

# REPRESENTING CHEMICAL STRUCTURES USING OWL AND DESCRIPTION GRAPHS

by

Joanna Kathleen Hastings

submitted in accordance with the requirements

for the degree of

Master of Science

in the subject

Computer Science

at the

UNIVERSITY OF SOUTH AFRICA

Supervisor: Katarina Britz

Joint Supervisor: Tertia Hörne

Joint Supervisor: Robert Stevens

November 2010

---

---

# Acknowledgements

Many thanks to my supervisors, Katarina Britz, Tertia Hörne and Robert Stevens, for their input, invaluable discussions, and patience. Thanks for the valuable input into the project from Ulrike Sattler and for the early ground work laid in this area by Matthew Horridge and Duncan Hull. Thanks also to Michel Dumontier and Colin Batchelor for their expert advice.

Thanks to all my colleagues at the European Bioinformatics Institute, in particular Marcus Ennis and Kirill Degtyarenko, for their patience in answering questions; and Christoph Steinbeck and Paula de Matos, for allowing me the time to pursue my studies and for supporting my participation in this and many other fascinating projects.

Finally, this work would not be possible without the continuing love and support of my partner, Sven Bergmann, and my children Aidan and Aliya, who endure with good grace the excess of what should be their time that I spend in front of my computer.

*The whole is more than the sum of its parts.*

—ARISTOTLE (384 BC–322 BC)

---

---

# Abstract

Objects can be said to be structured when their representation also contains their parts. While OWL in general can describe structured objects, description graphs are a recent, decidable extension to OWL which support the description of classes of structured objects whose parts are related in complex ways. Classes of chemical entities such as molecules, ions and groups (parts of molecules) are often characterised by the way in which the constituent atoms of their instances are connected via chemical bonds. For chemoinformatics tools and applications, this internal structure is represented using chemical graphs. We here present a chemical knowledge base based on the standard chemical graph model using description graphs, OWL and rules. We include in our ontology chemical classes, groups, and molecules, together with their structures encoded as description graphs. We show how role-safe rules can be used to determine parthood between groups and molecules based on the graph structures and to determine basic chemical properties. Finally, we investigate the scalability of the technology used through the development of an automatic utility to convert standard chemical graphs into description graphs, and converting a large number of diverse graphs obtained from a publicly available chemical database.

**Keywords:** chemistry, ontology, description graphs, rules, OWL, chemical structures, structured objects

---

---

# Contents

<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xii</b>
<b>List of Acronyms</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Research questions . . . . .	2
1.2 Out of Scope . . . . .	3
1.3 Statement of the contributions . . . . .	3
1.4 Structure . . . . .	4
<b>2 Some chemistry background</b>	<b>5</b>
2.1 Chemical entities . . . . .	6
2.2 Properties of chemical entities . . . . .	12
2.3 Chemical substances . . . . .	13
<b>3 Chemical Structures</b>	<b>15</b>
3.1 Representation . . . . .	18
3.1.1 MOLfile . . . . .	18
3.1.2 Chemical Markup Language . . . . .	19
3.1.3 SMILES . . . . .	21
3.1.4 InChI and InChIKey . . . . .	22
3.2 Sources of chemical data . . . . .	22
<b>4 Description Logics</b>	<b>25</b>
4.1 Logical representation . . . . .	26
4.2 Semantic networks and frames . . . . .	27
4.3 From semantic networks to Description Logics . . . . .	29
4.3.1 A note about syntax . . . . .	30
4.4 The definition of complex concepts . . . . .	31
4.4.1 Constructors . . . . .	32
4.4.2 The TBox . . . . .	33
4.5 Knowledge about individuals . . . . .	34
4.5.1 Defining a Description Logics (DL) knowledge base . . . . .	35

---

4.6	Semantics . . . . .	36
4.7	Reasoning . . . . .	36
4.7.1	Terminological inferences . . . . .	36
4.7.2	Assertional inferences and query answering . . . . .	37
4.7.3	Open world semantics . . . . .	38
4.7.4	Decidability and complexity . . . . .	39
4.8	Basic DL languages . . . . .	40
<b>5</b>	<b>Rules</b>	<b>41</b>
5.1	Semantics . . . . .	42
5.2	Reasoning . . . . .	43
<b>6</b>	<b>OWL - the Web Ontology Language</b>	<b>45</b>
6.1	Background . . . . .	45
6.1.1	A note about terminology . . . . .	45
6.2	Expressivity and Web Ontology Language (OWL) flavours . . . . .	46
6.3	Notes on semantics . . . . .	47
6.4	OWL Syntaxes . . . . .	47
6.5	Tools for working with OWL ontologies . . . . .	48
6.6	Semantic Web Rule Language . . . . .	49
<b>7</b>	<b>Representing chemical structures in OWL</b>	<b>51</b>
7.1	Structured Objects . . . . .	51
7.2	Representation in TBox . . . . .	52
7.3	Representation in ABox . . . . .	54
7.4	Representation with Rules . . . . .	55
<b>8</b>	<b>Description Graphs</b>	<b>57</b>
8.1	Definition . . . . .	57
8.2	Decidability of Reasoning . . . . .	58
8.2.1	Boundedness of graphs . . . . .	58
8.2.2	Separation of properties . . . . .	59
8.2.3	Acyclicity . . . . .	59
8.3	Rules . . . . .	60
8.3.1	Definition . . . . .	61
8.4	Implementation . . . . .	62

---



---

<b>9</b>	<b>Software design and implementation</b>	<b>65</b>
9.1	Overview . . . . .	65
9.2	Architecture . . . . .	66
9.2.1	Downloadable files . . . . .	67
9.3	Software libraries . . . . .	67
9.3.1	Chemical data conversion . . . . .	68
9.3.2	Programmatic manipulation of OWL ontologies . . . . .	68
9.3.3	Reasoning over graph-enriched knowledge bases . . . . .	69
9.4	Implementation . . . . .	69
9.4.1	Ontology . . . . .	69
9.4.2	Description Graphs . . . . .	69
9.4.3	Rules . . . . .	72
<b>10</b>	<b>Results</b>	<b>75</b>
10.1	Classification based on substructures . . . . .	76
10.2	Determination of chemical properties from graphs . . . . .	77
10.3	Scalability of reasoning over knowledge base . . . . .	78
<b>11</b>	<b>Discussion</b>	<b>81</b>
11.1	Structure-based chemical classification by expert systems . . . . .	81
11.2	Related work . . . . .	85
11.3	Comparison of our research to previous efforts . . . . .	88
<b>12</b>	<b>Conclusion</b>	<b>91</b>
	<b>Bibliography</b>	<b>93</b>
	<b>Appendices</b>	<b>99</b>
<b>A</b>	<b>Ontology Syntax Formats</b>	<b>101</b>
<b>B</b>	<b>Source Code Listing</b>	<b>105</b>
B.1	Conversion of chemical graphs . . . . .	105
B.2	Creating the ontology . . . . .	108
B.3	Adding rules to the ontology . . . . .	110
B.4	Performing the reasoning . . . . .	112
<b>C</b>	<b>OWLED paper</b>	<b>115</b>

---

# List of Figures

2.1	Some example uses of the science of chemistry . . . . .	6
2.2	Isotopes of Hydrogen . . . . .	7
2.3	Copper(2) Sulphate Salt - an ionically bound crystal structure . . . . .	8
2.4	Carbon compounds - methane, hexane, isohexane and cyclohexane . . . . .	10
2.5	Carbon compounds - ethane, ethene and ethyne . . . . .	10
2.6	Chiral molecules - S-camphor and R-camphor . . . . .	13
2.7	Diamond - a single large covalently bound molecule . . . . .	13
3.1	Chemical skeleton graphs and coordinates . . . . .	16
3.2	R group structural class definition for Carboxylic Acid . . . . .	17
3.3	Connection table for paracetamol from (Indiana Cheminformatics Educa- tion Portal, 2010) . . . . .	19
3.4	MOLfile format for Paracetamol . . . . .	20
3.5	CML format for Paracetamol . . . . .	21
4.1	A semantic network populated with some animal concepts . . . . .	28
6.1	Protégé OWL ontology editing environment . . . . .	48
7.1	Structured objects in chemistry – hexane, cyclohexane, paracetamol, aspirin	52
7.2	Unintended infinite tree model . . . . .	54
7.3	Some molecules which share a cyclohexane skeleton . . . . .	55
8.1	Description graph representation of cyclohexane molecule . . . . .	58
8.2	Large molecules and polymer representation . . . . .	59
8.3	Description Graphs and Rules Syntax . . . . .	62
9.1	Overall architecture . . . . .	67
9.2	Core ontology structure . . . . .	70
9.3	Illustration of the cyclobutane description graph . . . . .	71
9.4	Some cyclic compounds . . . . .	73
9.5	Classification based on parthood . . . . .	74
10.1	Ontology hierarchy with test classes before classification execution . . . . .	75
10.2	Ontology hierarchy with test classes after reasoning . . . . .	76
10.3	Performance results of classification . . . . .	78

---

11.1 D-threonine and L-threonine . . . . .	83
11.2 Polycyclic molecular cages . . . . .	84
11.3 Structural relations in the ChEBI ontology . . . . .	87

# List of Tables

6.1 Syntax OWL and DL . . . . .	46
10.1 Results of performance evaluation – No generated rules . . . . .	79
10.2 Results of performance evaluation – Including generated rules . . . . .	79

# List of Acronyms

**DL** Description Logics

**KR** Knowledge Representation

**OWL** Web Ontology Language

**OBO** Open Biological and Biomedical Ontologies

**FOL** First-order logic

**TBox** Terminology box

**ABox** Assertion box

**2D** two dimensional

**3D** three dimensional

**DNA** Deoxyribonucleic acid

**RNA** Ribonucleic acid

**CML** Chemical markup language

**PDB** Protein Data Bank

**XML** eXtensible Markup Language

**SMILES** Simplified Molecular Input Line Entry System

**InChI** IUPAC International Chemical Identifier

**CAS** Chemical Abstracts Service

**ChEBI** Chemical Entities of Biological Interest

**RDF** Resource Description Framework

**SWRL** Semantic Web Rule Language

**CDK** Chemistry Development Kit

**QSAR** Quantitative Structure-Activity Relationship

---

**GUI** Graphical user interface

**API** Application programming interface

**IUPAC** International Union for Pure and Applied Chemistry

# 1

## Introduction

Recent years have lead to an explosion in the availability of data in the chemistry domain, just as in many other scientific domains. With this information explosion, however, retrieving *relevant* information from the deluge of available information becomes an ever harder problem. Computers have yet to replicate or even come close to the insight of human experts, but computational processing is essential to filter the available information so as to better facilitate the work of human experts. There is thus a clear need for the development of more usable computational systems, which use some of the knowledge available to human experts in order to more efficiently filter irrelevant information and extract relevant information for display to human experts, and to perform some of the work of human experts, thus enabling the user to cope with the higher volumes. Computer systems which make use of expert domain knowledge, encoded in a computable format, have been termed ‘expert systems’, and many such systems are in active use in chemistry research today ([Hemmer, 2008](#)).

These expert systems rely on computable, tractable encodings of human domain knowledge. Many different formats and systems have been developed for encoding and storing such knowledge. These include relational databases which store “flat” fact-like knowledge in structured tables; rule systems which store knowledge about actions or situations which are triggered by certain preconditions being met in “if-then” type rules; graph-like *semantic networks* which structure data in the form of nodes connected by interrelationships between the nodes; and, more recently, logic-based *ontologies* which encode the meaning of data through formal axioms. Such semantic networks and ontologies are becoming widespread tools within the academic community in many scientific domains, an example of which is the successful Gene Ontology ([The Gene Ontology Consortium, 2000](#)).

In order to provide the maximum benefit to the academic community, expert systems

which are developed, and the domain knowledge which is encoded, must be developed as open data and provided in formats compliant with open standards. Two of the primary ontology formats which are in widespread use within the academic community are the Open Biological and Biomedical Ontologies (OBO) format ([The Gene Ontology Consortium, 2010](#)), which encodes knowledge in a semantic network-style formalism, and the W3C standard OWL language ([Smith \*et al.\*, 2010](#)), which encodes axiomatic, logic-based ontologies in a variety of different levels of expressivity and syntaxes for serialization.

Objects can be said to be structured when their representation also contains their parts. Many objects in the domain of chemistry are structured in this fashion. While OWL in general can describe structured objects, *description graphs* are a recent, decidable extension to OWL which support the description of classes of structured objects whose parts are related in complex ways beyond the level at which OWL is capable of modelling ([Motik \*et al.\*, 2008c,b,a](#)).

Chemical structures are represented as chemical graphs, which encode the connections between parts of the chemical entity ([Trinajstić, 1992](#)). A classic cheminformatics application on such chemical graphs is chemical classification based on the presence or absence of substructures within the structures. OWL, as it currently stands, is incapable of representing the required complex structures, particularly cycles ([Cuenca Grau \*et al.\*, 2008](#)). However, the required structures could be represented in description graphs, and it is the subject of this dissertation to evaluate precisely this approach.

The remainder of this introduction is organised as follows. In the next section, we present the research questions which we address in this dissertation. Section 1.2 describes some related aspects that are not directly addressed by this work. Section 1.3 presents the main contributions and, finally, Section 1.4 describes how this dissertation is organized.

## 1.1 Research questions

The goal of this research is to evaluate the utility, applicability, and practical constraints in the modelling of complex chemical structures in the description graphs extension to the standard OWL language.

In this dissertation, we will present a method for transforming chemical graphs into description graphs, and apply this method to create an OWL knowledge base of chemical entities enhanced with the structures of the chemical entities as description graphs. We will consider to what extent the formalism of description graphs, together with rules for



expressing conditionality, supports the type of reasoning which domain experts would expect from a structure-enhanced chemical knowledge base, such as classification based on chemical structures, and determination of chemical properties based on the structures. Finally, we will assess the scalability of the technology by evaluating the times taken to reason over knowledge bases of varying sizes.

The research questions which guide our implementation are:

- Can chemical entities be classified based on their substructures from description graphs?
- Can basic chemical properties be determined from the description graphs?
- How scalable is the resulting knowledge base?

## 1.2 Out of Scope

*Description graph editing tools.* Currently, in the project, the description graphs are created programmatically and added to the knowledge base. An improvement to the method we used would be a facility which enabled the creation of description graphs through a graphical user interface. However, such a utility is out of scope of this dissertation.

## 1.3 Statement of the contributions

As a result of the work presented in this dissertation, the following contributions can be highlighted:

- Implementation of a description graph-enriched knowledge base for chemical structures.
- Evaluation of the description graph formalism and the scalability of reasoning.

In addition to these contributions, one paper presenting the findings of this dissertation was produced:

- J. Hastings, M. Dumontier, M. Horridge, D. Hull, U. Sattler, R. Stevens, C. Steinbeck, T. Hörne and K. Britz (2010), Representing chemical structures using OWL, Description Graphs, and Rules. In *Proceedings of OWL Experiences and Directions (OWLED 2010)*, San Francisco, CA.

This paper is reproduced in Appendix C.

My contributions to the paper were:

- Designing the translation between chemical data and description graphs, including selecting the source format, modelling information in terms of the target language expressivity and defining the mapping.
- Developing the software which generated the OWL ontology, the description graphs from chemical structures and the rules, adding these to the knowledge base, and performing the reasoning.
- Writing the initial draft of the paper and revising it after discussion and review by the other authors.

The other authors' contributions were as follows: MH and DH provided the OWL API for Description Graphs and assisted with specifying the software libraries required for the implementation; RS, US, TH and KB assisted with the project conception and early specification; RS and CS assisted with the chemistry-specific domain modelling; MD presented the paper at the conference; and all authors read and reviewed the final paper.

## 1.4 Structure

The structure of this dissertation is as follows:

Relevant chemistry background is given in Chapter 2 and an overview of the chemical graph formalism and standard syntaxes in Chapter 3. Chapter 4 gives an introduction to Description Logics, the knowledge representation formalism on which OWL is based.

Chapter 5 describes the rules formalism which allows the representation of dynamic, procedural knowledge. Chapter 6 gives an overview of the OWL language including syntax and semantics, and Chapter 7 describes the core research problem, that is, the limitation of OWL alone for the representation and reasoning with complex structured objects.

Chapter 8 presents the description graphs extension to OWL and in Chapter 9 we describe the implementation of the software for the project, showing how description graphs can be used to implement a knowledge base about chemical structures. In Chapter 10 we present the results of our research, followed by a discussion and comparison to related work in Chapter 11.

Finally, we present our conclusions in Chapter 12.

# 2

## Some chemistry background

In this chapter, we present an introduction to the domain of chemistry which will provide the background and the domain-specific terminology which we will use throughout the remainder of this dissertation.

Chemistry is the science which studies the structure and interactions of the atoms and molecules from which all of the matter around us is constituted, both living and non-living. Chemical entities are the atoms, molecules, and substances which are the objects of study by chemists. The study of these entities and their properties and interactions has lead to many of the beneficial developments of the modern world, including life-saving modern medicines, such as antibiotics and anti-cancer drugs (Figure 2.1, a). It has also lead to the development of novel materials such as plastics, synthetic fabrics, strong metal alloys such as stainless steel (Figure 2.1, c), and the complex controlled chemical reactions which underlie the batteries which now power many ubiquitous devices such as cellular phones and cameras, and drive the colourful displays of these devices (Figure 2.1, b). The methods of studying the structure of chemical entities have been applied in the life sciences in the field of molecular biology, where the chemical structure of complex biological molecules such as proteins have been studied together with their mechanisms of interaction both with each other and with smaller molecules (Figure 2.1, d)<sup>1</sup>.

While traditionally an experimental science, advances in theoretical chemistry have allowed increasingly complex predictions to be made about molecules and reactions which have not been observed experimentally. This allows scientists to avoid or defer expensive experimental investigations under certain circumstances. Central to theoretical chemistry are techniques for the modelling and representation of chemical entities, and the complex mathematical theories which are developed based on these models. Computers

---

<sup>1</sup>Images taken from Wikimedia Commons, except the protein structure which is taken from PDB record 1c0f(PDB, 2010).



**Figure 2.1** Some example uses of the science of chemistry

are used for virtual experiments on theoretical or predicted chemical structures, and they are used to help determine the structures of chemicals contained in samples isolated from nature.

As more and more complex calculations have been relegated to the superior computational speed and accuracy of computers, the field of chemoinformatics, which is the application of computers to solve problems in chemistry, has developed many algorithms and techniques which are in common use in chemical and pharmaceutical industries today. But in addition, the domain of chemistry, with its plethora of difficult and even computationally intractable problems, has led to advances in computing, with the field of chemoinformatics being one of the drivers. It provided early test cases for the development of complex stochastic and heuristic computational techniques which are now widely used in Artificial Intelligence (Trinajstić, 1992).

## 2.1 Chemical entities

All of matter is composed from tiny particles called atoms. Atoms, in turn, are composed from small, solid, positively charged nuclei, surrounded by clouds<sup>2</sup> of negatively charged electrons. Electrons are retained by the atom due to the electrostatic attraction between the negatively charged electrons and the positively charged nucleus. An atom which is neutral overall has the same number of electrons as protons, since electrons and protons have equal but opposite charges.

Atoms have different types depending on the number of protons in the nucleus, which is called the atomic number. A hydrogen atom, for example, has one proton in its nucleus,

---

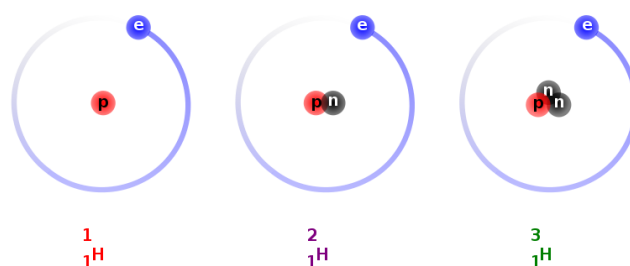
<sup>2</sup>We here present a simplified description of atoms, deferring questions on the precise quantum mechanical nature of electron orbitals, since this will not be relevant for our purposes.

thus has atomic number one. Carbon, on the other hand, has six protons in the nucleus, thus has atomic number six. Atoms which have the same atomic number are defined as being of the same chemical element, and share many important properties. There are 112 known elements. Different numbers of neutrons in the nucleus of atoms of the same element lead to atoms of the same types but with different weights, called isotopes (Figure 2.2 illustrates the isotopes of Hydrogen). However, the reactivity is much the same for different isotopes, as the reactivity is largely driven by electrostatic factors which depend on the protons and electrons, and only somewhat affected by the total mass, which depends on the number of protons and neutrons together, of the nucleus. Electrons are arranged around the nucleus of the atom in regions called orbitals, each of which can house a maximum of two electrons. The outermost orbitals form the location of the reactivity of the atom.

**Definition 2.1.** An *atom* is a small, massive, positively charged nucleus surrounded by a cloud of negatively charged electrons. In an atom of neutral charge, there are as many protons in the nucleus of the atom as there are electrons in the electron cloud.

**Definition 2.2.** An *isotope* is the name for the different variants of atoms which have the same atomic number, i.e. the number of protons in the nucleus, but different numbers of neutrons. For example, Carbon has atomic number 6 but isotopes with mass number, i.e. number of protons and neutrons together, 12, 13, and 14.

**Definition 2.3.** An *ion* is an atom with a non-zero charge, either positive or negative. Charge occurs in integral units (...,-2,-1,0,1,2,...) which count the number of extra, or missing, electrons surrounding that atom relative to the number of protons in the nucleus.

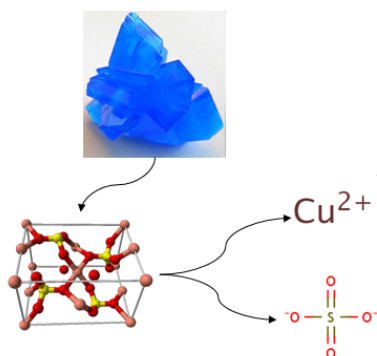


**Figure 2.2** Isotopes of Hydrogen

Atoms themselves are very small and the number of different kinds of atoms is not large enough to account for the range and diversity of chemical substances we observe

---

around us. However, atoms commonly join together semi-permanently in a process called chemical bonding to form molecules, which consist of multiple atoms joined together in stable configurations. There are many different ways for atoms to join together through a wide range of bond strengths and flexibilities. The strongest bond is the *covalent bond*, which is when two atoms share a pair of electrons in an energetically stable orbit between them. Units of covalently bound atoms are called *molecules*. *Ionic bonding* occurs when electrostatic attraction brings together a positively charged atom or molecule and a negatively charged atom or molecule.



**Figure 2.3** Copper(2) Sulphate Salt - an ionically bound crystal structure

Molecules joined together by ionic bonding are often called salts and form solids which are structured and crystalline. Figure 2.3 illustrates the crystal structure of Copper(2) Sulphate, an ionic crystalline solid. Many other, weaker forms of bonding or interaction between chemical entities do exist, such as metal bonding, hydrogen bonding and Van der Waals interactions, but we will discount these for the remainder of this document as being somewhat outside of the scope of the current requirements for chemical ontology. Covalent bonds may be single (one shared pair of electrons between the atoms), double (two shared pairs of electrons), triple (three shared pairs of electrons) or aromatic. *Aromatic* is a term used to refer to the situation in conjugated cycles of alternating single and double bonds, where the valence electrons seem to form a resonance pattern which distributes the electron orbitals stably around the ring structure.

**Definition 2.4.** A *covalent bond* is a bond between two atoms which consists of a shared pair of valence electrons positioned between the two atoms. Each atom generally contributes one electron to the shared pair on bond formation.

The concept of bonding is useful for a formal definition of a molecule in terms of

---

atoms and bonds. Informally, we can view a molecule as consisting of a set of connected atoms. We must therefore define connection, and this we do, in terms of covalent bonds.

**Definition 2.5.** A set of atoms are said to be *connected* if each atom can be reached from each other atom from a path which traverses only the bonds between atoms, i.e. without jumping from an atom to another atom without there being a bond present between the two.

Note that we will consider only covalent bonds as forms of connection in the remainder of this document, but it is plausible that a different treatment would extend the definition of connection to other weaker forms of chemical bonding as needed.

Now we are ready to define a molecule in terms of its atoms and their connection.

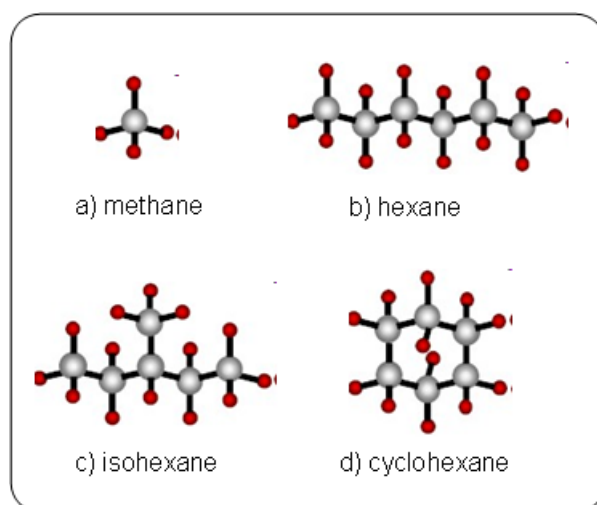
**Definition 2.6.** A *molecule* is a set of atoms which are connected (that is, each atom is connected to the whole) by covalent bonds. The term is generally used only to refer to entities which have a neutral charge overall, with *molecular entity* being used when the overall charge is not necessarily neutral.

**Definition 2.7.** A *molecular ion* is a molecular entity which has an overall non-zero charge.

The field of organic chemistry, in contrast to inorganic chemistry, is the domain of chemistry interested in the study of Carbon-containing molecules. Historically, the term ‘organic’ was used because it was concerned with the study of the molecules involved in the processes of living organisms. However, it was determined that these molecules almost universally contained Carbon, and the field was later generalised to include all Carbon-containing molecules whether biological or synthetic. The size and flexibility of the molecules which form the subject matter of organic chemistry extends far beyond what has been achieved in inorganic chemistry. This is largely due to the remarkable flexibility of Carbon in terms of the bonds it is able to form with neighbouring atoms.

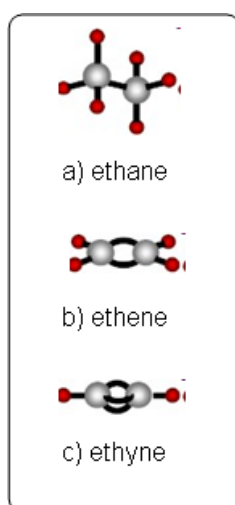
Carbon has four bonding electrons in its *valence shell*, which means that it can form four bonds with other atoms, each bond consisting of one of the Carbon atom’s electrons and one of the bonding atom’s electrons. A carbon atom’s bonds will be distributed evenly over the atom’s surface, forming a tetrahedron, as illustrated in Figure 2.4 (a).

What differentiates organic from inorganic chemistry is the many ways Carbon can bond to other atoms including other Carbon atoms. Carbon atoms bonded to other Carbon atoms can form chains such as in Figure 2.4 (b) (hexane), branched chains (Figure 2.4 (c), isohexane), and rings (Figure 2.4 (d), cyclohexane). Furthermore, neighboring Carbon



**Figure 2.4** Carbon compounds - methane, hexane, isohexane and cyclohexane

atoms can form double and triple bonds in addition to single bonds, as illustrated in Figure 2.5 (a), (b) and (c).



**Figure 2.5** Carbon compounds - ethane, ethene and ethyne

Carbon and Hydrogen thus chain together through Carbon-Carbon bonds to form complex structural skeletons for organic molecules. These skeletons may additionally have other kinds of atoms attached, and those are crucial for rendering the differences in bioactivity of the different kinds of organic molecule, but the main bulk of organic molecules is generally made up of the Carbon skeleton or backbone (Carpi, 2003).

**Definition 2.8.** A skeleton of an organic compound is the chains, branches and/or rings of carbon atoms that form the basis of the structure of an organic molecule.



Molecules may clearly be divided into parts, each of which may be interesting in its own right. It is common to refer to the atoms within the molecule, but also to refer to parts of the molecule which are especially interesting (by virtue of explaining the reactivity, or structural composition, of the molecule) as *groups*. A group is a molecular part together with one or more attachment points, which represent the location at which the group is bonded to the whole. In this document the only molecular parts we will consider will be those that are themselves internally connected (each atom within the part can be reached by every other atom via a path which traverses bonds). In a more general treatment structurally disconnected parts might be considered, such as pharmacophores, which are a set of groups or features of a molecule which are spatially dispersed but which together are responsible for a certain kind of biological or pharmacological activity, usually described in terms of the parts and the angles and distances between them.

In organic chemistry, groups which are attached to the main carbon-hydrogen skeleton of the molecule are commonly termed functional groups as they are usually the sites of highest reactivity of the molecule and responsible for many of the molecule's properties.

**Definition 2.9.** A *group* is a defined linked collection of one or more atoms, within a molecular entity.

**Definition 2.10.** A *functional group* is an atom, or a group of atoms that has similar chemical properties whenever it occurs in different compounds. It defines the characteristic physical and chemical properties of families of organic compounds (McNaught and Wilkinson, 1997).

We find it convenient to refer to “chemical entities”, to be understood as any of the different named individual chemical entities mentioned here, together with others which we have not defined here but which include *radicals* (a molecular entity with an unpaired electron; the unpaired electron makes the entity highly reactive) and *residues* (a kind of group, having two attachment points rather than one, commonly used to refer to the amino acids which chain together to form polypeptide chains such as Deoxyribonucleic acid (DNA) and Ribonucleic acid (RNA)).

**Definition 2.11.** A *chemical entity* is any constitutionally or isotopically distinct atom, molecule, ion, ion pair, radical, radical ion, complex, conformer etc., identifiable as a separately distinguishable entity (McNaught and Wilkinson, 1997).

## 2.2 Properties of chemical entities

In this section we define some properties of chemical entities which may be relevant for chemical classification.

Aromaticity is the name used to refer to compounds which contain at least one cyclic part (ring) which is composed of an alternating pattern of single and double bonds<sup>3</sup>. Rings of this type are unusually stable; far more stable than the double bonds they are composed of would usually warrant. Furthermore, their atoms show even spacing, while ordinarily double bonds have a shorter bond length than single bonds. It is now understood that aromatic cycles are actually forming a conjugated system with the bonds “shared” across the ring, such that each bond can be thought of as being composed of 1.5 shared electron pairs rather than one or two.

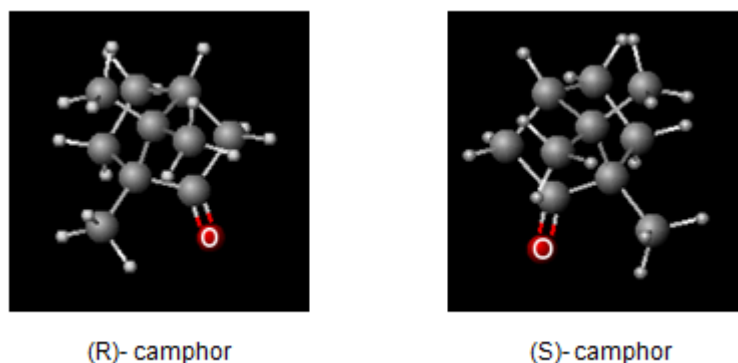
**Definition 2.12.** *Aromaticity* is a chemical property in which a conjugated ring exhibits a stabilization stronger than would be expected by the stabilization of conjugation alone. It can be considered a manifestation of cyclic delocalization and of resonance.

The stereochemistry of a molecule refers to the spatial arrangement of the atoms within the molecule. Sometimes, a molecule can have the same composition, even to the level of the bonds between individual atoms, and still have different properties, by virtue of being shaped differently. Molecules constituted by identical atoms are termed *isomers*, and molecules constituted by identical atoms and bonds, but nevertheless displaying different spatial arrangements, are termed *stereoisomers*. Molecules are termed *chiral* when they lack an internal plane of symmetry and have a mirror image stereoisomer form. The mirror images are called *enantiomers*. Figure 2.6 illustrates two chiral molecules. Notice that they are mirror images of each other. However, should the one be rotated, the relevant hydroxy group would in fact be pointing upwards from the plane of the remainder of the molecule, not downwards.

A term commonly used to describe molecules in organic chemistry is *saturation*: molecules may be described as being either saturated or unsaturated. Here, what is meant by saturated is that the molecule contains only single bonds in the hydrocarbon skeleton. A molecule with double or triple bonds in the skeleton, such as benzene, is described as unsaturated. This categorisation is useful for describing several properties of molecules such as the reactivity and the structural rigidity.

---

<sup>3</sup>This is a slight simplification of the full story, as not all rings with alternating single and double bonds are aromatic. The exact criteria for deducing aromaticity are given by Hückel’s law. For details see (Trinajstić, 1992).

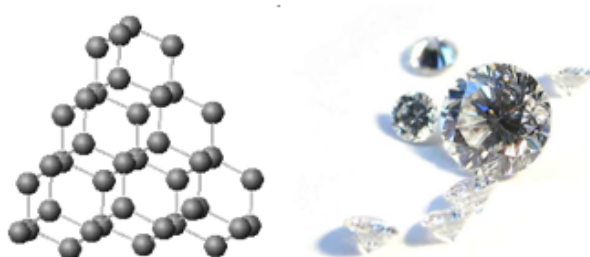


**Figure 2.6** Chiral molecules - S-camphor and R-camphor

Finally a property of molecules which is often referred to explicitly in organic chemistry is the presence of, and number of, cycles and fused cycles. A *cycle* is a bond path through a molecule that returns to the same atom; two cycles are *fused* if they share at least one bond (but are nevertheless distinct cycles, that is, there is at least one bond which is not shared).

## 2.3 Chemical substances

Larger molecules, called macromolecules, are sometimes formed in extended covalently bound units. The division between small molecules and macromolecules is vague, however, and at least somewhat dependent on perspective and context. Diamonds, for example, as illustrated in Figure 2.7, consist entirely of a single huge covalently bound molecule. [DNA](#), the molecule on which the genetic code of life is encoded, also consists of a single long covalently bound molecular chain. Other biochemical macromolecules include proteins and the many different kinds of [RNA](#).



**Figure 2.7** Diamond - a single large covalently bound molecule

Aggregates of molecules, atoms, ions or other kinds of chemical entities are called chemical substances or chemical species. A chemical substance usually consists of very many individual chemical entities. For the most part, chemical substances are what we deal with in everyday life and in the laboratory. It is usually very difficult to isolate or observe a particular individual molecule, although advances in microscopy have made this possible in recent years. Chemical substances may be homogeneous (consisting of individuals of only one kind) or heterogeneous (mixed). Many properties commonly ascribed to chemical entities are best understood as being properties of substances, for example state (solid, liquid, gas), or boiling or melting point, do not make sense as a property of a single molecule, but only of an aggregate (substance).

**Definition 2.13.** A *chemical substance* is an aggregate collection of chemical entities.

In the next chapter we discuss chemical structures and present the chemical graph, a commonly used formalism for describing chemical structures in organic chemistry.

# 3

## Chemical Structures

In this chapter, we present chemical structures as special properties of complex chemical entities, and discuss the chemical graph formalism. We present the representational formats which are commonly used to represent and manipulate the structural information of chemical entities in Section 3.1, and publicly available sources of data on chemical structures in Section 3.2.

The term *chemical structure* refers to the connection of atoms within a complex chemical whole. For example, molecules are composed of atoms connected by hydrogen bonds, and salts are composed of charged ions connected by ionic bonds.

Graph theory, as a branch of mathematics, is concerned with mathematical entities which describe the way in which objects are connected<sup>1</sup>. The principal entity in graph theory is the graph, which includes a set of objects as vertices, and the binary relations between those objects as arcs, or edges, of the graph. Graph theory has found many applications in chemistry where graphs can be used to represent many different chemical objects including molecules, reactions, crystals and clusters (Trinajstić, 1992). The particular type of chemical graph which will be the focus here is the molecular graph, which represents the constitution of a molecule in terms of its atoms and bonds.

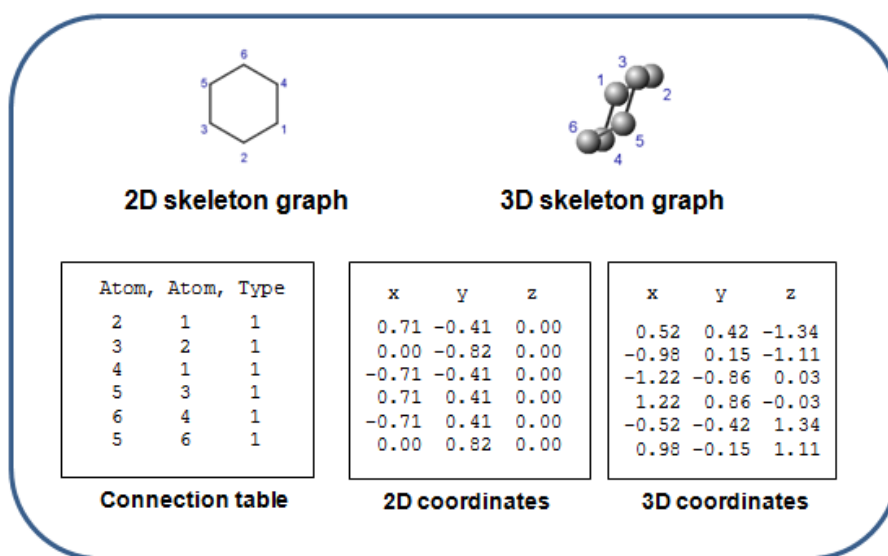
The chemical, i.e. molecular and constitutional, graph describes the atomic connectivity within a molecule in terms of labelled nodes for the atoms or groups within the molecule, and labelled edges for the (usually covalent) bonds between the atoms or groups. The graph, strictly speaking, thus encodes only the constituents and their bonds. However, the representational formalism is usually extended to include other information such as idealised two dimensional (2D) or three dimensional (3D) coordinates for the atoms, bond order (single, double, triple) and type (e.g. aromatic). In diagrammatic

---

<sup>1</sup>Parts of this section on chemical graphs first appeared in (Hastings *et al.*, 2010).

representation, bond order is illustrated by the number of parallel lines connecting atoms – one line for a single bond, two lines for a double bond, and so on – and aromatic bonds are represented by either a single line plus a dashed line or an alternating pattern of single and double lines. Chirality, i.e. stereochemistry, is illustrated by a triangle-shaped ‘wedged’ bond which may be solid (indicating the direction up) or dashed (indicating the direction down). Figure 3.1 illustrates examples of chemical graphs, 2D and 3D coordinates, and the corresponding 2D and 3D visualisations as are commonly used in chemistry.

Note that hydrogen nodes, and the edges linking them to their nearest neighbouring atoms, are not explicitly displayed in Figure 3.1. This is common in the representation of chemicals by graphs, since the presence and location of hydrogen atoms in the molecule can be inferred from the nature and connection of the remaining atoms. Also notice that carbon atoms, which form the ‘corners’ of the images as displayed, are not labelled as such. These representational economies are due to the prevalence of carbon and hydrogen in organic chemistry, thus allowing these efficiencies which have been introduced for clarity of depiction and efficiency in storage. Hydrogen-suppressed graphs of this form are called *skeleton graphs*.



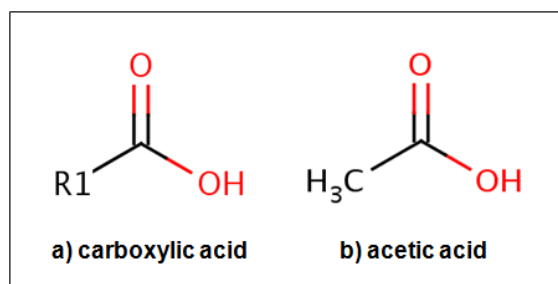
**Figure 3.1** Chemical skeleton graphs and coordinates

The coordinates given in Figure 3.1 are given in ‘coordinate space’ with respect to the axes of the images, that is, they are unitless. Further chemical detail which can be encoded in various different representational formalisms includes charge (positive or negative), radicals (unpaired electrons) and isotopes; types of bonds (single, double, triple,

---

or aromatic<sup>2</sup>) and chirality<sup>3</sup>.

Chemical graphs simplify the complex nature of a molecule, and enable the schematic visualisation of molecules that are commonly used by chemists. They also enable various useful predictions to be made about physical and chemical properties of molecules which are based on connectivity. They are crucial tools in the domain of chemoinformatics, allowing accurate computable representation and identification of chemical entities. Chemical graphs also allow for the representation of classes of chemical entities, where a *chemical class* should be understood as a set of molecules which share a common element and potentially also a common reactivity or other interesting chemical feature. Chemical classes are often illustrated by means of one or more ‘R’ groups being included in the chemical graph, which may be replaced with any chemical group attached at that point. Figure 3.2 shows an ‘R’ group representation of the class of carboxylic acids. The group labelled R1 in Figure 3.2 (a) may be replaced with any other molecular group. For example, in acetic acid (vinegar) illustrated in Figure 3.2 (b), R1 is replaced by a methyl group (CH<sub>3</sub>).



**Figure 3.2** R group structural class definition for Carboxylic Acid

Much of organic chemistry can be expressed by using the graph formalism. However, it should be noted that chemical graphs do not describe everything in chemistry. They do not work for non-pairwise bonding as found in metal coordination compounds, ionic bonding as found in minerals, and hydrogen bonding which is responsible for base pairing in DNA and RNA. It is also impossible to draw a chemical graph that defines broad classes such as ‘organic molecule’, or ‘carbohydrate’, and similarly most of the families of naturally-occurring chemical compounds. Furthermore, there are molecules such as D-glucose that rapidly interconvert (tautomerize) between forms with very different

---

<sup>2</sup>In the chemical graph formalism, aromaticity may be deduced from an alternating pattern of single and double bonds in a conjugated cyclic system.

<sup>3</sup>Chirality can be ‘hacked’ into the 2D connection table by creating special sorts of edges (‘up’ and ‘down’), or not specially represented but following from the properties of the Cartesian coordinates of the atoms in a 3D connection table

---

(ring-like and chain-like) chemical graphs.

Nevertheless, a broad range of chemical entities are able to be represented with chemical graphs by various conventions which are widely used and agreed on for approximate representation within the limits of the graph formalism. For example, polymers are usually represented by partial, schematic chemical graphs, in which the structure of the repeating unit is explicitly represented together with linkage points on either end. Ionic bonding is not considered as resulting in a single molecule, thus, salts are represented as two (or more) separate chemical graphs, on each of which a non-zero charge is specified.

In this section we have introduced the chemical graph formalism, and in the next section we discuss specific technical encoding formats which have been developed around this formalism.

## 3.1 Representation

Together with the development of chemical graph theory and the theoretical operations used in predictions of properties, standards for the computational representation of the graphs themselves were developed<sup>4</sup>. The core representational formalism is as a decorated connection table ([Gasteiger and Engel, 2003](#)), with both atoms and their connections (bonds) being assigned additional properties to further describe the molecule. Other, more compressed forms of representation have been developed, such as the popular line notations which compactly represent the structure while allowing for efficient computer-based searching.

Figure 3.3 illustrates the connection table formalism that underlies all of the following specific representation formats. Each atom in the chemical structure graph is given a numeric identifier in an atom table, and then bonds are illustrated as connections between atom numbers illustrated in a bond table.

Note that the numbering of the atoms in a molecule needs to be reproducible if molecular representations are to be compared. Algorithms thus exist which generate unique (canonical) numberings for the atoms within molecules in a consistent fashion.

### 3.1.1 MOLfile

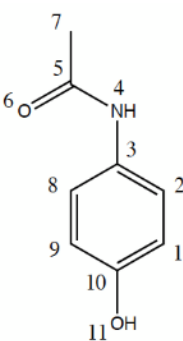
The MOLfile format is owned by the Elsevier MDL company ([MDL, 2010](#)), and is the most commonly used format for chemical data exchange. It is a flat ASCII text file with

---

<sup>4</sup>Parts of this section on representation of chemical graphs first appeared in ([Hastings and The ChEBI Team, 2010](#)).



Atom Number	Atom Type
1	C
2	C
3	C
4	N
5	C
6	O
7	C
8	C
9	C
10	C
11	O

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	0	0	2	0
2	1	0	2	0	0	0	0	0	0	0	0
3	0	2	0	1	0	0	0	1	0	0	0
4	0	0	1	0	1	0	0	0	0	0	0
5	0	0	0	1	0	2	1	0	0	0	0
6	0	0	0	0	2	0	0	0	0	0	0
7	0	0	0	0	1	0	0	0	0	0	0
8	0	0	1	0	0	0	0	0	2	0	0
9	0	0	0	0	0	0	0	2	0	1	0
10	2	0	0	0	0	0	0	0	1	0	1
11	0	0	0	0	0	0	0	0	0	1	0

**Figure 3.3** Connection table for paracetamol from ([Indiana Cheminformatics Education Portal, 2010](#))

a specific format, consisting of an *atom table*, describing the atoms contained in the chemical entity, and a *bond table*, describing the bonds between the atoms. Both the atom table and the bond table are extended with additional properties including the isotope and charge of the individual atoms, and the bond order and type of the bonds.

Figure 3.4 shows the MOLfile for the 2D representation of paracetamol.

### 3.1.2 Chemical Markup Language

Chemical markup language (CML), is an extensible eXtensible Markup Language (XML)-based representation of the same information which is encoded in the MOLfile format ([Murray-Rust \*et al.\*, 2001](#)). The benefit of using CML rather than the MOLfile is that, like most flat-file-based formats, the MOLfile format is inflexible and vulnerable to minor formatting errors rendering the entire file incomprehensible by software (i.e. it is very ‘brittle’). Furthermore, it is difficult to extend since with any change to the file format, all software and tools must be redeveloped to work with the new format. This is avoided with an extensible XML format such as CML in which the meaning of the data is encoded in the file together with the data, rather than being specified only by a very specific

```

Marvin_03190821382D
11 1 0 0 0 0 999 V2000
0.7145 0.4125 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.7145 -0.4125 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.8250 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 -0.8250 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-0.7145 -0.4125 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-0.7145 0.4125 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1.4288 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.7145 0.0000 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.7145 0.0000 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 1.6500 0.0000 N 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 -1.6499 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 1 1 0 0 0 0 0
1 3 2 0 0 0 0 0
10 3 1 0 0 0 0 0
4 2 2 0 0 0 0 0
4 11 1 0 0 0 0 0
4 5 1 0 0 0 0 0
6 5 2 0 0 0 0 0
3 6 1 0 0 0 0 0
7 8 1 0 0 0 0 0
8 9 2 0 0 0 0 0
8 10 1 0 0 0 0 0
M END
  
```

**Figure 3.4** MOLfile format for Paracetamol

formatting location within the file, thus allowing formatting rearrangements without loss of interpretability.

The formatted text in Figure 3.5 shows the **CML** for the **3D** representation of paracetamol. The core information content of a **CML** file is equivalent to, and losslessly interconvertible with, the MOLfile representation. But the **CML** contains additional meta-data in the preamble of the file which was not contained in the MOLfile representation – for example, formula and systematic name – which is possible because the **CML** is extensible.

This **CML** file includes three-dimensional coordinates is included below. This file was adapted from that provided as the structure of a molecule found bound to a protein structure in the Protein Data Bank (**PDB**) (**PDB**, 2010), molecule code ‘TYL’; hence the availability of three-dimensional coordinates, as the **PDB** is a repository for the **3D** structures of proteins together with bound ligands.

Notice that the **CML** representation of the molecule is more verbose than the MOLfile, and indeed atoms and bonds have been abbreviated in this illustrative image. As each data item is explicitly encoded in **XML**, it allows the meaning of the data to be much more readily interpreted, whereas the MOLfile is almost uninterpretable unless you know the meaning of the different columns and blocks in the file format beforehand.

```

<?xml version="1.0" encoding="UTF-8"?>
<cml xsi:schemaLocation="http://cml.sourceforge.net/schema/cmlCore.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      dictRef="ebiMolecule:ebiMoleculeDict.xml"
      xmlns:ebiMolecule="http://www.ebi.ac.uk/felics/molecule">
  <molecule id="TYL" formalCharge="0">
    <identifier dictRef="ebiMolecule:inchi">
      InChI=1/C8H9NO2/c1-6(10)9-7-2-4-8(11)5-3-7/h2-5,11H,1H3,(H,9,10)/f/h9H
    </identifier>
    <identifier dictRef="ebiMolecule:systematicName">
      N-(4-hydroxyphenyl)acetamide
    </identifier>
    <formula dictRef="ebiMolecule:stereoSmiles">
      CC(=O)Nc1ccc(O)cc1
    </formula>
    <formula dictRef="ebiMolecule:nonStereoSmiles">
      O=C(Nc1ccc(O)cc1)C
    </formula>

    <atomArray>
      <atom id="C1" elementType="C" hydrogenCount="0" formalCharge="0"
            x3="0.462" y3="0.191" z3="0.227"/>
      <atom id="C2" elementType="C" hydrogenCount="1" formalCharge="0"
            x3="-0.832" y3="0.424" z3="-0.216"/>
      <atom id="C3" elementType="C" hydrogenCount="1" formalCharge="0"
            x3="-1.132" y3="0.312" z3="-1.559"/>
      <atom id="C4" elementType="C" hydrogenCount="0" formalCharge="0"
            x3="-0.14" y3="-0.033" z3="-2.465"/>
      <atom id="C5" elementType="C" hydrogenCount="1" formalCharge="0"
            x3="1.153" y3="-0.266" z3="-2.022"/>
      <...>
    </atomArray>
    <bondArray>
      <bond atomsRefs2="C1 C10" order="4"/>
      <bond atomsRefs2="C10 O" order="1"/>
      <bond atomsRefs2="C9 C10" order="4"/>
      <bond atomsRefs2="C8 C9" order="4"/>
      <bond atomsRefs2="C3 C8" order="4"/>
      <...>
    </bondArray>
  </molecule>
</cml>

```

Dictionary reference

Metadata about the compound

Atoms, with x, y, z coordinates

Bonds between named atoms from atomArray section

Figure 3.5 CML format for Paracetamol

### 3.1.3 SMILES

Simplified Molecular Input Line Entry System (**SMILES**) is a line notation (a typographical method using printable characters) for entering and representing molecules and reactions. The original **SMILES** specification was developed by Arthur Weininger and David Weininger in the late 1980s. It has since been modified and extended by others, most notably by Daylight Chemical Information Systems Inc. ([Daylight, inc., 2010](#)).

The **SMILES** format caters for unique representations of distinct chemical entities. A drawback is that the **SMILES** output can depend on the program that generates it.

The **SMILES** for paracetamol is:

---

```
CC(=O)Nc1ccc(O)cc1
```

---

Here again, hydrogens are implicit. Also, to reduce the space of the representation, single bonds are also implicit between neighbouring atoms. Atoms are numbered when necessary to illustrate where the same atom appears again (i.e. when the molecule contains a cycle). Brackets indicate branching, and '=' indicates a double rather than a single bond. Lowercase letters indicate aromaticity.

The [SMILES](#) representation retains a high degree of human readability while compressing the structural encoding of the molecule into as few characters as possible. However, as we mentioned before, different algorithms exist which produce different [SMILES](#) codes for the same molecule, so care must be taken to cite the implementation used when storing structures of molecules primarily as [SMILES](#).

### 3.1.4 InChI and InChIKey

The IUPAC International Chemical Identifier ([InChI](#)) is a non-proprietary, structured textual identifier for chemical entities ([IUPAC, 2010](#)) which is generated by an algorithm from a MOLfile representation of the chemical entity. The purpose of the [InChI](#) is to provide an identifier for the chemical entity independently of how it is drawn, in contrast to the MOLfile for a particular structure, which does differ depending on how the structure is drawn. The generated [InChI](#) identifier is not intended to be read and understood by humans, but is particularly useful for computational matching of chemical entities, such as when a computer has to align two different sets of data derived from different sources.

The [InChIKey](#) is a hashed key for the [InChI](#), which allows easier database lookups as it has fewer characters and is of a specified, invariant length of 14 characters.

The [InChI](#) and [InChIKey](#) for paracetamol are:

---

```
InChI=1/C8H9NO2/c1-6(10)9-7-2-4-8(11)5-3-7/h2-5,11H,1H3,(H,9,10)/f/h9H  
InChIKey=RZVAJINKPMORJF-BGGKNDAXCW
```

---

In this section we showed some commonly used representational formalisms for chemical graphs. In the next section we discuss the main publicly available sources for chemical data, including chemical structures in one or more of the above representation formalisms.

## 3.2 Sources of chemical data

The bioinformatics community has developed a policy of open access and open data since its inception. This is in sharp contrast to the situation in the field of cheminformatics, which has traditionally been a closed-access area. Chemical structures and associated data were published in expensive journals and curated into commercial, expensive databases such as Chemical Abstracts Service ([CAS](#)), provided by the American Chemical Society ([CAS, 2010](#)), and Beilstein, provided by Elsevier since 2007 ([Beilstein, 2010](#)). In recent

years this has been changing with more and more chemical data being brought into the public domain. This has been in part due to the efforts of the bioinformatics community, which needed access to chemistry data to support systems-wide integrative research. It has also been in part due to the joint efforts of pharmaceutical companies to reduce the expense of pre-competitive research, since pharmaceutical companies have historically each expensively maintained their own database of chemicals for pre-competitive research (Marx, 2009).

In 2004, two complementary open access databases were initiated by the bioinformatics community, Chemical Entities of Biological Interest (ChEBI) (de Matos *et al.*, 2010) and PubChem (Sayers, 2005). PubChem, containing 26 million compound structures, serves as automated repository on the structures and biological activities of small molecules. ChEBI is a manually annotated database of small molecules containing around 550000 compound structures. Both resources provide their chemical structures in several different formats and include additional useful information such as names and calculated chemical properties, and therefore are sources of chemical structural data for primary academic research in many domains.

ChEBI additionally provides a chemical ontology, which is discussed further in Chapter 11. Additional publicly available resources for chemistry information which are becoming more widely used are ChemSpider (Williams, 2008), which provides a integrated cheminformatics search platform across many publicly available databases, and Wikipedia Chemistry (Wikipedia, 2010). Many smaller databases also exist, often dedicated to particular topic areas or types of chemicals. For a full listing of publicly available chemistry data sources, see (CHEMBIOGRID, 2010). For a discussion see (Williams, 2009).



# 4

## Description Logics

In this chapter, we describe the Description Logics formalism for knowledge representation. We start with the historical development of logical representation in Section 4.1 and then describe the development of semantic networks in Section 4.2. We show how a First-order logic (FOL) formalism can be used to give a semantics to semantic networks in Section 4.3. The definition of complex concepts is discussed in Section 4.4 and the specification of the properties of individuals in a knowledge base in Section 4.5. In Section 4.6 we characterise the semantics of DLs. Section 4.7 gives an overview of the standard reasoning tasks and methods. Finally, selected important basic DLs are presented in Section 4.8.

Knowledge representation is the branch of artificial intelligence that is concerned with the representation of human knowledge in a form that renders it accessible for computational processing (Nardi and Brachman, 2007). The goal is to provide descriptions of the world in different domain areas that can be used effectively by intelligent applications. Intelligence in this context refers to the ability of the system to derive the *implicit consequences* of the facts which are explicitly provided to it.

Historical approaches to knowledge representation can be divided into those which were developed within the formal predicate logic, with inferences drawn by verifying logical consequences of the axioms captured, and others which were modelled around ideas emerging from studies in cognitive psychology of how humans encoded knowledge in interconnected semantic networks.

## 4.1 Logical representation

The logic-based approach to knowledge representation uses the tools of formal logic and logical deduction to represent and reason over knowledge expressed as logical axioms, in particular the predicate, or first-order logic (FOL). In the following section, we follow the terminology and definitions introduced in [Ben-Ari \(2001\)](#).

**FOL** is used to model axioms about *predicates*, which are relations that operate on *values* from a particular *domain*.

**Definition 4.1.** A predicate is a relation  $R$  that takes  $n$  values from a domain  $D$  and maps them into the boolean domain  $\{T, F\}$ . The number of values that a predicate takes from the domain is called the *arity* of the predicate. Predicates taking one value are called *unary* predicates, those taking two values are called *binary* predicates, those taking three values are called *ternary* predicates, and so on.

A formula in FOL is built up from *predicate symbols*, which represent relations over some domain of interest; *variables*, which range over values from the domain; *constants*, which represent domain elements, the usual boolean connectives, as well as two *quantifiers*,  $\forall$  and  $\exists$ .

The quantifiers used in predicate logic are  $\forall$  (for all), which means that the formula within the scope of the quantifier must hold for *all* values within the domain; and  $\exists$  (there exists), which means that the formula within the scope of the quantifier must hold for *some* value within the domain. For example,

$$\forall x \forall y \text{ Mother}(x, y) \longrightarrow (\text{Female}(x) \cap \text{Parent}(x, y))$$

means that for every  $x$  and  $y$  within our domain of interest (which in this case is human beings), if  $x$  is the *Mother* of  $y$ , then we can derive that  $x$  is *Female*, and furthermore that  $x$  is the *Parent* of  $y$ . This is another example,

$$\forall x \exists y \text{ Parent}(y, x)$$

means that for all  $x$  (human beings, in our domain) there is some  $y$  such that  $y$  is the parent of  $x$ . Note that the following similar-seeming formulation would *not* be true in our domain,

$$\forall x \exists y \text{ Parent}(x, y)$$

as this states that for all  $x$  there exists some  $y$  such that  $x$  is the parent of  $y$ , and this is clearly not the case, since there are many people who do not have children in their lifetimes.

Formally, the semantics of formulae in **FOL** are given by assigning *Interpretations* to the formulae. An interpretation assigns values to the variables and constants of a formula



from entities in the domain, such that the truth value of the formula can be evaluated.

**Definition 4.2.** Given  $U$ , a set of formulas such that  $\{p_1, \dots, p_m\}$  are all the predicate letters and  $\{a_1, \dots, a_k\}$  are all the constant symbols appearing in  $U$ , an *interpretation*  $I$  is a triple

$$(D, \{R_1, \dots, R_n\}, \{d_1, \dots, d_n\}),$$

where  $D$  is a *non-empty* domain,  $R_i$  is an assignment of an  $n_i$ -ary relation on  $D$  to the  $n_i$ -ary predicate letter  $p_i$  and  $d_i \in D$  is an assignment of an element of  $D$  to the constant  $a_i$  (Ben-Ari, 2001).

The first-order predicate logic provides a powerful framework for encoding knowledge which provides many tools for making deductions or inferences from the expressed knowledge. For example, from a statement that all parents are tired, and that John is a parent, we can infer that John is tired. Formally,

$$\forall x \text{ Parent}(x) \longrightarrow \text{Tired}(x)$$

$$\text{Parent}(\text{JOHN})$$

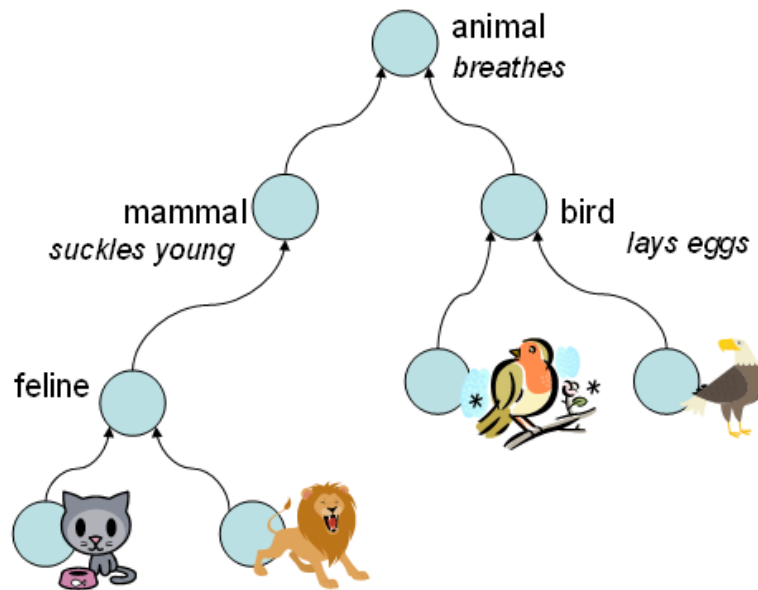
The system can then conclude that

$$\text{Tired}(\text{JOHN})$$

## 4.2 Semantic networks and frames

Semantic networks consist of entities and connections between entities. They were based on studies in cognitive psychology, and considered to be close to the way that knowledge is actually stored in the human brain. Evidence for the interconnected network storage model is given by experiments on human cognition which measured that the response time for assessing category statements (such as ‘a robin is an animal’) increased as the relative hierarchical distance between the subject (robin) and the object (animal) increased (Sternberg, 2003). It was also found that the response time for retrieving properties associated with concepts increased the more general the property was. That is, it took less time to answer ‘do robins have red breasts’ than it did to answer ‘do robins breathe’, where the latter (breathing) can be seen as a property that is general to a large number of living creatures, and thus it might be stored at a further distance in the cognitive knowledge network (being associated with ‘animal’ rather than with ‘robin’).

Figure 4.1 illustrates the core connection type in a semantic network, namely the hierarchical connection, which is called ‘IS-A’. It connects more specific entities to more general entities. The benefit of the hierarchical organisation is that properties can be



**Figure 4.1** A semantic network populated with some animal concepts

inherited by all descendents of a general entity, which allows for a compact and concise representation. For example, robin ‘IS-A’ bird and bird ‘IS-A’ animal. Other relationships can be used to specify any kind of property relevant for encoding knowledge in the domain.

Inheritance of properties from general to specific entities in the hierarchy is one form of inference from explicit to implicit knowledge within semantic network structures that can be performed by intelligent applications. Additional forms of inference can make use of additional relationships in order to traverse links between entities and deduce new facts based on the structure of the network.

In addition to a position in the graph, both hierarchically and in terms of custom relationships to specify properties, entities in a semantic network may have a prescribed *internal structure*. For example, an entity *Person* may have prescribed attributes *age* and *name*. Such structured entities are referred to as *Frames*, where the frame can be thought of as a template for an entity in that domain. Since an attribute of a frame can itself be another frame, this representation is immediately very powerful in what it can represent.

Due to their similarity to the supposed structure of cognitive representations, their compactness, and their expressive power, semantic networks and frames became very popular as forms of knowledge representation (Nardi and Brachman, 2007). However, they provided no *semantics* for the knowledge encoded, which hindered the development

of computational tools and algorithms to work with knowledge represented in that format, provide consistency checking, and computationally draw inferences from the explicit knowledge. Furthermore, in the absence of formal semantics, system implementations diverged in their interpretation of the underlying data, with the result that different systems behaved differently even when presented with very similar underlying network structures and relationships.

An important step in the direction of providing semantics and formal rigour to semantic networks and frames was the recognition that the core features of frames and semantic networks could be given a semantics within **FOL** by regarding the entities within the network as *unary predicates* and the relationships between entities as *binary predicates*.

It was then recognised that encoding the features of the semantic networks did not require the full expressive power of **FOL**, but that such a characterisation of semantic networks used a *fragment* of **FOL**. In fact, different properties of semantic networks lead to different fragments of **FOL**. Each fragment has distinct properties and profiles with respect to drawing computational inferences, and many optimisations were able to be introduced to solve specific reasoning problems for specific fragments which would not have applied to full general **FOL** reasoners.

It is from research in this area that the field of *Description Logics* was born. Initially, research began under the label *terminological systems* to emphasise that logical formalisms were being used to create models for the terminology and relationships in a particular domain, following on from semantic networks. However, over time, the focus gradually shifted to the logical properties of the different fragments, and the term *Description Logics* became popular.

## 4.3 From semantic networks to Description Logics

The core elements in a **DL** knowledge base are *concepts* and *roles*. The concepts represent entities in a domain, corresponding to the nodes in a semantic network. The roles represent relations between concepts, corresponding to the links in a semantic network. Considered as a fragment of **FOL**, concepts can be viewed as *unary predicates* and roles as *binary predicates*<sup>1</sup> (Nardi and Brachman, 2007).

When modelling an extract of the given example semantic network in Figure 4.1,

---

<sup>1</sup>Note that some expressive **DLs** do admit roles with arity greater than two, but most **DLs** do not, so for purposes of this introduction we find it sufficient to define roles as having arity two.

we might have concepts such as *Feline* and *Mammal*, and roles such as *has\_child* and *suckles*.

Regarding the concepts and roles as unary and binary predicates in a FOL interpretation, taking values from a domain  $D$ , we might be interested in modelling, for example, that mammals suckle their young. This can be represented in FOL as:

$$(\forall x \forall y)[(Mammal(x) \wedge Mammal(y) \wedge has\_child(x, y)) \rightarrow suckles(x, y)]$$

The hierarchical relationship ‘IS-A’, which is the backbone of the hierarchical network-based approach to knowledge representation, can also be given semantics within a logical framework. Recall that ‘IS-A’ in a semantic network connects more specific entities to more general entities, and properties are inherited from more general to more specific entities in the hierarchy. But this hierarchical relationship corresponds to *set inclusion* in the semantics of FOL. The entities can be interpreted as unary predicates picking out a set of members from a domain. The more specific entities pick out a subset of members of the domain relative to that picked out by the more general entity. In the case of our example *Feline* as a more specific entity and *Mammal* as a more general entity, we have that *Feline* ‘IS-A’ *Mammal* in our semantic network, and this can be represented in FOL as:

$$\forall x, Feline(x) \rightarrow Mammal(x)$$

That is, the set of all elements of the domain for which *Feline* is true, is a subset of that for which *Mammal* is true:

$$\{x^I \in D : Feline^I(x^I)\} \subseteq \{y^I \in D : Mammal^I(y^I)\}$$

where  $x$  and  $y$  are variables in the FOL language, *Feline* and *Mammal* are unary predicate symbols, and  $I$  is an interpretation with domain  $D$ . The unary predicate *Feline* is interpreted as the set of felines by  $I$ , and the predicate *Mammal* is interpreted as the set of mammals, and in the interpretation, the set of felines is a subset of the set of mammals.

DLs thereby give a precise semantics to the meaning of nodes and links in semantic networks. However, DLs are far more expressive than semantic networks, since DLs provide formal *concept constructors* which allow complex logical *definitions* for concepts in the domain to be built from the basic building blocks provided by the language and the atomic (undefined) roles and concepts within the domain (Nardi and Brachman, 2007).

After a brief digression to remark on the syntax which will be used in the remainder of this dissertation, we will give more detail about concept definitions in Section 4.4.

### 4.3.1 A note about syntax

DLs commonly use an abbreviated syntax compared to FOL, by making implicit the universal quantification of variables. That is, given concepts such as *Feline* and *Mammal* in the previous example, and an assertion such as:

$$(\forall x)Feline(x) \rightarrow Mammal(x)$$

we instead refer simply to *Feline*, *Mammal*, dropping the reference to variables altogether, written as:

$$Feline \sqsubseteq Mammal$$

The DL convention therefore makes implicit the universal quantification of statements.

We will use the term *extension* of a concept expression to refer to the members of the domain for which a particular concept expression is true. That is, the extension of the *Mammal* concept is the set of all members of the domain which are mammals.

Similarly, role expressions are commonly expressed in a variable-free syntax as  $\forall R.B$ .

## 4.4 The definition of complex concepts

The power of DLs lies in the ability to define *complex concepts* from the building blocks of the DL language and the basic (*atomic*) concepts and roles defined in the domain. Let us assume we have defined atomic concepts *A* and *B* in our domain and an atomic role *R*. We can then *define* new concepts in terms of these atomic concepts and roles using the constructors of the DL language (Nardi and Brachman, 2007).

For example, we can define a new concept in terms of the *conjunction* of atomic concepts, such as:

$$C \equiv A \sqcap B$$

This means that the extension of the concept *C* is precisely those domain elements that are in the extension of *A* and also in the extension of *B*.

Alternatively, we can define a new concept in terms of the *disjunction* of atomic concepts, such as:

$$C \equiv A \sqcup B$$

This means that the extension of the concept *C* consists of those domain elements that are either in the extension of *A* or in the extension of *B*.

Concepts may also be defined in terms of roles by using either *value* or *existential* restrictions on the role fillers. Recall that roles are binary predicates between members of the domain. An existential restriction, expressed as  $\exists R.A$ , picks out those domain

members which are in the relationship  $R$  with *some* member of the extension of the concept  $B$ .

A value restriction, expressed as  $\forall R.A$ , picks out those domain members where, *if* they are in the relationship  $R$  with another domain member, then the other domain member is part of the extension of the concept  $B$ . The value restriction is therefore a restriction on the *filler* of the role to *only* members of the extension of  $B$ .

For example, the following concept definition picks out the set of all domain members for which, if they participate in the role  $R$ , the target of the role is a member of the extension of concept  $B$ :

$$C \equiv \forall R.B$$

In fact, not only can complex concepts be defined in terms of *atomic* concepts, but the same apparatus can be used to define complex concepts in terms of *other complex concepts*. That is, if there is a complex concept  $C$  which has already been defined in terms of atomic concepts (or roles), we can make use of that complex concept in definitions of other complex concepts in the knowledge base. This allows increasingly complex and sophisticated definitions of concepts in the domain to be constructed in increasing levels of complexity.

#### 4.4.1 Constructors

Conjunction, disjunction, value and existential restrictions are all examples of **DL constructors** for the definition of complex concept expressions. A wide variety of such constructors are available, and different **DL** languages are distinguished by offering different combinations of such constructors, which leads to different complexity profiles for various reasoning tasks<sup>2</sup>.

While we do not attempt to give an exhaustive listing of all possible **DL** constructors, we note several together with brief descriptions, as a reference for use in the remainder of this dissertation.

There are special constructors that refer to the most general concept in any domain, the *top* concept, and the most specific concept in any domain, the *bottom* concept (Baader and Nutt, 2007):

- *Universal (or Top) concept*:  $\top$ , the extension of which includes all domain elements in any interpretation.

---

<sup>2</sup>Hence the plural in the name ‘Description Logics’

- *Bottom concept*:  $\perp$ , the extension of which never includes any domain member in any interpretation (i.e. it is equivalent to the set theoretic  $\emptyset$ ).

Other concept constructors include (Baader and Nutt, 2007):

- *Disjunction (or)*:  $C \sqcup D$ , equivalent to set theoretic *union*.
- *Conjunction (and)*:  $C \sqcap D$ , equivalent to set theoretic *intersection*.
- *Complement (not)*:  $\neg C$ , equivalent to set theoretic *negation*.
- *Value restriction*:  $\forall R.C$ , which specifies that the filler of the role  $R$ , where it exists, must be of type  $C$ .
- *Existential restriction (limited)*:  $\exists R.\top$ , which specifies that at least one relation of type  $R$  must be specified, without restricting the type of the filler of the relation.
- *Existential restriction (fully qualified)*:  $\exists R.C$ , which specifies that at least one relation of type  $R$  must be specified with some member of type  $C$ .
- *Number restriction (at-most)*:  $\leq nR$ , which specifies that the *number* of fillers for role  $R$  should be less than or equal to  $n$ .
- *Number restriction (at-least)*:  $\geq nR$ , which specifies that the *number* of fillers for role  $R$  should be more than or equal to  $n$ .

Concept constructors commonly found only in very expressive DLs include (Calvanese and De Giacomini, 2007):

- *Qualified number restrictions*:  $\leq nR.C$  or  $\geq nR.C$ , allow the expression of arbitrary cardinality constraints on roles with role fillers belonging to the specified concept  $C$ .
- *Complex role constructors*: including role inclusion  $R \sqsubseteq R'$  and inverse roles  $R^-$
- *Role-value maps*: including containment role-value maps  $R_1 \subseteq R_2$

#### 4.4.2 The TBox

Concept constructors are used to provide definitions for concepts in a domain. These concept definitions constitute the *terminology* of the domain. The terminology is defined

---

in the *Terminology box* (*TBox*), one of two components comprising a *DL* knowledge base. (The second component is the *ABox*, which is discussed in the next section.)

*Atomic concepts* (denoted here by letters *A* and *B* – those which have no definition, the most basic terminology of the domain – are part of the vocabulary of the *TBox*, as are the *atomic roles*. Complex concept definitions, which build on the atomic concepts and roles, are of the form  $C \equiv D$ , where *D* is a complex concept expression formed from one of the available constructors for the *DL* language and the available concepts (both defined and atomic) and roles<sup>3</sup>.

Concept definitions provide *necessary and sufficient* conditions on membership in the extension of the concept, a form of definition which is much stronger than the definitions available in many other forms of knowledge representation, which typically include only *necessary* conditions (Nardi and Brachman, 2007). However, should concept definitions in terms of necessary and sufficient conditions not be possible, it is still possible to capture necessary conditions in a *TBox* by means of *concept inclusion* statements of the form  $C \sqsubseteq D$ . These can be used to capture hierarchical relationships between concepts in a domain, when full definitions cannot be provided.

The *TBox* thus consists of statements of the form

- concept definition  $C \equiv D$ , and
- concept inclusion  $C \sqsubseteq D$ .

For many reasoning tasks making use of a *DL* knowledge base, it is important that the *TBox* has the property of being *acyclic*, that is, that concepts are neither defined in terms of themselves, nor in terms of other concepts that indirectly refer to them (Nardi and Brachman, 2007). Each concept is furthermore assumed to be defined only once. This is always true in *DLs* that allow disjunction as a concept constructor, since multiple concept definitions can thereby be combined into one, but may need to be checked in *DLs* that do not. When combined, these restrictions ensure that every concept expression in the *TBox* can be expanded into a complex expression involving only atomic concepts, by recursively replacing defined concepts with their definitions.

The *TBox* provides the facility to represent knowledge about the entities and properties that are *general* in the domain being modelled. However, it does not provide the facility to make any assertions about *individuals*, that is, the specific members of the domain of interest.

---

<sup>3</sup>We do not here consider *DLs* which allow complex role constructors, although these would also form part of the *TBox*



## 4.5 Knowledge about individuals

Knowledge about the actual members of the domain of interest – the extension of the concepts defined in the **TBox** – is left entirely implicit in most references to concept definitions in **DL**, assisted by the compact, variable-free syntax which is in standard use. However, it is sometimes necessary to make statements about the world – the extension – within the knowledge base. Such extensional knowledge consists of statements about *individuals*, that is, specific members of the domain, and their membership in concepts and roles. For example, an assertion might state that a particular individual FLUFFY is a member of the extension of the concept *Feline*:

*Feline* (FLUFFY)

Another assertion might state that the individual FLUFFY has a child named PATCHES:

*has\_child*(FLUFFY, PATCHES)

Individuals can be *named*, and the **DL** system may or may not enforce a *unique name* assumption which ensures that no two individuals may have the same name without in fact being the same individual.

Assertions about individuals are expressed in the Assertion box (**ABox**), which is the second component of a **DL** knowledge base.

Note that the known individuals in the **DL** knowledge base are not assumed to constitute the full extent of possible individuals in the relevant domain. In contrast to many relational database systems, in **DL** knowledge bases something which is not explicitly expressed is assumed to be *not known*, rather than *known to not be*. This will be of great importance when considering the kinds of inferences which can be made from the assertions which are provided, as will be discussed further in the section on reasoning, Section 4.7.

### 4.5.1 Defining a **DL** knowledge base

We are now ready to give a formal definition for a **DL** knowledge base:

**Definition 4.3.** A knowledge base in a **DL** language consists of a set of atomic concepts  $A_i$ , a set of atomic roles  $R_i$ , a **TBox** and an **ABox**. The **TBox** contains the definitions of the concepts included in the knowledge base, defined through terminological axioms which take the form of logical equivalences –  $C \equiv D$  – or concept inclusions (statements of subsumption) –  $C \sqsubseteq D$  – where the concepts themselves are built up from the constructors available in that **DL** language. The **ABox** contains assertions about the

individuals (instances) that belong to those classes. That is, the **ABox** contains a finite set of assertions of the form  $C(a)$  (concept assertion) or  $R(a, b)$  (role assertion), where  $C$  is a concept definition,  $R$  a role name, and  $a$  and  $b$  are individual names (Baader and Nutt, 2007).

## 4.6 Semantics

Recall that one of the motivations for the initiation of research into **DLs** was the need to provide a more formal *semantics* to what had been popular but informally characterised network-based knowledge representation structures.

The semantics of a **DL** language is given by assigning an interpretation,  $I$ , which consists of a non-empty set  $\Delta^I$  (the domain of the interpretation), and an interpretation function which assigns to every atomic concept  $A$  a set  $A^I \subseteq \Delta^I$  and to every atomic role  $R$  a binary relation  $R^I \subseteq \Delta^I \times \Delta^I$  (Baader and Nutt, 2007).

This interpretation function is extended to concept descriptions by inductive definitions for each concept constructor, for example:

$$\begin{aligned} \top^I &= \Delta^I \\ \perp^I &= \emptyset \\ (\neg A)^I &= \Delta^I \setminus A^I \\ (C \sqcap D)^I &= C^I \cap D^I \\ (\forall R.C)^I &= \{ a \in \Delta^I \mid \forall b. (a, b) \in R^I \longrightarrow b \in C^I \} \\ (\exists R.\top)^I &= \{ a \in \Delta^I \mid \exists b. (a, b) \in R^I \} \\ (\exists R.C)^I &= \{ a \in \Delta^I \mid \exists b. (a, b) \in R^I \wedge b \in C^I \} \\ (\geq nR)^I &= \{ a \in \Delta^I : | \{ b \mid (a, b) \in R^I \} | \geq n \} \\ (\leq nR)^I &= \{ a \in \Delta^I : | \{ b \mid (a, b) \in R^I \} | \leq n \} \end{aligned}$$

An interpretation  $I$  *satisfies* an inclusion  $C \sqsubseteq D$  if  $C^I = D^I$  and an equality  $C \equiv D$  if  $C^I \subseteq D^I$ . An interpretation  $I$  is a *model* of an axiom or set of axioms if it satisfies these axioms. Two axioms (or sets of axioms) are *equivalent* if they share the same models (Baader and Nutt, 2007).

## 4.7 Reasoning

### 4.7.1 Terminological inferences

The main purpose of reasoning within a knowledge representation formalism is to derive *inferences* from the explicitly represented knowledge. For the knowledge expressed in the **TBox**, i.e. the terminological knowledge, the basic inference operation is *classification*, that is, placing concept expressions in the proper place with respect to the taxonomic hierarchy of concepts. This amounts to inferring *subsumption* ( $C \sqsubseteq D$ ) between concept expressions  $C$  and  $D$  (Baader and Nutt, 2007). Classification is accomplished by checking subsumption of a new concept against all the other concepts in the knowledge base and placing the new concept in between the most specific concept that it subsumes and the most general concepts that subsume it.

Further inferences are that two concept expressions might be found to be *equivalent*, written as  $C \equiv D$ , which means that  $C \sqsubseteq D$  and  $D \sqsubseteq C$ ; or they may be found to be *disjoint*, that is,  $C \sqcap D = \perp$ .

Another task that is required as the size of a knowledge base increases is the ability to automatically check for the *satisfiability* of each of the concept expressions, i.e. whether, for each concept in the knowledge base, there is an interpretation in which the concept has a non-empty extension. This assists the engineer in the maintenance and extension of the knowledge base. Satisfiability depends on there being no logical contradictions in the definitions of any of the concept expressions in the knowledge base.

Formally, we can define these properties as follows (Baader and Nutt, 2007):

**Definition 4.4.** A concept  $C$  is *satisfiable* with respect to a **TBox**  $T$  if there exists a model  $I$  of  $T$  such that  $C^I$  is nonempty. We say that  $I$  is a *model* of  $C$ .

**Definition 4.5.** A concept  $C$  is *subsumed* by a concept  $D$  with respect to a **TBox**  $T$  if  $C^I \subseteq D^I$  for every model  $I$  of  $T$ . This is written as  $C \sqsubseteq_T D$ .

**Definition 4.6.** Two concepts  $C$  and  $D$  are *equivalent* with respect to a **TBox**  $T$  if  $C^I = D^I$  for every model  $I$  of  $T$ . This is written as  $C \equiv_T D$ .

**Definition 4.7.** Two concepts  $C$  and  $D$  are *disjoint* with respect to a **TBox**  $T$  if  $C^I \cap D^I = \emptyset$  for every model  $I$  of  $T$ .

However, by the following observations, it turns out that all of the terminological inferences can be reduced to the problem of calculating subsumption (Baader and Nutt,

2007):

$C$  is unsatisfiable  $\Leftrightarrow C \sqsubseteq \perp$

$C$  and  $D$  are equivalent  $\Leftrightarrow C \sqsubseteq D$  and  $D \sqsubseteq C$

$C$  and  $D$  are disjoint  $\Leftrightarrow C \sqcap D \sqsubseteq \perp$

### 4.7.2 Assertional inferences and query answering

For **ABoxes**, that is, the extensional knowledge about individuals in the domain, reasoning provides the services of checking *consistency* of the assertional knowledge, that is, that it contains no logical contradictions. Formally, it can be defined as follows (Baader and Nutt, 2007):

**Definition 4.8.** An **ABox**  $A$  is *consistent* with respect to a **TBox**  $T$  if there is an interpretation  $I$  that is a model of both  $A$  and  $T$ . We simply say that  $A$  is consistent if it is consistent with respect to the empty **TBox**.

It is also useful for the system to be able to determine membership in the extension of concepts from the **TBox** for each individual in the **ABox**. This is known as *instance checking*, and provides the mechanism by which *query answering* is provided for **DL** knowledge bases. Instance checking amounts to the test whether an assertion of concept membership is entailed by an **ABox**. Formally, it can be defined as follows (Baader and Nutt, 2007):

**Definition 4.9.** An assertion  $\alpha$  is *entailed* by an **ABox**  $A$ , written  $A \models \alpha$  if every interpretation that satisfies  $A$  also satisfies  $\alpha$ .

Instance checking can be reduced to consistency checking by the following observation:

$A \models C(a) \Leftrightarrow A \cup \{\neg C(a)\}$  is inconsistent.

For query answering in support of applications, information *retrieval*, which is the task of finding for a given **ABox**  $A$ , all individuals  $a$  such that  $A \models C(a)$  for some query concept  $C$ , is needed.

### 4.7.3 Open world semantics

Relational databases also incorporate data conforming to some specification, in much the way that **DL ABoxes** allow the specification of individuals conforming to the conceptual specification of the **TBox**. However, an important distinction should be drawn between the semantics of **DL** knowledge bases and that of relational databases.

The distinction is that a database instance represents *only one* interpretation: exactly that interpretation which includes the individuals in the database data. However, an **ABox** represents *many different* interpretations, that is, all of its possible models. In a database instance, absence of information implies negative information – if something is not stated, it is not so. This is referred to as *closed world* semantics. In an **ABox**, absence of information only implies lack of knowledge, and negative assertions cannot be derived from the absence of positive assertions. This is referred to as *open world* semantics.

Open world semantics has important consequences for reasoning tasks including query answering. Since an **ABox** represents up to infinitely many interpretations, query answering requires nontrivial reasoning (Baader and Nutt, 2007), rendering it more algorithmically more complex in **DL** knowledge bases than in databases.

#### 4.7.4 Decidability and complexity

Since **DLs** are used to represent knowledge, it is important for application purposes that the inferencing mechanisms available in the languages should be *decision procedures*, that is, they should always terminate with an answer (yes or no) in every possible scenario (Baader and Nutt, 2007). This means that the standard reasoning tasks should be *decidable* in the underlying **DL** language. Investigating the decidability of reasoning in different **DL** languages has therefore been a core research area (Baader and Nutt, 2007).

However, even when reasoning problems in a particular **DL** language are shown to be decidable, it is not known how *much* time will be required to reason over knowledge bases to compute inferences. Decidability guarantees an answer in *finite* time, but makes no further statement about the amount of time and how it varies in the size of the knowledge base. To determine the amount of time that will be required to compute inferences in different **DL** languages, research into the *complexity* profile of the reasoning problems in that language is required. Complexity varies with the expressivity of the language – the less expressive the language (the fewer and simpler the constructors available in that language), the less complex the reasoning is in that language, but of course, the less elegant it is to model knowledge in real application areas.

An important property influencing the decidability and complexity of a **DL** language is whether it enjoys a form of the *tree model* property. That is, whether all knowledge bases modelled in the language have a *finite tree model* (Baader and Nutt, 2007), which means that they have an interpretation in the shape of a tree with a finite branching factor. The extent of the tree might be infinite, but the branching factor is finite. More formally:

**Definition 4.10.** A knowledge base in a **DL** language exhibits the *finite tree model*

property if for every concept  $C_0$  in the knowledge base, there exists a *finite* interpretation  $I$  that has the shape of a tree whose root belongs to  $C_0$ , whose depth is linearly bounded by the size of  $C_0$  and whose branching factor is bounded by the sum of the numbers occurring in at-least (minimum cardinality) restrictions in  $C_0$  plus the number of different existential restrictions in  $C_0$ .

## 4.8 Basic DL languages

Recall that DL languages consist of a set of atomic concepts (denoted by letters  $A$  and  $B$ ), a set of atomic roles (denoted by the letter  $R$ ), an allowed set of *concept constructors* and an allowed set of *role constructors*. Arbitrary concept descriptions are denoted by the letters  $C$  and  $D$ . In this section we provide the names and allowed constructors for some basic DL languages.

The DL language  $\mathcal{AL}$  (attributive language) includes the following constructors:

$$\begin{aligned}
 C, D \longrightarrow & A \mid \text{(atomic concept)} \\
 & \top \mid \text{(universal concept)} \\
 & \perp \mid \text{(bottom concept)} \\
 & \neg A \mid \text{(atomic negation)} \\
 & C \sqcap D \mid \text{(intersection)} \\
 & \forall R.C \mid \text{(value restriction)} \\
 & \exists R.\top \mid \text{(limited existential quantification)}
 \end{aligned}$$

The family of  $\mathcal{AL}$  languages can be extended with various constructors including

$$\begin{aligned}
 C, D \longrightarrow & \exists R.C \mid \text{(full existential quantification, denoted by } \mathcal{E} \text{ in the language name)} \\
 & C \sqcup D \mid \text{(union, denoted by } \mathcal{U} \text{ in the language name)} \\
 & \leq nR \text{ and } \geq nR \mid \text{(number restrictions, denoted by } \mathcal{N} \text{ in the language name)} \\
 & \neg C \mid \text{(negation of arbitrary concepts, denoted by } \mathcal{C} \text{ in the language name)}
 \end{aligned}$$

The DL languages  $\mathcal{SHIQ}$  and  $\mathcal{SHOIQN}$  are based on extending  $\mathcal{ALC}$  (that is,  $\mathcal{AL}$  with negation of arbitrary concepts) to include transitively closed primitive roles, which results in a logic called  $\mathcal{S}$  (Horrocks *et al.*, 2007), which is then further extended with features such as role inclusion (denoted  $\mathcal{H}$ ), nominals ( $\mathcal{O}$ ), inverse roles ( $\mathcal{I}$ ) and number

restrictions ( $\mathcal{N}$ ) which might be qualified ( $\mathcal{Q}$ ).

In the next chapter we will look in greater detail at one of the extensions to description logics which allows the specification of dynamic knowledge in knowledge bases, namely, rules.

# 5

## Rules

In this chapter, we describe rules as a form of knowledge representation and an extension to standard DLs, giving them semantics in terms of an epistemic operator in Section 5.1, and discussing some considerations surrounding reasoning with rules in Section 5.2.

DLs allow the representation of static terminological knowledge in the TBox and assertional knowledge about the world in the ABox. However, to represent knowledge which is *dynamic*, and which changes based on the status of knowledge about the world expressed in the current ABox, it is necessary to make use of an extension to ‘pure’ DL, in the form of *rules*.

Rules encode knowledge in the form if-then conditionalities,  $C \Rightarrow D$  (Baader and Nutt, 2007). Procedural rules, or *trigger rules*, model dynamic knowledge. The meaning of such a rule is that, if the condition specified in the antecedent is met, the consequent is asserted into the knowledge base. In this way, a rules-based system is constantly evolving as the program operates.

Dynamic knowledge is knowledge about a situation that changes over time. Depending on the purpose of a knowledge base, one approach to dealing with dynamic knowledge is to ‘freeze’ the situation at a specific point in time and model that point, effectively excluding the time dimension from the model. The tradeoff is that the model has to be manually updated every time something changes. This approach is the one followed in most DL knowledge bases. However, in some application scenarios, change over time may be integral to the purpose of the application, and therefore it may require a knowledge representation formalism that is able to represent change over time. Rules provide such a medium, and are often combined with DLs to achieve hybrid representational models.

A simple example of a dynamic property can be found in the domain of persons.



Consider a knowledge base about the persons currently working for an organisation. If the property *age* is included in the knowledge base, it would be tiresome for the ontology maintainer to keep updating the values for each person every time a person had a birthday. It would be much better to include a *rule* in the knowledge base of the general form *if* today's date matches the birth date of a person, *then* add one to the age of the person.

To take a slightly more complex example, imagine that the knowledge base is also extended with information about the salary of the employees of the organisation. The administrators, for various complicated tax reporting reasons, require a concept in the knowledge base for people considered HighEarnings based on whether their salary is in the top 10% of the company's salary range. This depends on all the *known individuals* in the knowledge base, and their salary values, rather than being a property of a general concept which can be expressed in the **TBox** of the knowledge base. This knowledge base could be modelled by including a *trigger rule* which, every time a new individual was added to the knowledge base, recalculated the salary range and classified all individuals as high earners or not based on the outcome.

Another kind of rule which can be used to extend a **DL** knowledge base is a *default rule*, which enable default reasoning in hierarchies. A default rule specifies default properties of a concept which can be overridden by more specific properties in a child concept in the hierarchies. For example, a default rule might be created to associate the property of being able to fly with individuals known to be birds, *unless* the specific type of bird that the individual is, is known to not be able to fly (e.g. penguins).

## 5.1 Semantics

Rules, in the most general sense, consist of conditionals of the form  $C \Rightarrow D$ .  $C$  is called the antecedent, and  $D$  the consequent. If the conditions expressed in the antecedent ( $C$ ) are known to be met in the knowledge base, then the consequent  $D$  is asserted. All rules are expressed and operate over *known individuals* in the knowledge base.

In order to provide a formal semantics for rules within a **DL** knowledge base, it is necessary to introduce an *epistemic* operator, that is, an operator which refers not to objects in the domain, but to what is *known* by the system about objects in the domain. A procedural rule is triggered only when the antecedent is met, that is, the assertional knowledge represented in the antecedent of the rule is *known* to the knowledge base (Baader and Nutt, 2007).

The epistemic operator is standardly denoted **K**. We can make use of it to construct

epistemic concepts:

$$C, D \longrightarrow \mathbf{KC}(\text{epistemic concept}) \quad (5.1)$$

The concept  $\mathbf{KC}$  denotes those objects for which the knowledge base knows that they are instances of  $C$ .

Now we can translate rules of the form  $C \Rightarrow D$  into inclusion axioms of the form  $\mathbf{KC} \sqsubseteq D$ . The  $\mathbf{K}$  operator has the effect that this axiom is only applicable to individuals for which the  $\mathbf{TBBox}$  and  $\mathbf{ABox}$  imply (i.e. the system knows) it is true that they are members of the concept  $C$ . We can now define a rule-extended knowledge base (Baader and Nutt, 2007):

**Definition 5.1.** A *rule knowledge base* is a triple  $K = (T, A, R)$  where  $T$  is a  $\mathbf{TBBox}$ ,  $A$  is an  $\mathbf{ABox}$ , and  $R$  is a set of rules written as inclusion axioms of the form 5.1.

Let us consider an example in a simple knowledge base expressing facts about an individual person named SUSAN,  $T = \exists \text{friend.Male}(\text{SUSAN})$ . If we query this knowledge base for persons who have male friends ( $\exists \text{friend.Male}$ ) we expect the answer SUSAN, since our knowledge base expresses precisely the fact that there is an individual SUSAN who has a male friend. On the other hand, if we query the database using the epistemic operator,  $\exists \text{friend.KMale}$ , the answer set is empty since there is no friend of SUSAN in the knowledge base who is *known* to be male. (Baader et al., 2007).

## 5.2 Reasoning

We define reasoning with rules procedurally in terms of the operation of the reasoner on a knowledge base. During reasoning, rules are applied to the knowledge base following a forward reasoning process, starting with the initial knowledge base  $K$ , a series of knowledge bases  $K^{(0)}, K^{(1)}, \dots$  are constructed, where  $K^{(i+1)}$  is obtained from  $K^{(i)}$  by adding a new assertion  $D(a)$  whenever  $R$  contains a rule  $C \Rightarrow D$  such that  $K^{(i)} \models C(a)$  holds (Baader and Nutt, 2007).

**Definition 5.2.** The *procedural extension* of a rule knowledge base is the knowledge base  $\bar{K} = (T, \bar{A})$  that is obtained from  $(T, A)$  by applying the rules as described.

This process eventually terminates since  $K$  contains only finitely many individuals and there are only finitely many rules in  $R$ . An important restriction is that the rule is not able to add new individuals to the knowledge base. It is easy to show that rules which

are able to add new individuals to the knowledge base can quickly lead to undecidability. Such a rule would only have to be defined recursively. For example, in a knowledge base of persons, a rule could be created which, when a new person is added to the knowledge base, *adds another* person in the consequent of the rule. In this fashion, the rule would never terminate as it would keep triggering itself.

In this chapter we have presented rules as a formalism for representing dynamic and default knowledge, both of which are not possible to model within the standard [DL](#) formalism. However, in this dissertation, we will make use of rules to express conditionality that is neither dynamic nor default; rather we will use the expressivity of rules as an additional tool to model static, but structured, conditionality. This is described further in [Chapter 9](#). In the next chapter we describe the popular ontology language for the semantic web, OWL, and in [Section 6.6](#) we describe the integration of a rule language with this ontology language.

# 6

## OWL - the Web Ontology Language

In this chapter, we present the syntax, semantics, and profiles of the Web Ontology Language, [OWL](#). We show how [OWL](#) is founded on description logics and how rules can also be included within [OWL](#).

### 6.1 Background

The Web Ontology Language, [OWL](#), is the W3C standardised format for encoding ontologies ([Smith et al., 2010](#)). [OWL](#) developed out of efforts to reconcile [DL](#) systems with developments in the semantic web ([Horrocks et al., 2007](#)). More and more data was being brought onto the semantic web in the form of the Resource Description Framework ([RDF](#)) which describes resources and the relationships between resources in terms of triples (with a triple representing an arbitrary connection in a graph) and is encoded in a web-friendly [XML](#) format. Ontology developers and researchers were finding that the use of [DL](#) facilitates powerful reasoning which aided and assisted the ontology development process. Early attempts to combine description logics with the semantic web were the OIL (Ontology Inference Language) and DAML (DARPA Agent Markup Language) projects. From these early independent projects, and a tight integration into the semantic web language [RDF](#), a W3C working group developed what is now known as [OWL](#), a unified ontology language for the semantic web ([Horrocks et al., 2007](#)).

#### 6.1.1 A note about terminology

In previous chapters, we have used the [DL](#) terminology *concept* and *role*. However, for historical reasons [OWL](#) uses *class* instead of *concept* and *object property* instead of *role*. In the remainder of this chapter and the remainder of the dissertation, we will use

the OWL terminology. However, it should be noted that, since OWL is founded on the semantics of DLs, a class in OWL is equivalent to a concept – either atomic or complex – in the underlying DL. OWL also admits individuals, therefore it corresponds to a full DL knowledge base.

Additional syntax and terminology discrepancies between OWL and DLs are listed in Table 6.1.

Construct	DL Syntax	OWL Syntax
<i>concept equivalence</i>	$C \equiv D$	<code>EquivalentClasses(<math>C, D</math>)</code>
<i>subsumption</i>	$C \sqsubseteq D$	<code>SubClassOf(<math>C, D</math>)</code>
<i>existential quantification</i>	$\exists R.C$	<code>C SubClassOf someValuesFrom <math>R</math></code>
<i>value restriction</i>	$\forall R.C$	<code>C SubClassOf allValuesFrom <math>R</math></code>

Table 6.1 Syntax OWL and DL

## 6.2 Expressivity and OWL flavours

OWL comes in three different “flavours” (or “species”), each corresponding to different levels of expressivity and computational tractability, and also different degrees of compatibility with RDF. These different flavours allow different rules specifying how concepts may be constructed. They also permit different DL constructors to be used. The different flavours of OWL are OWL-Lite, OWL-DL, and OWL-Full. OWL-Full allows the full expressivity of arbitrary RDF graphs to be encoded in an OWL ontology, but does not correspond to a DL language, and thus is not necessarily decidable, and will not be considered further here.

OWL-Lite is the least expressive of the different flavours of OWL, containing fewer constructors than OWL-DL (Horrocks *et al.*, 2007). OWL-DL is based on the DL language  $\mathcal{SHOIN}(\mathbf{D})$  and as such contains the following constructors, where the letters  $A$ ,  $R$  and  $o$  and represent names for classes, object properties, and individuals respectively.  $C$  represents an arbitrary class description.

- $A$  – atomic concepts
- $\top$  (owl:Thing) – top concept
- $\perp$  (owl:Nothing) – bottom concept
- $C_1 \sqcap \dots \sqcap C_n$  – intersection (AND)
- $C_1 \sqcup \dots \sqcup C_n$  – union (OR)
- $\neg C$  – negation (complement)
- $\{o_1\} \sqcup \dots \sqcup \{o_n\}$  – one of

$\exists R.C$  – some values from

$\forall R.C$  – all values from

$R.o$  – has value

$\geq nR$  – minimum cardinality

$\leq nR$  – maximum cardinality

Note that **OWL**-DL contains additional constructors relating to datatypes, data ranges, and individuals, which are not listed above.

## 6.3 Notes on semantics

Like **DLs**, **OWL** is assigned an *open world* semantics, which means that inferences can only be drawn based on what is explicitly captured in the knowledge base; if it is stated that *Square* and *Circle* are the only subclasses of *Shape* in a particular knowledge base, and we know that some concept *A* *has\_shape* *C*, we nevertheless cannot infer that *C* is either a *Square* or a *Circle*, as we do not know anything about which other shapes may exist that have not been specified (i.e., in the open world). However, we can achieve this by asserting a value restriction -  $\forall \text{has\_shape}.(Square \sqcup Circle)$ . This is called a *closure axiom*.

**OWL** also does not subscribe to the *unique name assumption* (Smith *et al.*, 2010), which means that different names may refer to the same entity. To clarify the semantics around same and different entities, **OWL** includes the `owl:sameAs` property to state that two given named individuals have the same identity, and the `owl:differentFrom` property to state that two given named individuals have different identities.

## 6.4 OWL Syntaxes

Several syntaxes for serialization have been developed for **OWL**, including the RDF/XML serialization (Smith *et al.*, 2010) which is directly backwards compatible with **RDF**, as well as the Manchester syntax, which is more concise and allows for greater human readability<sup>1</sup>.

Examples of ontologies expressed in RDF/XML and in Manchester syntax are reproduced in Appendix A.

---

<sup>1</sup>[http://www.co-ode.org/resources/reference/manchester\\_syntax/](http://www.co-ode.org/resources/reference/manchester_syntax/)

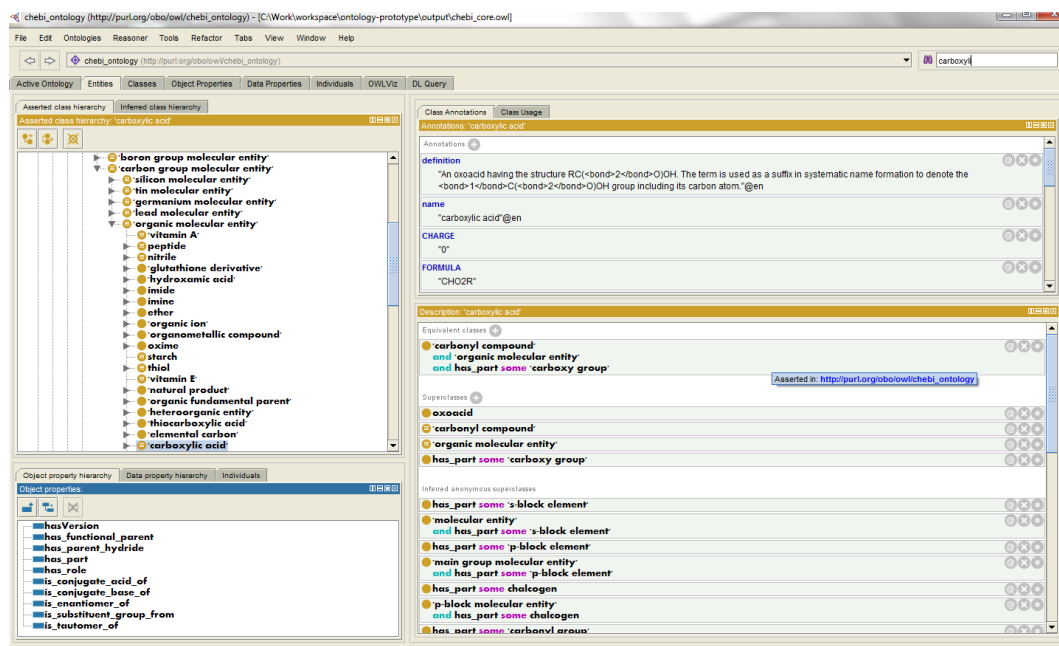


Figure 6.1 Protégé OWL ontology editing environment

## 6.5 Tools for working with OWL ontologies

The primary editor that is used to maintain OWL is Protégé-OWL<sup>2</sup>. Protégé provides an extensive framework for ontology development. It facilitates concept editing, property editing, individual editing, reasoner integration and ontology visualisation. Integral to the Protégé application is the possibility for developing plug-ins, or custom extensions, to suit user specific needs. Figure 6.1 shows a screenshot of an ontology (ChEBI) being edited in Protégé.

Functionality for working with and maintaining OWL ontologies programmatically is provided by the OWL API, a Java library which provides an extensive toolkit for developing and maintaining ontologies, including programmatically invoking reasoning (Horridge and Bechhofer, 2009). The OWL API underlies the Protégé-OWL application since Protégé version 4.

Many different *reasoners* have been developed which work with OWL ontologies, for example Fact++ (Tsarkov and Horrocks, 2006), Racer (Haarslev and Möller, 2003), Pellet (Sirin *et al.*, 2007) and HermiT (Shearer *et al.*, 2008), each of which supports different ranges of DL operators and may have different performance profiles on different reasoning tasks.

<sup>2</sup><http://protege.stanford.edu/overview/protege-owl.html>

## 6.6 Semantic Web Rule Language

OWL is extended with a *rule language* in order to express conditional if-then (epistemic) knowledge in the form of rules. As a result, many of the publicly available OWL reasoners are able to perform epistemic reasoning with rules.

The Semantic Web Rule Language (SWRL) is a rule language for the semantic web. A description of an RDF based syntax and semantics for rules is given for the SWRL in Horrocks *et al.* (2010). This is further extended with a syntax for rules included directly within OWL ontologies in Glimm *et al.* (2009).

The semantics of a collection of rules is given by interpreting the rules as first-order logical implications (Horrocks *et al.*, 2010). Rules are made up of atoms, that is, atomic assertions. Atoms can be of the form  $C(x)$ ,  $P(x,y)$ , `sameAs(x,y)` or `differentFrom(x,y)`, where  $C$  is an OWL concept expression,  $P$  is an OWL property, and  $x,y$  are either variables, OWL individuals or OWL data values.

An empty antecedent (body of the rule) is treated as trivially true (i.e. satisfied by every interpretation), so the consequent (conclusion of the rule) must also be satisfied by every interpretation. An empty consequent is treated as trivially false (i.e., not satisfied by any interpretation), so the antecedent must also not be satisfied by any interpretation. Multiple atoms in a rule are treated as a conjunction (and) (Horrocks *et al.*, 2010).

OWL ontologies extended with arbitrary rules are rendered undecidable, because the rule extension may be infinite. To remedy this, a restricted expressivity can be employed in the form of *DL-safe* rules (Motik *et al.*, 2004). The key restriction which renders a rule DL safe is that no new existentially quantified variables can be introduced in the consequent of the rule. In OWL ontologies, this is achieved by forcing all variables appearing in the consequent of the rules to be known, named individuals in the OWL knowledge base.

In the next chapter, we start to examine structured objects, and assess the capacity of OWL to model structured objects.





# Representing chemical structures in OWL

In this chapter, we describe several possible approaches to representing chemical structures using OWL and rules, and discuss the shortcomings of each. This explains the research problem that provides the context for this dissertation, namely, the fact that neither OWL alone, nor OWL enhanced with rules, provide adequate support for modelling the internal structure of structured objects at the class level.

## 7.1 Structured Objects

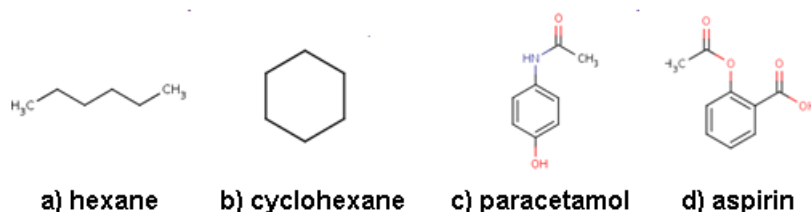
Objects can be said to be *structured* when they consist of parts connected in complex ways. A straight line, therefore, is not a structured object, since even if we consider it a set of connected parts such as smaller lines or points, they all are connected in a very straightforward way. If a line is divided, it nevertheless retains the property of ‘lineness’: it will divide into multiple lines, so doesn’t retain its unity as one thing, but each of the divided lines is itself a line. By contrast, a triangle is a structured object, since it is important that there are three sides and that each side is connected to two other sides at the edges. Without these structural properties of connection, that is if the triangle is divided, it ceases to be a triangle, although it might become a zigzag, or three separate lines.

Although most objects in our 3D world are structured in this sense, for the purpose of creating knowledge based systems, the structure of the object may or may not be important to model within the knowledge base. For example, we might describe the parts of a cell, as is done in the Gene Ontology ‘cellular component’ ontology, without ever referring to the structure of exactly how the parts are connected to each other within the cell. In anatomy, on the other hand, knowledge bases such as the Foundational Model

of Anatomy (Rosse and Jr, 2007) do concern themselves with attempts to model the structure of parts with respect to each other in their knowledge artefact. The reason for this is that, in the domain of anatomy, expert systems need to be able to draw inferences based on the structure of objects - a misplaced or disconnected part may be a sign that something is wrong with the organism (e.g. a missing kidney in an individual's anatomy).

Chemistry is another domain where the structure of the objects is crucial in order to derive correct inferences. A cyclohexane molecule, which consists of six carbon atoms bonded in a cycle with six single bonds, has very different properties to a hexane molecule, which consists of a straight chain of six carbon atoms. A hexane molecule, in turn, has very different properties to an isohexane molecule, which is branched. Figure 7.1 shows some examples of differently structured objects in chemistry.

While OWL provides an extensive collection of constructs for logic-based ontology development, decidability of reasoning problems—e.g., testing consistency of an ontology, satisfiability of classes or computing its inferred class hierarchy—is obtained by making sure that OWL has a *tree model property* (Vardi, 1996): in a nutshell, that means that every consistent ontology has a model, i.e., a state of affairs that satisfies all axioms in the ontology, whose relational structure looks like a tree. For this reason, OWL has not traditionally been able to describe arbitrarily structured objects, but only those which have structures which can be expressed in the shape of trees.



**Figure 7.1** Structured objects in chemistry – hexane, cyclohexane, paracetamol, aspirin

In the sections that follow, we evaluate several different approaches to representing the internal structure of chemical entities in OWL knowledge bases.

## 7.2 Representation in TBox

OWL can be used to describe objects consisting of an arbitrary, or infinite, number of parts, but it only allows for axioms connecting the parts in a *tree-like* manner. This is because OWL requires the tree model property (Vardi, 1996) in order to preserve the

decidability of reasoning (Motik *et al.*, 2008c).

The structured object in Figure 7.1 (b), cyclohexane, could be modelled in OWL at the class level (TBox) in the following manner:

$$\begin{aligned} \text{Cyclohexane}(M) \sqsubseteq & \exists.\text{has\_atom}.C1 \sqcap \exists.\text{has\_atom}.C2 \sqcap \exists.\text{has\_atom}.C3 \\ & \sqcap \exists.\text{has\_atom}.C4 \sqcap \exists.\text{has\_atom}.C5 \sqcap \exists.\text{has\_atom}.C6 \quad (7.1) \end{aligned}$$

$$\text{Cyclohexane}(M) \sqsubseteq \forall.\text{has\_atom}.(C1 \sqcup C2 \sqcup C3 \sqcup C4 \sqcup C5 \sqcup C6) \quad (7.2)$$

$$\text{Cyclohexane}(M) \sqsubseteq \forall.\text{has\_atom}.( \forall \text{has\_type}.Carbon ) \quad (7.3)$$

$$C1 \sqsubseteq \exists \text{has\_bond\_with}.C2 \sqcap \forall \text{has\_bond\_with}.C2 \quad (7.4)$$

$$C2 \sqsubseteq \exists \text{has\_bond\_with}.C3 \sqcap \forall \text{has\_bond\_with}.C3 \quad (7.5)$$

$$C3 \sqsubseteq \exists \text{has\_bond\_with}.C4 \sqcap \forall \text{has\_bond\_with}.C4 \quad (7.6)$$

$$C4 \sqsubseteq \exists \text{has\_bond\_with}.C5 \sqcap \forall \text{has\_bond\_with}.C5 \quad (7.7)$$

$$C5 \sqsubseteq \exists \text{has\_bond\_with}.C6 \sqcap \forall \text{has\_bond\_with}.C6 \quad (7.8)$$

$$C6 \sqsubseteq \exists \text{has\_bond\_with}.C1 \sqcap \forall \text{has\_bond\_with}.C1 \quad (7.9)$$

Equation 7.1 expresses the condition that a molecule  $M$  belongs to the class *Cyclohexane* if it contains six different atoms expressed with the *has\_atom* role. Equations 7.2 and 7.3 are the closure axioms which prevent this molecule from containing additional atoms, and constrain the type of the atoms to *Carbon* expressed with the *has\_type* role. Equations 7.4 – 7.9 then constrain the atoms to express the connections between the atoms in terms of the *has\_bond\_with* role. The combination of existential and value restriction is intended to faithfully represent the fact that each atom participates in a bond with the neighbouring atom in the cycle, and only that neighbouring atom. Note that the *has\_bond\_with* relation is not symmetric in this model, and this would mean that a faithful representation in this model would also need to include the bonds in the opposite direction, but this does not alter the utility of the example, since the problem with the model appears as soon as there is a cycle even if the edges are directed.

But what is the problem with this representation in OWL? It is that the representation is *underconstrained*, since in addition to a model that corresponds to the structure in which the atoms are connected as expected, there are additional models where the atom *C6* is connected to another individual atom of type *C1* – since *C1* is a class in the OWL TBox, nothing prevents the class from being instantiated by several different individuals. This model would thus include two individuals, both of type *C1*. But the second *C1* individual can have a bond with a second individual of class *C2*, and so on around the

entire cycle and into a round of third individuals, quickly resulting in an *infinite tree structure* for this model, as illustrated in Figure 7.2 (Motik *et al.*, 2008c). (Note that individual carbon atoms form the corners of this illustration, in skeleton graph depiction format as described in Chapter 3.)

Thus, a reasoner cannot make certain types of inferences about this molecule, such as the inference that an individual of this molecule class consists of six and only six individual atoms, and that these atoms are arranged in a closed bonding cycle.



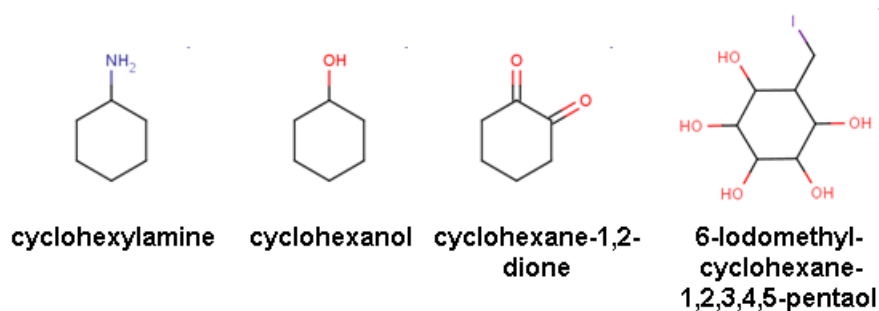
**Figure 7.2** Unintended infinite tree model

## 7.3 Representation in ABox

Figure 7.1 (b) (cyclohexane) could instead be represented in OWL using ABox assertions, in which the atoms could be represented as *individuals*, i.e. six single concrete carbon atoms. In this case the connection represented by the *has\_bond\_with* role assertions would be asserted between concrete atoms in the ABox and would thus not be subject to the misinterpretation that can apply if the cyclic structure is modelled at the class level (in the TBox).

Representation in the ABox, i.e. by modelling the structure of *individuals*, would surely suffice to adequately capture the structure of one concrete cyclohexane molecule. But this again precludes modeling structured data at the class level, that is, it precludes modeling structured aspects of commonality across *many* molecules. For instance, rather than cyclohexane itself, we may wish to model *cyclohexane-like* molecules, which contain a principal cyclohexane skeleton structure, as illustrated in Figure 7.3. To model the internal structure of a class of such molecules requires class-level modelling of the core skeleton structure.

In general in biomedical ontologies, most modelling is concerned with class-level concepts, that is, describing what is common to a set of objects of a certain kind. The



**Figure 7.3** Some molecules which share a cyclohexane skeleton

individual objects which can be found in the real world thus instantiate the class, and a model at the level of individuals does not therefore offer much utility. It may be the case that for molecules, one can waive this requirement due to the indistinguishability of individual molecules belonging to a given class (Lesk, 1980). However, if we model at the level of individuals we would lose the ability to construct models of molecule classes. A *molecule class* represents what is common to a group of related molecules, which may include elements of shared structure, even cyclic elements of shared structure. Nevertheless, individuals belonging to such a molecule class can differ dramatically in non-shared structural aspects, thus are clearly distinguishable from one another.

We can extend our earlier class-level OWL formulation of the atoms and connections within a cyclohexane molecule to a formulation which encodes the core skeleton of any cyclohexane-like molecule with the removal of the closure axiom 7.2; that is, we remove the restriction that the only atoms contained in the molecule are  $C1, \dots, C6$ . However, we still have not addressed the problem of the infinite tree model.

## 7.4 Representation with Rules

Another possibility is to model the structure of the structured objects in the knowledge base using rules, which are more expressive than OWL-DL but at the expense of decidability (Horrocks *et al.*, 2010). For example, the following rule can be used to ensure that

the final two connected carbon atoms from Figure 7.1 (b) are actually the same individual:

$$\begin{aligned}
 & \text{Cyclohexane}(m) \wedge \text{has\_atom}(m, c1) \wedge \text{has\_atom}(m, c2) \wedge \text{has\_atom}(m, c3) \\
 & \quad \wedge \text{has\_atom}(m, c4) \wedge \text{has\_atom}(m, c5) \wedge \text{has\_atom}(m, c6) \\
 & \quad \wedge \text{has\_bond\_with}(c1, c2) \wedge \text{has\_bond\_with}(c2, c3) \wedge \text{has\_bond\_with}(c3, c4) \\
 & \quad \wedge \text{has\_bond\_with}(c4, c5) \wedge \text{has\_bond\_with}(c5, c6) \wedge \text{has\_bond\_with}(c6, c7) \\
 & \quad \longrightarrow c7 = c1 \quad (7.10)
 \end{aligned}$$

If the rule specified in the antecedent is met (the chain of connected bonds between individual atoms in a cyclohexane molecule), the consequent of the rule will ensure that the first and seventh atoms in the connected chain of bonds are the same individual atom. This does seem to capture the required internal structure, but in a very roundabout way.

This approach seems somehow redundant, since the information is now encoded in the class specification and again in the rules, with the intended interpretation only accurate through the interaction of both. Such a modeling formalism is likely to be difficult to use (Motik *et al.*, 2008a). Also, the information thus specified is not available at the class level for purposes of classification – rules cannot encode whether a particular class is a subclass of another class nor provide a logical definition of the class in the ontology.

Bearing in mind that the inclusion of rules of this form for modelling structured objects leads to undecidability of reasoning, Motik *et al.* (2004) introduce *DL-safe rules*, which are restricted to refer only to *known* individuals within the knowledge base. We therefore consider whether a potential work-around for the problem of expressing the above rules and ensuring decidability would be to use *DL-safe rules*. But, in fact, we find that *DL-safe rules* cannot be used in this way, i.e. to model the structure of objects at the class level. This is because the restriction to apply only to known individuals within the knowledge base means that the above formulation cannot be modelled, since it has to apply to all cyclohexane molecules, and cannot be restricted to known individuals only.

Consequently, we have to conclude that rules will also not suffice to allow the internal structure of cyclic objects to be modelled at the class level in *OWL*.

*Description Graphs* are a formalism which has been introduced by Motik *et al.* (Motik *et al.*, 2008c,b,a) to address this weakness of *OWL* in representing structured objects, while still preserving the decidability of reasoning on ontologies containing such structured objects. In the next chapter, we describe the Description Graphs formalism.

# 8

## Description Graphs

In this chapter we present description graphs as an extension to the standard facilities available in [OWL](#) specifically targeted to modelling structured objects. We discuss how this addresses the shortcomings described in the previous chapter.

### 8.1 Definition

*Description graphs* are a representational formalism which has been introduced by [Motik et al. \(2008a,b,c\)](#) to address the weakness of [OWL](#) and traditional [DLs](#) in modelling structured objects, while still preserving the tractability of reasoning on ontologies containing such structured objects. However, in order to preserve the tractability of reasoning, some important constraints must be observed. These will be described in greater detail further on in this chapter.

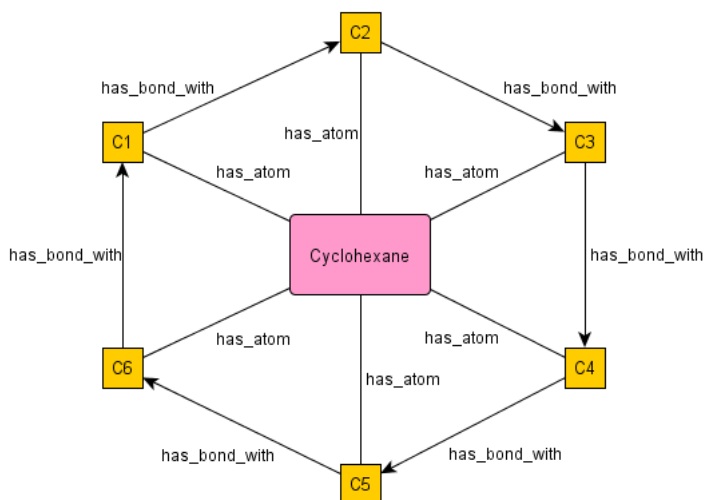
Structured objects in chemistry, such as molecules or the skeletons common to classes of molecules, are easily represented as connected graphs, as described in Chapter 3, i.e. in terms of atoms as nodes and bonds as the connections between them. They thus provide a natural use case for the description graph formalism. A description graph is a graph-like extension to the standard underlying [DL](#) formalism beneath [OWL-DL](#). Formally, we can define a description graph as follows:

**Definition 8.1.** A *description graph* is a directed graph  $G = (V, E, \lambda)$  in which each vertex  $i \in V$  is labeled with a set of (possibly negated) class names  $\lambda\langle i \rangle$ ; and each edge  $\langle i, j \rangle \in E$  is labeled with a set of atomic properties  $\lambda\langle i, j \rangle$ . Each description graph has a *main class*, which indicates the object whose structure is being modelled in the graph, and it is this main class that will be used to link to the remainder of the ontology that the



description graph is a part of (Motik *et al.*, 2008a).

For example, a description graph to represent a cyclohexane molecule might be constructed as illustrated in Figure 8.1. In this image, atoms are atomic classes, illustrated in orange square boxes, and the main class is denoted *Cyclohexane*, illustrated in the pink rectangle in the centre. Labelled lines represent atomic properties.



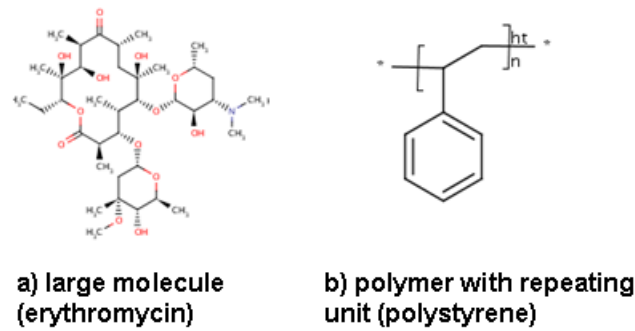
**Figure 8.1** Description graph representation of cyclohexane molecule

## 8.2 Decidability of Reasoning

At first glance, it would appear that allowing description graphs in ontologies leads to the same problem with undecidability of reasoning as the same representation of the structured data in **OWL** axioms or non-**DL**-safe rules. However, certain important restrictions allow for the development of a decidable reasoning algorithm for knowledge bases extended with description graphs. These are discussed below.

### 8.2.1 Boundedness of graphs

An important aspect of structured objects, which differentiates them from standard concepts usually modelled in **OWL** knowledge bases, is that the structure is usually modelled at a certain, fixed level of granularity; and at that level of granularity, the number of parts of the structure which are modelled is *bounded* (finite). In chemistry, chemical graphs are constructed to represent finite molecules, and molecules of arbitrary



**Figure 8.2** Large molecules and polymer representation

size such as polymers are not themselves modelled. Instead, a core repeating unit is chosen which characterises the structure of the larger molecule. This is illustrated in Figure 8.2.

Thus, although the number of required vertices in a description graph may be large in any particular domain, the size is bounded by some upper limit. This is crucial for ensuring termination of the reasoning algorithm developed by [Motik et al. \(2008a\)](#).

### 8.2.2 Separation of properties

For reasons described in [Motik et al. \(2008c\)](#), to ensure decidability of reasoning over the main **OWL** knowledge base, the properties which are used in the description graphs must not be referred to in the main ontology axioms. Since the main knowledge base consists of axioms referring to properties which range over the classes in the knowledge base, the full set of properties in the knowledge base has thus to be separated into *tree properties* and *graph properties*. This is known as the *strong separation requirement*.

The axioms in the **TBox** can propagate constraints across tree properties just like in standard **DLs**, and rules can be used to propagate constraints within a graph ([Motik et al., 2008c](#)).

### 8.2.3 Acyclicity

Although description graphs themselves are bounded, description graphs could lead to the ontology as a whole becoming undecidable if it is possible to create an infinite *chain* of description graph instances. This could be the case if a description graph implies another description graph in a way which creates a cycle of implication, for example if there is a chain of *graph specialisations* ([Motik et al., 2008a](#)):

**Definition 8.2.** A *graph specialisation* encodes the relationship between two graphs where one is the specialisation of the other. Graph specialisation is the equivalent of class subsumption, that is, given two graphs  $G_1$  and  $G_2$  with main classes  $C_1$  and  $C_2$  belonging to an ontology, if  $C_1 \sqsubseteq C_2$ , then  $G_1$  is a graph specialisation of  $G_2$ .

However, many domains can be adequately described by arranging the description graphs in an acyclic hierarchy. Motik *et al.* (2008a) thus introduce an acyclicity constraint on the description graphs which may be included in the knowledge base – not, that is, on the *internal structure* of the description graphs, but on the relationships *between* description graphs – such that a graph lower in the hierarchy may imply a graph higher in the hierarchy, but not the reverse. With this additional constraint, a decidable reasoning algorithm is obtained (Motik *et al.*, 2008a).

## 8.3 Rules

Due to the strong separation requirement which preserves decidability of reasoning in ontologies enhanced with description graphs, the properties which are used in the description graphs (i.e. the graph edges) may not be referred to in the main ontology axioms. The full set of properties in the knowledge base has thus to be separated into *tree properties* and *graph properties*. This provides a limitation in terms of the possibility for reasoning over the information encoded in the graphs, as the graph properties cannot be referred to in OWL axioms.

Thus, if *has\_atom* is a graph property, it is not possible to create an OWL axiom using it, such as the following:

```
SubClassOf(has_atom only (CarbonAtom or HydrogenAtom)) HydrocarbonMol
```

Without the possibility to refer to graph properties in OWL class axioms, it makes *classification* based on the internal structure of description graphs impossible without the addition of a construct different to the normal OWL terminological axioms. The construct which makes classification based on graph properties possible is *rules*.

Thus, classification based on properties of description graphs must be expressed with rules. Furthermore, these rules must be *role-safe*, which means safe with respect to the strong separation requirement, that is, they must not refer simultaneously to properties used in the graphs and those used in the OWL ontology axioms. Note that role safety for rules is not the same as DL safety. For DL safety, rules are restricted to refer only to known individuals in their consequents, while role safe rules can refer to arbitrary individuals in their consequents. On the other hand, for role safety, rules cannot refer

simultaneously both to graph properties and **OWL** properties, while **DL** safe rules have no such restriction. To maintain decidability of the overall knowledge base, therefore, it is required that the rules *either* refer only to graph properties and not object properties (role safety) *or* refer only to known individuals in their consequent (DL safety).

We thus introduce *graph rules* (Motik *et al.*, 2008b,a), as follows:

Let  $N_I$  and  $N_V$  be disjoint sets of *individuals* and *variables*.

An *atom* is of the form  $C(s)$  for a concept assertion,  $R(s, t)$  for a role assertion, where  $s, t \in N_I \cup N_V$ ,  $C$  is a concept, and  $R$  is a role.

A rule  $r$  is an expression of the form

$$B_1 \wedge B_2 \wedge \dots \wedge B_n \rightarrow H_1 \vee \dots \vee H_m \quad (8.1)$$

where  $B_i$  and  $H_j$  are atoms;  $n \geq 0$ , and  $m \geq 0$ .

The conjunction  $B_1 \wedge \dots \wedge B_n$  is called the *antecedent*, and the disjunction  $H_1 \vee \dots \vee H_m$  is called the *consequent*. A *graph rule* is a rule where all concepts and roles in the atoms are atomic, and that can also contain graph atoms of the form  $G(t_1, \dots, t_k)$ , for  $G$  an  $l$ -ary description graph and  $t_i \in N_I \cup N_V$  (Motik *et al.*, 2008b,a).

Rules form a natural medium for expressing *conditionality* about structured objects, which cannot be expressed in the ontology **TBox** (**OWL** axioms) if the overall ontology is to preserve decidability of reasoning. The structure of the structured object is, however, able to be expressed at the class level in the description graph, and rules can be used to support additional conditional inferencing based on the structure of the graphs in a knowledge base.

### 8.3.1 Definition

We are now in a position to provide a definition for a graph-extended knowledge base.

**Definition 8.3.** A graph-extended **OWL** knowledge base is a 4-tuple  $K = (T, G, P, A)$  where  $T$  is a set of **OWL** class axioms,  $G$  is a set of description graphs,  $P$  is a set of rules, and  $A$  is a set of **OWL** assertions.  $T$  is allowed to refer only to tree properties,  $G$  and  $P$  are allowed to refer only to the graph properties, and  $A$  is allowed to refer to both graph and tree properties (Motik *et al.*, 2008c,b,a).

## 8.4 Implementation

Description graphs are implemented as an extension to the OWL API ([Horridge and Bechhofer, 2009](#); [Glimm et al., 2009](#)), and reasoning over description graphs is implemented in the HermiT reasoner ([Shearer et al., 2008](#)). HermiT also supports reasoning with graph rules ([Glimm et al., 2009](#)), thus it allows construction of a full knowledge base about structured objects.

```

DGRule ::= 'DescriptionGraphRule' '(' {Annotation} 'Body' '(' {DGAAtom} ')'
        'Head' '(' {DGAAtom} ')' ')'

DGAAtom ::= 'ClassAtom' '(' ClassExpression IArg ')'
        | 'ObjectPropertyAtom' '(' ObjectPropertyExpression IArg IArg ')'

DGAxiom ::= 'DescriptionGraph' '(' {Annotation} DGNodes
        DGEEdges MainClasses ')'

DGNodes ::= 'Nodes' '(' NodeAssertion {NodeAssertion} ')'
NodeAssertion ::= 'NodeAssertion' '(' Class DGNode ')'
DGNode ::= IRI
DGEEdges ::= 'Edges' '(' EdgeAssertion {EdgeAssertion} ')'
EdgeAssertion ::= 'EdgeAssertion' '(' ObjectProperty DGNode DGNode ')'
MainClasses ::= 'MainClasses' '(' Class {Class} ')'

```

**Figure 8.3** Description Graphs and Rules Syntax

Description graphs are implemented in a functional style syntax. Description graphs consist of description graph axioms, each of which is a set of description graph nodes, a set of description graph edges, and a set of main classes. Description graph rules consist of a body and a head, each of which are composed of atoms referring to description graph properties. Figure 8.3 shows the functional style syntax for description graphs and rules, as set out in ([Glimm et al., 2009](#)).

For example, the graph representation of *Cyclohexane* illustrated in Figure 8.1, in functional syntax, is shown below.

```

DescriptionGraph(
  Nodes(
    NodeAssertion( Class Cyclohexane )
    NodeAssertion( Class C1 )
    NodeAssertion( Class C2 )
    NodeAssertion( Class C3 )
    NodeAssertion( Class C4 )
    NodeAssertion( Class C5 )
    NodeAssertion( Class C6 )
  )
  Edges(

```

```
EdgeAssertion( has_atom Cyclohexane C1 )
EdgeAssertion( has_atom Cyclohexane C2 )
EdgeAssertion( has_atom Cyclohexane C3 )
EdgeAssertion( has_atom Cyclohexane C4 )
EdgeAssertion( has_atom Cyclohexane C5 )
EdgeAssertion( has_atom Cyclohexane C6 )
EdgeAssertion( has_bond_with C1 C2 )
EdgeAssertion( has_bond_with C2 C3 )
EdgeAssertion( has_bond_with C3 C4 )
EdgeAssertion( has_bond_with C4 C5 )
EdgeAssertion( has_bond_with C5 C6 )
EdgeAssertion( has_bond_with C6 C1 )
)
MainClasses(
  Cyclohexane
)
)
```

---

In conclusion, description graphs are an extension to the standard ontology constructs which allows for decidable reasoning over structured objects. This is particularly applicable in the domain of chemistry, where chemists have long used graph theoretical representations of the structures of chemical entities. However, structured objects are widespread in biochemistry and the broader biomedical domain, and the description graph extension to the standard [OWL](#) suite of ontology modelling constructs seems promising for more accurate representation and reasoning over such structured objects.



# 9

## Software design and implementation

In this chapter, we present the design of the software which implements the empirical part of this dissertation, in the context of the research questions which we presented in Chapter 1. The overall technical architecture is discussed in Section 9.2, software dependencies in Section 9.3 and the implementation details in Section 9.4.

### 9.1 Overview

OWL ontologies, based on DLs, are becoming widespread tools to assist in knowledge based systems development throughout the life sciences. In chemistry, OWL is used in chemical ontologies such as ChEBI. However, as we have seen, due to their cyclic nature, OWL is not able to represent the complex structures of chemical entities at the class level in TBox axioms.

The purpose of this dissertation is to evaluate the utility and scalability of description graphs and rules for the representation of, and reasoning over, chemical structures in an enriched OWL knowledge base. To do this, we will make use of the chemical structures associated with chemical entities in the standard chemical graph format, and *convert* them into description graphs. Furthermore, we will create rules to allow conditional reasoning based on the structures in the graphs.

Recall that we begin with the following three questions, each of which represents functionality which is expected by domain experts from a structure-enhanced chemical knowledge base:

- Can chemical entities be classified based on their substructures?
- Can basic chemical properties be determined from the description graphs?



- How scalable is the resulting knowledge base?

We will address these questions in our software design by:

- Converting chemical structures into description graphs and associating them with an [OWL](#) chemical knowledge base;
- Using rules to test for the the structures of chemical *groups* (parts) as substructures in the converted chemical structures, and performing classification if the group is found;
- Formulating rules to test for basic structure-based chemical properties based on the information contained in the description graphs; and
- Creating a generator to add additional data in batches to the knowledge base and testing the performance in terms of time taken to reason over knowledge bases of different sizes.

## 9.2 Architecture

Figure 9.1 gives an overview of the architecture.

The knowledge base consists of:

1. A simple ontology describing classes pertaining to chemical entities,
2. Auto-generated description graphs from structures in the ChEBI database, and
3. Rules for structure-based classification.

Data from the [ChEBI](#) database ([de Matos \*et al.\*, 2010](#)) was used as a source for chemical structures. The full [ChEBI](#) database is available for download from <http://www.ebi.ac.uk/chebi/downloadsForward.do>. The chemical structures are in MOLfile format in the STRUCTURE field of the Structures table. See Chapter 3 for an explanation of this format.

The Chemistry Development Kit ([CDK](#)) was used to parse the chemical structures into the [CDK](#)'s internal Java object model, and from this the structural information was converted into description graphs. The description graphs were then combined with the base [OWL](#) ontology as well as the rules and the resulting knowledge base was used as input to the HermiT reasoner.

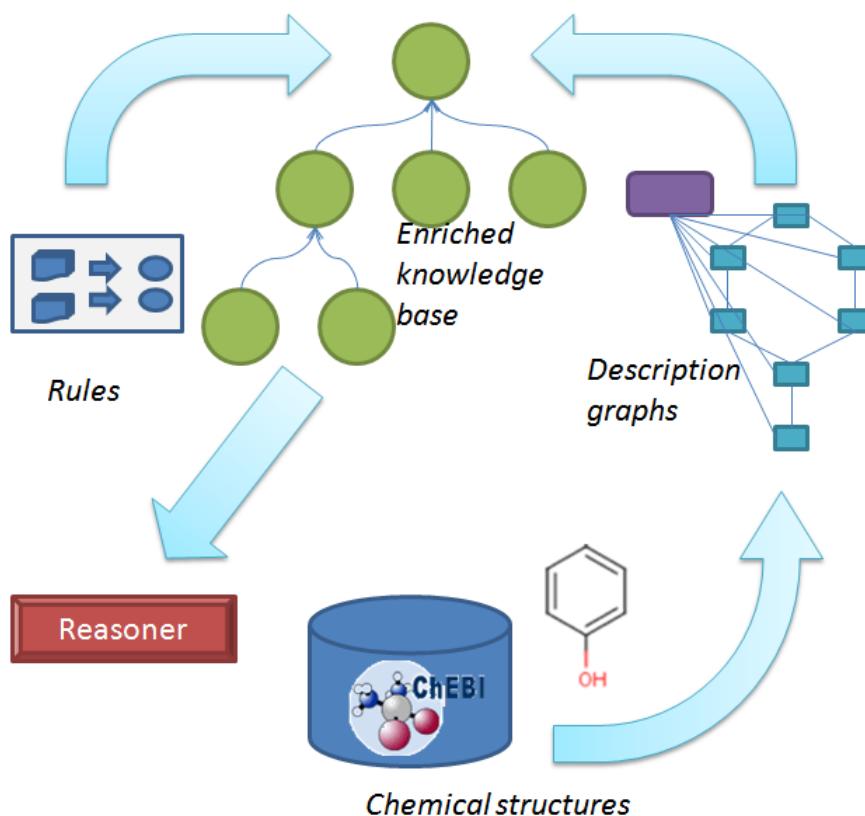


Figure 9.1 Overall architecture

### 9.2.1 Downloadable files

The ontology is available in two files. The main ontology is available at [http://www.ebi.ac.uk/~hastings/owled2010/chemistry\\_dgs\\_ontology.owl](http://www.ebi.ac.uk/~hastings/owled2010/chemistry_dgs_ontology.owl), and the graphs are available in a separate file at [http://www.ebi.ac.uk/~hastings/owled2010/chemistry\\_dgs\\_graphs.owl](http://www.ebi.ac.uk/~hastings/owled2010/chemistry_dgs_graphs.owl).

The software including all libraries and sources is available at <http://www.ebi.ac.uk/~hastings/owled2010/descgraphs.zip>.

In the next section, we describe the software libraries that we made use of in our implementation and the role that they played in our architecture.

## 9.3 Software libraries

The primary language of the implementation was Java, and we made use of several freely available Java open source libraries in support of our implementation. We briefly describe

these software libraries in this section.

### 9.3.1 Chemical data conversion

The [CDK](#) is a collaboratively developed open source software library providing implementations for many of the common cheminformatics problems. These range from Quantitative Structure-Activity Relationship ([QSAR](#)) descriptor calculations to [2D](#) and [3D](#) model building, input and output in different formats, SMILES parsing and generation, ring searches, isomorphism checking, and structure diagram generation ([Steinbeck et al., 2003, 2006](#)). It is written in Java and forms the backbone of a growing number of cheminformatics and bioinformatics applications.

For this implementation, the particular feature of the [CDK](#) that we made use of was the parsing of chemical structures in MOLfile format into the internal [CDK](#) chemical entity domain model.

### 9.3.2 Programmatic manipulation of [OWL](#) ontologies

Although Graphical user interface ([GUI](#))-based tools with which to manipulate [OWL](#) ontologies do exist, these tools do not yet provide support for recent extensions of which description graphs is an example. Since the tools which allow graphical manipulation of graph-enriched ontologies do not yet exist, it was necessary for this research that the ontology and graphs be created and interfaced with programmatically. The OWL API is an Application programming interface ([API](#)) for the creation, manipulation, and interfacing with [OWL](#) ontologies using the Java programming language ([Horridge and Bechhofer, 2009](#)). It is open source and has been developed primarily at the University of Manchester, providing an in-memory reference implementation for ontology manipulation. It also provides facilities for input and output of ontologies in several formats including the popular [OWL](#) Manchester syntax and [RDF](#). Finally it provides a standardised mechanism for passing ontologies to reasoners and interfaces to diverse reasoners including HermiT, which is the reasoner used in this research and described in [Section 9.3.3](#).

The OWL API formed the backbone of our implementation, allowing us to create the ontology, graphs and rules, pass the resulting knowledge base programmatically to our chosen reasoner, and receive the result after reasoning was completed.

### 9.3.3 Reasoning over graph-enriched knowledge bases

For reasoning over the graph-enriched knowledge base, we made use of the HermiT library<sup>1</sup> as this is the only reasoner that currently provides support for both description graphs and rules.

HermiT is a Java reasoner for [OWL](#) ontologies. Given an [OWL](#) file, HermiT is able to perform common reasoning tasks such as determining whether or not the ontology is consistent, identifying subsumption relationships between classes, and applying any rules specified in the knowledge base.

HermiT is based on a novel *hypertableau* calculus which provides much more efficient reasoning than earlier algorithms. Ontologies which previously required minutes or hours to classify can often be classified in seconds by HermiT. HermiT is the first reasoner able to classify a number of ontologies which had previously proven too complex for any available system to handle ([Shearer \*et al.\*, 2008](#)).

## 9.4 Implementation

In this section, we describe the structure of the implemented knowledge base.

### 9.4.1 Ontology

At the root of the ontology is the node ‘chemical entity’, beneath which are nodes for the primary division in kind of entity, namely ‘group’, ‘atom’, ‘molecule’, and ‘ion’. ‘Atom’ is further divided into the concrete types of atoms as per the periodic table, such as ‘carbon atom’ and ‘oxygen atom’.

An illustration of the overall structure of the core terms of the ontology is shown in [Figure 9.2](#).

### 9.4.2 Description Graphs

Description graphs were automatically generated<sup>2</sup> from a MOLfile connection table format [MDL \(2010\)](#). The standard MOLfile format consists of an atom table, which provides information about the atoms included in the molecule such as their types, and a

---

<sup>1</sup>Version 1.2.2 with slight customisation for input and output of graphs which is currently only partially supported by the HermiT library and the OWL API.

<sup>2</sup>Our software for this experiment is available in source and binary at <http://www.ebi.ac.uk/~hastings/owled2010/descgraphs.zip>



**Figure 9.2** Core ontology structure

bond table, which provides information about the bonds included in the molecule such as which atoms they connect and the bond order. Chapter 3 gives more information about the MOLfile format.

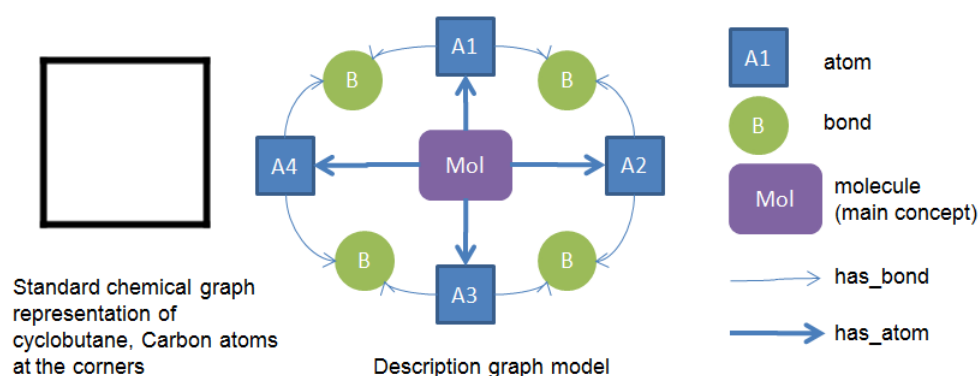
For example, here is the MOLfile format for a simple cyclobutane molecule:

```

Marvin 05170622152D

4 4 0 0 0 0          999 V2000
  0.4125   0.4125   0.0000 C   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 -0.4125   0.4125   0.0000 C   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 -0.4125  -0.4125   0.0000 C   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0.4125  -0.4125   0.0000 C   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2  1  1  0  0  0  0
4  1  1  0  0  0  0
3  2  1  0  0  0  0
4  3  1  0  0  0  0
M  END
  
```

Each description graph consists of a vertex for the description graph *main class* which is a subclass of ‘molecule’ in the ontology, a vertex for each atom which is a subclass of the atom type e.g. ‘carbon atom’ in the ontology, and a vertex for each bond which is a subclass of the bond type in the ontology e.g. ‘single’. Each atom vertex is connected to the molecule by the *has\_atom* property. Atom vertices are associated with bonds with *has\_bond*. Figure 9.3 shows an illustration of the description graph for cyclobutane.



**Figure 9.3** Illustration of the cyclobutane description graph

The description graph representation of cyclobutane which is added to the ontology is reproduced below in functional-style syntax. Note that 30377 is the ID for cyclobutane in the ChEBI database. Each atom is a node, and each bond is a node. Edges are *has\_bond* relationships between atoms and bonds and *has\_atom* relationships between the molecule (which is the main class) and the atoms.

```
DescriptionGraph (Nodes (
NodeAssertion(<30377_Atom_(C) 0> <30377_Atom_(C) 0>)
NodeAssertion(<30377_Atom_(C) 1> <30377_Atom_(C) 1>)
NodeAssertion(<30377_Atom_(C) 2> <30377_Atom_(C) 2>)
NodeAssertion(<30377_Atom_(C) 3> <30377_Atom_(C) 3>)
NodeAssertion(<30377_Bond_(SINGLE) 5> <30377_Bond_(SINGLE) 5>)
NodeAssertion(<30377_Bond_(SINGLE) 4> <30377_Bond_(SINGLE) 4>)
NodeAssertion(<30377_Bond_(SINGLE) 6> <30377_Bond_(SINGLE) 6>)
NodeAssertion(<30377_Bond_(SINGLE) 7> <30377_Bond_(SINGLE) 7>)
)
Edges (
EdgeAssertion(has_bond 30377_Atom_(C) 1 30377_Bond_(SINGLE) 4)
EdgeAssertion(has_bond 30377_Atom_(C) 0 30377_Bond_(SINGLE) 4)
EdgeAssertion(has_bond 30377_Atom_(C) 3 30377_Bond_(SINGLE) 5)
EdgeAssertion(has_bond 30377_Atom_(C) 0 30377_Bond_(SINGLE) 5)
EdgeAssertion(has_bond 30377_Atom_(C) 2 30377_Bond_(SINGLE) 6)
EdgeAssertion(has_bond 30377_Atom_(C) 1 30377_Bond_(SINGLE) 6)
EdgeAssertion(has_bond 30377_Atom_(C) 3 30377_Bond_(SINGLE) 7)
```

```
EdgeAssertion(has_bond 30377_Atom_(C)2 30377_Bond_(SINGLE)7)
EdgeAssertion(has_atom molecule_30377 30377_Atom_(C)0)
EdgeAssertion(has_atom molecule_30377 30377_Atom_(C)1)
EdgeAssertion(has_atom molecule_30377 30377_Atom_(C)2)
EdgeAssertion(has_atom molecule_30377 30377_Atom_(C)3)
)
MainClasses(<molecule_30377>
))
```

---

The vertices (but not the properties) of the description graphs are also classified in the main [OWL](#) ontology. The main class is classified beneath ‘molecule’ in the main ontology, the atoms beneath ‘atom’ and the bonds beneath ‘bond’.

The inclusion of cyclobutane in the main ontology is shown here in OWL Manchester syntax. Note that no relationship is asserted between this molecule and, for example, the class of cyclic molecules.

---

```
Class: <molecule_30377>
  Annotations:
    rdfs:label "cyclobutane"@en
  SubClassOf:
    <molecule>
```

---

Atoms are each classified beneath the relevant atom type, as follows:

---

```
Class: <30377_Atom_(C)3>
  SubClassOf:
    <carbon_atom>
```

---

And, each of the bonds is classified beneath the relevant bond type, as follows:

---

```
Class: <30377_Bond_(SINGLE)5>
  SubClassOf:
    <single_bond>
```

---

### 9.4.3 Rules

We implemented rules to classify chemical structures based on their composition and their connectivity. Rules were devised for the classification of *cyclic* compounds, which contained a cycle of connected atoms. Figure [9.4](#) shows examples of cyclic compounds.

Separate rules were created for cycles of different bond path lengths. For example, a rule to determine cycles of length three bonds is given in [9.1](#) below. Note that

---



**Figure 9.4** Some cyclic compounds

this was slightly simplified for readability. The full generated version also includes `DifferentFrom` statements to ensure non-trivial cycles.

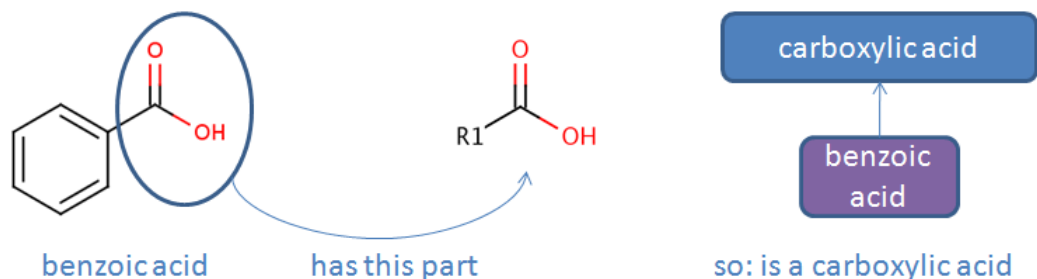
$$\begin{aligned}
 & \text{Molecule}(M) \wedge \\
 & \text{has\_atom}(M, A_1) \wedge \text{has\_atom}(M, A_2) \wedge \text{has\_atom}(M, A_3) \wedge \\
 & \text{Atom}(M, A_1) \wedge \text{Atom}(M, A_2) \wedge \text{Atom}(M, A_3) \wedge \\
 & \text{Bond}(M, B_1) \wedge \text{Bond}(M, B_2) \wedge \text{Bond}(M, B_3) \wedge \\
 & \text{has\_bond}(A_1, B_1) \wedge \text{has\_bond}(A_2, B_1) \wedge \\
 & \text{has\_bond}(A_2, B_2) \wedge \text{has\_bond}(A_3, B_2) \wedge \\
 & \text{has\_bond}(A_1, B_3) \wedge \text{has\_bond}(A_3, B_3) \wedge \\
 & \rightarrow \text{instanceOf}(M, \text{CyclicMolecule})
 \end{aligned} \tag{9.1}$$

where  $M$  is an arbitrary individual of type *Molecule*;  $A_1 - A_3$  are atoms;  $B_1 - B_3$  are bonds forming a cyclic path such that atom  $A_1$  is bonded to atom  $A_2$ ,  $A_2$  to  $A_3$  and  $A_3$  to  $A_1$  again. *CyclicMolecule* is a class in the ontology which has as members all molecules that contain a cycle.

Rules were also devised to determine *parthood* between chemical structures. If all atoms of  $A$  are atoms of  $B$ , and all bonds of  $A$  are bonds of  $B$ , then  $A$  is a *subgraph* of  $B$ . The term *group* is commonly used to denote arbitrary chemical parts, while the terms *molecule*, *ion* and so on refer to entire (complete) structures. Rules were devised for each group so as to identify these groups within the molecule. However, a consequence of the strong separation requirement is that a single rule cannot refer to both graph properties and tree properties. For this reason, even if we determine that a given graph is a subgraph of another graph, we cannot assert a relationship such as *has\_part* between the two main classes at the ontology level. A workaround for this is to create a class for every group, such that if the structure of the group is a subgraph of the structure of the molecule, then



the molecule can be classified as belonging to that particular class. This is illustrated in Figure 9.5.



**Figure 9.5** Classification based on parthood

Rules for parthood determination are of the form

$$\begin{aligned}
 & \text{Molecule}(M) \wedge \\
 & \text{has\_atom}(M, A_1) \wedge \dots \wedge \text{has\_atom}(M, A_n) \wedge \\
 & \text{Atom}(A_1) \wedge \dots \wedge \text{Atom}(A_n) \wedge \\
 & \text{Bond}(B_1) \wedge \dots \wedge \text{Bond}(B_m) \wedge \\
 & \text{has\_bond}(A_{i1}, B_{j1}) \wedge \dots \wedge \text{has\_bond}(A_{in}, B_{jm}) \\
 & \rightarrow \text{instanceOf}(M, \text{Class})
 \end{aligned}
 \tag{9.2}$$

where  $M$  is an arbitrary individual of type *molecule*;  $A_1 - A_n$  are *group* atoms; bonds exist between the group atoms  $A_{i1} - A_{in}$  and  $A_{j1} - A_{jn}$ , and *Class* is a class the identity of which depends on the group used to generate the rule, for example ‘carboxylic acid’ for the ‘carboxy group’.

The properties used as the graph edges (*has\_atom*, *has\_bond*) are available for use in the rules, as long as a rule does not mix graph properties with properties used in [OWL](#) axioms in the main ontology.

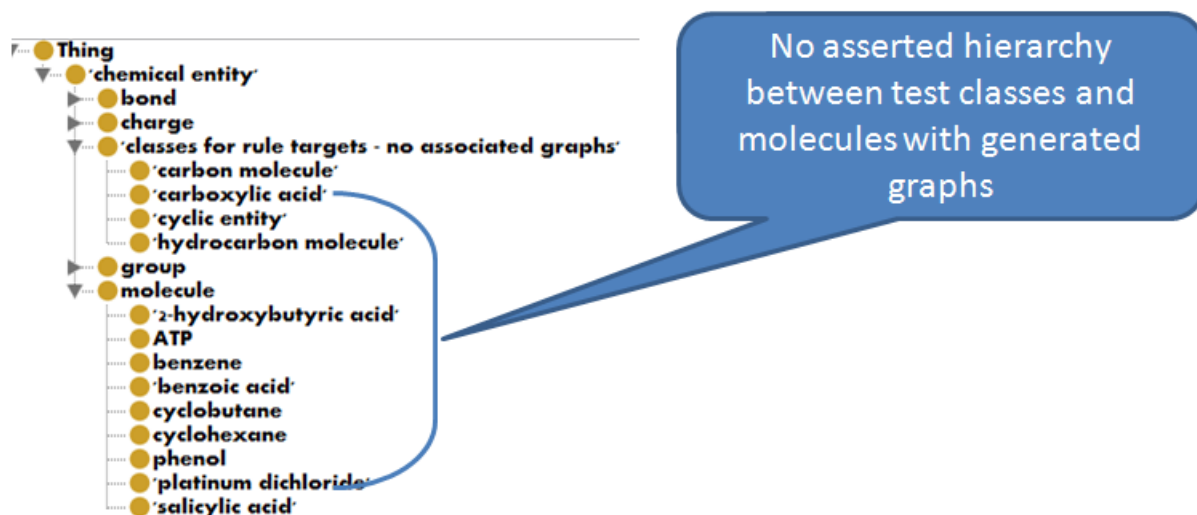
# 10

## Results

In this chapter we give the results of our project implementation and discuss the difficulties which were encountered during the implementation, design choices which were debated, and experience which was gained.

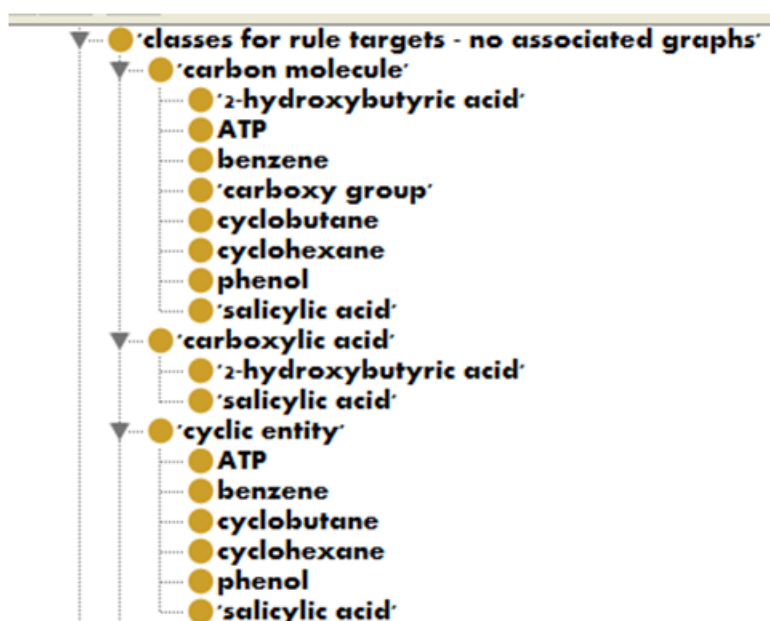
Our primary result was to establish whether reasoning with the rules over the combined knowledge base enriched with description graphs did indeed result in classification of description graph-enriched classes. These were successfully classified as cyclic molecules and as classes containing specific defined groups such as carboxylic acids<sup>1</sup>.

Figure 10.1 shows the asserted ontology before classification on the graphs and rules was executed, and Figure 10.2 shows the inferred hierarchy.



**Figure 10.1** Ontology hierarchy with test classes before classification execution

<sup>1</sup>The resulting inferred ontology is available at [http://www.ebi.ac.uk/~hastings/owled2010/chemistry\\_dgs\\_inferred.owl](http://www.ebi.ac.uk/~hastings/owled2010/chemistry_dgs_inferred.owl)



**Figure 10.2** Ontology hierarchy with test classes after reasoning

For example, an extract of the inferred ontology showing membership of the class for cyclobutane (with ID 30377) in the class of cyclic molecules is shown below. Recall that the only asserted class membership for this class was to the *molecule* class.

```

<Class rdf:about="molecule_30377">
  <rdfs:subClassOf rdf:resource="carbon_molecule"/>
  <rdfs:subClassOf rdf:resource="cyclic_entity"/>
  <rdfs:subClassOf rdf:resource="molecule"/>

```

We find this result positive in terms of overcoming the previously explicit limitation of [OWL](#) knowledge bases in expressing arbitrarily structured objects at the class level and performing classification based on the structure.

In the next sections we address the specific research questions we asked and discuss some limitations and challenges both in the approach in general and in this research project in particular.

## 10.1 Classification based on substructures

As presented in the previous chapter, rules were created to determine *parthood* between chemical structures, that is, to test whether all atoms of A are atoms of B, and all bonds of A are bonds of B. This was done, for example, for the highly reactive *Carboxy* group which forms the reactive definition of the class of *carboxylic acids*.

Recall that the consequence of the strong separation requirement for the rule implementation is that a single rule cannot refer to both graph properties and tree properties. For this reason, even if we determine that a given graph is a subgraph of another graph, we cannot assert a relationship such as *has\_part* between the two main classes at the ontology level. The workaround that we implemented for this is to create a class for every group, such that if the group’s structure is a subgraph of the molecule’s structure, then the molecule can be classified as belonging to that class. One such class which we created in the ontology was the class of carboxylic acids. We were able to create a rule which performed classification based on parthood by testing, in the antecedent of the rule, for the graph substructure matching the carboxy group, and then asserting, in the consequent of the rule, that the main class of the description graph was a subclass of the carboxylic acid class.

Within our knowledge base, this rule tested for the Carboxy group substructure and was able to classify graph-enriched classes representing molecules as subclasses of carboxylic acid when the test successfully matched for a given graph.

## 10.2 Determination of chemical properties from graphs

The primary chemical property for which we encoded rules in our knowledge base was *cyclicity*. Cyclic molecules in general display many different properties (such as reactivity and stability) when compared to their linear counterparts (Zero *et al.*, 2009), and as a result chemists frequently classify molecules into different classes depending on the number and size of the cycles present.

We therefore created rules for determining cycles of connected atoms within molecules, and these, together with the description graphs and the reasoner, were able to automatically classify the graph-enriched classes as cyclic within the ontology.

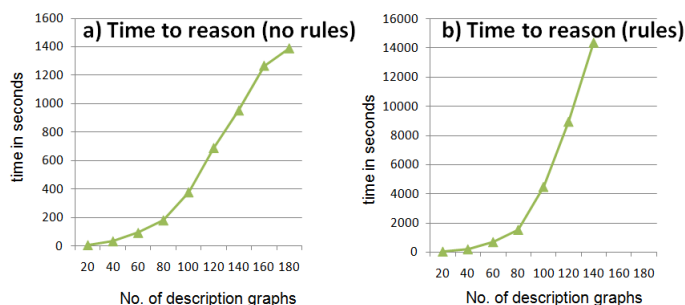
One limitation of the current approach is that each different cycle length requires a different rule. This is because each atom and bond in the cyclic path has to be expressed by a different individual in the rule antecedent (together with *differentFrom* clauses to ensure that different atoms are matched within the description graph for the molecule). We generated different rules to test graph cycles from length three to eight bonds in the cyclic path.

A further, and more pressing, limitation, is that the types of conditionality that can be expressed in rules potentially provide the facility for only a limited set of chemical properties relative to those required by chemists. For example, it is difficult to express rules

that must apply to *all* atoms from a given molecule's graph without specifically naming those atoms, since there is no `forAll` operator for the rules implemented alongside **OWL** as described in Chapter 6, Section 6.6. It is therefore not possible to test for *hydrocarbon* molecules, that is, those which contain *only* Carbon or Hydrogen atoms, since this would require the facility not only to match for certain atoms within the molecule, but to ensure that there are no additional atoms within the molecule which do not match. In **OWL** axioms, this type of restriction is accomplished with a *closure axiom* in terms of a *value restriction* ( $\forall R.C$ ).

### 10.3 Scalability of reasoning over knowledge base

To evaluate the performance, we executed reasoning (ontology classification) over iteratively increasing sizes of the knowledge base, both with and without rules. The results are summarised in Figure 10.3<sup>2</sup>.



**Figure 10.3** Performance results of classification

We did not attempt to control the size of the graphs which we randomly selected for inclusion into our knowledge base, but note that the average size of a molecule in the **CHEBI** database is around 30 atoms<sup>3</sup>.

We first present the results for reasoning over a graph-enriched knowledge base without any generated rules. This tests the capacity of the reasoner to reason over the **OWL** knowledge base, together with description graphs, but without any *conditionality* based on the graphs. Additional **OWL** axioms may be expressed in the ontology, making use of the normal **OWL** constructors and properties. Table 10.1 gives the times to reason over the graph-enriched but not rule-enriched knowledge base.

<sup>2</sup>Tested on a Dell twin core laptop.

<sup>3</sup>Excluding hydrogen atoms, which are commonly left implicit, as these can be ‘added back’ by calculations to determine their predicted positions. See Chapter 3 for further details.

No. Of Graphs	No. Of Axioms	Time to reason (seconds)
20	371	8
40	674	35
60	1015	94
80	1277	182
100	1844	379
120	2371	689
140	2677	952
160	2749	1266
180	2862	1388

**Table 10.1** Results of performance evaluation – No generated rules

We then added in the generated rules, which provide the ability to perform classification based on properties (such as being cyclic) determined from the structure of the description graphs, and re-tested the scalability of reasoning together with the rules. Table 10.2 gives the times to reason over the graph and rule-enriched knowledge base.

No. Of Graphs	No. Of Axioms	Time to reason (seconds)
20	371	34
40	674	193
60	1015	687
80	1277	1535
100	1844	4479
120	2371	8952
140	2677	14380

**Table 10.2** Results of performance evaluation – Including generated rules

The scalability of the reasoner against the knowledge base enriched with description graphs appears workable, with reasoning time growing to a maximum of 23 minutes (1388 seconds) for a knowledge base enriched with 180 graphs. However, including the graphs alone – without rules – does not allow for any classification based on the information encoded in the graphs. The rules are thus essential to expose the structure in the graphs to the reasoner. Unfortunately, we find that reasoning over the knowledge base enriched with graphs and rules appears to grow very rapidly into unmanageable durations, with the highest duration that we recorded for a knowledge base enriched with 140 graphs taking four hours (14380 seconds) to classify. This scalability is affected dramatically by the number and complexity of the generated rules, therefore this would appear to be a limiting factor in following our approach in a more complete fashion, where many chemical properties and subgraph relations might be expected to be included in the same

---

knowledge base to adequately model domain knowledge.

# 11

## Discussion

In this chapter, we discuss this research within the broader context of related research and the requirements of expert systems in the chemistry domain in general. We begin by discussing the expectations on expert systems performing chemical classification and to what extent our research has shown, at least in principle, that the description graph approach will be able to fulfill these expectations (Section 11.1). We then discuss related work in Section 11.2 and finally summarise the relationship of our work with these historical efforts in Section 11.3.

While sources of chemical data abound at an ever-increasing rate, relatively little of that has heretofore become available in the form of structured ontologies or other logic-based artificial intelligence ‘knowledge’ representations.

### 11.1 Structure-based chemical classification by expert systems

While the chemical graph formalism presents a method by which chemical structural information is readily encoded in several available standard computable formats, nevertheless there remains a gap in the broader area of chemical knowledge representation, since the chemical graph formalism only applies to fully defined structures. Representation of chemical structures with the chemical graph does not directly aid in the classification of those structures. Classification and interpretation of chemical structures is a specialised task, with chemists studying for years to master the highly developed chemical classification system in widespread use, and furthermore to understand the nuances and many



## 11.1. STRUCTURE-BASED CHEMICAL CLASSIFICATION BY EXPERT SYSTEMS

---

different variants in use within different specialised communities. We will take a closer look at the main features of chemical classification systems.

Chemical classes, the objects found within a chemical classification system, group together chemical entities in such a way as to allow meaningful scientifically relevant groups to be created and described. Ideally, all members of a chemical class should share important dispositional properties such as chemical reactivity. In fact, almost the only method of classification which was available to historical chemists, before compound structures were well understood, was based on the observation of reactivity through means of performing controlled reactions between different substances. Much of this historical classification is still inherited today and is taught in chemistry classes and reproduced in textbooks. Knowledge about the structural features which form the underlying *causes* of the shared dispositional properties (where such existed) was only determined later. However, now that chemical structures are well described (within the limits of the chemical graph formalism), many more structural features are able to be used for chemical class definitions. We can distinguish between useful chemical classes and irrelevant chemical classes. It is clear that classes such as “All compounds containing exactly 12 atoms” share no such dispositional properties, thus do not represent a useful chemical class for that reason<sup>1</sup>.

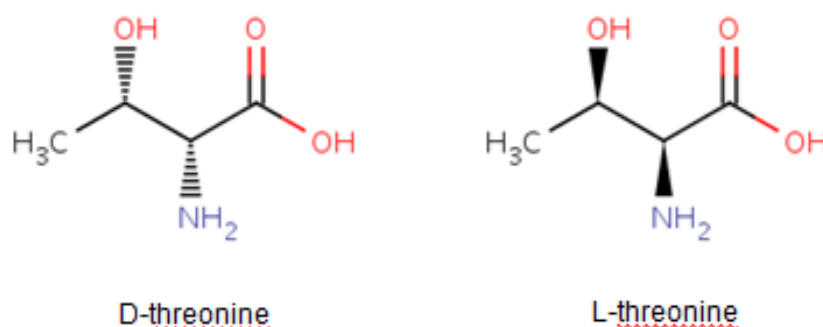
We here consider the typical class definitions which can be found in the traditional chemical classes in common use in chemistry, and try to analyse the features required for such definitions. The first element of chemical class definitions, providing a complete definition when it can be specified, is a full chemical graph for that class of molecules. The limits of the chemical graph formalism have been discussed earlier. Furthermore, we note that chemical graphs can be specified for completely stereochemically specified entities, and for completely stereochemically unspecified entities, but relative configurations of stereogenic centers cannot be specified using traditional chemical graph representation formalisms. For example, ‘allothreonine’ [rel-(2R,3R)-2-amino-3-hydroxybutanoic acid] and ‘threonine’ [rel-(2R,3S)-2-amino-3-hydroxybutanoic acid] are compounds with a relative configuration of stereogenic centres, thus for which a graph cannot currently be drawn. Figure 11.1 illustrates completely stereochemically specified forms of threonine. The *wedge bond* convention (dashed or solid) illustrates that the attached groups are going in or out of the plane of the remainder of the molecule. What cannot be represented in the current graph formalism is a relative arrangement of these - if the first is up, the second is

---

<sup>1</sup>Although, for technical reasons, such specifications may be valuable. Consider the database of all synthetically accessible, drug-like small molecules with up to 13 atoms, GDB-13(Blum and Reymond, 2009).

### 11.1. STRUCTURE-BASED CHEMICAL CLASSIFICATION BY EXPERT SYSTEMS

down - or conversely, if the first is down, the second is up - for example. However, we would be able to express this arrangement in *rules* within a graph enriched knowledge base such as the one we used.



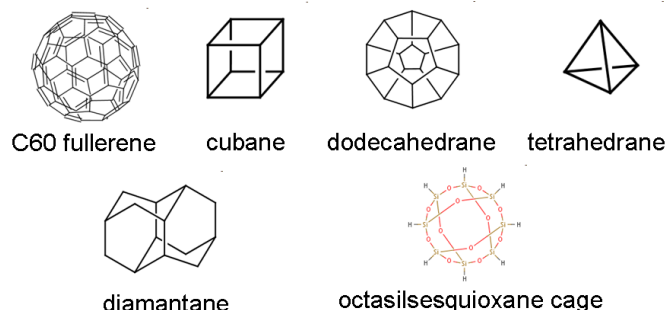
**Figure 11.1** D-threonine and L-threonine

The next element which is common in chemical class definitions is the composition in terms of which groups are attached (which may be simple atoms) and how many of them are attached, i.e. the cardinality. For example, ‘tricarboxylic acid’ can be defined as a compound containing exactly three carboxy groups. This type of classification, as we have illustrated, is able to be expressed in a graph enriched [OWL](#) knowledge base such as we have designed in this research.

A core structural element used in class definitions in chemistry is the *skeleton* of the molecule. For example, the class ‘metalloporphyrins’ is defined as any compound containing a porphyrin skeleton and a metal atom. Note that as the term is commonly used in chemistry, a skeleton is not always a straightforward substructure, since bond order (e.g. single or double) may vary, and bonds may even be added or removed while retaining the same ‘skeleton’. However, for our present purposes it will be sufficient to restrict the definition of skeleton to the stronger one which implies that the skeleton must be a substructure of the molecule it is a skeleton of, and thereby we are able to perform this type of classification within our knowledge base as illustrated by the successful classification based on the carboxy group substructure.

Another element commonly used in class definitions is the number and arrangement of rings (cycles) in a *ring system* which is a part of the molecule. A ring system is composed of multiple individual rings connected in a complex fashion. For example, the classes ‘ring assembly’ and ‘polycyclic cage’ (Figure [11.2](#)) both refer, in their definitions, to numbers and arrangements of rings in the molecule. We have illustrated our ability to classify graph-enriched classes representing molecules in terms of the number and size of

cycles which are present, and this is an important step in the direction of defining classes based on such complex ring arrangements.



**Figure 11.2** Polycyclic molecular cages

Straightforward chemical properties such as charge and unpaired electrons are used to define broad classes of molecules such as ion and radical. These may also apply at a lower level of classification, such as ‘aromatic diazonium ion’. Aromaticity and saturation are other properties commonly found in class definitions. Our approach allows for the classification based on straightforward properties of chemical entities. Unfortunately, some properties of chemical entities cannot be expressed in the rules formalism due to the limitation of not having a construct for *closure axioms* within the rules, as discussed in the previous chapter.

Finally, an interesting element of chemical class definitions is based on structural formulae, which is problematic to depict with existing graph-based tools. For example, ‘alkane’ is defined as “an acyclic branched or unbranched hydrocarbon having the general formula  $C_nH_{2n+2}$ .” These types of structural formulae also appear to be beyond the reach of the rules formalism at present.

Our goal in this research project has been restricted to testing whether the [OWL](#), description graphs and rules approach is able to perform classification based on chemical properties and substructures. We have not therefore been able to completely evaluate the extent to which our approach allows for the expression of each of these important properties as discussed here. However, we have seen that many of the more important properties relevant for chemical classification do appear possible, and it might be a direction for future research to extend the evaluation and implement some of these properties. The *scalability* concerns as regarding reasoning with rules are of paramount concern to the future of our approach, as is the limitation in terms of the kinds of conditionality which can be expressed in rules.

## 11.2 Related work

[Armengol and Plaza \(2005\)](#) describe an ontology-like, formal encoding of chemical structural features using *feature terms* which are based on the International Union for Pure and Applied Chemistry (IUPAC) nomenclature recommendations. Key to their approach is the representation of the main structural unit of a chemical entity plus the explicit representation of the additions and modifications to that structural unit. This is indeed similar to the approach taken in chemical nomenclature, where common structural units are given trivial or semisystematic ‘common’ names, and the names of more complicated molecules are built up from these parts by explicitly denoting the additions and modifications to the structural units which do have common names which constitute the entire molecule. This makes the resulting names far more memorable and comprehensible for humans than an approach based on fully systematic specification of all atoms and attachments within the molecule would be.

Armengol and Plaza specifically consider classifications based on hydrocarbon skeletons, number and type of ring systems, and functional groups. The feature terms with which they encode the structure of molecules for classificatory purposes consist of object names and relational ‘features’ which relate other objects to the objects being defined in different ways depending on the name of the feature. For example, they define the molecule ‘4-hexen-1-yne’, illustrated below, as

---

```
( define ( unsaturated-hydrocarbon 4-hexen-1-yne )  
  ( cyclic? false )  
  ( main-group hexane )  
  ( p-bonds ( define ( p-bond )  
    ( bond triple )  
    ( position one )  
  )  
    ( define ( p-bond )  
      ( bond double )  
      ( position four )  
    )  
  )  
)  
)
```

---

Here, the feature terms are ‘cyclic’, ‘main-group’, ‘p-bonds’, etc., each of which has a definition and takes values of a certain type only. The feature term ‘cyclic’ admits Boolean values, while the ‘main-group’ feature term takes a hydrocarbon as a value. Types are thus specified in terms of the features they have, and features can be inherited between types in a hierarchy (they include hydrocarbon as a child of organic molecule,

---

for example). Particularly interesting is their treatment of positional information about attachment of interesting parts to the ‘main-group’, or skeleton. They further provide for the specification of *relative* positions via the ‘relative-position’ feature term, in which the distance between two attachments is specified.

The authors used their encoding to correlate features of molecules with toxicology, and found that their approach compared favourably with existing approaches based on standard cheminformatics techniques<sup>2</sup>. However, they did not present a computational implementation for deriving feature terms from chemical structure specifications, and it is not clear to what extent automated reasoning over these non-standardised descriptions could be supported by currently available reasoning tools.

Within the context of the field of bioinformatics and the success of the Gene Ontology ([The Gene Ontology Consortium, 2000](#)) in assisting integration and interpretation of large scale bioinformatics data, the [ChEBI](#) ontology was developed by the European Bioinformatics Institute alongside their database of small molecules of biological interest ([de Matos \*et al.\*, 2010](#)). The [ChEBI](#) ontology provides a classification for all entities in [ChEBI](#) according to their physical composition and chemical structures. The relationships used in [ChEBI](#) include:

- *is a*, which is used to relate a more specialised entity to a more general one which subsumes it, such as L-alanine residue (CHEBI:46217) is a alanine residue (CHEBI:32441).
- *has part*, which is used to indicate the constituents of a composite entity, such as diclofenac sodium (CHEBI:4509) has part diclofenac(1-) (CHEBI:48311).
- *has role*, which is used to link a molecular entity to a role which it performs in some contexts, such as kanamycin A sulfate (CHEBI:6109) *has role* antibacterial drug (CHEBI:36047). (We will not consider the *has role* relationship further as it is not structure-based.)
- various chemical relationships including *is enantiomer of*, *is tautomer of*, and *acid/base* relations.

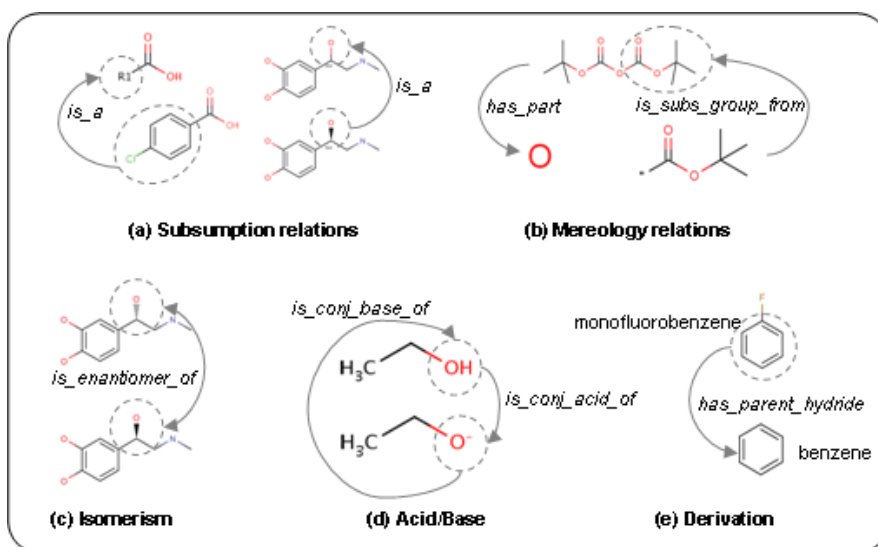
See Figure 11.3<sup>3</sup> for an illustration of the structural relations used in [ChEBI](#).

[Villanueva-Rosales and Dumontier \(2007\)](#) developed an ontology of chemical functional groups encoded in [OWL](#), which was able to be used to automatically link chemical

---

<sup>2</sup>For example, [QSAR](#) descriptors, which assign numeric values to algorithmically calculated properties of the structures of molecules which are believed to correlate with activity.

<sup>3</sup>Figure 11.3 first appeared in ([Hastings \*et al.\*, 2010](#)).



**Figure 11.3** Structural relations in the ChEBI ontology

entities defined by means of graphs to ontological classes defined by means of functional groups. They made use of a cheminformatics software application to extract functional groups from molecules with chemical structures specified in standard cheminformatics representation formalisms. While not as rich as the feature term representation described above, and not as large as the ChEBI ontology, the Dumontier representation allows powerful use to be made of the relatively compact knowledge base of functional groups in terms of the classification of arbitrary molecules in a defined hierarchy.

Cheminformatics tools and techniques do already exist to detect (for example) substructure relationships between different chemical graphs (a substructure is when one graph forms a wholly subsumed part of another graph), and graph isomorphism (which is when two graphs, differently depicted, actually represent the same molecule). Cheminformatics algorithms can also calculate many interesting structures and features of chemical entities represented as chemical graphs, such as charge, mass, and various “descriptors” which provide quantitative measures of certain aspects of the bioactivity of chemical entities.

However, no algorithms or tools yet exist which extensively interrelate cheminformatics algorithms, representation formalisms, and ontology, such that the former could provide input into the latter. One exception is the Dumontier ontology we described earlier, but that is limited to a fixed number of functional groups, providing only a limited vocabulary for defining chemical classes. Furthermore, it is clear that many of the elements of chemical class definitions that fall into this ‘gap’ area, rely on these

chemical graphs, or aspects (features) of graphs, which may even be expressed in terms of structural formulae.

## 11.3 Comparison of our research to previous efforts

Our results have shown that it is possible to create a chemical knowledge base using [OWL](#), description graphs and rules. The main strength of our approach is the direct encoding of complex structures at the class level in the ontology, and the encoding of rules for determining properties such as being cyclic, which are not able to be expressed as [OWL](#) axioms. We thereby show that this approach allows properties to be calculated by the reasoner rather than requiring these to be pre-computed and added to the asserted hierarchy of the ontology.

The main weaknesses are the limitations of rules for arbitrary property encoding and in particular the lack of quantification operators; and that there seems to be a scalability performance problem with using rules in this fashion. Pragmatically, the performance of the system was not where it would need to be to be able to handle the thousands or even millions of chemical graphs that are included in public databases. However, if ontologies are restricted to particular sub-domain areas of limited size, this might not be too much of a limitation.

[Armengol and Plaza \(2005\)](#) describe an ontology-like, formal encoding of chemical structural features using *feature terms*. Key to their approach is the representation of the main structural unit of a chemical entity and then the explicit representation of the additions and modifications to that structural unit. However, their knowledge base is not straightforwardly translatable into [OWL](#) and therefore it is not clear to what extent a comparison can be made in terms of conclusions that can be drawn with a reasoner. Many such frame-based representations of knowledge were found on closer investigation to be undecidable ([Baader and Nutt, 2007](#)), while one of the main strengths of [DLs](#) in general and [OWL-DL](#) in particular is their guarantee of decidability.

The [ChEBI](#) ontology is a well-known ontology for chemical entities, providing a deep classification according to the physical composition and chemical structure of chemical entities. While containing an ever-growing number of chemical entities, [ChEBI](#) is maintained entirely by hand, with no automated link between the structure of the chemicals captured in the chemical database and the structural definition of ontology classes. As a result, the [ChEBI](#) database has been able to grow at a much faster rate than

the [ChEBI](#) ontology, with the sizes currently<sup>4</sup> at around 550000 for the database and 22000 for the ontology. Chemical structures are exported into the ontology as *annotations* in the [InChI](#) format ([IUPAC, 2010](#)).

[Villanueva-Rosales and Dumontier \(2007\)](#) developed an [OWL](#) ontology for the classification of chemical compounds based on the presence of specified chemical functional groups. A key aspect of the approach was that tree-like expressions specified the necessary and sufficient conditions for functional groups such that the taxonomy of functional groups would be discovered on reasoning, thus reducing the burden of curating such an ontology. However, they were unable to express *arbitrary* structure at the class level due to the known limitations of [OWL](#).

Taking the desiderata of chemical ontology as the ability to center the knowledge base around an accurate representation of the structures of chemical entities, and to automatically determine the properties of chemical entities from those structures within the knowledge base, we find that the description graphs and rules extensions to [OWL](#) are a big step forward on the standard OWL language for this purpose.

---

<sup>4</sup>As of Release 69.





# 12

## Conclusion

In this chapter we summarise the dissertation, the results and the conclusions. We also state opportunities for future work.

For this research project we investigated the applicability of using [OWL](#), description graphs, and rules to implement a structure-enriched knowledge base for chemicals with classification based on the chemical structures and rules. This work contributes to the evaluation of new [OWL](#)-related technology towards the requirements of expert systems within the chemistry application domain. We gave an outline of background information both in terms of the domain of chemistry and in terms of knowledge representation and in particular [DLs](#). Our results showed that the approach is feasible and does allow classification based on certain properties of chemical entities, but significant limitations include the scalability of reasoning with the rules and the limitation on the kind of structure-based properties that can be tested for within the rule formalism.

Cheminformatics tools and techniques do already exist to detect chemical properties and subgraphs / graph isomorphisms, and the [CDK](#) ([Steinbeck \*et al.\*, 2006](#)) provides a well-developed open source library of such algorithms. These well-developed and optimised graph manipulation algorithms already in widespread use in the field of cheminformatics could provide input into the relatively new development of graph-enriched ontologies.

Future research in this area could investigate whether different representation strategies and/or rule implementations could alleviate the performance overhead in reasoning with the rules. Explicitly listed as out of scope for the purposes of this dissertation, another avenue of future research would be to implement a system to allow visualisation of the chemical description graphs being created within a visual ontology editing tool such as Protégé. The proof of concept we have presented here should be extended with

---

implementations for rules to determine several additional chemical properties. Finally, closer integration with the cheminformatics work which has already been done into the logic-based ontology domain might be achieved by the incorporation of a ‘chemical datatype’ into [OWL](#) based on [InChI](#) strings.

# Bibliography

- Armengol, A. and Plaza, E. (2005). An ontological approach to represent molecular structure information. In J. L. Oliviera, editor, *ISMBA 2005*, volume 3745 of *LNBI*, pages 294–304.
- Baader, F. and Nutt, W. (2007). *Basic Description Logics*, volume The Description Logic Handbook, chapter 2. Cambridge University Press, Cambridge, UK, 2nd edition.
- Baader, F., Küsters, R., and Wolter, F. (2007). *Extensions to Description Logics*, volume The Description Logic Handbook, chapter 5. Cambridge University Press, Cambridge, UK, 2nd edition.
- Beilstein (2010). Crossfire Beilstein Database.  
<http://info.crossfirebases.com/beilsteinacquisitionpressreleasemarch607.pdf>.
- Ben-Ari, M. (2001). *Mathematical Logic for Computer Science*. Springer-Verlag, London, Great Britain, 2nd edition.
- Blum, L. C. and Reymond, J.-L. (2009). 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, **131**, 8732–8733.
- Calvanese, D. and De Giacom, G. (2007). *Expressive Description Logics*, volume The Description Logic Handbook, chapter 5. Cambridge University Press, Cambridge, UK, 2nd edition.
- Carpi, A. (2003). Organic chemistry: An introduction. *Visionlearning*, **CHE-2**.  
[http://www.visionlearning.com/library/module\\_viewer.php?mid=60](http://www.visionlearning.com/library/module_viewer.php?mid=60).
- CAS (2010). Chemical Abstracts Service. <http://www.cas.org/>.
- CHEMBIOGRID (2010). CHEMBIOGRID: chemistry databases on the web.  
<http://www.chembiogrid.org/related/resources/about.html>.
- Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., and Sattler, U. (2008). OWL 2: The next step for OWL. *Journal of Web Semantics*, **4**, 309–322.
- Daylight, inc. (2010). <http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html>.

- de Matos, P., Alcántara, R., Dekker, A., Ennis, M., Hastings, J., Haug, K., Spiteri, I., Turner, S., and Steinbeck, C. (2010). Chemical Entities of Biological Interest: an update. *Nucl. Acids Res.*, **38**, D249–D254.
- Gasteiger, J. and Engel, T. (2003). *Chemoinformatics: A Textbook*. Wiley-VCH, Germany.
- Glimm, B., Horridge, M., Parsia, B., and Patel-Schneider, P. F. (2009). A syntax for rules in OWL 2. In *Proc. of OWL Experiences and Directions 2009 (OWLED 2009)*.
- 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*, pages 27–36, Sanibel Island, Florida, USA.
- Hastings, J. and The ChEBI Team (2010). The ChEBI tutorial.  
<http://www.ebi.ac.uk/chebi/tutorialForward.do>.
- Hastings, J., Batchelor, C., Steinbeck, C., and Schulz, S. (2010). What are chemical structures and their relations? In A. Galton and R. Mizoguchi, editors, *Proceedings of the 6th Formal Ontology in Information Systems conference*, Toronto, Canada.
- Hemmer, M. C. (2008). *Expert Systems in Chemistry Research*. CRC Press/Taylor and Francis Group, Boca Raton, FL.
- Horridge, M. and Bechhofer, S. (2009). The OWL API: A Java API for working with OWL 2 ontologies. In R. Hoekstra and P. F. Patel-Schneider, editors, *Proc. of OWL Experiences and Directions 2009 (OWLED 2009)*.
- Horrocks, I., Patel-Schneider, P. F., McGuinness, D. L., and Welty, C. A. (2007). *OWL: a Description Logic Based Ontology Language for the Semantic Web*, volume The Description Logic Handbook, chapter 14. Cambridge University Press, Cambridge, UK, 2nd edition.
- Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2010). SWRL: A semantic web rule language combining OWL and RuleML.  
<http://www.w3.org/Submission/SWRL/>.
- Indiana Cheminformatics Education Portal (2010). Representation of 2D structures on computers.  
<http://icep.wikispaces.com/Representation+of+2D+structures+on+computer>.
-

- IUPAC (2010). The IUPAC International Chemical Identifier (InChI).  
<http://www.iupac.org/inchi/>.
- Lesk, A. M. (1980). On the Gibbs paradox: what does indistinguishability really mean?  
*J. Phys. A: Math. Gen.*, **13**, L111–L114.
- Marx, V. (2009). Tear down this firewall: Pharma scientists call for a pre-competitive approach to bioinformatics. *GenomeWeb BioInform News*.  
<http://www.genomeweb.com/informatics/tear-down-firewall-pharma-scientists-call-pre-competitive-approach-bioinformatic?page=show>.
- McNaught, A. D. and Wilkinson, A. (1997). *IUPAC Compendium of Chemical Terminology, 2nd ed. (the "Gold Book")*. Blackwell Scientific Publications, Oxford. XML on-line corrected version: <http://goldbook.iupac.org> (2006-) created by M. Nic, J. Jirat, B. Kosata; updates compiled by A. Jenkins.
- MDL (2010). MDL. <http://www.mdl.com/company/about/history.jsp>.
- Motik, B., Sattler, U., and Studer, R. (2004). Query answering for OWL-DL with rules. In *Proc. ISWC-2004*, pages 549–563.
- Motik, B., Cuenca Grau, B., Horrocks, I., and Sattler, U. (2008a). Modeling ontologies using OWL, description graphs, and rules. In *Proc. of the 5th OWLED Workshop on OWL: Experiences and Directions*, Karlsruhe, Germany.
- Motik, B., Cuenca Grau, B., Horrocks, I., and Sattler, U. (2008b). Representing structured objects using description graphs. In *Proc. of the 11th Int. Joint Conf. on Principles of Knowledge Representation and Reasoning (KR 2008)*. AAAI Press.
- Motik, B., Cuenca Grau, B., and Sattler, U. (2008c). Structured objects in OWL: representation and reasoning. In *Proc. of the 17th International World Wide Web Conference (WWW 2008)*, Beijing, China. ACM.
- Murray-Rust, P., Rzepa, H. S., and Wright, M. (2001). Development of Chemical Markup Language (CML) as a system for handling complex chemical content. *New J. Chem.*, **2001**, 618–634.
- Nardi, D. and Brachman, R. J. (2007). *An introduction to Description Logics*, volume The Description Logic Handbook, chapter 1. Cambridge University Press, Cambridge, UK, 2nd edition.
-

- PDB (2010). The world wide protein data bank. <http://www.wwpdb.org/>.
- Rosse, C. and Jr, M. J. (2007). The foundational model of anatomy ontology. In A. Burger, D. Davidson, and R. Baldock, editors, *Anatomy Ontologies for Bioinformatics: Principles and Practice*, pages 59–117. Springer, London.
- Sayers, E. (2005). PubChem: An entrez database of small molecules. *NLM Tech Bull.*, **2005 Jan-Feb**.
- Shearer, R., Motik, B., and Horrocks, I. (2008). HermiT: A highly-efficient OWL reasoner. In C. Dolbear, A. Ruttenberg, and U. Sattler, editors, *Proceedings of the 5th Workshop on OWL: Experiences and Directions*, Karlsruhe, Germany.
- Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, **5**, 51–53.
- Smith, M. K., Welty, C., and McGuinness, D. L. (2010). The web ontology language. <http://www.w3.org/TR/owl-guide/>.
- Steinbeck, C., Han, Y., Kuhn, S., Horlacher, O., Luttmann, E., and Willighagen, E. (2003). The Chemistry Development Kit (CDK): An open-source Java library for chemo- and bioinformatics. *ChemInform*, **34**(21).
- Steinbeck, C., Hoppe, C., Kuhn, S., Floris, M., Guha, R., and Willighagen, E. L. (2006). Recent developments of the Chemistry Development Kit (CDK) - an open-source Java library for chemo- and bioinformatics. *Curr. Pharm. Des.*, **12**, 2111–2120.
- Sternberg, R. J. (2003). *Cognitive Psychology*. Thomson Wadsworth, Belmont, CA, USA, 3rd edition.
- The Gene Ontology Consortium (2000). Gene ontology: tool for the unification of biology. *Nat. Genet.*, **25**, 25–9.
- The Gene Ontology Consortium (2010). The OBO language, version 1.2. [http://www.geneontology.org/GO.format.obo-1\\_2.shtml](http://www.geneontology.org/GO.format.obo-1_2.shtml).
- Trinajstić, N. (1992). *Chemical graph theory*. CRC Press, Florida, USA.
- Tsarkov, D. and Horrocks, I. (2006). FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, pages 292–297. Springer.

- Vardi, M. Y. (1996). Why is modal logic so robustly decidable? In *Proc. DIMACS Workshop*, volume 31, pages 149–184.
- Villanueva-Rosales, N. and Dumontier, M. (2007). Describing chemical functional groups in OWL-DL for the classification of chemical compounds. In *Proc. of OWL: Experiences and Directions (OWLED 2007)*.
- Wikipedia (2010). Chemistry. <http://en.wikipedia.org/wiki/Chemistry>.
- Williams, A. (2008). ChemSpider and its expanding web: Building a structure-centric community for chemists. *Chemistry International*, **30**.
- Williams, A. J. (2009). A perspective of publicly accessible/open-access chemistry databases. *Drug Discovery Today*, **13**, 495–501.
- Zero, P., Plucinski, F., and Mazurek, A. P. (2009). Theoretical comparison of molecular properties of linear and cyclic glycine derived peptides and their phosphor analogues. *Journal of Molecular Structure: THEOCHEM*, **915**(1-3), 182 – 187.





# Appendices

---

---



## Ontology Syntax Formats

The following extract illustrates an [OWL](#) class definition in [RDF-XML](#) syntax (taken from an [OWL](#) version of [ChEBI](#)).

---

```
<owl:Class rdf:about="#CHEBI_9943">
  <rdfs:subClassOf rdf:resource="#CHEBI_23480"/>
  <rdfs:subClassOf rdf:resource="#CHEBI_32876"/>
  <FORMULA>C17H27NO2</FORMULA>
  <crossReference
    >PubMed citation:15109646</crossReference>
  <crossReference
    >KEGG COMPOUND accession:C07187</crossReference>
  <synonym xml:lang="en"
    >1-(2-(dimethylamino)-1-(4-methoxyphenyl)ethyl)cyclohexanol</synonym>
  <crossReference
    >PubMed citation:9871604</crossReference>
  <crossReference
    >PubMed citation:1976813</crossReference>
  <status>Checked</status>
  <MASS>277.40182</MASS>
  <crossReference
    >CAS Registry Number:93413-69-5</crossReference>
  <iupacName xml:lang="en"
    >1-[2-(dimethylamino)-1-(4-methoxyphenyl)ethyl]cyclohexanol</iupacName>
  <synonym xml:lang="en"
    >1-[2-Dimethylamino-1-(4-methoxy-phenyl)-ethyl]-cyclohexanol</synonym>
  <synonym xml:lang="en"
    >1-[2-Dimethylamino-1-(4-methoxy-phenyl)-ethyl]-cyclohexanol (HCl)</synonym>
  <synonym xml:lang="en"
    >1-[2-Dimethylamino-1-(4-methoxy-phenyl)-ethyl]-cyclohexanol (HCl)</synonym>
  <crossReference
    >Beilstein Registry Number:4234848</crossReference>
  <crossReference
    >PubMed citation:15990295</crossReference>
  <SMILES
    >COc1ccc(cc1)C(CN(C)C)C1(O)CCCC1</SMILES>
  <synonym xml:lang="en">EFFEXOR</synonym>
```

---

---

```

<InChIKey
  >InChIKey=PNVNVHUZROJLTJ-UHFFFAOYAJ</InChIKey>
<crossReference
  >PubMed citation:14640559</crossReference>
<crossReference
  >PubMed citation:14971904</crossReference>
<CHARGE>0</CHARGE>
<crossReference
  >DrugBank accession:DB00285</crossReference>
<synonym xml:lang="en">Elafax</synonym>
<crossReference
  >PubMed citation:10891117</crossReference>
<crossReference
  >PubMed citation:17725338</crossReference>
<InChI
  >InChI=1/C17H27NO2/c1-18(2)13-16(17(19)11-5-4-6-12-17)14-7-9-15
    (20-3)10-8-14/h7-10,16,19H,4-6,11-13H2,1-3H3</InChI>
<synonym xml:lang="en">VENLAFAXINE</synonym>
<synonym xml:lang="en">Venlafaxine</synonym>
<dc:name xml:lang="en">venlafaxine</dc:name>
<synonym xml:lang="en">venlafaxine</synonym>
<synonym xml:lang="es">venlafaxina</synonym>
<synonym xml:lang="fr">venlafaxine</synonym>
<synonym xml:lang="la">venlafaxinum</synonym>
</owl:Class>

```

---

The following illustrates the same concept expression in the more compact, Manchester [OWL](#) syntax:

---

```

Class: chebi_ontology:CHEBI_9943

SubClassOf:
  chebi_ontology:CHEBI_23480,
  chebi_ontology:CHEBI_32876

Facts:
  dc:name "venlafaxine"@en,
  chebi_ontology:CHARGE "0",
  chebi_ontology:FORMULA "C17H27NO2",
  chebi_ontology:InChI "InChI=1/C17H27NO2/c1-18(2)13-16(17(19)11-5-4-6-12-17)
    14-7-9-15(20-3)10-8-14/h7-10,16,19H,4-6,11-13H2,1-3H3",
  chebi_ontology:InChIKey "InChIKey=PNVNVHUZROJLTJ-UHFFFAOYAJ",
  chebi_ontology:MASS "277.40182",
  chebi_ontology:SMILES "COc1ccc(cc1)C(CN(C)C)C1(O)CCCCC1",
  chebi_ontology:crossReference "Beilstein Registry Number:4234848",
  chebi_ontology:crossReference "CAS Registry Number:93413-69-5",
  chebi_ontology:crossReference "DrugBank accession:DB00285",
  chebi_ontology:crossReference "KEGG COMPOUND accession:C07187",
  chebi_ontology:crossReference "PubMed citation:10891117",
  chebi_ontology:crossReference "PubMed citation:14640559",

```

---

---

```

chebi_ontology:crossReference  "PubMed citation:14971904",
chebi_ontology:crossReference  "PubMed citation:15109646",
chebi_ontology:crossReference  "PubMed citation:15990295",
chebi_ontology:crossReference  "PubMed citation:17725338",
chebi_ontology:crossReference  "PubMed citation:1976813",
chebi_ontology:crossReference  "PubMed citation:9871604",
chebi_ontology:iupacName
    "1-[2-(dimethylamino)-1-(4-methoxyphenyl)ethyl]cyclohexanol"@en,
chebi_ontology:status  "Checked",
chebi_ontology:synonym
    "1-(2-(dimethylamino)-1-(4-methoxyphenyl)ethyl)cyclohexanol"@en,
chebi_ontology:synonym
    "1-[2-Dimethylamino-1-(4-methoxy-phenyl)-ethyl]-cyclohexanol"@en,
chebi_ontology:synonym
    "1-[2-Dimethylamino-1-(4-methoxy-phenyl)-ethyl]-cyclohexanol (HCl)"@en,
chebi_ontology:synonym
    "1-[2-Dimethylamino-1-(4-methoxy-phenyl)-ethyl]-cyclohexanol (HCl)"@en,
chebi_ontology:synonym  "EFFEXOR"@en,
chebi_ontology:synonym  "Elafax"@en,
chebi_ontology:synonym  "VENLAFAXINE"@en,
chebi_ontology:synonym  "Venlafaxine"@en,
chebi_ontology:synonym  "venlafaxina"@es,
chebi_ontology:synonym  "venlafaxine"@en,
chebi_ontology:synonym  "venlafaxine"@fr,
chebi_ontology:synonym  "venlafaxinum"@la

```

---

---

---

# B

## Source Code Listing

In this appendix we provide relevant extracts from the source code used in the project implementation in Java to illustrate our solution to central problems. All source code is downloadable in the file <http://www.ebi.ac.uk/~hastings/owled2010/descgraphs.zip>.

### B.1 Conversion of chemical graphs

The following source code uses the [CDK](#) to parse chemical graphs expressed in MOLfile format and creates description graphs for them. This extract is taken from the file `MolfileToDGConverter.java` in the downloadable source code.

---

```
MDLV2000Reader reader = new MDLV2000Reader(new StringReader(molFile)) ;
ChemFile chemFile = (ChemFile) reader.read(new ChemFile());
IChemModel model = chemFile.getChemSequence(0).getChemModel(0);
IAtomContainer cont = model.getMoleculeSet().getAtomContainer(0);
try {
    AtomContainerManipulator.percieveAtomTypesAndConfigureAtoms(cont);
} catch (NullPointerException npe) {
    npe.printStackTrace();
}

List<OWLClass> atoms = new ArrayList<OWLClass>();
Map<Integer, OWLClass> bonds = new HashMap<Integer, OWLClass>();
Map<OWLClass, DescriptionGraph.Edge[]> bondsToEdges =
    new HashMap<OWLClass, DescriptionGraph.Edge[]>();

//Mapping of IAtomContainer to AtomicConcept vertices: atoms and bonds are each atomic concepts
AtomicConcept[] concepts = new
    AtomicConcept [cont.getAtomCount()+cont.getBondCount()+1];

//Mapping to edges: bonds
DescriptionGraph.Edge[] edges =
```

---



## B.1. CONVERSION OF CHEMICAL GRAPHS

---

```
new DescriptionGraph.Edge[(cont.getBondCount()*2)+cont.getAtomCount()];

//Start concepts: in this case just one for the molecule itself
Set<AtomicConcept> startConcepts = new HashSet<AtomicConcept>();
AtomicConcept molecule = AtomicConcept.create(
    DGOwlUtilities.ontologyIRI+"#molecule_"+title/*+"("+label+")"*/);
OWLClass clsMainConcept = dataFactory.getFactory().getOWLClass(
    IRI.create(DGOwlUtilities.ontologyIRI+"#molecule_"+title));
startConcepts.add(molecule);

int nodeIndex = 0;
Map<IAtom,Integer> atomIndexMap = new HashMap<IAtom, Integer>();
Map<IBond,Integer> bondIndexMap = new HashMap<IBond, Integer>();

//AtomicConcepts for atoms
while (nodeIndex < cont.getAtomCount()) {
    IAtom atom = cont.getAtom(nodeIndex);
    AtomicConcept atomicConcept =
        AtomicConcept.create(DGOwlUtilities.ontologyIRI
            +"#"+title+"_Atom_"+atom.getSymbol()+")"+nodeIndex);
    concepts[nodeIndex] = atomicConcept;
    atomIndexMap.put(atom,nodeIndex);
    nodeIndex++;

    IRI atomIri = IRI.create(atomicConcept.getIRI());
    OWLClass atomClass = dataFactory.getFactory().getOWLClass(atomIri);
    atoms.add(atomClass);           //save the vertex for later

    if (atom.getSymbol().equals("C")) {
        dataFactory.getManager().addAxiom(dataFactory.getOntology(),
            dataFactory.getFactory().getOWLSubClassOfAxiom(
                atomClass, dataFactory.getClsCarbonAtom()));
    } else if (atom.getSymbol().equals("N")) {
        dataFactory.getManager().addAxiom(dataFactory.getOntology(),
            dataFactory.getFactory().getOWLSubClassOfAxiom(
                atomClass, dataFactory.getClsNitrogenAtom()));
    } else if (atom.getSymbol().equals("O")) {
        dataFactory.getManager().addAxiom(dataFactory.getOntology(),
            dataFactory.getFactory().getOWLSubClassOfAxiom(
                atomClass, dataFactory.getClsOxygenAtom()));
    } else if (atom.getSymbol().equals("H")) {
        dataFactory.getManager().addAxiom(dataFactory.getOntology(),
            dataFactory.getFactory().getOWLSubClassOfAxiom(
                atomClass, dataFactory.getClsHydrogenAtom()));
    } else if (atom.getSymbol().equals("S")) {
        dataFactory.getManager().addAxiom(dataFactory.getOntology(),
            dataFactory.getFactory().getOWLSubClassOfAxiom(
                atomClass, dataFactory.getClsSulfurAtom()));
    } else if (atom.getSymbol().equals("Co")) {
        dataFactory.getManager().addAxiom(dataFactory.getOntology(),
            dataFactory.getFactory().getOWLSubClassOfAxiom(
                atomClass, dataFactory.getClsCobaltAtom()));
    } else if (atom.getSymbol().equals("Pt")) {
```

---

## B.1. CONVERSION OF CHEMICAL GRAPHS

---

```
dataFactory.getManager().addAxiom(dataFactory.getOntology(),
    dataFactory.getFactory().getOWLSubClassOfAxiom(
        atomClass, dataFactory.getClsPlatinumAtom()));
} else if (atom.getSymbol().equals("P") ) {
    dataFactory.getManager().addAxiom(dataFactory.getOntology(),
        dataFactory.getFactory().getOWLSubClassOfAxiom(
            atomClass, dataFactory.getClsPhosphorusAtom()));
} else if (atom.getSymbol().equals("Cl") ) {
    dataFactory.getManager().addAxiom(dataFactory.getOntology(),
        dataFactory.getFactory().getOWLSubClassOfAxiom(
            atomClass, dataFactory.getClsChlorineAtom()));
} else {
    dataFactory.getManager().addAxiom(dataFactory.getOntology(),
        dataFactory.getFactory().getOWLSubClassOfAxiom(
            atomClass, dataFactory.getClsAtom()));
}
}

//AtomicConcepts for bonds
while ( (nodeIndex-cont.getAtomCount()) < cont.getBondCount() ) {
    IBond bond = cont.getBond((nodeIndex-cont.getAtomCount()));
    AtomicConcept atomicConcept =
        AtomicConcept.create(DGowlUtilities.ontologyIRI
            +"#"+title+"_Bond_"+bond.getOrder()+"")+nodeIndex);
    concepts[nodeIndex] = atomicConcept;
    bondIndexMap.put(bond,nodeIndex);
    IRI bondIri = IRI.create(atomicConcept.getIRI());
    OWLClass bondClass = dataFactory.getFactory().getOWLClass(bondIri);
    bonds.put(nodeIndex, bondClass);        //save the vertex for later

//Bond order only covering single, double and triple
if (bond.getOrder().toString().equals("SINGLE") ) {
    dataFactory.getManager().addAxiom(dataFactory.getOntology(),
        dataFactory.getFactory().getOWLSubClassOfAxiom(
            bondClass, dataFactory.getClsSingleBond()));
} else if (bond.getOrder().toString().equals("DOUBLE") ) {
    dataFactory.getManager().addAxiom(dataFactory.getOntology(),
        dataFactory.getFactory().getOWLSubClassOfAxiom(
            bondClass, dataFactory.getClsDoubleBond()));
} else if (bond.getOrder().toString().equals("TRIPLE") ) {
    dataFactory.getManager().addAxiom(dataFactory.getOntology(),
        dataFactory.getFactory().getOWLSubClassOfAxiom(
            bondClass, dataFactory.getClsTripleBond()));
} else {
    dataFactory.getManager().addAxiom(dataFactory.getOntology(),
        dataFactory.getFactory().getOWLSubClassOfAxiom(
            bondClass, dataFactory.getClsBond()));
}

    nodeIndex++;
}

//AtomicConcept for molecule (DG) itself
concepts[nodeIndex] = molecule;        //add the molecule itself
```

---

```
//edges link ATOMS to BONDS; therefore TWO edges for each bond
int edgeIndex = 0;
for (int i=0; i < cont.getBondCount(); i++) {
    DescriptionGraph.Edge[] edgesForBond = new DescriptionGraph.Edge[2];

    IBond bond = cont.getBond(i);
    //atom(0) has bond with bond(i)
    edges[edgeIndex] = new DescriptionGraph.Edge(
        dataFactory.getHasBondWithRole(),
        atomIndexMap.get(bond.getAtom(0)),
        bondIndexMap.get(bond));
    edgesForBond[0] = edges[edgeIndex];
    edgeIndex++;
    //atom(1) has bond with bond(i)
    edges[edgeIndex] = new DescriptionGraph.Edge(
        dataFactory.getHasBondWithRole(),
        atomIndexMap.get(bond.getAtom(1)),
        bondIndexMap.get(bond));
    edgesForBond[1] = edges[edgeIndex];
    edgeIndex++;

    bondsToEdges.put( bonds.get(bondIndexMap.get(bond)) , edgesForBond);
}

//join the molecule to the atoms
for (int i=0; i<cont.getAtomCount(); i++) {
    edges[edgeIndex] = new DescriptionGraph.Edge(
        dataFactory.getHasAtomRole(),
        nodeIndex,
        i);          //one for each atom
    edgeIndex++;
}
```

---

## B.2 Creating the ontology

The following source code shows an extract of the code used to create the main ontology and add classes to it with the [OWL](#) API. The extract was taken from the file `DGOntologyCreator.java` in the downloadable source code.

---

```
//chemical entity
IRI classIRI = IRI.create(DGOwlUtilities.ontologyIRI + "#chemical_entity");
dataFactory.setClsChemEnt(factory.getOWLClass(classIRI));

OWLAnnotationProperty rdfLabel = factory.getRDFSLabel();
OWLAnnotation labelAnn = factory.getOWLAnnotation(rdfLabel,
    factory.getOWLStringLiteral("chemical entity", DGOwlUtilities.ENGLISH));
OWLAxiom ax = factory.getOWLAnnotationAssertionAxiom(classIRI, labelAnn);
```

---

```
manager.addAxiom(ontology, ax);
manager.addAxiom(ontology, factory.getOWLSubClassOfAxiom(dataFactory.
    getClsChemEnt(), factory.getOWLThing()));

//molecule
classIRI = IRI.create(DGOwlUtilities.ontologyIRI + "#molecule");
dataFactory.setClsMolecule(factory.getOWLClass(classIRI));

labelAnn = factory.getOWLAnnotation(rdfLabel, factory.
    getOWLStringLiteral("molecule", DGOwlUtilities.ENGLISH));
ax = factory.getOWLAnnotationAssertionAxiom(classIRI, labelAnn);
manager.addAxiom(ontology, ax);
manager.addAxiom(ontology, factory.getOWLSubClassOfAxiom(
    dataFactory.getClsMolecule(), dataFactory.getClsChemEnt()));

classIRI = IRI.create(DGOwlUtilities.ontologyIRI + "#test_molecule");
dataFactory.setClsTestMolecule(factory.getOWLClass(classIRI));

labelAnn = factory.getOWLAnnotation(rdfLabel, factory.getOWLStringLiteral(
    "classes for rule targets - no associated graphs", DGOwlUtilities.ENGLISH));
ax = factory.getOWLAnnotationAssertionAxiom(classIRI, labelAnn);
manager.addAxiom(ontology, ax);
manager.addAxiom(ontology, factory.getOWLSubClassOfAxiom(
    dataFactory.getClsTestMolecule(), dataFactory.getClsChemEnt()));

dataFactory.setClsCarboxylicAcid(factory.getOWLClass(
    IRI.create(DGOwlUtilities.ontologyIRI + "#molecule_" + 33575)));
labelAnn = factory.getOWLAnnotation(rdfLabel, factory.getOWLStringLiteral(
    "carboxylic acid", DGOwlUtilities.ENGLISH));
ax = factory.getOWLAnnotationAssertionAxiom(dataFactory.
    getClsCarboxylicAcid().getIRI(), labelAnn);
manager.addAxiom(ontology, ax);
manager.addAxiom(ontology, factory.getOWLSubClassOfAxiom(
    dataFactory.getClsCarboxylicAcid(), dataFactory.getClsTestMolecule()));

//cyclic entity
classIRI = IRI.create(DGOwlUtilities.ontologyIRI + "#cyclic_entity");
dataFactory.setClsCyclicEntity(factory.getOWLClass(classIRI));

labelAnn = factory.getOWLAnnotation(rdfLabel, factory.
    getOWLStringLiteral("cyclic entity", DGOwlUtilities.ENGLISH));
ax = factory.getOWLAnnotationAssertionAxiom(classIRI, labelAnn);
manager.addAxiom(ontology, ax);
manager.addAxiom(ontology, factory.getOWLSubClassOfAxiom(
    dataFactory.getClsCyclicEntity(), dataFactory.getClsTestMolecule()));
```

---

## B.3 Adding rules to the ontology

The following source code uses the [OWL](#) API to create a rule for the detection of cycles in description graphs. This extract is taken from the file `RuleGenerator.java` in the downloadable source code.

---

```
//MainConcept in this case is any molecule or group
//(i.e. chemical entity)
//we create a variable X3 for three length cycles, which
//is a child of any chemical entity
SWRLVariable varX = dataFactory.getFactory().getSWRLVariable(
    IRI.create("x"+j ));
SWRLClassAtom varXClassAtom = dataFactory.getFactory().
    getSWRLClassAtom(dataFactory.getClsMolecule(), varX);
bodyAtoms.add(varXClassAtom);

OWLObjectProperty hasAtomAsProperty = dataFactory.getFactory().
    getOWLObjectProperty(IRI.create(dataFactory.getHasAtomRole().getIRI()));
OWLObjectProperty hasBondWithProperty = dataFactory.getFactory().
    getOWLObjectProperty(IRI.create(dataFactory.getHasBondWithRole().getIRI()));

List<SWRLVariable> atomVars = new ArrayList<SWRLVariable>();
List<SWRLVariable> bondVars = new ArrayList<SWRLVariable>();

//Create atom and bond variables
for (int x=1; x<=j; x++) {
    //an atom 'is' an atom
    SWRLVariable varAtom = dataFactory.getFactory().getSWRLVariable(
        IRI.create("atom"+x ));
    SWRLClassAtom varAtomClassAtom = dataFactory.getFactory().
        getSWRLClassAtom(dataFactory.getClsAtom(), varAtom);
    bodyAtoms.add(varAtomClassAtom);
    SWRLDifferentIndividualsAtom differentFrom = dataFactory.
        getFactory().getSWRLDifferentIndividualsAtom(varAtom, varX);

    bodyAtoms.add(differentFrom);

    atomVars.add(varAtom);

    //the molecule 'has' the atom
    SWRLObjectPropertyAtom propertyAtom = dataFactory.getFactory().
        getSWRLObjectPropertyAtom(hasAtomAsProperty, varX, varAtom);
    bodyAtoms.add(propertyAtom);

    //the bond 'is' a bond
    SWRLVariable varBond = dataFactory.getFactory().getSWRLVariable(
        IRI.create("bond"+x ));
    SWRLClassAtom varBondClassBond = dataFactory.getFactory().
        getSWRLClassAtom(dataFactory.getClsBond(), varBond);
    bodyAtoms.add(varBondClassBond);
```

---

### B.3. ADDING RULES TO THE ONTOLOGY

---

```
        SWRLDifferentIndividualsAtom differentFrom2 = dataFactory.getFactory().
            getSWRLDifferentIndividualsAtom(varAtom, varBond);
        bodyAtoms.add(differentFrom2);

        bondVars.add(varBond);
    }

    //Now we created all atoms and bonds, link them up. First: all different
    for (int x=0; x<j; x++) {
        for (int y=x+1; y<j; y++) {
            SWRLVariable varAtom = atomVars.get(x);
            SWRLVariable varAtomOther = atomVars.get(y);

            SWRLDifferentIndividualsAtom differentFrom = dataFactory.
                getFactory().getSWRLDifferentIndividualsAtom(varAtom, varAtomOther);
            bodyAtoms.add(differentFrom);

            SWRLVariable varBond = bondVars.get(x);
            SWRLVariable varBondOther = bondVars.get(y);

            SWRLDifferentIndividualsAtom differentFromBonds = dataFactory.
                getFactory().getSWRLDifferentIndividualsAtom(varBond, varBondOther);
            bodyAtoms.add(differentFromBonds);
        }
    }

    //Now: Bonds
    for (int x=0; x<j; x++) {
        //has bond with the next atom in the chain
        SWRLVariable varAtom = atomVars.get(x);
        SWRLVariable varAtomBondedWith = (x==(j-1)?atomVars.get(0):atomVars.get(x+1));
        SWRLVariable varBond = bondVars.get(x);

        SWRLObjectPropertyAtom atomHasBondOne = dataFactory.getFactory().
            getSWRLObjectPropertyAtom(hasBondWithProperty, varAtom, varBond);
        SWRLObjectPropertyAtom atomHasBondTwo = dataFactory.getFactory().
            getSWRLObjectPropertyAtom(hasBondWithProperty, varAtomBondedWith, varBond);

        bodyAtoms.add(atomHasBondOne);
        bodyAtoms.add(atomHasBondTwo);
    }

    //THE CONCLUSION:

    //X is cyclic
    SWRLClassAtom varXCyclic = dataFactory.getFactory().getSWRLClassAtom(
        dataFactory.getClsCyclicEntity(), varX);
    headAtoms.add(varXCyclic);

    SWRLRule rule2 = dataFactory.getFactory().getSWRLRule(bodyAtoms, headAtoms);
    rules.add(rule2);
```

---

## B.4 Performing the reasoning

The following extract, taken from `MyHermitOWLReasoner.java`, shows how the ontology, together with the description graphs, is passed to the reasoner, extending the `HermiT` implementation.

---

```
public static OWLReasoner createHermitOWLReasoner(
    OWLOntology ontology, Configuration configuration,
    Set<MyDescriptionGraph> descriptionGraphs) {
    Set<DescriptionGraph> graphs = new HashSet<DescriptionGraph>();
    for (MyDescriptionGraph graph : descriptionGraphs) {
        graphs.add((DescriptionGraph) graph);
    }

    MyHermitOWLReasoner hermit = new MyHermitOWLReasoner(
        configuration, ontology.getOWLOntologyManager(),
        ontology, graphs, configuration.reasonerProgressMonitor) {
        protected OWLOntology m_rootOntology = null;
        protected boolean changed = false;

        private final OWLOntologyChangeListener ontologyChangeListener =
            new OWLOntologyChangeListener() {
                public void ontologiesChanged(List<? extends OWLOntologyChange> changes)
                    throws OWLException {
                    changed = true;
                }
            };

        protected void initOWLAPI(OWLOntology rootOntology) {
            m_rootOntology = rootOntology;
            m_rootOntology.getOWLOntologyManager().addOntologyChangeListener(
                ontologyChangeListener);
            super.initOWLAPI(rootOntology);
        }

        public void dispose() {
            m_rootOntology.getOWLOntologyManager().removeOntologyChangeListener(
                ontologyChangeListener);
            m_rootOntology = null;
            super.dispose();
        }

        public OWLOntology getRootOntology() {
            return m_rootOntology;
        }

        public void flush() {
            if (changed) {
                loadOntology(m_rootOntology.getOWLOntologyManager(), m_rootOntology, null);
                changed = false;
            }
        }
    };
}
```

---

```
};  
hermit.initOWLAPI(ontology);  
return hermit;  
}
```

---

The following extract, taken from file `DGOntologyReasoner.java` in the downloadable source code, shows how the results were collected from the reasoner and examined.

---

```
OWLReasoner reasoner = dataFactory.getReasoner();  
if (printResults) System.out.println("Starting reasoning");  
long startTime = System.currentTimeMillis();  
reasoner.prepareReasoner();  
long endTime = System.currentTimeMillis();  
int seconds = (int) (endTime - startTime)/1000;  
if (printResults) System.out.println("Reasoning took "+seconds  
    +" seconds. Ontology is consistent: "+reasoner.isConsistent());  
  
if (printResults) {  
    Node<OWLClass> equivNothing = reasoner.getEquivalentClasses(  
        dataFactory.getFactory().getOWLNothing());  
    System.out.println("Got "+equivNothing.getSize()+  
        " clses equivalent to owl:Nothing");  
    for (OWLClass cls : equivNothing.getEntities()) {  
        System.out.println("cls: "+cls+ " is equivalent to owl:Nothing");  
    }  
  
    NodeSet<OWLClass> organicChlds = reasoner.getSubClasses(  
        dataFactory.getClsCarbonEntity(), false);  
    System.out.println("Got "+organicChlds.getFlattened().size()+  
        " clses classified under carbon molecule");  
    for (OWLClass cls : organicChlds.getFlattened()) {  
        System.out.println("cls: "+cls+ " is a carbon molecule");  
    }  
  
    NodeSet<OWLClass> hcChlds = reasoner.getSubClasses(  
        dataFactory.getClsHydroCarbon(), false);  
    System.out.println("Got "+hcChlds.getFlattened().size()+  
        " clses classified under hydrocarbon molecule");  
    for (OWLClass cls : hcChlds.getFlattened()) {  
        System.out.println("cls: "+cls+ " is a hydrocarbon molecule");  
    }  
  
    NodeSet<OWLClass> cyclicChlds = reasoner.getSubClasses(  
        dataFactory.getClsCyclicEntity(), false);  
    System.out.println("Got "+cyclicChlds.getFlattened().size()+  
        " clses classified under cyclic molecule");  
    for (OWLClass cls : cyclicChlds.getFlattened()) {  
        System.out.println("cls: "+cls+ " is cyclic ");  
    }  
}
```

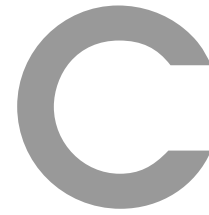
---



```
    }

    NodeSet<OWLClass> carboxylicChlds = reasoner.getSubClasses(
        dataFactory.getClsCarboxylicAcid(), false);
    System.out.println("Got "+carboxylicChlds.getFlattened().size()
        +" clses classified under carboxylic acid");
    for (OWLClass cls : carboxylicChlds.getFlattened()) {
        System.out.println("cls: "+cls+ " is a carboxylic acid (33575) ");
    }
}
```

---



## OWLED paper

In this appendix, we reproduce in full the accepted paper which was presented in the 2010 Web Ontology Language Experiences and Directions conference (OWLED 2010), which was held in San Francisco on the 21st and 22nd June 2010.

The slides which were used in the presentation are available at <http://www.slideshare.net/micheldumontier/owled-descgraphs>.

The proceedings were published in CEUR-WS, and the online version is available at [http://ceur-ws.org/Vol-614/owled2010\\_submission\\_13.pdf](http://ceur-ws.org/Vol-614/owled2010_submission_13.pdf).

The paper is reproduced here with original formatting.

# Representing Chemicals using OWL, Description Graphs and Rules

Janna Hastings<sup>1,2,3\*</sup>, Michel Dumontier<sup>4</sup>, Duncan Hull<sup>1</sup>, Matthew Horridge<sup>5</sup>,  
Christoph Steinbeck<sup>1</sup>, Ulrike Sattler<sup>5</sup>, Robert Stevens<sup>5</sup>, Tertia Hörne<sup>2</sup>, and  
Katarina Britz<sup>2,3</sup>

<sup>1</sup> European Bioinformatics Institute, UK

<sup>2</sup> University of South Africa

<sup>3</sup> Meraka Institute, South Africa

<sup>4</sup> Carleton University, Canada

<sup>5</sup> University of Manchester, UK

**Abstract.** Objects can be said to be structured when their representation also contains their parts. While OWL in general can describe structured objects, description graphs are a recent, decidable extension to OWL which support the description of classes of structured objects whose parts are related in complex ways. Classes of chemical entities such as molecules, ions and groups (parts of molecules) are often characterised by the way in which the constituent atoms of their instances are connected via chemical bonds. For chemoinformatics tools and applications, this internal structure is represented using chemical graphs. We here present a chemical knowledge base based on the standard chemical graph model using description graphs, OWL and rules. We include in our ontology chemical classes, groups, and molecules, together with their structures encoded as description graphs. We show how role-safe rules can be used to determine parthood between groups and molecules based on the graph structures and to determine basic chemical properties. Finally, we investigate the scalability of the technology used through the development of an automatic utility to convert standard chemical graphs into description graphs, and converting a large number of diverse graphs obtained from a publicly available chemical database.

**Key words:** chemistry, ontology, description graphs, rules

## 1 Introduction

Objects can be said to be structured when their representation also contains their parts. While OWL in general can describe structured objects, description graphs are a recent, decidable extension to OWL which support the description of classes of structured objects whose parts are related in complex ways [1–3].

Classes of chemical entities such as molecules, ions and groups (parts of molecules) are often characterised by the way in which the constituent atoms of

---

\* To whom correspondence should be addressed, [hastings@ebi.ac.uk](mailto:hastings@ebi.ac.uk).

their instances are connected via chemical bonds. For example, a cyclic hydrocarbon such as benzene is characterised as six carbon atoms, each of which is connected to two other carbon atoms in such a way that it forms a single cycle (or ring). For various cheminformatics applications, chemical structures are represented as chemical graphs, comprising of atoms as vertices and bonds as edges. These can be encoded as connection tables [4].

A classic chemoinformatic application is chemical classification by comparing all substructures such general descriptions are subsumed by more complex and refined substructures. The Web Ontology Language (OWL), as it currently stands, is incapable of representing the required complex structures, particularly cycles [7]. The chemical graph formalism has previously been reported as a candidate application for substructure classification using description graphs [5, 6].

In this paper, we present a method for transforming chemical graphs into description graphs, and apply this method to create an OWL knowledge base of chemical entities enhanced with the structures of the chemical entities as description graphs. We will consider to what extent the formalism of description graphs, together with rules for expressing conditionality, supports the type of reasoning which domain experts would expect from a structure-enhanced chemical knowledge base, such as classification based on chemical structures, and determination of chemical properties based on the structures. Finally, we assess the scalability of the technology by evaluating the times taken to reason over knowledge bases of varying sizes.

## 2 Background

### 2.1 OWL 2, Description Graphs, and Rules

OWL 2 [7] is the latest release of the Web Ontology Language (OWL) family of languages. While OWL provides an extensive collection of constructs for logic-based ontology development, decidability of reasoning problems—e.g., testing consistency of an ontology, satisfiability of classes or computing its inferred class hierarchy—is obtained by making sure that OWL has a *tree model property* [8]: in a nutshell, that means that every consistent ontology has a model, i.e., a state of affairs that satisfies all axioms in the ontology, whose relational structure looks like a tree. For this reason, OWL has not traditionally been able to describe arbitrarily structured objects, but only those which had structures which could be expressed in the shape of trees. *Description Graphs* are a formalism which has been introduced by Motik et al. [1–3] to address this weakness of OWL in representing structured objects, while still preserving the decidability of reasoning on ontologies containing such structured objects.

A description graph is a directed graph  $G = (V, E, \lambda)$  in which each vertex  $i \in V$  is labeled with a set of (possibly negated) class names  $\lambda(i)$ ; and each edge  $\langle i, j \rangle \in E$  is labeled with a set of atomic properties  $\lambda(i, j)$ . Each description graph has a *main class*, which indicates the object whose structure is being

modelled in the graph, and it is this main class that will be used to link to the remainder of the ontology that the description graph is a part of.

In order to preserve the decidability of reasoning, some important constraints must be observed within a graph-enhanced knowledge base [1]. For our purposes, the most significant of these is that the properties which are used in the description graphs (i.e. the graph edges) must not be referred to in the main ontology axioms, which is known as the *strong separation* requirement. The full set of properties in the knowledge base has thus to be separated into *tree properties* and *graph properties*. This provides a limitation in terms of the possibility for reasoning over the information encoded in the graphs, as the graph properties cannot be referred to in OWL axioms, an example of which might be `SubClassOf(has_atom only (CarbonAtom or HydrogenAtom)) HydrocarbonMol`.

Thus chemical classification must be expressed with rules. Further, these rules must be role-safe, that is, they must not refer simultaneously to properties used in the graphs and those used in the OWL ontology axioms.

A graph-extended OWL knowledge base is thus a 4-tuple  $K = (T, G, P, A)$  where  $T$  is a set of OWL class axioms,  $G$  is a set of description graphs,  $P$  is a set of rules, and  $A$  is a set of OWL assertions.  $T$  is allowed to refer only to tree properties,  $G$  and  $P$  are allowed to refer only to the graph properties, and  $A$  is allowed to refer to both graph and tree properties [1–3].

## 2.2 Chemical entities and graphs

At the molecular level, all of matter is composed of *atoms* of different kinds (such as Carbon and Oxygen) joined together through chemical bonds of different strengths. Covalent bonds (the strongest kind of chemical bond) join atoms together into composite units called *molecules*. Chemical entities are usually categorised into chemical classes by virtue of sharing common substructure or activity. An example of a chemical class is ‘carboxylic acids’, which groups together all molecules that share the important carboxy functional group and therefore hold the disposition to behave similarly in certain chemical reactions involving that group.

The structure of a molecule is nicely represented by a chemical graph, which describes the atomic connectivity within a molecule in terms of labelled nodes for the atoms or groups within the molecule, and labelled edges for the (usually covalent) bonds between the atoms or groups [4]. The chemical graph formalism is widely used in the field of cheminformatics to calculate many properties of chemical entities. Chemical graphs are encoded in a variety of standard formats, prominent among which is the MOLFile connection table-based format [9].

## 3 Methods

The purpose of our experiment is to evaluate the utility and scalability of description graphs and rules for the representation of, and reasoning over, chemi-

cal structures. The knowledge base<sup>6</sup> consists of i) a simple ontology describing classes pertaining to chemical entities, ii) auto-generated description graphs from structures in the ChEBI database, and iii) rules for structure-based classification. We used the HermiT [11] reasoner<sup>7</sup> for reasoning about the ontology, description graphs and rules [5]. ChEBI [12] was used as a source for chemical structures, which were parsed using the Chemical Development Kit [13].

Our evaluation criteria considers the following three aspects expected by domain experts:

- Can chemical entities be classified based on their substructures?
- Can basic chemical properties be determined from the description graphs?
- How scalable is the resulting knowledge base?

We now describe the structure of the implemented knowledge base.

### 3.1 Ontology

At the root of the ontology is the node ‘chemical entity’, beneath which are nodes for the primary division in kind of entity, namely ‘group’, ‘atom’, ‘molecule’, and ‘ion’. ‘Atom’ is further divided into the concrete types of atoms as per the periodic table, such as ‘carbon atom’ and ‘oxygen atom’.

An illustration of the overall structure of the core terms of the ontology is shown in Figure 1.

### 3.2 Description Graphs

Description graphs were automatically generated<sup>8</sup> from a MOLfile connection table format [9]. The standard MOLfile format consists of an atom table, which provides information about the atoms included in the molecule such as their types, and a bond table, which provides information about the bonds included in the molecule such as which atoms they connect and their order (single, double, etc.).

Each description graph consists of a vertex for the description graph *main class* which is a subclass of ‘molecule’ in the ontology, a vertex for each atom which is a subclass of the atom type e.g. ‘carbon atom’ in the ontology, and a vertex for each bond which is a subclass of the bond type in the ontology e.g. ‘single’. Each atom vertex is connected to the molecule by the *has\_atom* property. Atom vertices are associated with bonds with *has\_bond*. Figure 2 shows an illustration of the description graph for cyclobutane.

<sup>6</sup> Ontology available in two files, the main ontology at [http://www.ebi.ac.uk/~hastings/owled2010/chemistry\\_dgs\\_ontology.owl](http://www.ebi.ac.uk/~hastings/owled2010/chemistry_dgs_ontology.owl) and the graphs at [http://www.ebi.ac.uk/~hastings/owled2010/chemistry\\_dgs\\_graphs.owl](http://www.ebi.ac.uk/~hastings/owled2010/chemistry_dgs_graphs.owl)

<sup>7</sup> Version 1.2.2 with slight customisation for input and output of graphs which is currently only partially supported by the HermiT library and the OWL API.

<sup>8</sup> Our software for this experiment is available in source and binary at <http://www.ebi.ac.uk/~hastings/owled2010/descgraphs.zip>



**Fig. 1.** Core ontology structure

The vertices (but not the properties) of the description graphs are also classified in the main OWL ontology. The main class is classified beneath ‘molecule’ in the main ontology, the atoms beneath ‘atom’ and the bonds beneath ‘bond’.

### 3.3 Rules

We implemented rules to classify chemical structures based on their composition and their connectivity. Rules were devised for the classification of *cyclic* compounds, which contained a cycle of connected atoms.

For example, a rule to determine cycles of length three atoms is (slightly simplified for readability, the full generated version also includes `DifferentFrom` statements to ensure non-trivial cycles)

$$\begin{aligned}
 & \text{Molecule}(M) \wedge \\
 & \text{has\_atom}(M, A_1) \wedge \text{has\_atom}(M, A_2) \wedge \text{has\_atom}(M, A_3) \wedge \\
 & \text{Atom}(M, A_1) \wedge \text{Atom}(M, A_2) \wedge \text{Atom}(M, A_3) \wedge \\
 & \text{Bond}(M, B_1) \wedge \text{Bond}(M, B_2) \wedge \text{Bond}(M, B_3) \wedge \wedge \\
 & \text{has\_bond}(A_1, B_1) \wedge \text{has\_bond}(A_2, B_1) \wedge \\
 & \text{has\_bond}(A_2, B_2) \wedge \text{has\_bond}(A_3, B_2) \wedge \\
 & \text{has\_bond}(A_1, B_3) \wedge \text{has\_bond}(A_3, B_3) \wedge \\
 & \rightarrow \text{instanceOf}(M, \text{CyclicMolecule})
 \end{aligned} \tag{1}$$

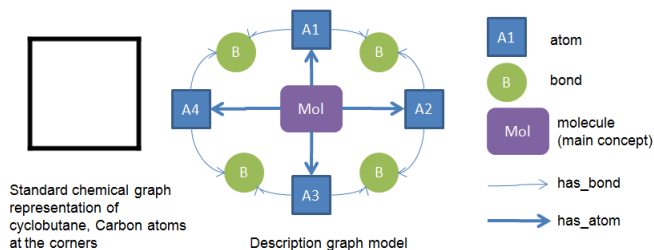


Fig. 2. Illustration of the cyclobutane description graph

Rules were also devised to determine *parthood* between chemical structures. If all atoms of A are atoms of B, and all bonds of A are bonds of B, then A is a *subgraph* of B. The term *group* is commonly used to denote arbitrary chemical parts, while the terms *molecule*, *ion* and so on refer to entire (complete) structures. Rules were devised for each group so as to identify these groups in the molecule. However, a consequence of the strong separation requirement is that a single rule cannot refer to both graph properties and tree properties. For this reason, even if we determine that a given graph is a subgraph of another graph, we cannot assert a relationship such as *has\_part* between the two main classes at the ontology level. A workaround for this is to create a class for every group, such that if the group's structure is a subgraph of the molecule's structure, then the molecule can be classified as belonging to that class. Rules for parthood determination are of the form

$$\begin{aligned}
 & \text{Molecule}(M) \wedge \\
 & \text{has\_atom}(M, A_1) \wedge \dots \wedge \text{has\_atom}(M, A_n) \wedge \\
 & \text{Atom}(A_1) \wedge \dots \wedge \text{Atom}(A_n) \wedge \\
 & \text{Bond}(B_1) \wedge \dots \wedge \text{Bond}(B_m) \wedge \\
 & \text{has\_bond}(A_{i1}, B_{j1}) \wedge \dots \wedge \text{has\_bond}(A_{in}, B_{jm}) \\
 & \rightarrow \text{instanceOf}(M, \text{Class})
 \end{aligned} \tag{2}$$

where M is an arbitrary individual of type *molecule*;  $A_1 - A_n$  are *group* atoms; bonds exist between the group atoms  $A_{i1} - A_{in}$  and  $A_{j1} - A_{jn}$ , and **Class** is a class the identity of which depends on the group used to generate the rule, for example 'carboxylic acid' for the 'carboxy group'.

The properties used as the graph edges (*has\_atom*, *has\_bond*) are available for use in the rules, as long as a rule does not mix graph properties with properties used in OWL axioms in the main ontology.

In the next section, we present the results of reasoning over the knowledge base.

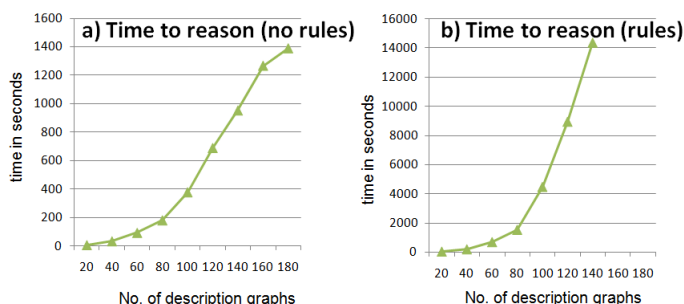


## 4 Results

Reasoning with the rules over the combined knowledge base resulted in classification of description graph-enriched classes as cyclic molecules and as classes containing specific defined groups such as carboxylic acids<sup>9</sup>. We find this result positive in terms of overcoming the previously explicit limitation of OWL knowledge bases in expressing arbitrarily structured objects at the class level and performing classification based on the structure.

However, we acknowledge that the types of conditionality that can be expressed in rules potentially provide the facility for only a limited set of chemical properties relative to those required by chemists. For example, it is difficult to express rules that must apply to *all* atoms from a given molecule’s graph without specifically naming those atoms, since there is no **forAll** operator in SWRL.

To evaluate the performance, we executed reasoning over iteratively increasing sizes of the knowledge base, both with and without rules. The results are summarised in Figure 3<sup>10</sup>. We do not attempt to control the size of the graphs which we randomly selected for inclusion into our knowledge base, but note that the average size of a molecule in the ChEBI database is around 30 atoms<sup>11</sup>.



**Fig. 3.** Performance results of classification

The scalability of the reasoner against the knowledge base enriched with description graphs appears workable, with reasoning time growing to a maximum of 23 minutes (1388 seconds) for a knowledge base enriched with 180 graphs. However, including the graphs alone – without rules – does not allow for any classification based on the information encoded in the graphs. The rules are this essential to expose the structure in the graphs to the reasoner. Unfortunately, we find that reasoning over the knowledge base enriched with graphs and rules

<sup>9</sup> The resulting inferred ontology is available at [http://www.ebi.ac.uk/~hastings/owled2010/chemistry\\_dgs.inferred.owl](http://www.ebi.ac.uk/~hastings/owled2010/chemistry_dgs.inferred.owl)

<sup>10</sup> Tested on a Dell twin core laptop.

<sup>11</sup> Excluding hydrogen atoms, which are commonly implicit, as these can be ‘added back’ by calculations to determine their predicted positions.

appears to grow very rapidly into unmanageable durations, with the highest duration that we recorded for a knowledge base enriched with 140 graphs taking four hours (14380 seconds) to classify. This scalability is affected dramatically by the number and complexity of the generated rules, therefore this would appear to be a limiting factor in following our approach in a more complete fashion, where many chemical properties and subgraph relations might be reasonably expected to be included in the same knowledge base.

## 5 Discussion

Our results have shown that it is possible to create a chemical knowledge base using OWL, description graphs and rules. The main strength of our approach is the direct encoding of complex structures at the class level in the ontology, and the encoding of rules for determining properties such as being cyclic, which are not able to be expressed as OWL axioms. We thereby show that this approach allows properties to be calculated by the reasoner rather than requiring these to be pre-computed and added to the asserted hierarchy of the ontology.

The main weaknesses are the limitations of rules for arbitrary property encoding and in particular the lack of quantification operators; and that there seems to be a scalability performance problem with using rules in this fashion. Pragmatically, the performance of the system was not where it would need to be to handle thousands or even millions of chemical graphs as are included in public databases. However, if ontologies are restricted to particular sub-domain areas of limited size, this might not be too much of a limitation.

Other approaches for partially including chemical structural information in knowledge bases have been described in recent years. Armengol and Plaza (2005) [14] describe an ontology-like, formal encoding of chemical structural features using *feature terms*. Key to their approach is the representation of the main structural unit of a chemical entity and then the explicit representation of the additions and modifications to that structural unit. However, their knowledge base is not straightforwardly translatable into OWL and therefore it is not clear to what extent a comparison can be drawn in terms of conclusions that can be drawn with a reasoner.

The ChEBI ontology is a well known ontology for chemical entities, providing a deep classification according to the physical composition and chemical structure of chemical entities. While containing an ever-growing number of chemical entities, ChEBI is maintained entirely by hand, with no automated link between the structure of the chemicals captured in the chemical database and the structural definition of ontology classes. As a result, the ChEBI database has been able to grow at a much faster rate than the ChEBI ontology, with the sizes currently<sup>12</sup> at around 550000 for the database and 22000 for the ontology. Chemical structures are exported into the ontology as *annotations* in the InChI [15] format.

---

<sup>12</sup> As of Release 69.

In 2007, Dumontier and Villaneuva-Rosales developed an OWL ontology for the classification of chemical compounds based on the presence of specified chemical functional groups [16]. A key aspect of the approach was that tree-like expressions specified the necessary and sufficient conditions for functional groups such that the taxonomy of functional groups would be discovered on reasoning (thus reducing the burden of curating such an ontology). However, they were unable to express *arbitrary* structure at the class level, and they therefore used SWRL rules to classify instances having more sophisticated structures such as cycles.

Taking the desiderata of chemical ontology as the ability to center the knowledge base around an accurate representation of the structures of chemical entities, and to automatically determine the properties of chemical entities from those structures within the knowledge base, we find that the description graphs and rules extensions to OWL are a big step forward on the standard OWL language for this purpose.

## 6 Conclusion

Our approach uses OWL, description graphs, and rules to implement a structure-enriched knowledge base for chemicals with classification based on the chemical structures and rules. We see this work as a contribution to the evaluation of new OWL-related technology towards the requirements of the chemistry application domain.

Cheminformatics tools and techniques do already exist to detect chemical properties and subgraphs / graph isomorphisms, and the CDK [13] provides a well-developed open source library of such algorithms. These well-developed and optimised graph manipulation algorithms already in widespread use in the field of cheminformatics could provide input into the relatively new development of graph-enriched ontologies.

Next steps will be to investigate whether different representation strategies and/or rule implementations could alleviate the performance overhead in reasoning with the rules; to implement a system to allow visualisation of the chemical description graphs being created; to extend the rules to determine several additional chemical properties; and to investigate the incorporation of a ‘chemical datatype’ into OWL based on InChI strings.

## Acknowledgements

We acknowledge the detailed comments from three anonymous reviewers whose input helped to significantly improve the final result. We further wish to acknowledge invaluable discussions and suggestions from Kirill Degtyarenko, Stefan Schulz, Colin Batchelor, and Birte Glimm. This work has been partially supported by the BBSRC, grant agreement number BB/G022747/1 within the “Bioinformatics and biological resources” fund.

## References

1. Motik, B., Cuenca Grau, B., and Sattler, U. (2008) Structured Objects in OWL: Representation and Reasoning. In Proc. of the 17th International World Wide Web Conference (WWW 2008), Beijing, China, 21-25 April 2008. ACM.
2. Motik, B., Cuenca Grau, B., Horrocks, I. and Sattler, U. (2008) Representing Structured Objects using Description Graphs. In Proc. of the 11th Int. Joint Conf. on Principles of Knowledge Representation and Reasoning (KR 2008), AAAI Press, 2008.
3. Motik, B., Cuenca Grau, B., Horrocks, I. and Sattler, U. (2008) Modeling Ontologies using OWL, Description Graphs, and Rules. In Proc. of the 5th OWLED Workshop on OWL: Experiences and Directions, Karlsruhe, Germany, October 26-27, 2008.
4. N. Trinajstić. (1992) Chemical Graph Theory. CRC Press, Florida, USA.
5. Glimm, B., Horridge, M., Parsia, B., Patel-Schneider, P.F. (2009). A Syntax for Rules in OWL 2. In Proc. of OWL Experiences and Directions 2009 (OWLED 2009).
6. Konyk, M., De Leon, A., Dumontier, M. (2008) Chemical Knowledge for the Semantic Web. 2008. Proceedings of Data Integration in the Life Sciences (DILS2008), Lecture Notes in Computer Science. LNBI 5109:169-176, Evry, France.
7. Grau, B. and Horrocks, I. and Motik, B. and Parsia, B. and Patel-Schneider, P. and Sattler, U (2008) OWL 2: The next step for OWL. In Journal of Web Semantics, 4:6 309–322.
8. Vardi, M. Y. (1996) Why Is Modal Logic So Robustly Decidable? In Proc. DIMACS Workshop, volume 31, pages 149-184, 1996.
9. <http://www.mdl.com/company/about/history.jsp>, last accessed April 2010.
10. Horridge, M. and Bechhofer, S. (2009). The OWL API: A Java API for Working with OWL 2 Ontologies. In Proc. of OWL Experiences and Directions 2009 (OWLED 2009), R. Hoekstra and P. F. Patel-Schneider, eds.
11. Shearer, R., Motik, B. and Horrocks, I. (2008) HermiT: A Highly-Efficient OWL Reasoner. In Dolbear, C., Ruttenberg, A. and Sattler, U. (Eds.), Proceedings of the 5th Workshop on OWL: Experiences and Directions, Karlsruhe, Germany, October 26-27, 2008.
12. de Matos, P.; Alcantara, R.; Dekker, A.; Ennis, M.; Hastings, J.; Haug, K.; Spiteri, I.; Turner, S.; and Steinbeck, C. (2010) Chemical Entities of Biological Interest: an update. Nucl. Acids Res. 2010 38: D249–D254.
13. Steinbeck C., Hoppe C., Kuhn S., Floris M., Guha R., Willighagen E.L. (2006) Recent Developments of the Chemistry Development Kit (CDK) - An Open-Source Java Library for Chemo- and Bioinformatics. Curr. Pharm. Des. 2006; 12(17):2111-2120.
14. Armengol, A. and Plaza, E. (2005) An ontological approach to represent molecular structure information. In J.L. Oliveira et al. (Eds.): ISMBA 2005, LNBI 3745, pp. 294-304, 2005.
15. <http://www.iupac.org/inchi/>, last accessed April 2010.
16. Villanueva-Rosales, N. and Dumontier, M. (2007) Describing chemical functional groups in OWL-DL for the classification of chemical compounds. OWL: Experiences and Directions (OWLED 2007).