .
  ?p1 :isElemOf ?r2 .
  ?p3 :isElemOf ?r2 .
  ?p1 :isElemOf ?r3 .
  ?p4 :isElemOf ?r3 .
  ?r1 rdf:type :MSB_Rel .
  ?r2 rdf:type :MSB_Rel .
  ?r3 rdf:type :MSB_Rel .
  ?r1 :hasElldist ?c1 .
  ?r2 :hasElldist ?c2 .
  ?r3 :hasElldist ?c3 .
  FILTER (?p1 != ?p2 && ?p1 != ?p3 && ?p1 != ?p4 &&
    ?p2 != ?p3 && ?p2 != ?p4 &&
    ?p3 != ?p4 &&
    ?c1 = 0 && ?c2 = 0 && ?c3 = 0)
}

```

Figure 3.12: Query 5a: Objects with the same topology.


```

PREFIX [...]
SELECT DISTINCT ?q
WHERE {
  {
    SELECT DISTINCT ?q (COUNT(DISTINCT ?r) AS ?count)
    WHERE {?q :hasPart ?r}
    GROUP BY ?q
    HAVING (?count = 4)
  }

  ?q :hasPart ?p1 .
  ?q :hasPart ?p2 .
  ?q :hasPart ?p3 .
  ?q :hasPart ?p4 .
  ?p1 :isElemOf ?r1 .
  ?p2 :isElemOf ?r1 .
  ?p1 :isElemOf ?r2 .
  ?p3 :isElemOf ?r2 .
  ?p1 :isElemOf ?r3 .
  ?p4 :isElemOf ?r3 .
  ?r1 rdf:type :MSB_Rel .
  ?r2 rdf:type :MSB_Rel .
  ?r3 rdf:type :MSB_Rel .
  ?r1 :hasEllDist ?c1 .
  ?r2 :hasEllDist ?c2 .
  ?r3 :hasEllDist ?c3 .
  ?r1 :hasPar ?a1 .
  ?r2 :hasPar ?a2 .
  ?r3 :hasPar ?a3 .
  FILTER (?p1 != ?p2 && ?p1 != ?p3 && ?p1 != ?p4 &&
    ?p2 != ?p3 && ?p2 != ?p4 &&
    ?p3 != ?p4 &&
    ?c1 = 0 && ?c2 = 0 && ?c3 = 0 &&
    ?a1 <= 0.1 && ?a2 <= 0.1 && ?a3 <= 0.1)
}

```

Figure 3.13: Query 5b: Objects with the same topology and different geometric constraints.

Query 6: Categorization. In this example, a query (Fig. 3.14) is used to categorize an object by searching for the class containing the highest number of examples having the same characteristics (or a superset of them); in this sense a recognition by similarity (similar in principle to a k -nearest neighbor classification method) is performed. The obtained results are:

- MSB_Plier: 10 instances;
- MSB_Table: 8 instances;
- MSB_Bird: 7 instances;
- MSB_Bust: 1 instance;
- MSB_Octopus: 1 instance;
- MSB_Vase: 1 instance.

Thus, the object is recognized as belonging to the MSB_Plier class.

3.4.2 Evaluation

The experiments were performed on an Intel® Core™i7 2.00 GHz with 8 GB of RAM running the GNU/Linux operating system.

In the Table 3.1 the results for query #1 are reported; as they are almost identical across the two versions of the ontology, the reasons being that numerical or discretized quantities are not involved and that the `hasPart` relation is the same, they are shown only once. In the Table 3.2, the results of the execution of the queries described in the previous section are reported.

As it can be noted from the table, the first ontology needs a much longer time for queries (up to a factor of 500 more) with respect to the second ontology. Such behaviour shows that the use of datatypes and queries on numerical values does not scale well; this is due to the fact that filtering on numerical values is computationally expensive, thus numerical values should be partitioned for performance reasons whenever possible.

3.5 Discussion

The advantages in the use of a semantic approach for representing objects by part decomposition are mainly the ability to perform selective queries at the desired level of granularity, giving the possibility to query for partial matches, and to retrieve the most likely class of an object by counting the similar examples for each class, while retaining the ability to overcome the

```

PREFIX [...]
SELECT DISTINCT ?q
WHERE {
  {
    SELECT DISTINCT ?q (COUNT(DISTINCT ?r) AS ?count)
    WHERE {?q :hasPart ?r}
    GROUP BY ?q
    HAVING (?count = 5)
  }
  ?q rdf:type ?t .
  ?t rdfs:subClassOf :MSB_Object .
  ?q :hasPart ?p1 .
  ?q :hasPart ?p2 .
  ?q :hasPart ?p3 .
  ?q :hasPart ?p4 .
  ?q :hasPart ?p5 .
  ?p1 :isElemOf ?r1 .
  ?p1 :isElemOf ?r2 .
  ?p1 :isElemOf ?r3 .
  ?p1 :isElemOf ?r4 .
  ?p2 :isElemOf ?r1 .
  ?p3 :isElemOf ?r2 .
  ?p4 :isElemOf ?r3 .
  ?p5 :isElemOf ?r4 .
  ?r1 :hasBBconn ?c1 .
  ?r2 :hasBBconn ?c2 .
  ?r3 :hasBBconn ?c3 .
  ?r4 :hasBBconn ?c4 .
  FILTER (?p1 != ?p2 && ?p1 != ?p3 && ?p1 != ?p4 && ?p1 != ?p5 &&
    ?p2 != ?p3 && ?p2 != ?p4 && ?p2 != ?p5 &&
    ?p3 != ?p4 && ?p3 != ?p5 &&
    ?p4 != ?p5 &&
    ?c1 = 1 && ?c2 = 1 && ?c3 = 1 && ?c4 = 1)
}

```

Figure 3.14: Query 6: Class retrieval using number of matching examples.

n	Exec. time (ms)	# results
2	104	20
3	60	16
4	46	13
5	34	30
6	21	30
7	12	7
8	10	20
9	8	17
10	7	10
11	8	13
12	5	1
13	4	0
14	4	0
15	4	2
16	4	2
17	4	2
18	4	6
19	4	1

Table 3.1: Results for objects having n parts with varying n .

#	Exec. time numerical (ms)	Exec. time discretized (ms)	# results
1	100	72	54
2	257	32	12
3	2070	137	18
4	2597	111	70
5	75381	2297	17
6	197244	412	7

Table 3.2: Results of the example queries.

ambiguity caused by a topology-only graph representation. The semantic representation provides an explicit formalization of the used concepts and makes it extendable with other features and descriptors; although it would be possible to use standard subgraph matching algorithms directly on the model parts, this advantage would be lost. We have not used any semantic labels on parts such as Leg or Wing, the reason being that we were interested in a perception-like task which can be possibly supported by fully automated algorithms, while semantic labelling instead involves a further (and higher level) step to be performed by humans; such labels can anyway be added later on by human annotators or annotating algorithm which exploit part decomposition similarity.

The main drawback of this approach is that, although it is possible to obtain an automatic part decomposition, it is rarely possible to consider it reliable; this becomes even more evident when using real data such as noisy point clouds rather than precise 3D models. While the retrieval system is efficient, the overall performance of the system depends on the performance of the acquisition and shape decomposition stage. Furthermore, especially in the case of articulated objects, it is difficult to generalize some of the properties due to the high variability of the examples; for dealing with these cases, it would be useful to add the concept of *joint* between two different parts as in SRDL.

Following the criteria highlighted in the first chapter, this implementation can be analyzed as follows:

- Representation: concepts are used for representing the object classes as they are defined in the two datasets (the first one having no taxonomic structure, the second one offering different levels of hierarchy) and geometric characteristics of parts in the second ontology; individuals are used for representing single instances of such classes, single parts composing each instance from the first dataset and reified relations in the first version of the ontology; object properties relate objects having a part decomposition with their parts in both the ontologies, then reified relations with their elements in the first ontology and parts among them in the second ontology; datatype properties relate objects with their numerical quantities.
- Reasoning tasks: complete materialization (involving classification and realization) offered by a rule reasoner associated to the triple store.
- Expressivity: $\mathcal{AL}\mathcal{I}(\mathcal{D})$ for the first ontology, $\mathcal{AL}\mathcal{HI}(\mathcal{D})$ for the second one (\mathcal{I} is used because of inverse properties and symmetric properties

such as `connectedTo` and `intersecting`, \mathcal{H} because some object properties, e.g. related to parallelness, are arranged as subproperty of a more general property, e.g. `hasParallelRelationWith`);

- Storage: OWLIM-Lite¹³, a semantic repository including a rule-based reasoner supporting RDFS, OWL-Lite and OWL 2 QL/RL and a set of query languages including SPARQL 1.1; it has been chosen because the instances have to be stored efficiently for retrieval.
- Queries: performed in SPARQL 1.1 on the inferred ontology, in order to support exact qualified cardinality queries, transitive properties (via property paths) etc.
- Extensions: no extensions have been used, but it is possible to augment the ontology with annotations (for example using tools such as Fuzzy2OWL¹⁴).

3.5.1 Future work

An extension of this work, which is subject of current research, is the integration within a broader knowledge base such as the PRAXICON (see Chap. 5). As the PRAXICON knowledge base has been derived by cognitive experiments involving humans, the description of objects and their characteristics makes use of semantic features; as the segmentations we have used have been created by humans, it would be possible to map them to the mereological information stored within the PRAXICON. In order to produce significant results and to be able to generalize, this integration process has anyway to deal with the availability of a rich description of objects on the PRAXICON side and of a variety of decomposed objects with a high inter-class and intra-class variability; thus, the most limiting issue at the moment is given by coverage on both sides.

The integration with the PRAXICON offers another challenge, namely the integration of purely geometric information about objects with qualitative labels such as `long` or `small` assigned by humans. In order to use a knowledge base containing metric information such as the one presented in this chapter, labels have to be grounded to numerical quantities in some way. In Chap. 4 we will discuss how this problem can be faced with the use of a fuzzy extension of description logics, while in Chap. 5 we will see how to use and compare metric information at a lower level (using numerical quantities rather than labels).

¹³<http://owlim.ontotext.com/display/OWLIMv52/OWLIM-Lite>

¹⁴<http://gaia.isti.cnr.it/straccia/software/FuzzyOWL>

Chapter 4

Description logics and grasping

Although a rich literature already exists dealing with the problem of grasping known and unknown objects based on their geometry and on grasp suitability measures, representation of human (and humanoid) grasp types is still an open research problem. The aim of this chapter is to provide a way to find a suitable qualitative grasp for an object using geometric and metric information as described in the previous chapters. We will discuss the possibility to use expressive description logics to deal with the problem of grasping, thus extending the way grasping and manipulation can be represented and used in knowledge representation frameworks; in particular, we will discuss the use of a fuzzy extension of description logics to qualitatively describe a taxonomy of human grasp types.

As the focus of this thesis on logical formalisms, this work provides not only an evaluation of the suitability of a semantic representation for a robotic-specific application, but also a test bench for the usage of fuzzy description logics out of their usual domain of application, using data to derive fuzzy membership function to provide additional generalization capabilities.

4.1 State of the art

The study of grasping and manipulation has a long tradition in robotics: as it is a very complex human activity, involving many aspects of cognition from perception to action, it is still a hot research topic whose aim is to mimic such human ability on robots and to make it as much autonomous as possible.

The analysis of human grasps brought to the development of several

grasping taxonomies, whose purpose is to group grasps according to different aspects such as the *opposition types* (the directions in which forces can be applied by the hand on the object, usually measured with respect to the palm), the *power/precision requirements* (for distinguishing the possible uses of a certain grasp) and the *virtual fingers* (groups of fingers working together as functional units). In [Feix et al.,] a taxonomy along with a dataset built from grasping experiments is presented; another dataset showing results of grasping simulations using objects from the Princeton dataset is presented in [Goldfeder et al., 2009].

The eigengrasp approach has been introduced in [Ciocarlie et al., 2007, Ciocarlie and Allen, 2009] to reduce the dimensionality of grasp configurations by using “grasping units” which can be combined together to represent more complex grasps; the main idea is that, as a grasp is a vector in the space of the degrees of freedom of a hand (position and rotation of its links and joints), its principal components obtained through principal component analysis (PCA, see [Jolliffe, 2002]) can be combined linearly to approximate all the grasp postures, thus reducing the dimension of the space to search. Another approach whose popularity has been increasing in the last decades is the use of *affordances* [Gibson, 1975], which can be seen as “possibilities” offered by the environment and the objects to execute a grasp (see for example [Ek et al., 2010, Barck-Holst et al., 2009]). Other approaches for grasp learning include data-driven generalizations (matching hand shape to object shape) [Li et al., 2007], shape analysis and hand preshaping [Bard et al., 1991] and grasp understanding [Palm et al., 2009]. The selection of a preshape is important because, as it is possible to perform similar actions on objects within the same category, only a subset of grasps is needed; preshape selection avoids a full-space search in the space of possible grasps by providing information on mechanical and physical constraints of the hand.

Teaching a robot how to grasp objects can be done in different ways, depending on how much a human teacher is involved in the process; Programming-by-Demonstration (see e.g. [Zollner et al., 2002, Cypher and Halbert, 1993, Ekvall and Kragic, 2005, Kang and Ikeuchi, 1994]), for instance, is an approach where the teacher executes a movement which is tracked and then replicated on a robot. Several works include not only real settings for manipulation but also simulated environments (see e.g. [Aleotti and Caselli, 2010a, Ogata and Takahashi, 1994]).

On the symbolic and semantic side, several approaches have been tried including knowledge bases and rules [Iberall et al., 1988] and primitive-grasp tables [Bekey et al., 1993]. Symbol extraction from manipulation process

has been experimented as well, for example in [Chinellato et al., 2007]. Other approaches to semantic grasping and manipulation can be found in [Aleotti and Caselli, 2012, Aleotti and Caselli, 2010b], where part decomposition and object representation using graphs are used to generalize grasps; a similar approach using superquadric fitting has been presented in [Varadarajan and Vincze, 2011]. None of these works anyway makes use of description logics or semantic query languages.

KnowRob, on the other hand, represents information on grasp capabilities with taxonomies such as the following:

```

GraspingSomething
  PowerGrasp
    ExtensionType
    LargeDiameter
  PrecisionGrasp
    PalmarPinch
    PrecisionSphere
  IntermediateGrasp
    IntermediateExtension

```

while information about robot hands is expressed via SRDL as a series of ABox individuals denoting the URDF links and joints such as `pr2.l_gripper.l_finger_joint`, `pr2.l_gripper.l_finger_link`, `pr2.l_gripper.l_finger_tip_joint`, `pr2.l_gripper.l_finger_tip_link` etc., related through object properties such as `precedingLink`, `succeedingLink`, `precedingJoint`, `succeedingJoint` and so on, for example:

```
succeedingLink(pr2.l_gripper.l_finger_tip_joint, pr2.l_gripper.l_finger_tip.link)
```

```
precedingLink(pr2.l_gripper.l_finger_tip_joint, pr2.l_gripper.l_finger.link)
```

Also, the joints have an orientation matrix represented by 36 datatype properties having floating-point numbers as range and `matrixElement` as superproperty.

4.2 Representation issues

When attempting to provide a semantic representation of a grasp type, several problems have to be taken into account:

- Can a grasp type be represented only in terms of its shape? Does it have sense?
- What attributes are necessary and interesting for the “representation” of a grasp? Are such attributes “ambiguous”?
- Which is the level of detail needed for describing the object to be grasped?
- Is this representation scalable and efficient?
- How can it be related to other features of an object, a hand or a robot?
- How can it be used to take into account physical constraints (e.g. the weight of an object, the torque, etc.)?

In our previous works [Vitucci et al., 2010b, Vitucci, 2011] we faced some of these issues, focusing on the level of expressivity needed for a flexible representation of both the grasp types and of some classes of objects, the latter derived directly from an image processing step; in particular, we explored fuzzy DLs for representing objects by image part decomposition in order to exploit such representation for manipulation tasks. In this chapter we analyze the problem of grasp representation, trying to provide an answer to the remaining issues and some ideas on possible extensions.

Although a semantic representation is not sufficient alone for reliable grasp planning, it is useful for adding a level of detail to grasp representations when they are integrated in a semantic framework. Some of the attributes which can be efficiently represented using a semantic formalism include the approximate shape and size of a specific grasp: this information is enough to reduce the number of possible grasps to simulate during the planning phase, as it rules out the highly unlikely ones and gives some degree of preference on the others. As this representation is obtained directly from the data, it can be easily adapted to a different formalism; the main advantage is in that it makes the inclusion of other sources information (such as topological information for objects or their physical properties) easy, provided that such sources are defined conveniently as discussed in Chap. 3.

4.3 Implementation

4.3.1 Dataset

For our experiments we used the taxonomy and the dataset¹ presented in [Feix et al.,], because it is the most up-to-date and freely available dataset along with a grasp taxonomy and it simple to be used with Matlab or Octave. More specifically, the dataset contains information about a series of grasping experiments performed by five subjects in two different trials (under the same conditions and used respectively for training and testing in the related works) on a series of specific objects such as a CD, a tennis ball or a cylindrical object; Table 4.1 lists all the objects used in the experiments along with their geometry and size.

Each experiment includes the whole trajectory of the hand, from a resting position (where the hand lies on a table) to the holding position (after performing the grasp and holding the object over the support) and then back to the resting position; six magnetic sensors located on the five fingers and on the dorsum of the hand are used for tracking. The information registered include the position of each finger and its rotation matrix, either raw or transformed with respect to the dorsum. Examples of the trajectories recorded from the sensors are shown in Fig. 4.1, showing both raw trajectories of all the fingers plus the dorsum and trajectories of the fingers with respect to the dorsum. We used the untransformed dataset containing 100 samples; furthermore, differently from [Palm et al., 2009] we are only interested in the position the fingers assume when the grasp is completed, thus we only consider the positions in the highest point of the trajectory.

4.3.2 The grasping ontologies

We created two different ontologies: the first one is a “standard” ontology derived from [Feix et al., 2009]; each of the 31 grasp types has been represented as a class, and the grasping category (power, precision or intermediate), the opposition type (palm, pad or side) and the virtual fingers involved (including the palm, as a “special” finger) are all represented as classes.

The fuzzy ontology, instead, is automatically built from the data (see Sec. 4.3.3) and includes the qualitative size and shape of the objects that can be grasped by each grasp; it can include the standard ontology as well, but for the sake of simplicity (and for performance issues on the current version of

¹Several versions of the dataset are available on <http://grasp.xief.net/dataset.htm>

Grasp n.	Grasp name	Object	Size
1	Large Diameter	Cylinder	11cm dia
2	Small Diameter	Cylinder	3cm dia
3	Medium Wrap	Cylinder	3cm dia
4	Adducted Thumb	Cylinder	3cm dia
5	Light Tool	Cylinder	1cm dia
6	Prismatic 4 Finger	Cylinder	1cm dia
7	Prismatic 3 Finger	Cylinder	1cm dia
8	Prismatic 2 Finger	Cylinder	1cm dia
9	Palmar Pinch	Coin	
10	Power Disk	Mini CD	8cm dia, 2mm high
11	Power Sphere	Tennis Ball	67mm dia
12	Precision Disk	CD	12cm dia, 2mm high
13	Precision Sphere	Tennis Ball	67mm dia
14	Tripod	Golf Ball	43mm dia
15	Fixed Hook	Cylinder	3cm dia
16	Lateral	Credit Card	
17	Index Finger Extension	Cylinder	3cm dia
18	Extension Type	Plate	
19	Writing Tripod	Cylinder	1cm dia
20	Parallel Extension	Box	4 cm thick
21	Adduction Grip	Cylinder	1cm dia
22	Tip Pinch	Cube	5mm dia
23	Lateral Tripod	Bottle Cap	
24	Sphere 4 Finger	Tennis Ball	67mm dia
25	Quadpod	Golf Ball	43mm dia
26	Sphere 3 Finger	Tennis Ball	67mm dia
27	Stick	Cylinder	1cm dia
28	Palmar	Plate	
29	Ring	Cylinder	64mm dia
30	Ventral	Cylinder	1cm dia
31	Inferior Pincer	Golf ball	43mm dia

Table 4.1: Objects used in the grasping experiments with their sizes.

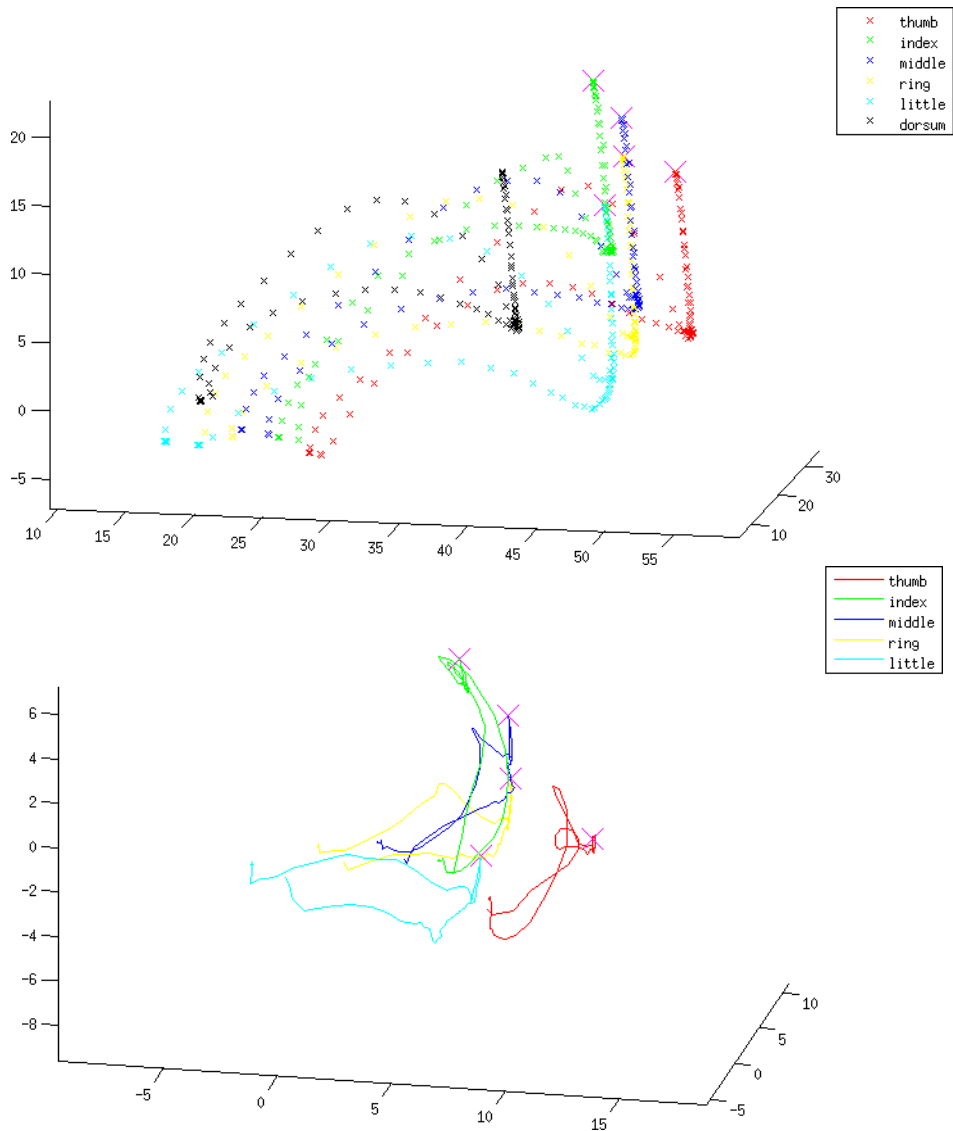


Figure 4.1: Untransformed and transformed trajectories of the finger tips.

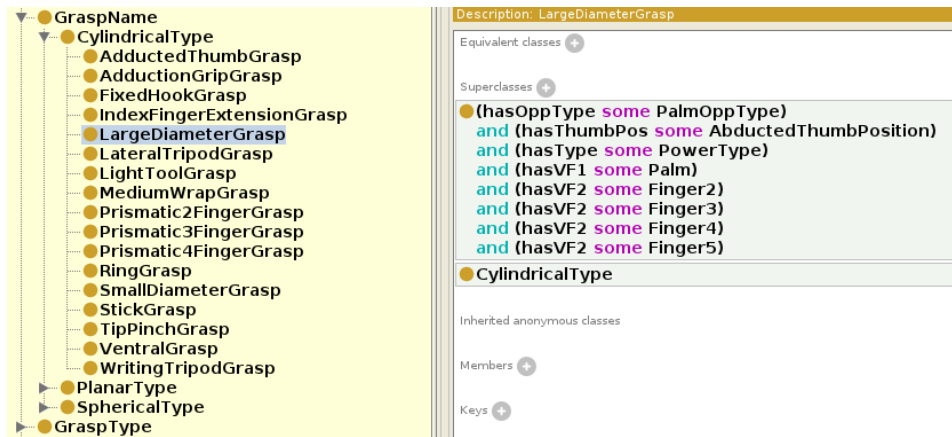
the reasoner) we omit such information. Fig. 4.2 shows an excerpt from both the ontologies.

4.3.3 Use grasp features to describe objects

Our aim is to deal with variation in an object size and shape in a smooth way. We do not want to impose explicit (“crisp”) boundary values to every grasp but rather express it with a suitability value for the object to grasp; for this reason, we use a fuzzy formulation both of the objects properties and of the grasp types.

As it is possible to see from Tab. 4.1 the used objects have different shapes, some of them being cylindrical, some other spherical and so on; this means that, for most of the grasps, not all the dimensions of an object are important: for example, grasps executed on cylindrical objects are not affected by the length of the grasped cylinders (in the hypothesis that the object is well balanced and forces do not play a major role), provided that their length is at least comparable with the distance between the index finger tip and the little finger tip and that the grasp position is not relevant (i.e. a power grasp can be executed in the middle or close to any of the ends of a big cylinder). Grasps belonging to the taxonomy can be described giving to their “predominant” dimensions a higher weight, meaning that those dimensions will be preferred over the remaining dimensions to decide the degree of suitability of the grasp with respect to an object; thus, the advantage of using a fuzzy reasoner such as fuzzyDL is that weighted axioms are supported, so it is possible to give a different degree of preference to every feature.

As we mentioned earlier on, and extending the approach presented in [Bekey et al., 1993], grasp measures can be used to “ground” the concepts related to size (i.e. length, width and height); in other words, measures obtained by some selected grasps can be regarded as “representative” for the linguistic concepts related to size. As an example, a Large Diameter grasp can be considered representative for the concept of “big size” to which we refer as **VeryHigh**. Although it would be possible to use predefined linguistic labels, saying for example that the membership function **VeryHigh** is a triangular function defined in the interval $[9cm, 11cm]$ with maximum degree of truth at $10cm$, this would not reflect the shape of a specific hand and would require a normalization to have the same meaning for every hand; in fact, not only grasps are performed in a different way depending on the hand, but different hands pose different mechanical constraints. In addition, deriving the membership functions from the data makes the process data-driven and



```
(define-fuzzy-concept VeryLow
  triangular(0.0000, 30.0000, 0.0000, 0.3620, 1.7114))
(define-fuzzy-concept Low
  triangular(0.0000, 30.0000, 0.3620, 1.7114, 3.7643))
(define-fuzzy-concept Medium
  triangular(0.0000, 30.0000, 1.7114, 3.7643, 8.3681))
(define-fuzzy-concept High
  triangular(0.0000, 30.0000, 3.7643, 8.3681, 12.1212))
(define-fuzzy-concept VeryHigh
  triangular(0.0000, 30.0000, 8.3681, 12.1212, 14.1858))

(implies (w-sum
  (0.0833 (some hasR1 No))
  (0.0833 (some hasR2 Yes))
  (0.0833 (some hasR3 No))
  (0.1500 (some hasFirstDim VeryLow))
  (0.3000 (some hasSecondDim VeryHigh))
  (0.3000 (some hasThirdDim VeryHigh)))
  LargeDiameterGrasp)

(implies (w-sum
  (0.0833 (some hasR1 No))
  (0.0833 (some hasR2 Yes))
  (0.0833 (some hasR3 No))
  (0.1500 (some hasFirstDim High))
  (0.3000 (some hasSecondDim Medium))
  (0.3000 (some hasThirdDim Medium)))
  SmallDiameterGrasp)
```

Figure 4.2: Example of the two grasping ontologies.

automatic.

In order to relate grasp data to the linguistic labels describing size and shape of objects and grasps, several steps have to be performed:

1. define a “measure” for every grasp, i.e. define how to measure the size of an object wrapped by every grasp from numeric data: for example, if the relevant dimension in the Inferior Pincer grasp is the radius of the sphere it wraps, we need a way to estimate the same dimension of the wrapped object starting from the data, i.e. to define “how large” is an Inferior Pincer grasp when performed by *this* specific hand: in this case, a good approximation can be achieved considering the (Euclidean) distance between the thumb tip and the index finger tip;
2. choose a representative grasp for every concept of size: although the choice can be made on an arbitrary basis, we can use grasp measures to find clusters and then select any grasp belonging to each of these groups;
3. build the membership functions from the measures of the selected grasps, i.e. decide what kind of function and which parameters to assign to each of them;
4. evaluate the membership functions for every grasp to express each grasp in terms of the linguistic labels.

Grasp measures and important dimensions As the dataset contains a whole sequence, we have to select a single sample (or a small set of samples) which represents the position of the fingers *while* the hand is grasping the object; from this sample, which we choose as the time in which most of the fingers are located at their maximum height with respect to the rest position, we extract the positions of the finger tips and of the dorsum and calculate their pairwise distances. We also calculate the distance between each pair of finger tips at the beginning of the experiment to obtain information such as the length of the hand etc. As in the dataset the fingers are ordered as “index, thumb, wrist, middle, ring, little”, to avoid confusion before any calculation we rearrange them as “thumb, index, middle, ring, little, wrist”. An example of the evolution of the distance between each pair of finger tips is shown in Fig. 4.3.

As we have mentioned, the distances between finger tips are used for calculating the characteristics of each grasp type; the “dimensions” of a grasp are intended as the dimensions of the object it can be applied to rearranged in descending order, i.e. if the size of an object aligned to the axis is

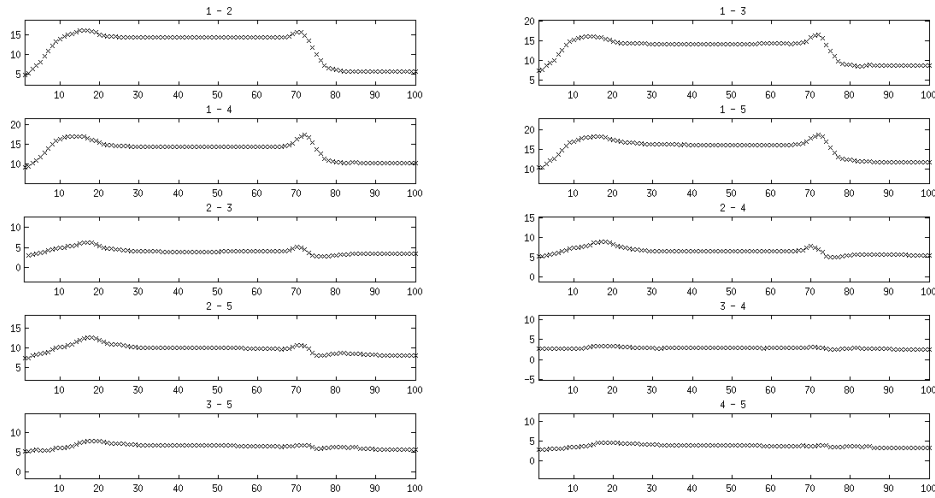


Figure 4.3: Distances between finger tips for the Large Diameter grasp (subject 1, trial 2).

Grasp	Main dimension(s)	Measures
Large Diam.	2, 3 (cylinder radius)	$\langle D_{1,2}, D_{1,3}, D_{1,4}, D_{1,5} \rangle - 2fT$
Small Diam.	2, 3 (cylinder radius)	$\langle D_{6,2}, D_{6,3}, D_{6,4}, D_{6,5} \rangle - pT - fT$
Tripod	1, 2, 3 (sphere radius)	$D_{1,3} - 2fT$
Parallel Ext.	3 (plane thickness)	$\langle D_{1,2}, D_{1,3}, D_{1,4}, D_{1,5} \rangle - 2fT$

Table 4.2: Measures for each grasp. $D_{i,j}$ is the distance between finger tip i and finger tip j (finger 6 being the palm), $\langle D_{i,j}, D_{k,l} \rangle$ is the average between the distances $D_{i,j}$ and $D_{k,l}$, fT is the finger thickness, pT is the palm thickness.

$\langle 4cm, 7cm, 3cm \rangle$ it will be rearranged as $\langle 7cm, 4cm, 3cm \rangle$. Furthermore, as we mentioned each grasp has an associated “vector of importance” to identify the relevant dimensions of an object it can be applied to; for instance, a vector such as $[0, 1, 1]$ means that for this grasp the relevant dimensions of the object to be grasped are the least significant two.

It is important to note that the importance vectors and the measures are performed in two different phases, although they are related to each other: the importance vectors define *what* dimensions are more discriminative for a grasp, while the grasp measure define *how* to calculate them and, if possible, how to approximate the less important dimensions. Examples of importance vectors and measures are shown in Tab. 4.2.

The grasp measures are calculated qualitatively, and as we will see later it is not necessary to have a precise measure. For the same reason, as the used distance is Euclidean, it is not necessary to transform the finger tip position

and the dorsum position to obtain respectively the real finger tip and palm center positions, but it is enough to subtract the finger and palm thicknesses from the calculated distance. Tab. 4.3 shows the error when comparing the grasp measures with the real objects used in the experiments; as it can be seen measures for the same grasp can be very different not only between two different subjects but even between two different trials by the same subject, therefore some measures can be less reliable for some specific types of grasp.

Grasp	Subject 1		Subject 2		Subject 3		Subject 4		Subject 5	
	Trial 1	Trial 2	Trial 1	Trial 2	Trial 1	Trial 2	Trial 1	Trial 2	Trial 1	Trial 2
1	0.1146	0.1102	0.1035	0.1019	0.0091	0	0.0349	0.0354	0.0763	0.0639
2	1.1085	0.2431	0.0429	0.0714	1.2284	0.0219	0.2384	0.2508	0.2026	0.2043
3	0.3515	0.3036	0.8774	0.1714	0.1985	0.3246	0.4653	0.4001	0.2099	0.1911
4	0.2573	0.2594	0.1796	0.2021	0.1510	0.1795	0.4170	0.3730	0.2594	0.2362
6	0.6341	1.0234	0.8142	0.9874	1.0353	0.8983	0.2895	0.2851	1.2653	1.1677
7	1.1014	0.9546	1.0934	1.2828	0.8972	0.9796	0.7610	0.7827	0.8723	0.9800
8	0.7151	0.9563	1.4658	0.6622	0.9078	0.9191	1.0577	0.9144	1.0108	0.8947
9	2.1871	3.1813	4.7878	6.6508	3.1994	2.6536	4.1107	35.1508	6.1399	5.0693
11	0.2620	0.1230	0.1723	0.0697	0.0523	0.0897	0.0259	0.0096	0.0665	0.0757
12	0.6788	0.0551	0.0952	0.0578	0.0485	0.0569	0.0340	0.0187	0.0473	0.0437
13	0.6839	0.0414	0.0654	0.0086	0.0856	0.0847	0.0905	0.0708	0.1161	0.0255
14	0.5654	0.0709	0.0940	0.0813	0.1508	0.2047	0.0203	0.0567	0.0445	0.0123
15	1.2292	0.2583	0.0640	0.0758	0.2370	0.1269	0.2843	0.2623	0.2315	0.2210
17	0.3898	0.7826	0.3187	0.8967	0.4063	0.5201	0.4962	0.2773	1.6100	0.4597
18	0.2496	0.6429	0.1829	0.1604	0.0631	0.2097	0.3697	0.6632	0.2528	0.0503
20	0.5739	0.3025	0.0216	0.0335	0.2825	0.2404	1.6179	0.3146	1.4766	1.4658
21	3.2917	0.7111	3.1596	2.7871	1.2061	1.1956	0.6248	0.5087	1.6329	2.0357
22	0.4019	0.2951	13.9136	0.2760	0.3710	0.4411	0.3874	0.5688	0.9189	0.8820
24	0.0959	0.0515	0.2087	0.2317	0.0146	0.0120	0.0177	0.0242	0.0356	0.0082
25	0.1144	0.2178	0.0873	0.1085	0.1423	0.1486	0.1092	0.1524	0.0390	0.0577
26	0.1316	0.0647	0.2006	0.2133	0.0251	0.0170	0.0775	0.0332	0.0349	0.0337
28	0.0400	0.1346	0.0344	0.1165	1.6767	1.6335	1.8939	0.0885	0.1721	0.1505
29	0.0981	0.0607	0.3014	0.3075	0.1074	0.0188	0.0919	0.0060	0.0393	0.0151
31	0.1139	0.2018	0.8116	0.8617	0.2582	0.2251	0.0693	0.1769	0.4509	0.5668

Table 4.3: Errors between estimated and real measures, calculated as $e = (m - r)/r$, where m is the grasp measure and r is the real measure of the object on its important dimension. Light cyan is used for highlighting the “small” grasps, while light red for the grasps showing significant differences among subjects and light yellow for grasps showing significant differences between two trials of the same subject.

For defining the important dimensions, we divide the 31 grasps in 4 categories:

- cylindrical grasps (most important dimensions: the 2 smallest ones, corresponding to the cylinder radius): Large Diameter, Small Diameter, Medium Wrap, Adducted Thumb, Light Tool, Prismatic 4 Finger, Prismatic 3 Finger, Prismatic 2 Finger, Fixed Hook, Index Finger Extension, Writing Tripod, Adduction Grip, Tip Pinch, Lateral Tripod, Stick, Ring, Ventral;
- spherical grasps (most important dimensions: all the dimensions, corresponding to the sphere radius): Power Sphere, Precision Sphere, Tripod, Sphere 4 Finger, Quadpod, Sphere 3 Finger, Inferior Pincer;
- disk grasps (most important dimensions: the 2 biggest ones): Power Disk, Precision Disk;
- planar grasps (most important dimension: the smallest one): Palmar Pinch, Lateral, Extension Type, Parallel Extension, Palmar.

Anyway, we exclude the following grasps from further analysis because it is not possible to estimate the object dimensions for them: Light Tool, Power Disk, Lateral, Lateral Tripod, Stick, Ventral. It is important to note here that the terms “spherical” and “cylindrical” are used in a broad sense, thus the first one is a synonym for “equidimensional” (so cubes are “spherical” as well) and the second one is a synonym for “elongated” (so a prolate ellipsoid or a parallelepiped would be “cylindrical” as well).

Building the membership functions Table 4.4 and Fig. 4.4 show the grasps selected as representative for each linguistic label related to the size; Table 4.2 shows the measures for some of the grasps, while Table 4.3 shows the difference in 5 different experiments (all the people at the second trial) between the estimated measures and the real measures. As it can be seen, the chosen measures are better in some trials than in others; this can happen either because, after the first grasp, a subject performs the second grasp in a different way (altering the position of the fingers) or because the understanding of the way to execute a specific grasp is different among the subjects, so that some descriptors (which are qualitatively obtained from the shapes suggested in the taxonomy) are better in some cases and worse in some others. It can be noted that the performances are generally worse for “small” grasps and better for “big” grasps, suggesting that the approximation of a size is more difficult when the measured dimension is small because of the sensitivity to the relative position of the sensors.

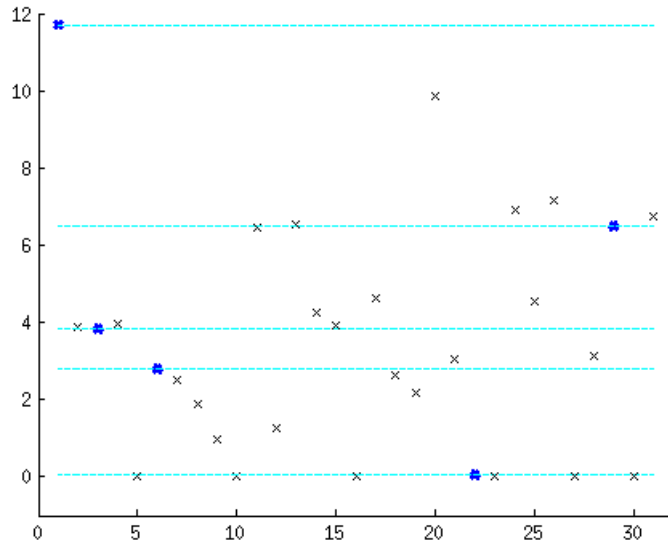


Figure 4.4: Representative grasp measures. The highlighted points represent the most significant measure for the chosen grasps (on the x axis, having a value between 1 and 31).






Linguistic label	Grasp	
VeryLow	Tip Pinch	
Low	Prismatic 2 Fingers	
Medium	Medium Wrap	
High	Ring	
VeryHigh	Large Diameter	

Table 4.4: Chosen representative grasps (images taken from [Feix et al., 2009]).

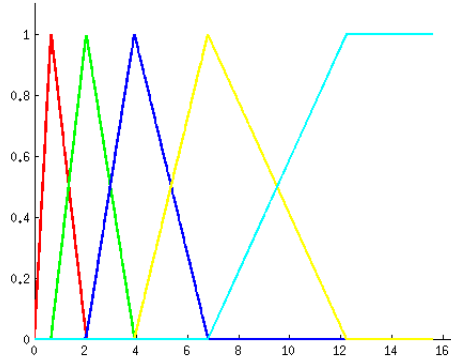


Figure 4.5: Example of generated membership functions.

We select five linguistic labels (namely **VeryHigh**, **High**, **Medium**, **Low** and **VeryLow**) and we build the related fuzzy membership functions. All the functions are triangular except for the **VeryHigh** function which is trapezoidal (the **VeryLow** function is not trapezoidal because there should be a minimum size for an object to be graspable, thus values very close to 0 are not suitable); being M_{range} the maximum value for any dimension, their parameters are calculated as follows:

- the parameter b is the measure of the representative grasp;
- the parameter a is calculated as $a = b_p$, where b_p is the b parameter of the previous membership function (0 for **VeryLow**);
- the parameter c is calculated as $c = b_n$, where b_n is the b parameter of the next membership function (M_{range} for **VeryHigh**);
- the **VeryHigh** function has an additional parameter $d = c = M_{range}$, because the maximum value of a **VeryHigh** dimension cannot exceed the maximum range.

The choice of such shapes for the membership functions is due to the data: if we take a single measure for every grasp, it is natural to assign such value the maximum degree of truth and to assign the minimum degree of truth in correspondence of the value of the previous and next measure. As an alternative it is possible to use both the trials by the same subject to define an interval where the function has the maximum degree of truth, but as it can be seen from Tab. 4.3 two trials by the same subject can have completely different measures.

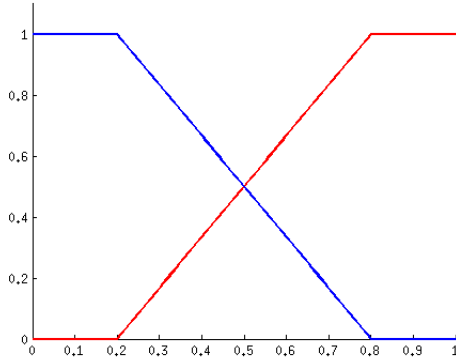


Figure 4.6: Membership functions related to the object shape.

From these measures all the grasp data are “converted” to membership functions; in this way, for example, a grasp on a planar object can be described in terms of the dimension of another grasp. As the values for the fuzzy functions are automatically derived, these concepts can be “personalized” depending on the real hand shape to be used; thus, what is **High** for one hand can be **High** or **Medium** for another one.

We also need a way to define approximately the shape of an object, i.e. beside its size we need to know whether the object is cylindrical, spherical or planar in order to select the correct family of grasps. Such characteristics are calculated from the bounding box dimensions as follows:

- $R_1 = \frac{b}{a}$
- $R_2 = \frac{c}{b}$
- $R_3 = \frac{c}{a}$

Each of these features is evaluated through the two predefined membership functions *Yes* and *No* (Fig. 4.6), measuring the “similarity” of two dimensions through their ratio as follows:

- *Yes*: trapezoidal function with parameters [0.2, 0.8, 1, 1]
- *No*: trapezoidal function with parameters [0, 0, 0.2, 0.8].

Building the axioms These pieces of information have to be combined to obtain fuzzy axioms giving appropriate weights to each of the features,

that is to the three features related to the shape (R_1 , R_2 and R_3) and to the three features related to the size (length, width and height); additionally, an axiom can include the type of the grasp (namely Power, Precision or Intermediate). Thus, an axiom has the following form:

$$W_{shape} \sum_{i=1}^3 (w_{ratio_i} MF_{ratio_i}(x)) + W_{size} \sum_{j=1}^3 (w_{size_j} MF_{size_j}(x)) \implies \text{Class}(x)$$

with the constraint:

$$W_{shape} \sum_{i=1}^3 w_{ratio_i} + W_{size} \sum_{j=1}^3 w_{size_j} = 1$$

where W_{shape} and W_{size} are the fixed overall weights given respectively to the shape and to the size membership functions, while w_{ratio_i} and w_{size_j} are the three weights for each dimension ratio and each dimension size.

The choice of using weights instead of operators such as AND is due to the fact that weights offer a way to balance the effect of “false” membership function, where an AND (translated in a min function according to Zadeh logic) would make the whole axiom false.

4.3.4 Use object features to retrieve grasps

For retrieving suitable grasps, fuzzy axioms like the following can be used:

```
(implies (w-sum
  (0.0833 (some hasR1 No))
  (0.0833 (some hasR2 Yes))
  (0.0833 (some hasR3 No))
  (0.1500 (some hasFirstDim VeryHigh))
  (0.3000 (some hasSecondDim VeryHigh))
  (0.3000 (some hasThirdDim VeryHigh)))
  LargeDiameterGrasp)
```

This axiom means that an object, to be grasped by a Large Diameter grasp, has to have all **VeryHigh** dimensions and a cylindrical shape (as R2, the ratio between the second and third dimension, is close to one while the other two ratios are close to 0). Although it would be more clear to express the fuzzy axioms as in the formula above (i.e. as the weighted sum of the shape and size components, which in turn are weighted sums), for performance reasons it is more convenient to write fuzzyDL axioms in this way (i.e. including

the weights of the shape and size components in the weight of every single membership function).

Some taxonomic (crisp) information can be added to these axioms: for example a Large Diameter grasp is a Power Type grasp, so this information can be added using an AND operator (because either a grasp is a Power Type or it is not, according to the taxonomy) or as an additional weighted term (if the type of a grasp should not have much importance with respect to its geometric characteristics).

4.4 Experiments

In the next examples we proceed to a qualitative evaluation of the use of fuzzy axioms to find suitable grasps for each part of an object. For every object part we select the two preferred grasps and discuss the result; it is important to note that, as we want to exploit the object information presented in Chap. 3, we focus on the geometric characteristics rather than on the mereotopological structure, thus we do not introduce any constraints on the position of the parts composing an object: they are considered free, i.e. not attached to each other. In Fig. 4.7 the objects used in the experiments, whose results are reported in Tab. 4.6 to 4.11, are shown. For each table several scales along with the corresponding measures for the selected object part are reported, then the best two grasps both with $[W_{shape} = 0.25, W_{size} = 0.75]$ and with $[W_{shape} = 0.5, W_{size} = 0.5]$ are reported. The tables are limited to five parts per object for space reasons. The abbreviations for the grasp names are show in Tab. 4.5.

In the first example (the airplane) it is possible to see that for the first part (the body of the plane) the most suitable grasps are cylindrical and the degree of suitability of the found grasps is rather high; for the remaining parts (the wings and the stabilizers) the preferred grasps are planar. It can be noted that the highest degrees of satisfiability are reached on average when the model is scaled by a factor of 25, that is when its dimensions are closer to the dimensions of the hand used for creating the used membership functions.

In the second example (the bearing) for the first part (the biggest one, represented in red in the figure) a Power Sphere grasp is considered more suitable when the scale is 10, while a Precision Disk grasp is considered better when the scale is 25 and 50: the reason is that, although the dimensions of the object are almost equal to each other thus the shape is inferred to be spherical, the size in the second case is much more similar to the size of the important dimensions of a Precision Disk grasp, keeping into account that

the third dimension in the case of disk grasps is considered less important. In the third case, instead, the unrealistically high suitability for a Precision Disk grasp is due to the fact that the maximum for the `VeryHigh` membership function has been set to a value of 30, thus all the three dimensions are considered to be `VeryHigh`.

In the third and fourth case (the chair and the table) for the first part (the seat and the table top respectively) the preferred grasps are planar, while for the legs the preference is for cylindrical grasps.

In the fifth case (the cup) for the first part (the container) the preferred grasps are spherical or disk-type depending on the dimensions as in the case of the bearing; in this case it is possible to see that when the scale is high, even if the shape of the object is spherical, the grasps have a very low suitability: this happens because the dimensions exceed the limit for the `VeryHigh` membership function so that no membership function becomes even partially true. For the second part (the handle) the found grasps, although having rather high suitability degrees, are not much reliable for a real grasp, the reason being that no grasp on a concave object is defined in the taxonomy; this is a case which shows that information on the bounding box and the (convex) shape alone are not sufficient for finding a suitable grasp.

In the sixth case (the vase) for the first part (the bowl) a spherical grasp is preferred when the scale is 10, while a cylindrical grasp is preferred when the scale is 25; again, this depends not only on the shape of the object but also on its size. For the second part (the neck), instead, cylindrical grasps are preferred.

LargeDiameter	LD	SmallDiameter	SD
MediumWrap	MW	AdductedThumb	AT
LightTool	LT	Prismatic4Finger	P4F
Prismatic3Finger	P3F	Prismatic2Finger	P2F
PalmarPinch	PP	PowerDisk	PD
PowerSphere	PS	PrecisionDisk	PrD
PrecisionSphere	PrS	Tripod	T
FixedHook	FH	Lateral	L
IndexFingerExtension	IFE	ExtensionType	ET
WritingTripod	WT	ParallelExtension	PE
AdductionGrip	AG	TipPinch	TP
LateralTripod	LTr	Sphere4Finger	S4F
Quadpod	Q	Sphere3Finger	S3F
Stick	S	Palmar	P
Ring	R	Ventral	V
InferiorPincer	IP		

Table 4.5: Grasp names abbreviations.

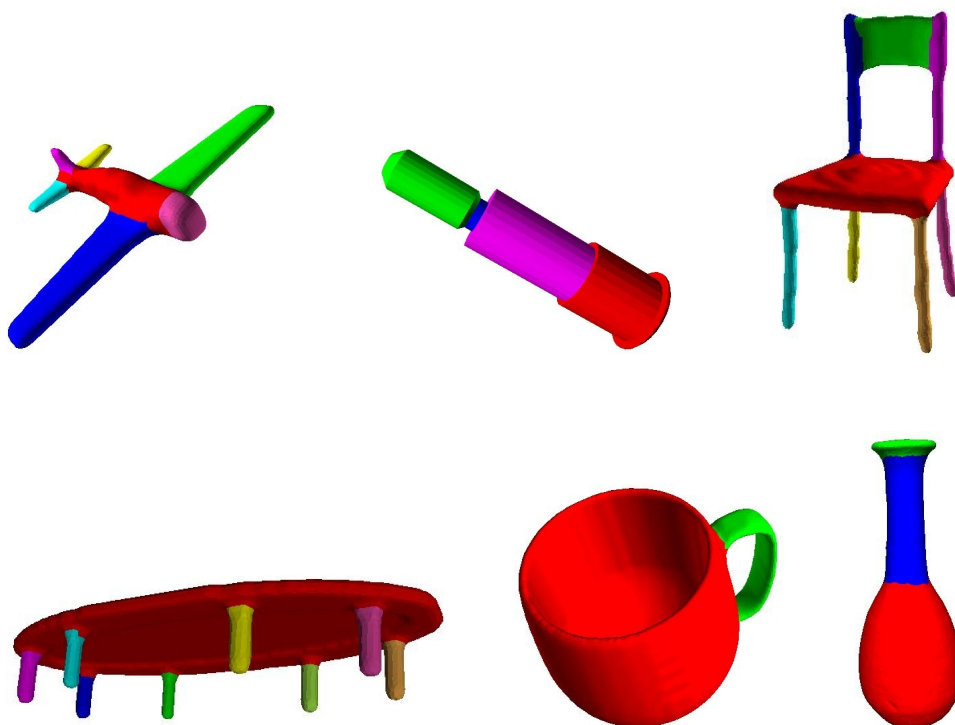
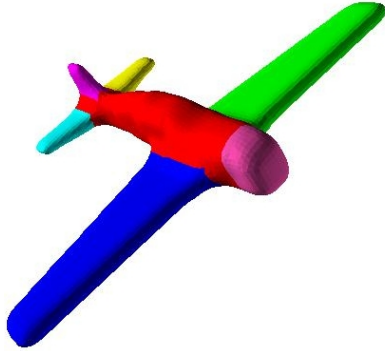
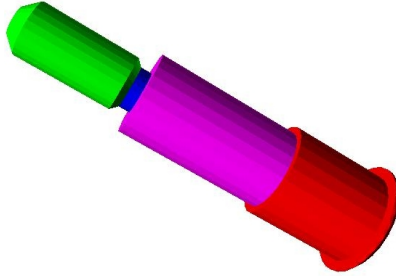


Figure 4.7: Objects used in the experiments.



Scale	Dimensions	$W_{shape} = 0.25, W_{size} = 0.75$		$W_{shape} = W_{size} = 0.5$	
1.00	[1.31 0.30 0.27]	TP (0.5086)	PP (0.4208)	TP (0.6663)	LD (0.4908)
5.00	[6.56 1.49 1.35]	P2F (0.7204)	P4F (0.5824)	P2F (0.8074)	P4F (0.7155)
10.00	[13.12 2.97 2.71]	P4F (0.7356)	P3F (0.7356)	P4F (0.8176)	P3F (0.8176)
25.00	[32.79 7.44 6.77]	R (0.8074)	PS (0.5563)	R (0.8655)	LD (0.5147)
50.00	[65.58 14.87 13.53]	LD (0.8454)	PE (0.6862)	LD (0.8908)	PE (0.5723)
1.00	[0.86 0.44 0.12]	PP (0.5081)	TP (0.3886)	PP (0.6055)	TP (0.4367)
5.00	[4.31 2.18 0.58]	PP (0.6015)	P2F (0.4291)	PP (0.6679)	P2F (0.4637)
10.00	[8.61 4.36 1.16]	SD (0.4862)	MW (0.4862)	PP (0.5895)	ET (0.5446)
25.00	[21.53 10.89 2.89]	ET (0.7074)	P (0.7074)	ET (0.7385)	P (0.7385)
50.00	[43.06 21.79 5.78]	LD (0.4332)	PrD (0.3515)	ET (0.5002)	PE (0.5002)
1.00	[0.88 0.46 0.12]	PP (0.5207)	TP (0.4027)	PP (0.6172)	TP (0.4428)
5.00	[4.40 2.31 0.62]	PP (0.6335)	TP (0.4181)	PP (0.6924)	TP (0.4531)
10.00	[8.80 4.63 1.24]	PP (0.4587)	ET (0.4519)	PP (0.5759)	ET (0.5714)
25.00	[21.99 11.57 3.10]	ET (0.6780)	P (0.6780)	ET (0.7221)	P (0.7221)
50.00	[43.98 23.15 6.20]	LD (0.4308)	R (0.3696)	ET (0.5051)	PE (0.5051)
1.00	[0.26 0.16 0.08]	PP (0.3287)	TP (0.2396)	PP (0.4577)	ET (0.3580)
5.00	[1.29 0.82 0.38]	PP (0.6513)	TP (0.5650)	PP (0.6728)	TP (0.5485)
10.00	[2.57 1.64 0.75]	PP (0.6372)	TP (0.4900)	PP (0.6634)	TP (0.4985)
25.00	[6.43 4.10 1.88]	ET (0.5807)	P (0.5807)	ET (0.6258)	P (0.6258)
50.00	[12.86 8.20 3.75]	AT (0.5535)	IFE (0.5535)	AT (0.5408)	IFE (0.5408)
1.00	[0.28 0.22 0.08]	PP (0.3864)	TP (0.2346)	PP (0.5425)	ET (0.4274)
5.00	[1.39 1.11 0.41]	PP (0.6656)	TP (0.4816)	PP (0.7286)	TP (0.4461)
10.00	[2.78 2.22 0.82]	PP (0.6085)	P4F (0.3987)	PP (0.6906)	ET (0.4642)
25.00	[6.96 5.56 2.04]	ET (0.6642)	P (0.6642)	ET (0.7277)	P (0.7277)
50.00	[13.92 11.12 4.08]	PrD (0.6589)	AT (0.5259)	ET (0.6072)	PE (0.6072)

Table 4.6: Airplane.



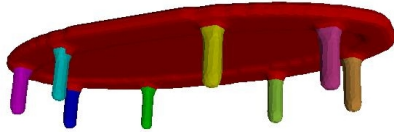
Scale	Dimensions	$W_{shape} = 0.25, W_{size} = 0.75$		$W_{shape} = W_{size} = 0.5$	
1.00	[0.55 0.55 0.54]	PP (0.7114)	TP (0.5872)	PP (0.5854)	PrD (0.5827)
5.00	[2.77 2.76 2.68]	T (0.5326)	Q (0.5326)	T (0.6884)	Q (0.6884)
10.00	[5.53 5.52 5.35]	PS (0.6561)	PrS (0.6561)	PS (0.7707)	PrS (0.7707)
25.00	[13.83 13.80 13.38]	PrD (0.8500)	LD (0.8333)	PrD (0.9000)	LD (0.6667)
50.00	[27.66 27.61 26.77]	PrD (0.8500)	LD (0.8333)	PrD (0.9000)	LD (0.6667)
1.00	[0.55 0.26 0.26]	PP (0.4523)	TP (0.4156)	TP (0.5094)	PP (0.4126)
5.00	[2.76 1.30 1.30]	TP (0.4888)	P4F (0.4597)	TP (0.5583)	P4F (0.5388)
10.00	[5.51 2.60 2.60]	P2F (0.6730)	P4F (0.5897)	P2F (0.6811)	P4F (0.6255)
25.00	[13.78 6.51 6.51]	R (0.8663)	PS (0.6107)	R (0.8099)	PS (0.6192)
50.00	[27.55 13.02 13.02]	LD (0.9243)	PE (0.8333)	LD (0.8486)	PrD (0.7181)
1.00	[0.20 0.20 0.05]	PP (0.3603)	ET (0.2325)	PP (0.5502)	ET (0.4649)
5.00	[0.99 0.99 0.26]	PP (0.6388)	TP (0.4294)	PP (0.7358)	ET (0.4649)
10.00	[1.98 1.98 0.52]	PP (0.6040)	P4F (0.3737)	PP (0.7126)	ET (0.4649)
25.00	[4.95 4.95 1.30]	PP (0.4684)	ET (0.4465)	PP (0.6222)	ET (0.6076)
50.00	[9.90 9.90 2.60]	ET (0.7158)	P (0.7158)	ET (0.7872)	P (0.7872)
1.00	[0.70 0.39 0.39]	PP (0.5895)	TP (0.5125)	TP (0.5424)	PP (0.5041)
5.00	[3.50 1.95 1.95]	P4F (0.7199)	P3F (0.7199)	P4F (0.6806)	P3F (0.6806)
10.00	[7.00 3.91 3.91]	SD (0.8932)	MW (0.8932)	SD (0.7962)	MW (0.7962)
25.00	[17.50 9.77 9.77]	PrD (0.6474)	LD (0.6298)	PrD (0.6753)	LD (0.6206)
50.00	[35.00 19.53 19.53]	LD (0.7505)	PE (0.6833)	LD (0.7011)	PE (0.5667)

Table 4.7: Bearing.



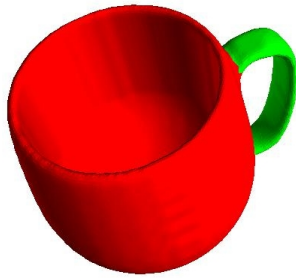
Scale	Dimensions	$W_{shape} = 0.25, W_{size} = 0.75$		$W_{shape} = W_{size} = 0.5$	
1.00	[0.87 0.80 0.20]	PP (0.6376)	TP (0.4457)	PP (0.7430)	ET (0.4769)
5.00	[4.33 4.01 1.01]	PP (0.5714)	T (0.5498)	PP (0.6989)	ET (0.5550)
10.00	[8.67 8.01 2.01]	ET (0.7703)	P (0.7703)	ET (0.8315)	P (0.8315)
25.00	[21.67 20.03 5.03]	PrD (0.6949)	ET (0.5385)	ET (0.6769)	PE (0.6769)
50.00	[43.34 40.06 10.06]	PE (0.5095)	LD (0.2667)	PE (0.6576)	PP (0.4769)
1.00	[0.47 0.31 0.10]	PP (0.4610)	TP (0.3089)	PP (0.5874)	ET (0.4200)
5.00	[2.35 1.54 0.51]	PP (0.6155)	TP (0.4587)	PP (0.6903)	TP (0.4641)
10.00	[4.70 3.09 1.02]	PP (0.5393)	T (0.4067)	PP (0.6396)	ET (0.5005)
25.00	[11.74 7.72 2.54]	ET (0.6993)	P (0.6993)	ET (0.7462)	P (0.7462)
50.00	[23.48 15.44 5.08]	PrD (0.6838)	LD (0.5687)	ET (0.6200)	PE (0.6200)
1.00	[0.83 0.15 0.12]	TP (0.3724)	PP (0.3297)	TP (0.5813)	LD (0.4995)
5.00	[4.14 0.74 0.59]	TP (0.8027)	PP (0.6323)	TP (0.8681)	PP (0.5330)
10.00	[8.28 1.47 1.18]	P2F (0.6535)	TP (0.5960)	P2F (0.7687)	TP (0.7303)
25.00	[20.70 3.68 2.94]	AT (0.8087)	IFE (0.8087)	AT (0.8722)	IFE (0.8722)
50.00	[41.39 7.36 5.88]	R (0.7230)	PS (0.4774)	R (0.8150)	SD (0.5629)
1.00	[0.81 0.13 0.11]	TP (0.3640)	PP (0.3246)	TP (0.5760)	LD (0.5000)
5.00	[4.05 0.67 0.56]	TP (0.8050)	PP (0.6206)	TP (0.8700)	PP (0.5249)
10.00	[8.10 1.34 1.12]	TP (0.6321)	P2F (0.6179)	TP (0.7548)	P2F (0.7452)
25.00	[20.25 3.35 2.80]	AT (0.7346)	IFE (0.7346)	AT (0.8230)	IFE (0.8230)
50.00	[40.50 6.69 5.61]	R (0.7172)	PS (0.4727)	R (0.8115)	SD (0.5885)
1.00	[0.72 0.11 0.11]	TP (0.3496)	PP (0.3245)	TP (0.5664)	LD (0.5000)
5.00	[3.58 0.55 0.53]	TP (0.7478)	PP (0.5770)	TP (0.8319)	LD (0.5000)
10.00	[7.17 1.09 1.06]	TP (0.6743)	P2F (0.5757)	TP (0.7828)	P2F (0.7172)
25.00	[17.92 2.73 2.64]	P4F (0.7892)	P3F (0.7892)	P4F (0.8595)	P3F (0.8595)
50.00	[35.84 5.46 5.28]	R (0.5548)	SD (0.5452)	R (0.7032)	SD (0.6968)

Table 4.8: Chair.



Scale	Dimensions	$W_{shape} = 0.25, W_{size} = 0.75$		$W_{shape} = W_{size} = 0.5$	
1.00	[1.98 1.00 0.10]	PP (0.3934)	TP (0.3918)	PP (0.5413)	TP (0.4266)
5.00	[9.91 5.02 0.49]	PP (0.5524)	TP (0.4392)	PP (0.6473)	ET (0.4762)
10.00	[19.83 10.05 0.99]	PrD (0.6360)	ET (0.5607)	ET (0.6528)	P (0.6528)
25.00	[49.56 25.12 2.47]	ET (0.7031)	P (0.7031)	ET (0.7478)	P (0.7478)
50.00	[99.13 50.25 4.94]	SD (0.3170)	MW (0.3170)	PP (0.4186)	ET (0.4186)
1.00	[0.23 0.09 0.07]	TP (0.2819)	PP (0.2199)	TP (0.4643)	LD (0.4144)
5.00	[1.16 0.45 0.36]	TP (0.5808)	PP (0.5424)	TP (0.6635)	PP (0.4897)
10.00	[2.31 0.90 0.71]	TP (0.7379)	PP (0.6474)	TP (0.7683)	PP (0.5597)
25.00	[5.78 2.25 1.78]	P2F (0.8149)	P4F (0.7177)	P2F (0.8196)	P4F (0.7548)
50.00	[11.55 4.50 3.56]	AT (0.8212)	IFE (0.8212)	AT (0.8237)	IFE (0.8237)
1.00	[0.25 0.09 0.08]	TP (0.2882)	PP (0.2210)	TP (0.4706)	LD (0.4177)
5.00	[1.23 0.44 0.42]	TP (0.6057)	PP (0.5644)	TP (0.6822)	PP (0.4902)
10.00	[2.46 0.88 0.84]	TP (0.7177)	PP (0.5984)	TP (0.7569)	PP (0.5129)
25.00	[6.15 2.19 2.09]	P2F (0.8880)	P4F (0.7713)	P2F (0.8705)	P4F (0.7926)
50.00	[12.30 4.38 4.19]	AT (0.8812)	IFE (0.8812)	AT (0.8659)	IFE (0.8659)
1.00	[0.24 0.09 0.09]	TP (0.2835)	PP (0.2213)	TP (0.4569)	LD (0.4020)
5.00	[1.18 0.45 0.44]	TP (0.6133)	PP (0.5863)	TP (0.6769)	PP (0.5035)
10.00	[2.36 0.90 0.88]	TP (0.6952)	PP (0.5808)	TP (0.7314)	PP (0.4998)
25.00	[5.91 2.25 2.20]	P2F (0.8410)	P4F (0.7368)	P2F (0.8287)	P4F (0.7592)
50.00	[11.82 4.50 4.40]	AT (0.8276)	IFE (0.8276)	AT (0.8197)	IFE (0.8197)
1.00	[0.23 0.08 0.07]	TP (0.2844)	PP (0.2140)	TP (0.4738)	LD (0.4263)
5.00	[1.16 0.42 0.34]	TP (0.5694)	PP (0.5254)	TP (0.6638)	PP (0.4741)
10.00	[2.31 0.85 0.69]	TP (0.7601)	PP (0.6572)	TP (0.7909)	PP (0.5620)
25.00	[5.78 2.12 1.72]	P2F (0.8297)	P4F (0.7323)	P2F (0.8373)	P4F (0.7724)
50.00	[11.56 4.24 3.45]	AT (0.8370)	IFE (0.8370)	AT (0.8422)	IFE (0.8422)

Table 4.9: Table.



Scale	Dimensions	$W_{shape} = 0.25, W_{size} = 0.75$		$W_{shape} = W_{size} = 0.5$	
1.00	[1.39 1.38 1.31]	PP (0.4558)	P4F (0.3875)	PrD (0.5520)	PS (0.5000)
5.00	[6.96 6.91 6.54]	PS (0.9650)	PrS (0.9650)	PS (0.9767)	PrS (0.9767)
10.00	[13.93 13.82 13.09]	PrD (0.8500)	LD (0.8333)	PrD (0.9000)	LD (0.6667)
25.00	[34.82 34.54 32.71]	PS (0.2500)	PrD (0.2500)	PS (0.5000)	PrD (0.5000)
50.00	[69.64 69.08 65.42]	PS (0.2500)	PrD (0.2500)	PS (0.5000)	PrD (0.5000)
1.00	[1.04 0.45 0.25]	PP (0.5311)	TP (0.5053)	TP (0.5743)	PP (0.5454)
5.00	[5.22 2.27 1.27]	P2F (0.6425)	P4F (0.5742)	P2F (0.6658)	P4F (0.6202)
10.00	[10.44 4.53 2.53]	AT (0.5950)	IFE (0.5950)	AT (0.6341)	IFE (0.6341)
25.00	[26.11 11.33 6.33]	PrD (0.6393)	R (0.6287)	R (0.6565)	LD (0.6236)
50.00	[52.22 22.66 12.65]	LD (0.7781)	PE (0.7435)	LD (0.7561)	PE (0.6870)

Table 4.10: Cup.



Scale	Dimensions	$W_{shape} = 0.25, W_{size} = 0.75$		$W_{shape} = W_{size} = 0.5$	
1.00	[0.92 0.61 0.59]	PP (0.7591)	TP (0.6806)	PP (0.6195)	TP (0.6194)
5.00	[4.60 3.03 2.97]	T (0.6586)	Q (0.6586)	T (0.7179)	Q (0.7179)
10.00	[9.20 6.06 5.95]	PS (0.7118)	PrS (0.7118)	PS (0.7533)	PrS (0.7533)
25.00	[23.00 15.15 14.87]	LD (0.8742)	PE (0.8351)	PrD (0.8182)	LD (0.7485)
50.00	[45.99 30.31 29.74]	PE (0.5351)	LD (0.4242)	PE (0.4701)	LD (0.4485)
1.00	[0.34 0.34 0.11]	PP (0.4507)	TP (0.2918)	PP (0.5899)	ET (0.4341)
5.00	[1.71 1.71 0.54]	PP (0.6639)	TP (0.4043)	PP (0.7320)	ET (0.4341)
10.00	[3.41 3.41 1.09]	PP (0.5232)	T (0.4844)	PP (0.6382)	ET (0.5300)
25.00	[8.53 8.53 2.72]	ET (0.5979)	P (0.5979)	ET (0.6880)	P (0.6880)
50.00	[17.07 17.06 5.44]	PrD (0.7163)	LD (0.5333)	ET (0.6341)	PE (0.6341)
1.00	[0.70 0.29 0.28]	PP (0.4900)	TP (0.4557)	TP (0.5612)	PP (0.4393)
5.00	[3.50 1.43 1.40]	P4F (0.5286)	P3F (0.5286)	P4F (0.6098)	P3F (0.6098)
10.00	[7.00 2.86 2.80]	P2F (0.6795)	SD (0.5950)	P2F (0.7105)	SD (0.6541)
25.00	[17.50 7.16 7.01]	R (0.9103)	PS (0.6129)	R (0.8643)	PS (0.5956)
50.00	[34.99 14.32 14.02]	LD (0.7931)	PE (0.6845)	LD (0.7861)	PE (0.5690)

Table 4.11: Vase.

4.5 Discussion

Although there exist state-of-the-art grasping algorithms to deal with grasping tasks, the advantage of a logic formalization is to provide more information about specific grasps: this way, grasps can be described qualitatively and can be grounded to data using the same principles discussed in the previous chapter. In addition, grasps can be “personalized” easily depending on the available specific hands and high-level information coming from other sources (e.g. weight of the objects, characteristics of the material etc. which we have ignored) can be integrated in the definition of a grasp. An example of such flexibility can be the case a robot built from separate components, where knowledge about the world can be “plugged in” separately and then adapted to the specific embodiment as a top-down process.

The limitations we faced in this approach are related mostly to the intrinsically qualitative nature of the represented grasps together with their measures and the difficulty in finding qualitative measures whose performances are stable and reliable; moreover, we also faced some problems related to the tools we used (for example, fuzzyDL cannot handle too many axioms, otherwise it crashes).

Taking into account the description logics aspects we have been discussing in Chap. 2, in this application we obtained the following:

- Representation: classes are used for representing the different kinds of grasps, thus they are defined in terms of TBox axioms; instances are used for representing single objects; datatype properties are used for representing the geometric properties of an object.
- Reasoning tasks: realization, instance checking.
- Expressivity: $\mathcal{ALCF}(\mathcal{D})$ (\mathcal{C} is used for disjointness of classes such as `PowerType` and `PrecisionType`, while \mathcal{F} is used for the datatype properties describing different features);
- Storage: custom format due to the extension used (translatable in OWL 2 with annotations with appropriate parsers).
- Queries: custom format due to the extension used, which uses the concept of minimum degree of satisfiability of an instance, i.e. the minimum degree of truth it is inferred to belong to a certain class.
- Extensions: fuzzy extension to the $\mathcal{SHIN}(\mathcal{D})$ logic with fuzzy operators and weighted sums.

4.5.1 Future work

Several extensions are possible for this approach, the first one being the inclusion of more geometric details about objects in order to deal with concave geometries as well and, especially, the integration of other resources related to objects and materials; furthermore, calculating the grasp suitability can be useful to provide an automatic evaluation of the suggested grasps. Finally, a dataset oriented to these experiments, where the measures can be calculated at runtime, should be developed.

Chapter 5

Description logics and cognitive architectures

The field of cognitive robotics is very broad: as its aim is to design artificial systems which are able to act out of structured environments by mimicking human cognitive abilities such as perception, memory and reasoning, many disciplines such as neuroscience, psychology, cognitive science, linguistics and computer science are involved together for producing reliable and efficient models of cognition to be used on robots. The stimuli we receive as human beings are different as they come from different sources, thus, in order to replicate our abilities to make sense of such a wealth of information, robots should be able to integrate them. From this point of view the study of *multimodal information* from a robotic perspective has gained much importance in the last decades; the field is more general than *multisensor fusion* in that the sources of data are not only limited to perception, but rather all the available information sources such as external databases, dialogues with humans and higher level knowledge are combined to provide “intelligent” behaviour.

In this chapter we will explore the use of description logics in a cognitive architecture along with their advantages and their limitations. We will see how several assumptions which hold in other domains cannot be made here, then we will discuss how such limitations can be dealt with and we will show some examples.

5.1 State of the art

Several cognitive architectures have been proposed so far, the oldest and most “traditional” being SOAR [Laird et al., 1987] (which has roots in ar-

tificial intelligence and historically has been a symbolic architecture, but it has been extended to include sub-symbolic processing which are possibly useful for robot control) and ACT-R [Anderson et al., 2004] (which has its roots in cognitive psychology and it has been inspired by theories of human memory).

There are several problems in adopting such architectures in robotics: first of all a deep understanding of a cognitive architecture for practical applications is a difficult task, as it is necessary to understand the basics of human cognition and the theory underlying the adopted models; within the robotic domain, in particular, it is possible to obtain better performances using ad-hoc solution which would not make sense for humans. Furthermore, it is not always easy to generate the “right symbols” or to use low-level information in such architectures: for these tasks additional modules are required. Although there are several efforts in this direction such as [Laird et al., 2012] and [Hanford et al., 2009], the main question remains on whether systems such as SOAR and ACT-R can readily be adapted for robotic applications.

These architectures can be enriched with a *semantic memory*, which can be considered as a long-term concept-based memory containing general knowledge with the aim to provide some capability of reasoning and generalization; most often anyway this module makes only use of lexical information, meaning that it does not provide any grounding or reference to the real objects they refer to but rather to their descriptions. Common knowledge resources such as Cyc [Lenat, 1995] or ConceptNet [Liu and Singh, 2004] are respectively too broad (providing much uninteresting information but not real “common use” knowledge such as *Apples are round and green or red*) or lacking of a formal model (thus not easy to use for drawing inferences).

From a cognitive point of view it is not useful to rely only on symbolic models (using abstract information) on the one hand or statistical models (using big amounts of data) on the other, because both models offer only a partial view of cognition and tend to give less importance respectively to perceptual data and available high-level knowledge; the problem in this case is on how to provide an integration between the two models to take the advantages of both. The problem then becomes how to derive knowledge directly, using modality-specific features obtained from perception and organizing them in a conceptual system.

Perceptual symbol systems [Barsalou, 1999] have been introduced as a model of integration in this direction: in this model perception through sensory-motor systems is stored as patterns of activations in the brain, so that perceptual symbols associated to single perceptual components (such as the color and the shape) can be later retrieved and organized as concepts

in a conceptual system. In this case perceptual information is not just “attached” to symbols by means of a translation in a different language, because this would bring back the *symbol grounding problem* [Harnad, 1990] where a symbol has to be connected back to its perceptual representation and real entities in the world; on the contrary, symbols are amodal and represent subsets of a perceptual state. In order to be effectively usable, such representation should not only *record* perceptual information but also *interpret* it to distinguish specific instances of entities (like instances in the DL domain) from general categories (like classes in DLs).

Perceptual symbols are not to be considered immutable and discrete as they can change over time; as they are componential rather than holistic, in the sense that they regard “parts” of a perception rather than a whole indistinguished perception, they are intrinsically qualitative and need not represent specific entities but can represent classes of such entities. Perceptual symbols are multimodal, including not only all the sensory channels but also introspection and proprioception. The most important characteristic of perceptual symbols is that they offer a different approach to categorization: an instance is decided to belong to a category if it can be “simulated” by the concept related to such category using its own representation: although in [Barsalou, 1999] there are no details on how simulators can be actually implemented, this means that the decision on whether a real entity can be categorized by a certain concept has not to be taken on a logical basis but rather on a low-level basis (e.g. by comparing low-level representations). In this sense, *frames* are considered as “integrated systems of perceptual symbols to construct specific simulations of a category” [Barsalou, 1999, p. 14] and linguistic symbols are used to “index and control simulations” [Barsalou, 1999, p. 16]; this approach has the advantage to be adaptable to any kind of embodiment.

A consequence of this approach is the need for *semantic features*, that is a method to find out what kind of features can actually be derived from perception and used as symbols; such features should be nameable in order to be communicated to other people, but they should not be decided “a priori”. The purpose of works such as [Vinson and Vigliocco, 2008] and [Vatakis et al., 2012] is to obtain such features from humans when they are free to decide how to describe words or unknown objects; this way there is no bias from the designer of the system and it can be evaluated which features are considered more important and discriminative. In fact, the identification of useful and discriminative semantic features is an open research issue: as we mentioned before, this problem has been faced in the computer vision community as well (see e.g. [Farhadi et al., 2009]). An investigation on

the plausibility of semantic features through functional magnetic resonance imaging (fMRI), a technique for studying the internal behaviour of the brain and its activity, has been conducted in [Chang et al., 2011].

The problem of obtaining knowledge from perception and experience has been faced many times: Kuipers' works such as [Modayil and Kuipers, 2007, Kuipers, 2008] are focused on how to obtain concepts from a dynamical system perspective, while works such as [Schyns et al., 1998] and [Yee et al., 2011] focus on behavioural models and works such as [Griffith et al., 2012] and [Coelho et al., 2001] focus on the developmental perspective.

A different approach for integration is given by neurosymbolic research: in works such as [de Penning et al., 2011, Röhrbein et al., 2007, Kollia et al., 2010] the learning capabilities given by a connectionist approach such as neural networks are bridged with a symbolic layer through knowledge interpretation (i.e. representation of formal knowledge statements as neural networks).

On the other hand, the study of language together with action and perception has seen a huge research effort in the last decade (see for example [Glenberg and Kaschak, 2002, Cangelosi et al., 2010, Salvi et al., 2012, Stramandinoli et al., 2012, Caligiore and Fischer, , Pastra et al., 2011, Pastra and Aloimonos, 2012] and Sec. 5.1.1), where the perspective is focused on the integration of the language with perception and action within a unified cognitive model.

5.1.1 The POETICON++ project

The POETICON++ project¹ has as its main aim to use natural language as a means for robots to generalize both learned and novel behaviours and perceptions; its core database called PRAXICON (see [Pastra et al., 2010, Pastra, 2010, Pastra, 2008]) has the role of bridging a conceptual structure to low-level sensorimotor representations through the use of *embodied concepts*.

The PRAXICON is a semantic memory, thus it contains factual information and general knowledge; it is neither a procedural nor an episodic memory. Being an action-centric framework, it is centered on the definition of actions in terms of other related concepts such as movements and objects and their linguistic and sensorimotor representations; its main source of lexical information is WordNet, used together with language-related tools

¹<http://www.poeticon.eu>

(such as the ones used within the COSMOROE project²) and other sources of information including the results of cognitive experiments (such as the ones described in [Vatakis et al., 2012]).

Our contribution to the project was the development, evaluation and discussion of a knowledge base built on top of the PRAXICON database. The advantage in the use of a knowledge representation approach is its ability to formalize and provide standard reasoning services to a conceptual system; anyway, as it will be discussed later, a formal approach of this kind has several conceptual and technical shortcomings and proposes some new challenges.

5.2 Issues in semantic memory design

One of the problems in designing a semantic memory is related to the aspects a concept is supposed to capture: if modal and amodal representations have to coexist, a way to integrate and organize them should be provided. A representation making use of perceptual symbols should exhibit both a conceptual structure and a suitable way to represent perceptual (sensorimotor) information; although a knowledge representation framework can be a natural choice for representing concepts and their relations and for integrating other sources of knowledge, several aspects have to be considered:

- How are concepts and relations among them acquired or generated?
- How are concepts organized? Do they create a taxonomy?
- How are concepts related to perceptions? Which representation do perceptions use?
- What level of information can be represented with such kind of language?

As we have seen, when knowledge is derived from “common sense” there might be a problem related to typicality and exceptions: as we have seen in Chap. 2 description logics do not allow for “partial inheritance”, hence classes cannot be used directly for representing concepts. Furthermore, information expressed in natural language is intrinsically vague: for dealing with it, language-based representations such as fuzzy logic might help to build concepts in a more natural way.

²<http://www.cosmoroe.eu>

The way concepts are related to perceptions from an engineering point of view depends on the implementation: neural networks, for example, can provide different layers of abstraction, but the results at each level are difficult to understand; neurosymbolic approaches offer a more direct way to encode symbolic knowledge using a connectionist approach, but they most often do not make use of “simple” neurons but rather of more complex structures.

Regarding the level of information which can be represented, a possible problem for example lies in the comparison between features: while a general solution involving concepts such as “comparison type” and “dimension of comparison” is more general (and it is under investigation within the PRAXICON project), a DL-inspired solution (i.e. the use of specific properties) can be more easily implemented: as an example, when comparing two colors for deciding which one is “more red”, in the first case the comparison would be of the type “more than” and the dimension would be the red channel in the RGB representation, while in the second case a `moreRedThan` property would be needed. The choice depends on the level at which the comparison is performed, in this case on whether the equality check is performed within the ontology (e.g. using SPARQL queries as in the examples we will discuss later on) or not.

5.3 Implementation

In the following our implementation of the PRAXICON database will be presented. As the term “concept” constitutes the core of the PRAXICON, in order to avoid confusion we will be referring to PRAXICON concepts as “concepts” and to DL concepts as “classes”.

5.3.1 PRAXICON overview

The PRAXICON is essentially a concept-centric database: it represents concepts as entities having relations with other entities, be them relations with other concepts, with attributes or with different modalities. As it has been mentioned in the introduction, such relations define both the conceptual structure of the memory and its embodied perceptual attributes; this integration aims at including necessary and sufficient information for the reconstruction of every concept, which means that a concept should be retrieved by giving its representation in terms of language or of any other modality. It is important to note that not every concept is (and has to be) grounded: some concepts may lack of a linguistic, visual or motoric representation, as it happens in the case of *movement-object complexes*, which are concepts

representing a specific movement performed with a specific tool but lack of a specific linguistic representation.

The PRAXICON includes a reasoner (not to be confused with a semantic reasoner) which computes scores on the possible outcomes of the *generalization* process, which is one of the core functions of the PRAXICON: if a concept is related to another concept via a “container-content” relation (e.g. the concept of “cup” is related to the concept of “coffee” because a cup can contain coffee), the former can be substituted to the latter in a search task (e.g. while a robot is looking for “coffee” the reasoner suggests to look for a “cup” instead). This process improves the “creativity” of an autonomous system by providing it an intelligent way of reacting to the failure of a task. The PRAXICON reasoner computes a score on these alternative solutions depending on semantic closeness of concepts (the length of the path linking them), the presence of all the needed relations in an intersection of relations, the substitution of concepts with other concepts (such as the use of *coffee cup* in place of *coffee* thanks to a relation container-content) and so on. As such heuristics are still under development, in our experiments we will only show what the available answers are and discuss about their validity.

Concepts in the PRAXICON can have four types, namely **Entity**, **Movement**, **Feature** and **Abstract**; also, the idea of *basic level* as a “low level of abstractness” is introduced: an abstract concept can be a basic level concept if it derives (directly or indirectly) from an entity or a movement concept, otherwise it is a non-basic level concept. The relations among concepts can be of different types, among which:

- between entities (type-token, part-whole, container-content, ...);
- between movements (type-token, step-process, ...);
- between descriptive features (type-token, part-whole, ...);
- between entities and movements (object-action, artifact-use, ...);
- between entity or movement and their descriptive features (has_shape, has_size, has_direction, has_velocity, ...).

The type-token relation is used to define a taxonomy among the concepts, so an axiom such as *A* type-token *B* means that the entity/movement/feature *A* is a “father concept” for *B*. Relations can be arranged in complex ways (see Fig. 5.1), namely in relation chains (similar to property chains in description logics), intersections of relations (relations which have to hold at the same time) and unions of intersections (possible alternatives among intersections

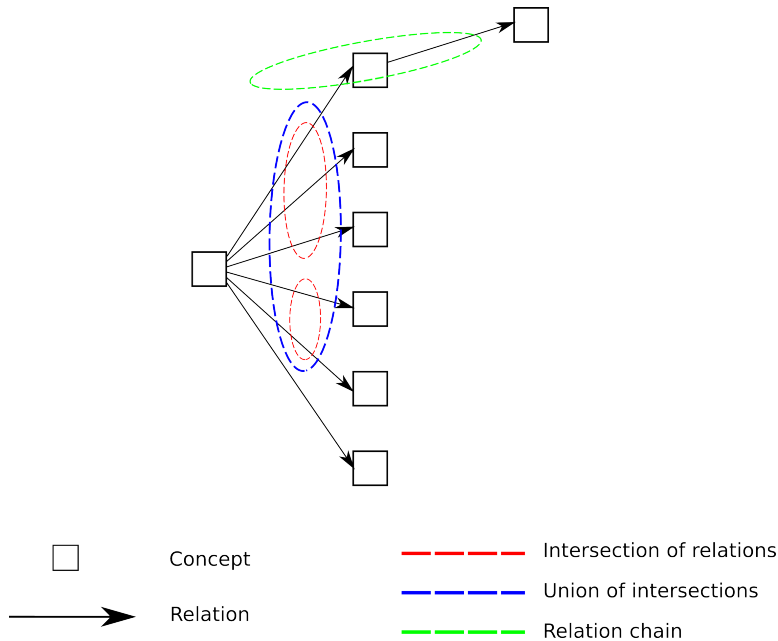


Figure 5.1: Example of PRAXICON relations and their arrangement.

of relations). A combination of relations can be *inherent* if it defines a necessary and sufficient condition for the entity it refers to (for example, an intersection of three relations of type *artifact-use*, *action-object*, *action-purpose* is inherent for movement concepts as they both imply each other).

A movement concept in the PRAXICON is represented as the combination of three concepts: the *purpose* of the movement (e.g. cutting), the used *tool* (e.g. a knife) and the affected *object* (e.g. the bread). The movement concept has an associated label such as *Cut with knife the bread* referring to these three concepts, and it is related to them by the three relations *action-purpose*, *use-artifact* and *action-object*.

For more and updated information on the PRAXICON structure, refer to the cited publications and to the project Web site.

5.3.2 Requirements

The formalization of the PRAXICON using DLs make use of instances rather than classes for representing concepts because:

- Many classes (ca. 80,000) make it difficult to use a standard reasoner; the only reasoner which can handle such number of classes is Snorocket, but it reasons on the OWL 2 EL profile, thus several constructs such as union and inverse relations are not available; furthermore, ABox

axioms and concrete datatypes are not available.

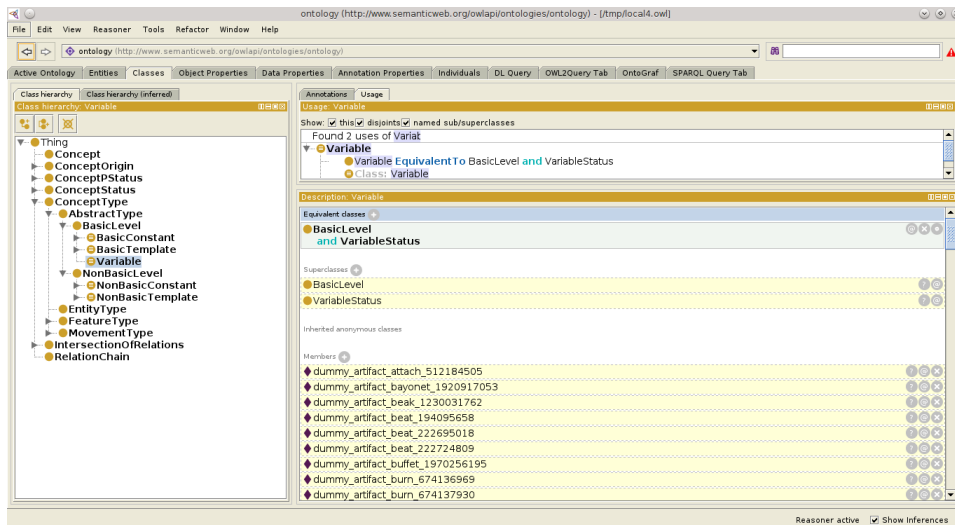
- Partial queries on the concepts for checking the “degree of satisfaction” (such as in intersections of relations, when not necessarily all of them should hold for result scoring purposes) should be possible; although this is possible with TBox axioms using “incremental intersections” of concepts, this is less practical.
- It is easier to extend concepts definitions and to create/modify them automatically using relations with new instances, while restrictions on classes are more difficult to create and maintain especially when they are nested.
- It should be possible to “count” things (e.g. the number of relations which hold etc.), i.e. a (weak) form of CWA should be available for computing scores.

More in detail, we added these constraints to the original database:

- only AbstractType concepts can be basic or non basic (for concrete concepts this distinction does not have much sense); also, only non basic abstract concepts might have no origin;
- a concept is “fully abstract” if it is non basic and it does not have any origin;
- the type of a template derives from the type of the associated movement concept;
- entities and features cannot be templates, which are reserved for movements;
- concepts having origin in an entity can have the status of a variable or a constant, while concepts having origin in a movement can have the status of a constant or a template and concepts with no origin can only be constant.

The concepts in the DL implementation have been renamed using a hash of the compound representation present in the original database as obtained from WordNet; for example, the concept *butter_knife%1:06:00::* becomes *butter_knife_1477250444* in the ontology.

Object properties are used to represent relations between concepts. As we discussed in Chap. 2, the token-type relation can be used as a subclass



axiom: although it would need to be transitive and to have other characteristics, this is not necessary to be formalized if SPARQL property paths are used. We added a role hierarchy with a general top property `relatedTo` as an ancestor property of all the PRAXICON relations, a `HasFeature` super-property covering concept features (having entities as a domain and features as a range) and its inverse `FeatureOf`; by adding domain and range axioms, subclasses of features are automatically created from the PRAXICON properties (e.g. the feature class `Shape` is created as the range of the `HasShape` property) so that PRAXICON feature concepts can be classified using more specific classes.

We used OWLIM 5.2³ as our semantic repository because it is suitable for big ABoxes, provides reasoning within OWL 2 RL and QL (thus supporting the constructs we need) and supports SPARQL 1.1 queries.

5.4 Experiments

As a visualization and testing bench we used Protégé 4.2 with a small version of the PRAXICON; the conversion of the database in an ontology has been performed with the OWLAPI library⁴.

In order to show the generalization capabilities, we ran several SPARQL queries against the ontology. Although we worked on the topic of semantic interpretation of sentences [Vitucci et al., 2012] and one of the PRAXICON modules is devoted to the understanding of a request expressed in natural

³<http://owlim.ontotext.com>

⁴<http://owlapi.sourceforge.net>

language, for the time being we ignore the syntactic and semantic analysis of the sentence, thus focusing on the structure of the knowledge base. The experiments conducted so far are composed of:

- a verbal request for an action, such as *Spread the butter with a knife*, expressed as a triplet $\langle movement, tool, object \rangle$ where *tool* and *object* can be empty;
- a set of objects in the scene, whose type, depending on the labels provided by the visual level output, can be specific (e.g. butter knife, tea spoon) or generic (e.g. knife, spoon), although the different levels of details should not be mixed;
- an expected decision in the form of a triplet $\langle movement, tool, object \rangle$ defining the kind of movement to execute, the tool to use and the object affected by the action (e.g. $\langle spread, butter, butterknife \rangle$).

In our experiments we suppose the vision system can provide any level of detail to match entries in the PRAXICON knowledge base (e.g. it can find teaspoons, not only spoons). Furthermore, we do not currently score solutions as the definition of scoring criteria making use of the distance between concept, the inherence of a relation and other heuristics is still under development; instead, we list all the possible solutions.

Spread the butter with a butter knife We search for the three lemmas *spread*, *butter* and *butter knife* as string values for the datatype property text, obtaining the instances `spread_456537824`, `butter_519714614`, `butter_knife_1477250444`; looking for the possible intersections of relations containing the three of them, the concept `spread_971071545` related to the movement of “spreading the butter with a butter knife” is retrieved and the search ends with the result $\langle spread, butter, butterknife \rangle$, which is a single movement already known by the robot; this is anyway a less common (and interesting) case as all the needed information is already known within the query, thus no generalization is needed. The corresponding SPARQL query is:

```
PREFIX [...]
SELECT DISTINCT ?n0
WHERE {
  ?n1 :text "spread"^^xsd:string . ?n1 :inChain ?c1 .
  ?n2 :text "butter"^^xsd:string . ?n2 :inChain ?c2 .
```

```

    ?n3 :text "butter knife"^^xsd:string . ?n3 :inChain ?c3 .
    ?c1 :isChainOf ?ior . ?c2 :isChainOf ?ior . ?c3 :isChainOf ?ior .
    ?ior :isIntersectionOf ?n0 .
}

```

The concepts are searched using the three lemmas which are stored in the variables ?n1, ?n2 and ?n3; these three concepts should belong to three relation chains ?c1, ?c2 and ?c3 belonging to the same intersection of relations ?ior. From such intersection the movement concept is found and stored in the variable ?n0.

Spread the butter with a knife If the previous query is used (after substituting `butter knife` with `knife`), this time no results are obtained: there is no direct relation between the action of spreading the butter and a “generic” knife. The search should be extended using a generalization of the tool *knife* (looking for both ancestors and successors of the concept), in order to find out that such a movement exists when it is performed with a butter knife, making it a possible substitute. In this case the only information which has been used is related to the taxonomy of the objects (namely, property paths with zero or more occurrences of the relations `TokenType` and `TypeToken` have been used). The corresponding SPARQL query is similar to the previous one but, as the previous one does not return any results, it is automatically extended:

```

PREFIX[...]
SELECT DISTINCT ?n0
WHERE {
    ?n1 :text "spread"^^xsd:string . ?n1 :inChain ?c1 .
    ?n2 :text "butter"^^xsd:string . ?n2 :inChain ?c2 .
    ?n3 :text "knife"^^xsd:string .
    {?n4 :TokenType* ?n3} UNION {?n4 :TypeToken* ?n3} .
    ?n4 :inChain ?c3 .
    ?c1 :isChainOf ?ior . ?c2 :isChainOf ?ior . ?c3 :isChainOf ?ior .
    ?ior :isIntersectionOf ?n0 .
}

```

Differently from the previous case, the concept found by the lemma *knife* is generalized using property paths; in fact, `:TokenType*` stands for “all the individuals found in a chain of `:TokenType` relations” (to find more

general concepts, as the token-type relation is a sort of *isA* relation) and `:TypeToken*` stands for “all the individuals found in a chain of `:TypeToken` relations” (to find more specific concepts).

Spread the butter with a dull/small object Here *dull* is not an entity but a feature, thus the search has to find entity having such feature. The feature concepts having as linguistic representation *dull* and *small* are searched, then concepts related to such features through any subproperty of the property `HasFeature` are searched: once again the butter knife is found, along with the *exact* relations linking it to its features (in this case `HasShape` and `HasSize` respectively). The corresponding SPARQL query is:

```
PREFIX [...]
SELECT DISTINCT ?n0
WHERE {
  ?n1 :text "spread"^^xsd:string . ?n1 :inChain ?c1 .
  ?n2 :text "butter"^^xsd:string . ?n2 :inChain ?c2 .
  ?n3 :text "dull"^^xsd:string . ?n4 ?r ?n3 .
  ?r rdfs:subPropertyOf :HasFeature . ?n4 :inChain ?c3 .
  ?c1 :isChainOf ?ior . ?c2 :isChainOf ?ior . ?c3 :isChainOf ?ior .
  ?ior :isIntersectionOfRelationsOf ?n0 .
}
```

The difference is in that the concept related to the lemma *dull* is not used directly, but it is used for finding the concepts which are related to it through the `HasFeature` relation (because it is a feature, not an entity).

Cut the turkey wing This is an interesting example because it shows how to generalize using a relation not with a feature but with another entity (the turkey wing is a part of a turkey, though the relation `HasPart` has to be used). For finding a concept linking *cut* and *turkey*, both the possible objects for *cut* and *turkey* itself have to be generalized: the only concept having both a definite object and a tool is *Cut the staff of life with bread knife*, and the only common ancestor for *turkey* and *staff of life* is *solid food*, which is a non-basic level abstract concept together with its descendants *meat*, *bird* and *poultry*. In this case the PRAXICON heuristics would not go up all the chain, because as we have seen it would stop at the first basic level abstract concept or, if needed, at its father concept. The corresponding SPARQL query is similar to the previous ones, except for it makes use of more property paths:

```

PREFIX [...]
SELECT DISTINCT ?n0
WHERE {
  ?n1 :text "cut"^^xsd:string . ?n1 :inChain ?c1 .
  ?n3 :text "turkey wing"^^xsd:string . ?n3 :PartOf ?n5 .
  {?n6 :TokenType* ?n5} UNION {?n6 :TypeToken* ?n5} .
  ?n6 :inChain ?c3 .
  ?n0 :ActionObject ?n10 .
  {?n11 :TokenType* ?n10} UNION {?n11 :TypeToken* ?n10} .
  {?n11 :TokenType* ?n6} UNION {?n11 :TypeToken* ?n6} .
  ?c1 :isChainOf ?ior .
  ?ior :isIntersectionOf ?n0 .
}

```

Stir the soup with available objects Teaspoon, Screwdriver, Wrench In this case, for using physical properties of objects, it is necessary to know the typical measures for every object (it is possible both to use numerical measures and discretized quantities); we obtain such features along with numerical measures from the object ontology described in Chap. 3.

In Fig. 5.2 the 3D models for the three objects are shown, while in Tab. 5.1 their typical real sizes are listed for comparison purposes; the first model is not present in the Princeton Dataset, so we added it from another source along with its typical measures. The only movement concept containing both *stir* and *soup* is *Stir the soup with a spoon*; while the teaspoon is semantically very close to the spoon required for stirring, if the dimensions of the object are more important, a screwdriver and a wrench will be “closer” to a (serving) spoon than a teaspoon, provided that the versions available in the scene have similar dimensions; in this case, the comparison between the objects is checked on the low-level (metric features) with a level of “closeness” c such that two objects are similar if $(dim_{1,i} - dim_{2,i}) < c \forall i = 1, 2, 3$.

The problem of deciding the weights to give respectively to semantic closeness and geometric closeness is difficult and it is still under research, thus for the time being we provide both the solutions without scoring them; although we think the score depends on the task to execute, we believe that a further step of knowledge extraction from examples or the inclusion of common sense knowledge is required: in this example, for instance, it might help to know that *For stirring movements, long objects are needed*



Figure 5.2: Object models.

object	length	width	height
spoon	20.003	4.309	3.6225
	20.161	4.3432	3.6513
	21.273	4.5826	3.8525
teaspoon	14.922	3.2147	2.7025
	15.558	3.3515	2.8175
	15.875	3.4199	2.875
screwdriver	9.525	1.1009	1.0054
	22.542	2.6055	2.3794
	27.543	3.1835	2.9072
wrench	17.1	4.6077	1.3063
	21.9	5.9011	1.673
	23.1	6.2245	1.7646
	27.9	7.5178	2.1313

Table 5.1: Objects real sizes.

(this information can possibly be extracted analyzing the semantic features of the objects used for stirring, if they are available for all of them) or that *Tools are dirty* and *Dirty objects cannot be used with food* (common sense knowledge, more difficult to obtain and formalize). The difficulty here is on the decision about “how far” the generalization can go: is a wrench an acceptable solution in this case? What score should be assigned to such solution?

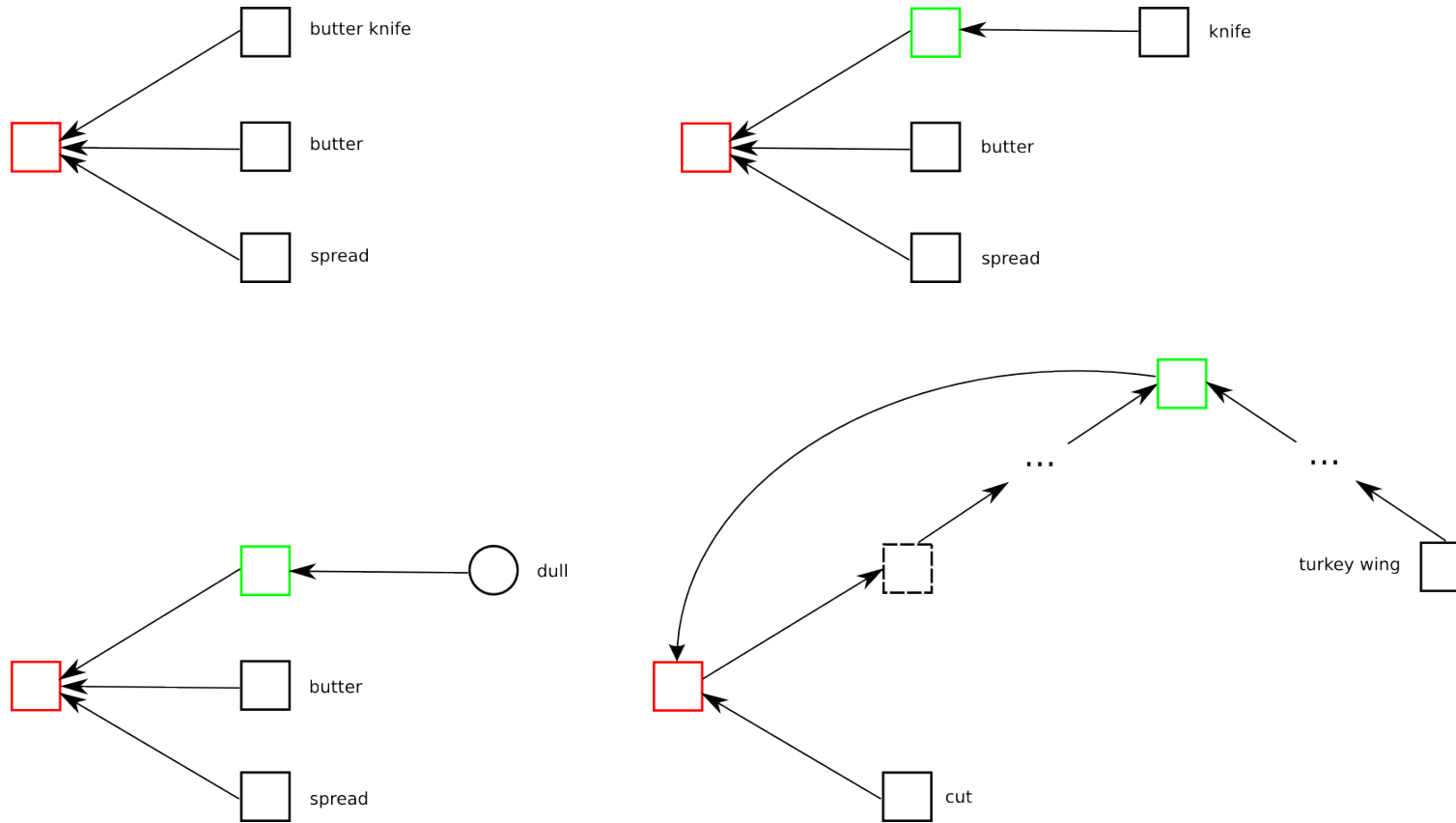


Figure 5.3: Graphical representation of the results of the queries. The squares denote entities and abstract concepts while circles denote features; the black color denotes the input concepts, while the red color denotes the movement concept to be given as an output, the green color denotes an intermediary concept that has to be found first and the dotted line denotes an intermediary concept.

5.5 Discussion

In this chapter we have presented a possible implementation of a semantic memory to be used within a cognitive architecture, building on the results and current efforts of another research project, to discuss about the problems affecting description logics and the opportunities offered by the use of powerful semantic query languages and scalable triple stores. Following the results from Chap. 2, the aspects related to a DL representation have been evaluated:

- Representation: the use of classes for PRAXICON concepts is not a suitable alternative for several reasons, namely the typicality/exceptions issues, the difficulty in building, storing and reasoning on expressive axioms when they are too many. More details later.
- Reasoning tasks: instance checking, instance retrieval.
- Expressivity: $\mathcal{ALHI}(\mathcal{D})$ (datatypes properties are currently used for linguistic representations).
- Storage: OWLIM 5.2, system capable of performing inference on OWL constructs via a rule-based reasoner while supporting a high number of instances (tens of millions of statements).
- Queries: using SPARQL 1.1 (taking advantage of property paths for taxonomic relations).
- Extensions: no extension used.

The main problems we dealt with in this chapter are related to the difficulty of using the description logic formalism “as is” in a cognitive context, because a conceptual approach making use of language and perceptions is prone to different interpretations (as we have seen, for example, in the meaning of the *isA* relation and in the representation and use of the perceptual features of the objects). While a representation making use of RDF graphs offers a solution both to representation and to querying problems, and it lets the knowledge base to be easily extended, some of the advantages of a “pure DL” solution (working mostly or only with a TBox) such as automatic classification are lost. The advantage of structuring information as a graph with respect to a classical database structure is evident: not only the graph structure of queries is more natural because of the use of multiple relations, but also the integration of different sources of information (such as in the case of PRAXICON and the object ontology) is easier; the most important

drawback of not using a database representation lies in the lack of attributes on relations, which make the representation heavier as the relations have to be reified and new “useless” individuals have to be introduced.

5.5.1 Future work

The realization of the PRAXICON is one of the aims of a current research project, therefore it is still under development and many extensions of the database have already been planned or are already under development; nevertheless, on the description logic perspective, several improvements can be made.

On the representation side, the concepts of intersection, relation chain and so on can be generalized as different combinations of relations in order to offer a more flexible way of formalizing such “relations between relations”; furthermore, a way to include common knowledge statements such as *Cups contain liquids* or *A spoonful is the volume of a chunk of butter* has to be found as it would greatly improve the PRAXICON reasoner’s generalization capabilities. The integration of other sources of information carrying different levels of representation is under investigation: we provided an example making use of our object ontology and we are currently studying how to integrate the grasp ontology as well, in order to realize a logic platform for dealing with the grasping problem. From the graphical point of view, as a way to let the PRAXICON be used by humans as well, a knowledge base graphical explorer can be developed on the model of the existing database navigator.

Chapter 6

Conclusions and future work

In this thesis we have reviewed the use of semantic technologies for applications and tasks related (and not limited) to the robotic domain; more in details, we have studied the issues related to the design of a knowledge base by means of languages based on description logics and its use with semantic query languages, the challenges this approach poses when used with information gathered from different levels of data and several possible solutions supported by examples. At the same time the “atypical” domain we have studied provided a testbench for such technologies, highlighting some interesting issues such as the scalability of a knowledge base, the efficiency of different representations, the power of the new SPARQL specifications and the performances of different reasoners while using different fragments of OWL.

From a robotic-related point of view we obtained as a result that semantic technologies can be used for linking different representations, and that it is possible to give semantics to low-level information albeit with some limitations. We have pointed out that description logics alone are not sufficient (or not suitable at all) for some tasks, and in some cases we found more suitable techniques for similar applications. In our work we dealt with three different levels of data, from the raw data used in grasping experiments to part decomposition in object modelling, then to high level concepts in the design of an embodied ontology.

The unifying thread of the thesis is the search for a balance between the expressivity of a knowledge representation language and the scalability required by a real world application, depending on the specific task at hand. More specifically, we have seen that a task such as 3D object modelling and retrieval, where the advantage of a semantic representation relies mostly on the graph-like representation of a shape than on the object’s character-

istics, has low expressivity needs and requires quite little reasoning effort, provided that the used storage is efficient and scalable and that a powerful query language is available. In this case the application might use a traditional database as well, but many facilities offered by a graph-based representation (such as the use of property paths) would then be lost and should be implemented in the business logic part.

On the other hand, a task such as grasp type selection has much higher expressivity needs because the formalized grasp types are “few” and do not change, thus they can be described in a more detailed way and the matching task can be reduced to an instance check reasoning service; moreover, as this task is “more robotic” as it draws its data directly from real-world numeric data, the higher expressivity given by fuzzy logics helps such representation to be more flexible and to take account of the vagueness of qualitative data. In this case there is no need of a complex query language or of a high degree of scalability, provided that a reasoner which can be used on such expressive logic is available.

Finally, the complexity of a cognitive architecture requires in a sense a compromise between expressivity and scalability: if the aim is to obtain a model of human reasoning capabilities and concept formation, it needs to be expressive enough to capture the richness of human descriptions (and as we have seen, up to a certain extent, the nuances of the language), but on the other hand it has to be scalable in order to deal with the huge number of concepts and relations among them. In some cases a “standard” language might not be well suited for expressing some cases (e.g in the case of exceptions), thus more ad-hoc solutions can be necessary. To summarize, this is the case in which more attention should be paid in selecting the tools to use, as they depend on each other and all together determine the extendability of the chosen approach.

Within the robotic domain scalability might be a concern only up to a certain extent; a robot typically does not have an unlimited number of skills, therefore it does not need to store and use huge amounts of different kinds of information. On the other hand, resources like object databases and semantic maps tend to grow fast as new dataset become available; furthermore, as the Web opens up the chance of using knowledge on-demand and semantic technologies make such resources available to and usable by automatic agents as well, scalability should be taken into account as a way to let robots exploit this wealth of information. In this scenario it might become more important to use different sources for validating existing knowledge rather than for adding new data, which is a promising research direction in all fields making use of huge sets of data.

6.1 Comparison with KnowRob

The KnowRob project, to the best of our knowledge, is the project having the most similar aims to ours. As it is available as an open source resource, we can make an in-depth comparison of the two approaches.

The KnowRob ontology¹ makes use of a *SHOIF(D)* logic, i.e.:

- \mathcal{S} : transitive properties (such as `parts`), full existential quantification (such as in the class `FoodAndBeverageOrganization`, which is a subclass of `∃hasDaytimeOpeningHours.DaytimeOpeningHours`) and disjunction (such as the class `BlackColor`, which is disjoint from the other subclasses of the class `ColoredThing`);
- \mathcal{H} : there exists a property taxonomy (e.g. the property `nextDetectionOfObject` is a subproperty of the property `nextEvent`, which in turn has the property `postEvents` as its superproperty);
- \mathcal{O} : some classes are defined as collections of instances (e.g. `Dirtyness` is an enumerated class containing instances `Dirty`, `ALittleDirty`, `Clean` and `Sterile`) and some properties are restricted to a single value (e.g. the class `ClosingSomething` has as a restriction class on the property `fromState` the instance `ObjectStateOpen`);
- \mathcal{I} : several properties have inverses (e.g. the inverse of the property `nextEvent` is `previousEvent`);
- \mathcal{F} : several properties have been declared functional (e.g. the properties `nextPointOnArmTrajectory` and `nextPointOnBaseTrajectory`).
- (\mathcal{D}): numbers and strings are used as concrete datatypes (e.g. the property `matrixElement` has floating-point numbers as range).

Furthermore, the SRDL2 ontology makes use of the \mathcal{Q} construct because of the axiom `ComponentComposition ⊑ ∃endLinkOfComposition.UrdfLink ⊓ = 1baseLinkOfComposition.UrdfLink`. Although OWL is used to represent knowledge formally so that classical reasoners such as Pellet and HermiT are used, the reasoning and storage/retrieval tasks are demanded to Prolog modules.

The main differences with KnowRob can be summarized as follows:

1. our work is more focused on ways to give semantics to and to execute queries on lower level information, while KnowRob is less interested

¹Downloaded from <http://ias.cs.tum.edu/kb/knowrob.owl> on July 10, 2012.

in a semantic representation of objects but more in the definition of processes, events and semantic maps;

2. KnowRob uses Prolog as a framework for dealing with RDF triples, while we use a triplestore equipped with a rule-based reasoner and a SPARQL query engine;
3. KnowRob uses SRDL for representing a robot’s model, while we focus on a more detailed logical representation based on grasp geometry for manipulation purposes; in fact, while within KnowRob the representation of objects in terms of geometric properties only is (correctly) considered not to be sufficient for grasping tasks, we explore a possible way to make use of such information (i.e. to match grasps and shapes);

while the main similarities are:

1. the general aim to build a modular architecture where different sources of information can be integrated and used to support each other;
2. the use of the concept of action class (a concept denoting a single movement);
3. the use of qualitative concepts and relations such as the “dirtiness” of a place in KnowRob and the “length” of an object in our discretized ontology;
4. the use of structural information such as the robot structure in KnowRob and the object structure in our ontology;
5. the use of numerical quantities to describe objects’ properties.

6.2 Future work

Our efforts of integration are directed to the realization of a unified framework to be used as a resource for cognitive robotics applications. Although building a complete system is a long-term objective of several current research projects and some results look rather promising, the path towards a complete architecture is still long. In this thesis we evaluated several components for such an architecture separately, each one with its peculiarities and challenges, and we used description logics as an interface. In order to be able to use the semantic memory provided by the PRAXICON to generalize over lower level concepts related to grasping abilities and object shapes, a better coverage is needed and the mapping methods between the different

modalities have to be evaluated carefully, both from a cognitive and from a technical point of view.

Several improvements can be made on each of the topics we discussed:

- automatic shape decomposition algorithms can be evaluated more deeply to make the acquisition of 3D models and the inclusion of their shape decomposition representation more scalable and robust;
- the evaluation of grasp suitability for each of the grasps suggested for an object can provide not only a quantitative way to evaluate such suggestions, but also additional information to be stored in the representation of an object (e.g., additional properties for an object can be related to the best part to be grasped and on the kind of grasp to use in that case);
- the representation of complex relation structures can be improved and become more flexible in order to be easily extended; furthermore, in order to improve the generalization capabilities, the use of external resources (such as the semantic role labelling module and the ontology of objects) has to be formalized completely and a suitable scoring function has to be developed.

Usable versions of the software realized for this thesis, used for building an ontology from object descriptors, for the guided discretization of datatype properties, for the automatic writing of structural queries from graphs and for the production fuzzyDL axioms from raw data are still under development and we plan to release them in the future.

Bibliography

- [Aleotti and Caselli, 2010a] Aleotti, J. and Caselli, S. (2010a). Grasp programming by demonstration in virtual reality with automatic environment reconstruction. *Virtual Reality*, pages 1–18.
- [Aleotti and Caselli, 2010b] Aleotti, J. and Caselli, S. (2010b). Grasp synthesis by 3D shape segmentation using Reeb graphs. In *IROS 2010 Workshop on grasp planning and task learning by imitation*.
- [Aleotti and Caselli, 2012] Aleotti, J. and Caselli, S. (2012). A 3D shape segmentation approach for robot grasping by parts. *Robotics and Autonomous Systems*, 60(3):358–366.
- [Anderson et al., 2004] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111:1036–1060.
- [Anguelov et al., 2004] Anguelov, D., Koller, D., Pang, H.-C., Srinivasan, P., and Thrun, S. (2004). Recovering articulated object models from 3D range data. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, UAI '04, pages 18–26, Arlington, Virginia, United States. AUAI Press.
- [Attene and Biasotti, 2011] Attene, M. and Biasotti, S. (2011). Geometric models with weighed topology. *Computers & Graphics*, 35(3):542 – 548.
- [Attene et al., 2007] Attene, M., Robbiano, F., Spagnuolo, M., and Falciديو, B. (2007). Semantic annotation of 3D surface meshes based on feature characterization. In *Proceedings of the semantic and digital media technologies 2nd international conference on Semantic Multimedia*, SAMT'07, pages 126–139, Berlin, Heidelberg. Springer-Verlag.
- [Attene et al., 2009] Attene, M., Robbiano, F., Spagnuolo, M., and Falciديو, B. (2009). Characterization of 3D shape parts for semantic annotation. *Comput. Aided Des.*, 41(10):756–763.

- [Baader et al., 2005] Baader, F., Brandt, S., and Lutz, C. (2005). Pushing the \mathcal{EL} envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05*, Edinburgh, UK. Morgan-Kaufmann Publishers.
- [Baader et al., 2008] Baader, F., Brandt, S., and Lutz, C. (2008). Pushing the el envelope further. In Clark, K. and Patel-Schneider, P. F., editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*.
- [Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, NY, USA.
- [Baader and Hollunder, 1995] Baader, F. and Hollunder, B. (1995). Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14(1):149–180.
- [Barck-Holst et al., 2009] Barck-Holst, C., Ralph, M., Holmar, F., and Kragic, D. (2009). Learning affordance relations for robotic grasping using probabilistic and ontological approaches. In *International Conference on Advanced Robotics*, Munich, Germany.
- [Bard et al., 1991] Bard, C., Troccaz, J., and VerCELLI, G. (1991). Shape analysis and hand preshaping for grasping. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 64–69 vol.1.
- [Barsalou, 1999] Barsalou, L. W. (1999). Perceptual symbol systems. *The Behavioral and brain sciences*, 22(4):577–609; discussion 610–60.
- [Bekey et al., 1993] Bekey, G., Liu, H., Tomovic, R., and Karplus, W. (1993). Knowledge-based control of grasping in robot hands using heuristics from human motor skills. *Robotics and Automation, IEEE Transactions on*, 9(6):709–722.
- [Bennett, 2002] Bennett, B. (2002). Physical objects, identity and vagueness. In Fensel, D., McGuinness, D., and Williams, M.-A., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Eighth International Conference (KR2002)*, pages 395–406, San Francisco, CA. Morgan Kaufmann.

- [Bennett, 2011] Bennett, B. (2011). Spatial vagueness. In Jeansoulin, R., Papini, O., Prade, H., and Schockaert, S., editors, *Methods for Handling Imperfect Spatial Information*. Springer-Verlag.
- [Bennett et al., 2000] Bennett, B., Cohn, A. G., Torrini, P., and Hazarika, S. M. (2000). A foundation for region-based qualitative geometry. In Horn, W., editor, *Proceedings of ECAI-2000*, pages 204–208.
- [Biederman, 1987] Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147.
- [Bilodeau and Bergevin, 2003] Bilodeau, G.-A. and Bergevin, R. (2003). Structural indexing extended to fuzzy graphs of 2D parts. In *International Conference on Image and Signal Processing (ICISP 2003)*, pages 61 – 68.
- [Bloch, 2006] Bloch, I. (2006). Spatial reasoning under imprecision using fuzzy set theory, formal logics and mathematical morphology. *International Journal of Approximate Reasoning*, 41(2):77–95.
- [Bobillo and Straccia, 2008] Bobillo, F. and Straccia, U. (2008). fuzzydl: An expressive fuzzy description logic reasoner. In *2008 International Conference on Fuzzy Systems (FUZZ-08)*, pages 923–930. IEEE Computer Society.
- [Caligiore and Fischer,] Caligiore, D. and Fischer, M. Vision, action and language unified through embodiment. *Psychological Research*, pages 1–6. 10.1007/s00426-012-0417-0.
- [Calvanese et al., 2007] Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. (2007). Tractable reasoning and efficient query answering in description logics: The DL-Lite family. 39(3):385–429.
- [Cangelosi et al., 2010] Cangelosi, A., Metta, G., Sagerer, G., Nolfi, S., Nehaniv, C., Fischer, K., Tani, J., Belpaeme, T., Sandini, G., Nori, F., Fadiga, L., Wrede, B., Rohlfing, K., Tuci, E., Dautenhahn, K., Saunders, J., and Zeschel, A. (2010). Integration of action and language knowledge: A roadmap for developmental robotics. *Autonomous Mental Development, IEEE Transactions on*, 2(3):167 –195.
- [Catalano et al., 2011] Catalano, C., Mortara, M., Spagnuolo, M., and Falcidieno, B. (2011). Semantics and 3D media: Current issues and perspectives. *Computers & Graphics*, 35(4):869 – 877.

- [Catalano et al., 2010] Catalano, C., Spagnuolo, M., Mortara, M., Faldieno, B., Stork, A., Koch, M., Alliez, P., Cazals, F., Yvenec, M., Huerst, W., Veltkamp, R., Pitikakis, M., Charbonnier, C., Assassi, L., Kim, J., Magnenat-Thalmann, N., Salamin, P., Thalmann, D., Dokken, T., and Quak, E. (2010). Focus K3D: Road map for Future Research.
- [Chang et al., 2011] Chang, K.-m. K., Mitchell, T., and Just, M. A. (2011). Quantitative modeling of the neural representation of objects: How semantic feature norms can account for fMRI activation. *NeuroImage*, 56(2):716–27.
- [Chen et al., 2009] Chen, X., Golovinskiy, A., and Funkhouser, T. (2009). A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3).
- [Chinellato et al., 2007] Chinellato, E., Morales, A., Cervera, E., and del Pobil, A. P. (2007). Symbol grounding through robotic manipulation in cognitive systems. *Robotics and Autonomous Systems*, 55(12):851–859.
- [Ciocarlie et al., 2007] Ciocarlie, M., Goldfeder, C., and Allen, P. (2007). Dimensionality reduction for hand-independent dexterous robotic grasping. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3270–3275. IEEE.
- [Ciocarlie and Allen, 2009] Ciocarlie, M. T. and Allen, P. K. (2009). Hand Posture Subspaces for Dexterous Robotic Grasping. *The International Journal of Robotics Research*, 28(7):851–867.
- [Coelho et al., 2001] Coelho, J., Piater, J., and Grupen, R. (2001). Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot. *Robotics and Autonomous Systems*, 37(2-3):195–218.
- [Cohn et al., 2006] Cohn, A., Hogg, D., Bennett, B., Devin, V., Galata, A., Magee, D., Needhap, C., and Santos, P. (2006). Cognitive vision: integrating symbolic qualitative representations with computer vision. In Christensen, H. I. and Nagel, H.-H., editors, *Cognitive Vision Systems: sampling the spectrum of approaches*, volume 3948 of *LNCS*, chapter 14, pages 221–246. Springer.
- [Cohn et al., 1993] Cohn, A. G., Randell, D. A., Cui, Z., and Bennett, B. (1993). Qualitative spatial reasoning and representation. In Carreté, N. P. and Singh, M. G., editors, *Qualitative Reasoning and Decision Technologies*, pages 513–522, Barcelona. CIMNE.

- [Cypher and Halbert, 1993] Cypher, A. and Halbert, D. C. (1993). *Watch what I do: Programming by demonstration*. The MIT Press, London, England.
- [Dartigues et al., 2007] Dartigues, C., Ghodous, P., Gruninger, M., Pallez, D., and Sriram, R. (2007). CAD/CAPP integration using feature ontology. *Concurrent Engineering*, 15(2):237–249.
- [Dasiopoulou and Kompatsiaris, 2010] Dasiopoulou, S. and Kompatsiaris, I. (2010). Trends and issues in description logics frameworks for image interpretation. In Konstantopoulos, S., Perantonis, S., Karkaletsis, V., Spyropoulos, C., and Vouros, G., editors, *Artificial Intelligence: Theories, Models and Applications*, volume 6040 of *Lecture Notes in Computer Science*, pages 61–70. Springer Berlin / Heidelberg.
- [de Penning et al., 2011] de Penning, L., d’Avila Garcez, A. S., Lamb, L. C., and Meyer, J.-J. C. (2011). A neural-symbolic cognitive agent for online learning and reasoning. In *IJCAI*, pages 1653–1658.
- [Dentler et al., 2011] Dentler, K., Cornet, R., ten Teije, A., and de Keizer, N. (2011). Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semant. web*, 2(2):71–87.
- [Dickinson et al., 1997] Dickinson, S. J., Bergevin, R., Biederman, I., Eklundh, J.-O., Munck-Fairwood, R., Jain, A. K., and Pentland, A. (1997). The Potential of Geons for Generic 3D Object Recognition. *Image and Vision Computing*, 15:277–292.
- [Edelman, 1997] Edelman, S. (1997). Computational theories of object recognition. In *Trends in Cognitive Science*, pages 296–304.
- [Ek et al., 2010] Ek, C. H., Song, D., Huebner, K., and Kragic, D. (2010). Exploring affordances in robot grasping through latent structure representation. In *European Conference on Computer Vision, Workshop on ‘Vision for Cognitive Tasks’*, Crete, Greece.
- [Ekvall and Kragic, 2005] Ekvall, S. and Kragic, D. (2005). Grasp Recognition for Programming by Demonstration. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 748–753. IEEE.
- [Falomir et al., 2011] Falomir, Z., Jiménez-Ruiz, E., Escrig, M. T., and Museros, L. (2011). Describing Images Using Qualitative Models and Description Logics. *Spatial Cognition & Computation*, 11(1):45–74.

- [Farhadi et al., 2009] Farhadi, a., Endres, I., Hoiem, D., and Forsyth, D. (2009). Describing objects by their attributes. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785.
- [Feix et al., 2009] Feix, T., Pawlik, R., Schmiedmayer, H., Romero, J., and Kragic, D. (2009). A comprehensive grasp taxonomy. In *Robotics, Science and Systems: Workshop on Understanding the Human Hand for Advancing Robotic Manipulation*.
- [Feix et al.,] Feix, T., Romero, J., Ek, C.-H., Schmiedmayer, H.-B., and Kragic, D. A metric for comparing the anthropomorphic motion capability of artificial hands. *Robotics, IEEE Transactions on*. to appear.
- [Gächter et al., 2008] Gächter, S., Harati, A., and Siegwart, R. (2008). Incremental object part detection toward object classification in a sequence of noisy range images. In *ICRA*, pages 4037–4042.
- [Gibson, 1975] Gibson, J. J. (1975). Affordances and behavior. In Reed, E. and Jones, R., editors, *Reasons for Realism: Selected Essays of James J. Gibson*, pages 410–411. Lawrence Erlbaum, Hillsdale, NJ, 1 edition.
- [Giorgi et al., 2007] Giorgi, D., Biasotti, S., and Paraboschi, L. (2007). Shape retrieval contest 2007: Watertight models track.
- [Glenberg and Kaschak, 2002] Glenberg, A. M. and Kaschak, M. P. (2002). Grounding language in action. *Psychonomic Bulletin & Review*, pages 558–565.
- [Goldfeder et al., 2009] Goldfeder, C., Ciocarlie, M., Dang, H., and Allen, P. K. (2009). The Columbia grasp database. In *IEEE Intl. Conf. on Robotics and Automation*.
- [Graves, 2008] Graves, H. (2008). Representing Product Designs Using a Description Graph Extension to OWL 2. In *OWLED*.
- [Griffith et al., 2012] Griffith, S., Sinapov, J., Sukhoy, V., and Stoytchev, A. (2012). A behavior-grounded approach to forming object categories: Separating containers from noncontainers. *IEEE T. Autonomous Mental Development*, 4(1):54–69.
- [Grosz et al., 2003] Grosz, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: combining logic programs with description logic. In *Proceedings of the 12th international conference on World Wide Web, WWW '03*, pages 48–57, New York, NY, USA. ACM.

- [Guarino and Welty, 2004] Guarino, N. and Welty, C. A. (2004). *An Overview of OntoClean*, chapter 8, pages 151–172. Springer.
- [Haarslev and Möller, 2003] Haarslev, V. and Möller, R. (2003). Racer: A core inference engine for the semantic web. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), located at the 2nd International Semantic Web Conference ISWC 2003, Sanibel Island, Florida, USA, October 20*, pages 27–36.
- [Hanford et al., 2009] Hanford, S., Janrathitikarn, O., and Long, L. (2009). Control of mobile robots using the SOAR cognitive architecture. *Journal of Aerospace Computing, Information, and Communication*, 6(2):69–91.
- [Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. In *Proceedings of the ninth annual international conference of the Center for Nonlinear Studies on Self-organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computing Networks on Emergent computation*, CNLS '89, pages 335–346, Amsterdam, The Netherlands, The Netherlands. North-Holland Publishing Co.
- [Hartanto and Hertzberg, 2009] Hartanto, R. and Hertzberg, J. (2009). On the benefit of fusing dl-reasoning with htn-planning. In *KI*, pages 41–48.
- [Haslhofer et al., 2011] Haslhofer, B., Momeni Roochi, E., Schandl, B., and Zander, S. (2011). Europeana RDF store report.
- [Hastings et al., 2010] Hastings, J., Dumontier, M., Hull, D., Horridge, M., Steinbeck, C., Stevens, R., Sattler, U., Hörne, T., and Britz, K. (2010). Representing Chemicals Using OWL, Description Graphs and Rules. In Sirin, E. and Clark, K., editors, *OWLED*, volume 614 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- [Horikoshi and Suzuki, 1993] Horikoshi, T. and Suzuki, S. (1993). 3D parts decomposition from sparse range data using information criterion. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, pages 168 –173.
- [Horrocks, 1998] Horrocks, I. (1998). The FaCT system. In de Swart, H., editor, *Proc. of the 2nd Int. Conf. on Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 307–312. Springer.

- [Horrocks and Graves, 2008] Horrocks, I. and Graves, H. (2008). Application of OWL 1.1 to Systems Engineering. In *OWL: Experiences and Directions (OWLED)*, page online.
- [Hudelot et al., 2008] Hudelot, C., Atif, J., and Bloch, I. (2008). Fuzzy spatial relation ontology for image interpretation. *Fuzzy Sets and Systems*, 159(15):1929–1951.
- [Hummel, 2000] Hummel, J. E. (2000). Where view-based theories break down: The role of structure in shape perception and object recognition. In *Cognitive Dynamics: Conceptual Change in Humans and Machines*.
- [Iberall et al., 1988] Iberall, T., Jackson, J., Labbe, L., and Zampano, R. (1988). Knowledge-based prehension: capturing human dexterity. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 82–87 vol.1.
- [Jolliffe, 2002] Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer, second edition.
- [Kalogerakis et al., 2010] Kalogerakis, E., Hertzmann, A., and Singh, K. (2010). Learning 3D mesh segmentation and labeling. *Components*, 29(July):1–12.
- [Kang and Ikeuchi, 1994] Kang, S. B. and Ikeuchi, K. (1994). Robot task programming by human demonstration: mapping human grasps to manipulator grasps. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 1, pages 97–104 vol.1.
- [Kassimi and beqqali, 2012] Kassimi, M. A. and beqqali, O. E. (2012). From 3D model data to semantics. *International Journal of Computer Science & Information Technology*, 3:1–17.
- [Keet and Artale, 2008] Keet, C. M. and Artale, A. (2008). Representing and reasoning over a taxonomy of part-whole relations. *Appl. Ontol.*, 3:91–110.
- [Klinov, 2008] Klinov, P. (2008). Pronto: A Non-monotonic Probabilistic Description Logic Reasoner. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *The Semantic Web: Research and Applications*, volume 5021 of *Lecture Notes in Computer Science*, chapter 66, pages 822–826. Springer Berlin Heidelberg, Berlin, Heidelberg.

- [Kollia et al., 2010] Kollia, I., Simou, N., Stafylopatis, A., and Kollias, S. (2010). Semantic image analysis using a symbolic neural architecture.
- [Kolovski et al., 2006] Kolovski, V., Parsia, B., and Katz, Y. (2006). Implementing OWL defaults. In Grau, B. C., Hitzler, P., Shankey, C., and Wallace, E., editors, *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- [Krötzsch, 2012] Krötzsch, M. (2012). OWL 2 Profiles: An introduction to lightweight ontology languages. In *Proceedings of the 8th Reasoning Web Summer School, Vienna, Austria, September 3–8 2012*. Springer.
- [Kuipers, 2008] Kuipers, B. (2008). Drinking from the firehose of experience. *Artif. Intell. Med.*, 44(2):155–170.
- [Kunze et al., 2011] Kunze, L., Roehm, T., and Beetz, M. (2011). Towards semantic robot description languages. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5589–5595, Shanghai, China.
- [Kurzhaniskiy and Varaiya, 2006] Kurzhaniskiy, A. A. and Varaiya, P. (2006). Ellipsoidal toolbox. Technical report, EECS, UC Berkeley.
- [Laird et al., 2012] Laird, J., Kinkade, K., Mohan, S., and Xu, J. (2012). Cognitive robotics using the SOAR cognitive architecture.
- [Laird et al., 1987] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). SOAR: an architecture for general intelligence. *Artif. Intell.*, 33(1):1–64.
- [Lawley and Bousquet, 2010] Lawley, M. and Bousquet, C. (2010). Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In Meyer, T., Orgun, M., and Taylor, K., editors, *Australasian Ontology Workshop 2010 (AOW 2010): Advances in Ontologies*, volume 122 of *CRPIT*, pages 45–50, Adelaide, Australia. ACS. Winner of Payne-Scott Best Paper Award.
- [Lemaignan et al., 2010] Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R., and Beetz, M. (2010). Oro, a knowledge management platform for cognitive architectures in robotics. In *IROS*, pages 3548–3553.
- [Lenat, 1995] Lenat, D. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

- [Li et al., 2007] Li, Y., Fu, J. L., and Pollard, N. S. (2007). Data-driven grasp synthesis using shape matching and task-based pruning. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):732–747.
- [Lien et al., 2006] Lien, J.-M., Keyser, J., and Amato, N. M. (2006). Simultaneous shape decomposition and skeletonization. In *Symposium on Solid and Physical Modeling*, pages 219–228.
- [Lim et al., 2011] Lim, G. H., Suh, I. H., and Suh, H. (2011). Ontology-based unified robot knowledge for service robots in indoor environments. *Trans. Sys. Man Cyber. Part A*, 41(3):492–509.
- [Liu and Singh, 2004] Liu, H. and Singh, P. (2004). Conceptnet: A practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226.
- [Lukasiewicz and Straccia, 2008] Lukasiewicz, T. and Straccia, U. (2008). Managing uncertainty and vagueness in description logics for the Semantic Web. *Web Semant.*, 6(4):291–308.
- [Mallenby and Bennett, 2007] Mallenby, D. and Bennett, B. (2007). Applying spatial reasoning to topographical data with a grounded ontology. In Fonseca, F., Rodrigues, M. A., and Levashkin, S., editors, *GeoSpatial Semantics, proceedings of the second international conference*, number 4853 in Lecture Notes in Computer Science, pages 210–227, Mexico City. Springer.
- [Marini et al., 2007] Marini, S., Spagnuolo, M., and Falcidieno, B. (2007). Structural Shape Prototypes for the Automatic Classification of 3D Objects. *IEEE Comput. Graph. Appl.*, 27(4):28–37.
- [Marr and Nishihara, 1978] Marr, D. and Nishihara, H. (1978). Representation and recognition of the spatial organization of three-dimensional shapes. In *Philosophical Transactions of the Royal Society of London B.*, volume 200, pages 269–294.
- [Marras et al., 2012] Marras, S., Bronstein, M. M., Hormann, K., Scateni, R., and Scopigno, R. (2012). Motion-based mesh segmentation using augmented silhouettes. *Graphical Models*, 74(4):164 – 172.
- [Modayil and Kuipers, 2007] Modayil, J. and Kuipers, B. (2007). Autonomous development of a grounded object ontology by a learning robot. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*.

- [Motik et al., 2008] Motik, B., Grau, B. C., and Sattler, U. (2008). The Representation of Structured Objects in DLs using Description Graphs. In Baader, F., Lutz, C., and Motik, B., editors, *Proc. of the 21st Int. Workshop on Description Logics (DL 2008)*, volume 353 of *CEUR Workshop Proceedings*, Dresden, Germany.
- [Mozos et al., 2011] Mozos, O. M., Marton, Z. C., and Beetz, M. (2011). Furniture Models Learned from the WWW – Using Web Catalogs to Locate and Categorize Unknown Furniture Pieces in 3D Laser Scans. *Robotics & Automation Magazine*, 18(2):22–32.
- [Natali et al., 2011] Natali, M., Biasotti, S., Patanè, G., and Falcidieno, B. (2011). Graph-based representations of point clouds. *Graph. Models*, 73(5):151–164.
- [Neumann and Möller, 2008] Neumann, B. and Möller, R. (2008). On scene interpretation with description logics. *Image and Vision Computing*, 26(1):82–101.
- [Ning et al., 2010] Ning, X., Li, E., Zhang, X., and Wang, Y. (2010). Shape decomposition and understanding of point cloud objects based on perceptual information. In *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications in Industry, VRCAI '10*, pages 199–206, New York, NY, USA. ACM.
- [Ogata and Takahashi, 1994] Ogata, H. and Takahashi, T. (1994). Robotic assembly operation teaching in a virtual environment. *Robotics and Automation, IEEE Transactions on*, 10(3):391–399.
- [Palm et al., 2009] Palm, R., Iliev, B., and Kadmiry, B. (2009). Recognition of human grasps by time-clustering and fuzzy modeling. *Robot. Auton. Syst.*, 57(5):484–495.
- [Pastra, 2008] Pastra, K. (2008). PRAXICON: The Development of a Grounding Resource. In *Proceedings of the 4th International Workshop on Human-Computer Conversation*.
- [Pastra, 2010] Pastra, K. (2010). From Lexicon to PRAXICON: language-action-image semantic relations. In Potagas, K., E. I., editor, *Conversations on Language and Action, Aiginiteion Series*, pages 143–172. Synapseis.

- [Pastra and Aloimonos, 2012] Pastra, K. and Aloimonos, Y. (2012). The minimalist grammar of action. In *Philosophical Transactions of the Royal Society B*, volume 367, pages 103–117.
- [Pastra et al., 2011] Pastra, K., Balta, E., Dimitrakis, P., and Karakatsiotis, G. (2011). Embodied language processing: A new generation of language technology. In *Language-Action Tools for Cognitive Artificial Agents*, volume WS-11-14 of *AAAI Workshops*. AAAI.
- [Pastra et al., 2010] Pastra, K., Dimitrakis, P., Balta, E., and Karakatsiotis, G. (2010). PRAXICON and its language-related modules. In *Companion Volume of the 6th Hellenic Conference on Artificial Intelligence (SETN)*, pages 27–32.
- [Röhrbein et al., 2007] Röhrbein, F., Eggert, J., and Körner, E. (2007). A cortex-inspired neural-symbolic network for knowledge representation. In *NeSy*.
- [Rom and Medioni, 1994] Rom, H. and Medioni, G. (1994). Part decomposition and description of 3D shapes. In *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 629–632 vol.1.
- [Salvi et al., 2012] Salvi, G., Montesano, L., Bernardino, A., and Santos-Victor, J. (2012). Language bootstrapping: Learning word meanings from perception-action association. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(3):660–671.
- [Schlenoff and Messina, 2005] Schlenoff, C. and Messina, E. (2005). A robot ontology for urban search and rescue. In *Proceedings of the 2005 ACM workshop on Research in knowledge representation for autonomous systems*, KRAS '05, pages 27–34, New York, NY, USA. ACM.
- [Schockaert et al., 2011] Schockaert, S., Makarytska, N., and De Cock, M. (2011). Fuzzy Methods on the Web: A Critical Discussion. In Cornelis, C., Deschrijver, G., Nachtgael, M., Schockaert, S., and Shi, Y., editors, *35 Years of Fuzzy Set Theory*, volume 261 of *Studies in Fuzziness and Soft Computing*, pages 237–266. Springer Berlin / Heidelberg.
- [Schyns et al., 1998] Schyns, P. G., Goldstone, R. L., and Thibaut, J. P. (1998). The development of features in object concepts. *The Behavioral and brain sciences*, 21(1):1–17; discussion 17–54.

- [Shamir, 2008] Shamir, A. (2008). A survey on Mesh Segmentation Techniques. *Computer Graphics Forum*, 27(6):1539–1556.
- [Shearer et al., 2008] Shearer, R., Motik, B., and Horrocks, I. (2008). Hermit: A Highly-Efficient OWL Reasoner. In Ruttenberg, A., Sattler, U., and Dolbear, C., editors, *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*, Karlsruhe, Germany.
- [Simou et al., 2007] Simou, N., Athanasiadis, T., Tzouvaras, V., and Kollias, S. (2007). Multimedia reasoning with f -*SHLN*. In *Proceedings of the Second International Workshop on Semantic Media Adaptation and Personalization, SMAP '07*, pages 44–49, Washington, DC, USA. IEEE Computer Society.
- [Sirin and Parsia, 2007] Sirin, E. and Parsia, B. (2007). SPARQL-DL: SPARQL Query for OWL-DL. In *In 3rd OWL Experiences and Directions Workshop (OWLED-2007)*.
- [Sirin et al., 2007] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53.
- [Stoilos et al., 2008] Stoilos, G., Stamou, G., Pan, J. Z., Simou, N., and Tzouvaras, V. (2008). Uncertainty Reasoning for the Semantic Web I. chapter Reasoning with the Fuzzy Description Logic f -*SHLN*: Theory, Practice and Applications, pages 262–281. Springer-Verlag, Berlin, Heidelberg.
- [Straccia, 2009] Straccia, U. (2009). Towards spatial reasoning in fuzzy description logics. In *Proc. of the 2009 IEEE International Conference on Fuzzy Systems*.
- [Stramandinoli et al., 2012] Stramandinoli, F., Marocco, D., and Cangelosi, A. (2012). The grounding of higher order concepts in action and language: A cognitive robotics model. *Neural Networks*, 32(0):165 – 173.
- [Sukumar et al., 2006] Sukumar, S., Page, D., Gribok, A., Koschan, A., and Abidi, M. (2006). Shape measure for identifying perceptually informative parts of 3D objects. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 679 –686.
- [Svensson and di Baja, 2002] Svensson, S. and di Baja, G. S. (2002). Using distance transforms to decompose 3D discrete objects. *Image and Vision Computing*, 20(8):529 – 540.

- [Tangelder and Veltkamp, 2007] Tangelder, J. W. H. and Veltkamp, R. C. (2007). A survey of content based 3D shape retrieval methods. *Multimedia Tools and Applications*, 39(3):441–471.
- [Tenorth and Beetz, 2009] Tenorth, M. and Beetz, M. (2009). KNOWROB - Knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 11-15, 2009, St. Louis, MO, USA*, pages 4261–4266. IEEE.
- [Tenorth and Beetz, 2012] Tenorth, M. and Beetz, M. (2012). A unified representation for reasoning about robot actions, processes, and their effects on objects. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal. Accepted for publication.
- [Tran et al., 2011] Tran, T., Ladwig, G., and Wagner, A. (2011). Approximate and Incremental Processing of Complex Queries against the Web of Data. In *DEXA (2)*, pages 171–187.
- [Tsarkov and Horrocks, 2006] Tsarkov, D. and Horrocks, I. (2006). FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer.
- [Tung and Schmitt, 2004] Tung, T. and Schmitt, F. (2004). Augmented Reeb graphs for content-based retrieval of 3D mesh models. *Proceedings Shape Modeling Applications*, pages 157–389.
- [van Kaick et al., 2011] van Kaick, O., Tagliasacchi, A., Sidi, O., Zhang, H., Cohen-Or, D., Wolf, L., and Hamarneh, G. (2011). Prior knowledge for part correspondence. *Computer Graphics Forum*, 30(2):553–562.
- [Varadarajan and Vincze, 2011] Varadarajan, K. and Vincze, M. (2011). Ontological knowledge management framework for grasping and manipulation. In *Proceedings of the Knowledge Representation for Autonomous Robots Workshop (ICRA 2011)*.
- [Varzi, 2007] Varzi, A. C. (2007). Spatial reasoning and ontology: Parts, wholes, and locations. In Aiello, M., Pratt-Hartmann, I. E., and van Benthem, J., editors, *Handbook of Spatial Logics*, pages 945–1038. Springer-Verlag.
- [Vasilakis et al., 2010] Vasilakis, G., Garc a-Rojas, A., Papaleo, L., Catalano, C. E., Robbiano, F., Spagnuolo, M., Vavalis, M., and Pitikakis,

- M. (2010). Knowledge-Based Representation of 3D Media. *International Journal of Software Engineering and Knowledge Engineering*, 20(5):739–760.
- [Vatakis et al., 2012] Vatakis, A., Pastra, K., and Dimitrakis, P. (2012). Acquiring object affordances through touch, vision, and language. In *13th International Multisensory Research Forum*.
- [Vinson and Vigliocco, 2008] Vinson, D. P. and Vigliocco, G. (2008). Semantic feature production norms for a large set of objects and events. *Behavior Research Methods*, 40(1):183–190.
- [Vitucci, 2011] Vitucci, N. (2011). Autonomous object manipulation: A semantic-driven approach. In *IJCAI*, pages 2858–2859.
- [Vitucci et al., 2010a] Vitucci, N., Neri, M. A., and Gini, G. (2010a). Semantic-aided visual grasping using a fuzzy description logic. In *Proceedings of the CogSys2010 (4th International Conference on Cognitive Systems)*, Zurich, Switzerland.
- [Vitucci et al., 2010b] Vitucci, N., Neri, M. A., and Gini, G. (2010b). Using $f-SHLN$ to represent objects: An aid to visual grasping. In *UniDL '10*, pages 80–87.
- [Vitucci et al., 2012] Vitucci, N., Neri, M. A., Tedesco, R., and Gini, G. (2012). Semanticizing Syntactic Patterns in NLP Processing Using SPARQL-DL Queries. In *OWLED*.
- [Wagan et al., 2009] Wagan, A., Godil, A., and Bres, S. (2009). 3D shape retrieval by visual parts similarity. In *Applied Imagery Pattern Recognition Workshop (AIPRW), 2009 IEEE*, pages 1–6.
- [Waibel et al., 2011] Waibel, M., Beetz, M., Civera, J., D’Andrea, R., Elfiring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., and van de Molengraft, R. (2011). RoboEarth. *Robotics Automation Magazine, IEEE*, 18(2):69–82.
- [Wang and Pan, 2007] Wang, S. and Pan, J. Z. (2007). Ontology-based Integration and Retrieval over Multiple Quantities - What if “Ovate leaves and often blue to purple flowers”. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, WI '07*, pages 388–394, Washington, DC, USA. IEEE Computer Society.

- [Yee et al., 2011] Yee, E., Huffstetler, S., and Thompson-Schill, S. L. (2011). Function follows form: activation of shape and function features during object identification. *Journal of experimental psychology. General*, 140(3):348–363.
- [Zhang et al., 2012] Zhang, X., Tutenel, T., Mo, R., Bidarra, R., and Bronsvoort, W. F. (2012). A method for specifying semantics of large sets of 3D models. In Richard, P., Kraus, M., Laramée, R. S., and Braz, J., editors, *GRAPP/IVAPP*, pages 97–106. SciTePress.
- [Zhang et al., 2003] Zhang, Y., Koschan, A., and Abidi, M. (2003). Superquadrics based 3D object representation of automotive parts utilizing part decomposition. In *Proc. SPIE 6th Int. Conf. on Quality Control by Artificial Vision*, pages 241–251.
- [Zollner et al., 2002] Zollner, R., Rogalla, O., Dillmann, R., and Zollner, M. (2002). Understanding users intention: programming fine manipulation tasks by demonstration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1114–1119.