D2.5.1 Specification of Coordination of Rule and Ontology Languages

Jeff Z.Pan (University of Manchester)
Enrico Franconi, Sergio Tessaris (Free University of Bozen-Bolzano)
Giorgos Stamou, Vassilis Tzouvaras (Centre for Research and Technology Hellas /Informatics and Telematics Institute)
Luciano Serafini (University of Trento)
Ian Horrocks, Birte Glimm (University of Manchester)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.5.1 (WP2.5)
We provide a unified framework in which the existing (and future) proposals of integrating different sorts of rule based language with the OWL DL Web ontology language can be compared. We are in particular interested in the axiom-based approach, as the SWRL proposal is a special case of it. We identify serval decidable sub-languages of SWRL with their complexity results and further explore several (including datatype predicate, fuzzy and context) extensions of SWRL.
Keyword list: description logics, ontology language, rule language, query language

Document Identifier	KWEB/2004/D2.5.1/v1.0
Project	KWEB EU-IST-2004-507482
Version	v1.0
Date	June 21, 2004
State	final
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

Centre for Research and Technology Hellas /Informatics and Telematics Institute
Free University of Bozen-Bolzano
Institut National de Recherche en Informatique et en Automatique
Learning Lab Lower Saxony
University of Manchester
University of Trento
Vrije Universiteit Amsterdam

Changes

Version	Date	Author(s)	Changes
0.1	21.04.04	Jeff Z Pan	creation
0.2	25.04.04	Ian Horrocks, Jeff Z Pan	adding Section 2.1: ORL
0.21	26.04.04	Ian Horrocks, Jeff Z Pan	renaming ORL as SWRL
0.22	29.04.04	Birte Glimm, Jeff Z Pan	adding Section 2.2: OWL-QL
0.3	11.05.04	Jeff Z Pan, Ian Horrocks	adding Chapter 3: A Predicate Extension
0.4	13.05.04	Luciano Serafini	adding Chapter 6: A Context Extension
0.5	14.05.04	Giorgos Stamou, Vassilis Tzouvaras, Jeff Z Pan	adding Chapter 5: A Fuzzy Extension
0.6	14.05.04	Enrico Franconi, Sergio Tessaris	adding Chapter 7: Rules and Queries in Ontologies: A Logical Framework
0.61	24.05.04	Enrico Franconi, Sergio Tessaris	Merging Chapter 2 and Chapter 7
0.7	25.05.04	Enrico Franconi, Sergio Tessaris	first final version of Rules and Queries in Ontologies: A Uniform Logical Framework
0.8	26.05.04	Jeff Z Pan	adding Chapter 1
1.0	11.06.04	Jeff Z Pan	adding Chapter 8

Executive Summary

In this report, we investigate the problem of combining ontologies with rule and query languages and provide a unified framework in which the existing (and future) proposals of integrating different sorts of rule based language with the OWL DL Web ontology language can be compared. Theorem 4 on page 11 show that under certain restrictions, the logical implication problem is equivalent in the three approaches described in Chapter 2.

We are in particular interested in the axiom-based approach, as the SWRL(Semantic Web Rule Language, developed by the Joint US/EU ad hoc Agent Markup Language Committee) proposal is a special case of it. As the logical implication in SWRL is obviously undecidable, we identify serval decidable sub-languages of SWRL with their complexity results.

We further explore several (including datatype predicate, fuzzy and context) extensions of SWRL, in order to meet various user requirements in the Semantic Web applications.

Contents

Chapter 1

Introduction

The need for integrating rules within the Semantic Web framework was clear since the early developments. However, up to the last few years, the research community focused its efforts on the design of the so called *Ontology Layer*. Nowadays, this layer is fairly mature in the form of Description Logics based languages such as the OWL Web Ontology Language [BvHH+04], which is now a W3C recommendation.

Although OWL adds considerable expressive power to the Semantic Web, it does have expressive limitations, particularly with respect to what can be said about properties. E.g., there is no composition constructor, so it is impossible to capture relationships between a composite property and another (possibly composite) property. The standard example here is the obvious relationship between the composition of the "parent" and "brother" properties and the "uncle" property. One way to address this problem would be to extend OWL with some form of "rules language".

In this chapter, we will first present some motivating examples to show the expressive power of rule languages. Then we will give a brief review on related work and describe the structure of the rest of this report.

1.1 Motivating Examples

In this section, some examples will be presented to illustrate what rules are as well as the power of rules. In these examples, we consider rules that have the form:

$$\mathsf{antecedent} \rightarrow \mathsf{consequent},$$

where both antecedent and consequent are conjunctions of atoms written $a_1 \wedge \ldots \wedge a_n$. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., $?x$). A formal definition of rules will be presented in Chapter 2.

Example 1 *A rule asserting that persons are younger than their parents could be written as follows:*

$$\mathsf{Person}(?p) \wedge \mathsf{dad}(?p, ?d) \wedge \mathsf{mum}(?p, ?m) \wedge \mathsf{age}(?p, ?pa) \wedge \mathsf{age}(?d, ?da) \wedge \mathsf{age}(?m, ?ma) \rightarrow <(?pa, ?da) \wedge <(?pa, ?ma),$$

where Person *is a class,* dad *and* mum *are object properties,* age *is a datatype property, and* $<$*, i.e. integer less-than, is a datatype predicate.* ♢

Rules with datatype predicates will be discussed in Chapter 5.

Note that rules with multiple atoms in the consequent could easily be transformed (via the Lloyd-Topor transformations [Llo87]) into multiple rules each with an atomic consequent. E.g., the rule given in Example 1 can be transformed into the following two rules:

$$\mathsf{Person}(?p) \wedge \mathsf{dad}(?p, ?d) \wedge \mathsf{mum}(?p, ?m) \wedge \mathsf{age}(?p, ?pa) \wedge \mathsf{age}(?d, ?da) \wedge \mathsf{age}(?m, ?ma) \rightarrow <(?pa, ?da),$$

$$\mathsf{Person}(?p) \wedge \mathsf{dad}(?p, ?d) \wedge \mathsf{mum}(?p, ?m) \wedge \mathsf{age}(?p, ?pa) \wedge \mathsf{age}(?d, ?da) \wedge \mathsf{age}(?m, ?ma) \rightarrow <(?pa, ?ma),$$

which assert that persons are younger than their dads and mums, respectively.

We called rules with an atomic consequent *atomic rules*.[1] Here is an example of an atomic rule.

Example 2 *The following rule asserts that one's parents' brothers are one's uncles:*

$$\mathsf{parent}(?x, ?p) \wedge \mathsf{brother}(?p, ?u) \rightarrow \mathsf{uncle}(?x, ?u),$$

where parent, brother *and* uncle *are all object properties.* ♢

Sometimes we need to specify a weight (with a truth value between 0 and 1) in an atom to represent the degree of confidence that an atom is true; we call rules containing atoms with weights *fuzzy rules*. Here is an example of a fuzzy rule.

Example 3 *The following rule asserts that if one's parents are rich in the degree of 0.8, then s/he is also rich:*

$$\mathsf{parent}(?x, ?p) \wedge \mathsf{Rich}(?p, 0.8) \rightarrow \mathsf{Rich}(?x),$$

where parent *is an object property,* Rich *is a class and 0.8 is the weight for atom* $\mathsf{Rich}(?p, 0.8)$. ♢

[1] cf. Chapter 2.

Fuzzy rules will be discussed in Chapter 6.

When we integrate a set of overlapping and heterogeneous ontologies, bridge rules between entities of different ontologies are often necessary. Here is an example of a bridge rule.

Example 4 *The following bridge rule asserts that* parents *(defined in ontology O_1) are* relatives *(defined in ontology O_2):*

$$1 : \mathsf{parent}(x, y) \xrightarrow{\sqsubseteq} 2 : \mathsf{relative}(x, y)$$

where parent *is an object property in the ontology O_1, while* relative *is an object property in the ontology O_2.*

We will introduce an extension of OWL that allows for bridge rules in chapter 7.

1.2 Related Work

In fact adding rules to Description Logic based knowledge representation languages is far from being a new idea. Several early Description Logic systems, e.g., Classic [PSMB+91, BPS94], included a rule language component. In these systems, however, rules were given a weaker semantic treatment than axioms asserting sub- and super-class relationships; they were only applied to individuals, and did not affect class based inferences such as the computation of the class hierarchy. More recently, the CARIN system integrated rules with a Description Logic in such a way that sound and complete reasoning was still possible [LR98]. This could only be achieved, however, by using a rather weak Description Logic (*much* weaker than OWL), and by placing severe syntactic restrictions on the occurrence of Description Logic terms in the (heads of) rules. Similarly, the DLP language proposed in [GHVD03] is based on the intersection of a Description Logic with horn clause rules; the result is obviously a decidable language, but one that is necessarily less expressive than either the Description Logic or rules language from which it is formed.

In recent years, more research has been devoted towards the integration of different sorts of rule based languages on top of the ontology layer provided by the OWL languages and in more general terms on top of a generic DL, and this work already produced some proposals for extending OWL languages. However, these proposals (see Chapter 2 for details) comes from different research communities, and often are difficult to compare because of the diverse underlying semantic assumptions.

In this report, we provide a unified framework in which the existing (and future) proposals can be compared. Using our framework, we show that – under the appropriate restrictions – there are strong correspondences among the proposals. This enables us to isolate interesting fragments of the proposed languages in which reasoning coincides.

Based on this framework, we further study the SWRL (Semantic Web Rule Language) developed by the Joint US/EU ad hoc Agent Markup Language Committee,[2] which is a special case of the axiom-based approach, by investigating several of its extensions, in order to meet various user requirements in Semantic Web applications.

1.3 Reader's Guide

The rest of this report is organised as follows: Chapter 2 provides a common framework for investigating the problem of combining ontologies with rule and query languages. Chapter 3 and 4 introduce two existing proposals of rule and query extensions of OWL, namely SWRL (Semantic Web Rule Language) and OWL-QL (OWL Query Language) respectively.

Chapter 5 introduces the datatype group approach to extend OWL datatyping with predicates and presents a predicate extension of SWRL, called SWRL-P, based on the datatype group approach.

Chapter 6 describes the basics of existing fuzzy set theory, presents a fuzzy extension for both OWL DL and SWRL and discuss how to provide reasoning support for the fuzzy extension of SWRL.

Chapter 7 proposes two alternative semantics for an OWL space (i.e., indexed set of OWL ontologies), presents a context extension of OWL and SWRL and discuss how to provide reasoning support for the context extension of OWL.

Chapter 8 concludes this report.

[2] See `http://www.daml.org/committee/` for the members of the Joint Committee.

Chapter 2

Rules and Queries with Ontologies: a Unified Logical Framework

2.1 Rule-extended Knowledge Bases

Let us consider a first-order function-free language with signature $\mathcal{A}$, and a description logic (DL) knowledge base Σ with signature subset of $\mathcal{A}$.

In this chapter we do not introduce any particular DL formalism. In our context, DL individuals correspond to constant symbols, DL atomic concepts and roles (and features) are unary and binary predicates in the case of a classical DL or the OWL language, and DL atomic n-ary relations correspond to predicates of arity n in the case of a $\mathcal{DLR}$-like DL. Note that the description logics we consider include those with datatypes (such as OWL DL [BvHH$^+$04]) or datatype expressions (such as OWL-E [PH04]).

A *term* is any constant in $\mathcal{A}$ or a variable symbol. If R is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms, $R(t_1, \ldots, t_n)$ is an *atom*, and an atom $R(t_1, \ldots, t_n)$ or a negated atom $\neg R(t_1, \ldots, t_n)$ are *literals*. A *ground* literal is a literal involving only constant terms. A set of ground literals is *consistent* if it does not contain an atom and its negation. If l is a literal, l or *not* l are *NAF-literals* (negation as failure literals). DL atoms, DL literals, and DL NAF-literals are atoms, literals, and NAF-literals whose predicates belong to the DL signature. A *rule* r may be of the forms:

$$
\begin{array}{rcll}
h_1 \wedge \ldots \wedge h_\ell & \leftarrow & b_1 \wedge \ldots \wedge b_m & \text{(classical rule)} \\
h_1 & :- & b_1 \wedge \ldots \wedge b_m \wedge \mathit{not}\ b_{m+1} \wedge \ldots \wedge \mathit{not}\ b_n & \text{(lp-rule)} \\
h_1 \wedge \ldots \wedge h_\ell & \Leftarrow & b_1 \wedge \ldots \wedge b_m & \text{(autoepistemic rule)}
\end{array}
$$

where $h_1, \ldots, h_\ell, b_1, \ldots, b_n$ are literals. Given a rule r, we denote by $H(r)$ the set $\{h_1, \ldots, h_\ell\}$ of *head* literals, by $B(r)$ the set of *body* literals $\{b_1, \ldots, b_n\}$, by $B^+(r)$ the set of *NAF-free* body literals $\{b_1, \ldots, b_m\}$, and by $B^-(r)$ the set of *NAF-negated* body literals $\{b_{m+1}, \ldots, b_n\}$. We denote by $\mathsf{vars}(\{l_1, \ldots, l_n\})$ the set of variables appearing

in the literals $\{l_1, \ldots, l_n\}$. The *distinguished variables* of a rule r are the variables that appear both in the head and in the body of the rule, i.e., $D(r) = \mathsf{vars}(H(r)) \cap \mathsf{vars}(B(r))$. A *ground rule* is a rule involving only ground literals. A rule is *safe* if all the variables in the head of the rule are distinguished. A *DL rule* is a rule with only DL literals. A set of literals is *tree-shaped* if its co-reference graph is acyclic; a co-reference graph includes literals and variables as nodes, and labelled edges indicate the positional presence of a variable in a literal. An *atomic* rule is a rule having a single literal in the head. A set of rules is *acyclic* if they are atomic and no head literal transitively depends on itself; a head literal h directly depends on a literal l if there is an atomic rule r with head h and with l part of the body $B(r)$. A set of rules is a *view* set of rules if each rule is atomic and no head literal belongs to the DL signature. A *rule-extended knowledge base* $\langle \Sigma, \mathcal{R} \rangle$ consists of a DL knowledge base Σ and a finite set $\mathcal{R}$ of rules.

2.2 The axiom-based approach

Let us consider a rule-extended knowledge base $\langle \Sigma, \mathcal{R} \rangle$ restricted to only classical rules.

Let I_Σ be a model of the description logics knowledge base Σ, i.e. $I_\Sigma \models \Sigma$. I is a model of $\langle \Sigma, \mathcal{R} \rangle$, written $I \models \langle \Sigma, \mathcal{R} \rangle$, if and only if I extends I_Σ with the interpretation of the non-DL predicates, and for each rule $r \in \mathcal{R}$ then

$$I \models \forall \mathbf{x}, \mathbf{y}. \exists \mathbf{z}. \left(\bigwedge B(r) \rightarrow \bigwedge H(r) \right)$$

where $\mathbf{x}$ are the distinguished variables of the rule $D(r)$, $\mathbf{y}$ are the non distinguished variables of the body $(\mathsf{vars}(B(r)) \setminus D(r))$, and $\mathbf{z}$ are the non distinguished variables of the head $(\mathsf{vars}(H(r)) \setminus D(r))$.

Let us define now the notion of logical implication of a ground literal l given a rule extended knowledge base: $\langle \Sigma, \mathcal{R} \rangle \models l$ if and only if $I \models l$ whenever $I \models \langle \Sigma, \mathcal{R} \rangle$. Note that the problems of DL concept subsumption and DL instance checking, and the problem of predicate inclusion (also called *query containment*) are all reducible to the problem of logical implication of a ground literal. Logical implication in this framework is undecidable, as it generalises the so-called *recursive CARIN* as presented in [LR98]. Logical implication in an axiom-based rule extended knowledge base remains undecidable even in the case of atomic negation-free safe DL rules with a DL having just the universal role constructor $\forall R.\, C$. Note that logical implication in an axiom-based rule extended knowledge base even with an empty TBox in Σ is undecidable (see, e.g., [BM02]).

In order to recover decidability, we reduce the expressivity of the approach in several ways; all the following restrictions disallow non DL predicates in the rules.

Theorem 1 *1. If we restrict the axiom-based approach to have only DL rules with tree shaped heads, tree shaped bodies and without negated atomic roles, the problem of*

logical implication in the rule extended knowledge base is NEXPTIME-complete with $\mathcal{ALCQI}$, OWL-Lite and OWL-DL as the underlying description logics knowledge base language.

2. *If in addition to the above conditions, constants are disallowed from the rules, the problem of logical implication in the rule extended knowledge base is EXPTIME-complete with any DL in EXPTIME (such as $\mathcal{ALCQI}$ or OWL-Lite) as the underlying description logics knowledge base language.*

3. *[LR98]: If we restrict the axiom-based approach to have only acyclic atomic negation-free safe DL rules with the $\mathcal{ALCNR}$ DL as the underlying description logics knowledge base language, the problem of logical implication is decidable in NEXPTIME.*

The SWRL proposal [HPS04] can be considered as a special case of the axiom-based approach presented above. SWRL uses OWL-DL or OWL-Lite as the underlying description logics knowledge base language (which admits data types), but it restricts the rule language to safe rules and without negated atomic roles. From the point of view of the syntax, SWRL rules are an extension of the abstract syntax for OWL DL and OWL Lite; SWRL rules are given an XML syntax based on the OWL XML presentation syntax; and a mapping from SWRL rules to RDF graphs is given based on the OWL RDF/XML exchange syntax. Logical implication in SWRL is still undecidable. The complexity results listed in Theorem 1 are applicable to SWRL as well.

Another way to make the axiom-based approach decidable is to reduce the expressivity of the DL, in order to disallow universal-like statements, while keeping rules cyclic.

In [LR98] it is shown that logical implication is decidable with atomic negation-free safe DL rules with the simple DL containing conjunction, disjunction, qualified existential, least cardinality and primitive negation.

In [CDGL$^+$04] a proposal is made of a very simple knowledge representation language, which captures the fundamental features of frame-based formalisms and of ontology languages for the semantic web; the precise definition of the language can be found in [CDGL$^+$04]. In this setting, it can be shown that the negation-free axiom-based approach is decidable, and the problem of logical implication of a ground literal is in EXPTIME, and it is polynomial in data complexity.

Conceptual graph rules [BM02] can be seen as a simple special case of an axiom-based rule extended knowledge base: CG-rules are negation-free, they do not have existential variables in the body, and Σ is TBox-free. Many decidable subclasses of CG-rules are special cases of the decidable cases presented above (but with Σ having a TBox); in particular, decidability of *range restricted CG-rules* is the TBox-free special case stated above [LR98] of atomic negation-free safe DL rules.

2.3 The DL-Log approach

Let us consider a rule-extended knowledge base $\langle \Sigma, \mathcal{R} \rangle$ where $\mathcal{R}$ is restricted to be a view set of lp-rules $\mathcal{P}$ (called a *program*).

The *non-DL Herbrand base* of the program $\mathcal{P}$, denoted by $\mathcal{HB}_{\mathcal{P}^-}$, is the set of all ground literals obtained by considering all the non-DL predicates in $\mathcal{P}$ and all the constant symbols from $\mathcal{A}$. An *interpretation* I wrt $\mathcal{P}$ is a consistent subset of $\mathcal{HB}_{\mathcal{P}^-}$. We say I is a *model* of a ground literal l wrt the knowledge base Σ, denoted $I \models_\Sigma l$, if and only if

- $l \in I$, when $l \in \mathcal{HB}_{\mathcal{P}^-}$
- $\Sigma \models l$, when l is a DL literal

We say that I is a model of a ground rule r, written $I \models_\Sigma r$, if and only if $I \models_\Sigma H(r)$ whenever $I \models_\Sigma b$ for all $b \in B^+(r)$, and $I \not\models_\Sigma b$ for all $b \in B^-(r)$. We denote with $\mathsf{ground}(\mathcal{P})$ the set of rules corresponding to the grounding of $\mathcal{P}$ with the constant symbols from $\mathcal{A}$. We say that I is a model of a rule-extended knowledge base $\langle \Sigma, \mathcal{P} \rangle$ if and only if $I \models_\Sigma r$ for all rules $r \in \mathsf{ground}(\mathcal{P})$; this is written as $I \models \langle \Sigma, \mathcal{P} \rangle$.

Let us define now the notion of logical implication of a ground literal l given a rule extended knowledge base: $\langle \Sigma, \mathcal{P} \rangle \models l$ if and only if $I \models_\Sigma l$ whenever $I \models \langle \Sigma, \mathcal{P} \rangle$. In the case of a NAF-free program, as well in the case of a program with stratified NAF negation, it is possible to adapt the standard results of datalog, which say that in these cases logical implication can be reduced to model checking in the (canonical) minimal model. So, if $I^{\mathcal{P}}_m$ is the minimal model of a NAF-free or stratified program $\mathcal{P}$, then $\langle \Sigma, \mathcal{P} \rangle \models l$ if and only if $I^{\mathcal{P}}_m \models_\Sigma l$.

In the case of an unrestricted program $\mathcal{P}$, an answer set semantics can be adopted to characterise logical implication. In this chapter we do not define the semantics of unrestricted rule extended knowledge bases; for a precise account, please refer to [Ros99, ELST04].

Theorem 2 *[ELST04]: The combined complexity of logical implication in a rule extended knowledge base with an EXPTIME-complete description logic (like, e.g., $\mathcal{ALCQI}$ or OWL-lite) is EXPTIME-complete in the case of NAF-free or stratified programs and it is NEXPTIME-complete in the unrestricted case. In a rule extended knowledge base with a NEXPTIME-complete description logic (like, e.g., $\mathcal{ALCQIO}$ or OWL-DL) the complexity is NEXPTIME-complete in the case of NAF-free programs and it is NP^{NEXP}-complete in the case of stratified programs and in the unrestricted case as well.*

In addition, it is possible to prove that the problem of logical implication of a DL literal in a rule extended knowledge base is independent on the presence of the program P. This means that the DL knowledge base is unaffected by the rule system, which can be seen as built on top of the DL knowledge base.

The DL-Log approach was first introduced with AL-Log. The AL-Log approach [DLNS98] is as a restriction of DL-Log. In fact, in AL-Log only view negation-free safe rules, whose DL predicates are only unary, with the $\mathcal{ALC}$ DL, are allowed. The complexity of logical implication is shown to be in NEXPTIME. [Ros99] extended AL-Log by allowing any DL predicate in the body of the rules. [ELST04] introduced DL-Log in the way we are presenting here.

An extension of DL-Log is the one where the recursive program is given a fixpoint semantics, which involves all individuals in the model, not only the ones in the Herbrand universe. In this extension, logical implication is undecidable with any DL having the ability to state at least atomic inclusion axioms between concepts [CR03]. It can be shown that, in the fixpoint based semantics, the DL-Log approach can be reconstructed by adding, for each rule, a special non-DL unary $top_{\mathcal{HB}}$ atom for each variable appearing in each DL literal of the rule, thus constraining the DL variables to be in the Herbrand universe anyway. Note also that in the case of acyclic rules, the fixpoint semantics coincide with the axiom-based semantics.

It is worthwhile mentioning at the end of this section three additional recent works that relate DLs with lp-rules: DLP [GHVD03] and [HMS04, Swi04]. In these papers it is shown how to *encode* the reasoning problem of a DL into a pure logic programming setting, i.e., into a rule extended knowledge base with a Σ without TBox. In the case of DLP, this is accomplished by encoding a severely restricted DL into a NAF-free negation-free DL program. In the two latter approaches, the full power of disjunctive logic programming is needed to perform the encoding of quite expressive DLs, at the cost of an exponential blow-up in space of the encoding.

2.4 The autoepistemic approach

Let us consider a rule-extended knowledge base restricted to autoepistemic rules.

Let I_Σ be a model, over the non empty domain Δ, of the description logics knowledge base Σ, i.e. $I_\Sigma \models \Sigma$. Let's define a variable assignment α in the usual way as a function from variable symbols to elements of Δ. A model of $\langle \Sigma, \mathcal{R} \rangle$ is a non empty set M of interpretations I, each one extending a DL model I_Σ with some interpretation of the non-DL predicates, such that for each rule r and for each assignment α for the distinguished variables of r the following holds:

$$\Big(\forall I \in M.\, I, \alpha \models \exists \mathbf{x}.\, \bigwedge B(r)\Big) \rightarrow \Big(\forall I \in M.\, I, \alpha \models \exists \mathbf{y}.\, \bigwedge H(r)\Big)$$

where $\mathbf{x}$ are the non distinguished variables of the body $(\mathsf{vars}(B(r)) \setminus D(r))$, and $\mathbf{y}$ are the non distinguished variables of the head $(\mathsf{vars}(H(r)) \setminus D(r))$.

Let us define now the notion of logical implication of a ground literal l given a rule extended knowledge base: $\langle \Sigma, \mathcal{R} \rangle \models l$ if and only if

$$\forall M.\,(M \models \langle \Sigma, \mathcal{R} \rangle) \rightarrow \forall I \in M.\,(I \models l)$$

The autoepistemic approach was first introduced by [DLN+98], with the goal of formalising the *constraint rules* implemented in many practical DL systems. Such rules, in fact, are simple to implement since they influence ABox reasoning, but leave TBox reasoning unaffected. These rules are also the basis of the recent formalisations of peer-to-peer systems [FKLS03]. As shown in [FKLS03], the autoepistemic semantics as defined above is equivalent to the context-based semantics of [GS98], and to the use of the autoepistemic operator, as defined, e.g., in [Rei92]. Using the results in [Mar99, GKWZ03], we can show that logical implication is decidable in the case of a rule extended knowledge base with DL rules with tree shaped body and heads, with the $\mathcal{ALC}$ DL; the precise complexity bounds are still unknown.

2.5 Queries

We now introduce the notion of a query to a rule extended knowledge base, that includes a DL knowledge base, a set of rules, and some facts.

Definition 3 *A* query *to a rule extended knowledge base is a (possibly ground) literal* $q_{\mathbf{x}}$ *with variables* $\mathbf{x}$ *(possibly empty). The* answer set *of* $q_{\mathbf{x}}$ *is the set of substitutions of* $\mathbf{x}$ *with constants* $\mathbf{c}$ *from* $\mathcal{A}$*, such that the grounded query is logically implied by the rule extended knowledge base, i.e.,*

$$\{\mathbf{c} \textit{ in } \mathcal{A} \mid \langle \Sigma, P \rangle \models q_{[\mathbf{x}/\mathbf{c}]}\}.$$

This definition of a query is based on the notion of *certain answer* in the literature and it is very general. Given a Σ, we define a *query rule* over Σ as a set of view rules together with a query literal selected from some head. In this way we capture the notion of a complex query expressed by means of a set of rules on top of an ontology.

The definition of query given above encompasses the different proposals of querying a DL knowledge base that have appeared in the literature. An important special case of a query rule is with view acyclic DL axiom-based rules, which is better known as a *conjunctive query* if each head literal appears only in one head, or *positive query* otherwise. Quite importantly, this restriction includes the seminal body of work on query answering with conjunctive queries (or with positive queries) with the very expressive $\mathcal{DLR}$ description logic (which includes $\mathcal{ALCQI}$) summarised in [CDGL00]. In this context, logical implication is EXPTIME-hard and in 2EXPTIME; in the case of a fixed finite domain (*closed domain assumption*) logical implication becomes coNP-complete in data complexity [CDGL00]. Practical algorithms for query answering have been studied in [THG02]. A proposal targeted towards the semantic web languages has been presented in [HT02].

Recently, the Joint US/EU ad hoc Agent Markup Language Committee has proposed an OWL query language called OWL-QL [FHH03], as a candidate standard language, which is a direct successor of the DAML Query Language (DQL). The query language is not fully formally specified, however it can be easily understood as allowing for conjunctive queries with distinguished variables (called *must-bind* variables) and non distinguished variables (called *don't-bind* variables). In addition, *may-bind* variables apparently provide the notion of a *possible* answer as opposed to the *certain* answer which has been adopted in this chapter. Query premises of OWL-QL allow to perform a simple form of local conditional query; this could be encoded as *assertions in DL queries* as introduced in [ELST04].

2.6 Comparing the three approaches

We first show in this section the conditions under which the three approaches coincide. This corresponds essentially to the case of negation-free view rule-extended knowledge bases with empty TBoxes.

Theorem 4 *If we restrict a rule extended knowledge base with classical rules to view negation-free DL rules with TBox-free Σ, a rule extended knowledge base with lp-rules to NAF-free negation-free DL programs with TBox-free Σ, and a rule extended knowledge base with autoepistemic rules to view negation-free DL rules with TBox-free Σ, the semantics of the rule extended knowledge base with classical rules, with lp-rules, and with autoepistemic rules coincide, i.e., the logical implication problem is equivalent in the three approaches.*

The above theorem is quite strict and it fails as soon as we release some assumption. We will show this by means of few examples. Consider the following knowledge base Σ, common to all the examples:

$$\texttt{is-parent} \doteq \exists \texttt{is-parent-of}$$
$$\texttt{my-thing} \doteq \texttt{is-parent} \sqcup \neg \texttt{is-father}$$
$$\texttt{is-parent-of(john, mary)}$$
$$\texttt{is-parent(mary)}$$

where we define, using standard DL notation, a TBox with the `is-parent` concept as anybody who is parent of at least some other person, and the concept `my-thing` as the union of `is-parent` and the negation of `is-father` (this should become equivalent to the top concept as soon as `is-father` becomes a subconcept of `is-parent`); and an ABox where we declare that John is a parent of Mary, and that Mary is parent of somebody.

Consider the following query rules, showing the effect of existentially quantified individuals coming from some TBox definition:

$\mathtt{Q_{ax}(x)} \leftarrow \mathtt{is\text{-}parent\text{-}of(x,y)}$
$\mathtt{Q_{lp}(x)} \text{ :- } \mathtt{is\text{-}parent\text{-}of(x,y)}$
$\mathtt{Q_{ae}(x)} \Leftarrow \mathtt{is\text{-}parent\text{-}of(x,y)}$

The query $\mathtt{Q_{ax}(x)}$ returns $\{\mathtt{john,\ mary}\}$; the query $\mathtt{Q_{lp}(x)}$ returns $\{\mathtt{john}\}$; the query $\mathtt{Q_{ae}(x)}$ returns $\{\mathtt{john,\ mary}\}$.

Consider now the query rules, which shows the impact of negation in the rules:

$\mathtt{Q_{ax}(x,y)} \leftarrow \neg\mathtt{is\text{-}parent\text{-}of(x,y)}$
$\mathtt{Q_{lp}(x,y)} \text{ :- } \neg\mathtt{is\text{-}parent\text{-}of(x,y)}$
$\mathtt{Q_{ae}(x,y)} \Leftarrow \neg\mathtt{is\text{-}parent\text{-}of(x,y)}$

The query $\mathtt{Q_{ax}(mary,\ john)}$ returns false; the query $\mathtt{Q_{lp}(mary,\ john)}$ returns true; the query $\mathtt{Q_{ae}(mary,\ john)}$ returns false.

Consider now the following alternative sets of rules, which show that autoepistemic rules, unlike the axiom-based ones, do not influence TBox reasoning:

$\mathtt{is\text{-}parent(x)} \leftarrow \mathtt{is\text{-}father(x)}$
$\mathtt{Q_{ax}(x)} \leftarrow \mathtt{my\text{-}thing(x)}$

$\mathtt{is\text{-}parent(x)} \Leftarrow \mathtt{is\text{-}father(x)}$
$\mathtt{Q_{ae}(x)} \Leftarrow \mathtt{my\text{-}thing(x)}$

In the first axiom-based case, the query $\mathtt{Q_{ax}(paul)}$ returns true; in the second autoepistemic case the query $\mathtt{Q_{ae}(paul)}$ returns false (we assume that `paul` is an individual in Σ).

Chapter 3

SWRL 0.5

Now that the OWL Web Ontology Language [BvHH+04] is a standard recommendation of the World Wide Web Consortium (W3C)[1], techniques to extend the use of OWL ontologies come more into focus. In this chapter, we present a concrete proposal for a rule extension of OWL.

The Joint US/EU ad hoc Agent Markup Language Committee[2] has proposed an OWL rule language called SWRL (Semantic Web Rule Language), a Horn clause rules extension to OWL.In this deliverable, we base our investigation of Semantic Web rule and query languages on SWRL 0.5 (or SWRL for short).[3] SWRL extends OWL in a syntactically and semantically coherent manner: the basic syntax for SWRL rules is an extension of the abstract syntax for OWL DL and OWL Lite; SWRL rules are given formal meaning via an extension of the OWL DL model-theoretic semantics; SWRL rules are given an XML syntax based on the OWL XML presentation syntax; and a mapping from SWRL rules to RDF graphs is given based on the OWL RDF/XML exchange syntax.

3.1 Abstract Syntax

The syntax for SWRL given in this section abstracts from any exchange syntax for OWL and thus facilitates access to and evaluation of the language. This syntax extends the abstract syntax of OWL described in the OWL Semantics and Abstract Syntax document [PSHH03a].

Names in the abstract syntax are RDF URI references [KC03]. These names may be abbreviated into qualified names, using one of the following namespace names:

[1] `http://www.w3.org`
[2] See `http://www.daml.org/committee/` for the members of the Joint Committee.
[3] `http://www.daml.org/2003/11/swrl/`

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

From the OWL Semantics and Abstract Syntax document [PSHH03a], an OWL ontology in the abstract syntax contains a sequence of annotations, axioms, and facts. Axioms may be of various kinds, for example, subClass axioms and equivalentClass axioms. SWRL extends axioms to also allow rule axioms, by adding the production:

axiom ::= rule

Thus a SWRL ontology could contain a mixture of rules and other OWL DL constructs, including ontology annotations, axioms about classes and properties, and facts about OWL individuals, as well as the rules themselves.

A rule axiom consists of an antecedent (body) and a consequent (head), each of which consists of a (possibly empty) set of atoms. Just as for class and property axioms, rule axioms can also have annotations. These annotations can be used for several purposes, including giving a label to the rule by using the rdf:label annotation property.

rule ::= 'Implies(' { annotation } antecedent consequent ')'
antecedent ::= 'Antecedent(' { atom } ')'
consequent ::= 'Consequent(' { atom } ')'

Informally, a rule may be read as meaning that if the antecedent holds (is "true"), then the consequent must also hold. An empty antecedent is treated as trivially holding (true), and an empty consequent is treated as trivially not holding (false). Non-empty antecedents and consequents hold iff all of their constituent atoms hold. As mentioned above, rules with multiple consequents could easily transformed (via the Lloyd-Topor transformations [Llo87]) into multiple rules each with a single atomic consequent.

Atoms in rules can be of the form C(x), P(x,y), Q(x,z), sameAs(x,y) or differentFrom(x,y), where C is an OWL DL description, P is an OWL DL *individual-valued* Property, Q is an OWL DL *data-valued* Property x,y are either variables or OWL individuals, and z is either a variable or an OWL data value. In the context of OWL Lite, descriptions in atoms of the form C(x) may be restricted to class names.

atom ::= description '(' i-object ')'
| individualvaluedPropertyID '(' i-object i-object ')'
| datavaluedPropertyID '(' i-object d-object ')'
| sameAs '(' i-object i-object ')'
| differentFrom '(' i-object i-object ')'

Informally, an atom C(x) holds if x is an instance of the class description C, an atom P(x,y) (resp. Q(x,z)) holds if x is related to y (z) by property P (Q), an atom sameAs(x,y) holds if x is interpreted as the same object as y, and an atom differentFrom(x,y) holds if x and y are interpreted as different objects.

Atoms may refer to individuals, data literals, individual variables or data variables. Variables are treated as universally quantified, with their scope limited to a given rule. As usual, only variables that occur in the antecedent of a rule may occur in the consequent (a condition usually referred to as "safety"). This safety condition does not, in fact, restrict the expressive power of the language (because existentials can already be captured using OWL someValuesFrom restrictions).

i-object ::= i-variable | individualID
d-object ::= d-variable | dataLiteral

i-variable ::= 'I-variable(' URIreference ')'
d-variable ::= 'D-variable(' URIreference ')'

3.2 Human Readable Syntax

Besides the abstract syntax, SWRL also provides an informal human readable syntax, in which a rule has the form:

$$\mathsf{antecedent} \rightarrow \mathsf{consequent},$$

where both antecedent and consequent are conjunctions of atoms written $a_1 \wedge \ldots \wedge a_n$. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., $?x$). Using this syntax, a rule asserting that the composition of parent and brother properties implies the uncle property would be written:

$$\mathsf{parent}(?a, ?b) \wedge \mathsf{brother}(?b, ?c) \rightarrow \mathsf{uncle}(?a, ?c). \tag{3.1}$$

If John has Mary as a parent and Mary has Bill as a brother, then this rule requires that John has Bill as an uncle.

3.3 Direct Model-Theoretic Semantics

The model-theoretic semantics for SWRL is a straightforward extension of the semantics for OWL DL given in [PSHH03a]. The basic idea is that we define *bindings*—extensions of OWL interpretations that also map variables to elements of the domain in the usual manner. A rule is satisfied by an interpretation iff every binding that satisfies the antecedent also satisfies the consequent. The semantic conditions relating to axioms and ontologies are unchanged, so an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology.

3.3.1 Interpreting Rules

From the OWL Semantics and Abstract Syntax document [PSHH03a] we recall that an abstract OWL interpretation is a tuple of the form

$$\mathcal{I} = \langle R, EC, ER, L, S, LV \rangle,$$

where R is a set of resources, $LV \subseteq R$ is a set of literal values, EC is a mapping from classes and datatypes to subsets of R and LV respectively, ER is a mapping from properties to binary relations on R, L is a mapping from typed literals to elements of LV, and S is a mapping from individual names to elements of $EC(\mathsf{owl:Thing})$.

Given an abstract OWL interpretation $\mathcal{I}$, a binding $B(\mathcal{I})$ is an abstract OWL interpretation that extends $\mathcal{I}$ such that S maps i-variables to elements of $EC(\mathsf{owl:Thing})$ and L maps d-variables to elements of LV respectively. An atom is satisfied by a binding $B(\mathcal{I})$ under the conditions given in Table 3.1, where C is an OWL DL description, P is an OWL DL *individual-valued* Property, Q is an OWL DL *data-valued* Property, x, y are variables or OWL individuals, and z is a variable or an OWL data value.

Atom	Condition on Interpretation
$C(x)$	$S(x) \in EC(C)$
$P(x, y)$	$\langle S(x), S(y) \rangle \in ER(P)$
$Q(x, z)$	$\langle S(x), L(z) \rangle \in ER(Q)$
sameAs(x, y)	$S(x) = S(y)$
differentFrom(x, y)	$S(x) \neq S(y)$

Table 3.1: Interpretation Conditions

A binding $B(\mathcal{I})$ satisfies an antecedent A iff A is empty or $B(\mathcal{I})$ satisfies every atom in A. A binding $B(\mathcal{I})$ satisfies a consequent C iff C is not empty and $B(\mathcal{I})$ satisfies every atom in C. A rule is satisfied by an interpretation $\mathcal{I}$ iff for every binding B such that $B(\mathcal{I})$ satisfies the antecedent, $B(\mathcal{I})$ also satisfies the consequent.

The semantic conditions relating to axioms and ontologies are unchanged. In particular, an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology; an ontology is consistent iff it is satisfied by at least one interpretation; an ontology O_2 is entailed by an ontology O_1 iff every interpretation that satisfies O_1 also satisfies O_2.

3.3.2 Example

Consider, for example, the "uncle" rule (3.1) from Section 3.2. Assuming that parent, brother and uncle are *individualvaluedPropertyID*s, then given an interpretation $\mathcal{I} = \langle R, EC, ER, L, S, LV \rangle$, a binding $B(\mathcal{I})$ extends S to map the variables $?a$, $?b$, and $?c$

to elements of $EC(\mathsf{owl:Thing})$; we will use a, b, and c respectively to denote these elements. The antecedent of the rule is satisfied by $B(\mathcal{I})$ iff $(a, b) \in ER(\mathsf{parent})$ and $(b, c) \in ER(\mathsf{brother})$. The consequent of the rule is satisfied by $B(\mathcal{I})$ iff $(a, c) \in ER(\mathsf{uncle})$. Thus the rule is satisfied by $\mathcal{I}$ iff for every binding $B(\mathcal{I})$ such that $(a, b) \in ER(\mathsf{parent})$ and $(b, c) \in ER(\mathsf{brother})$, then it is also the case that $(a, c) \in ER(\mathsf{uncle})$, i.e.:

$$\forall a, b, c \in EC(\mathsf{owl:Thing}).$$
$$((a, b) \in ER(\mathsf{parent}) \wedge (b, c) \in ER(\mathsf{brother})) \rightarrow (a, c) \in ER(\mathsf{uncle})$$

3.4 XML Concrete Syntax

It is useful to define XML serialisations for SWRL. Many possible XML encodings could be imagined (e.g., a RuleML based syntax as proposed in `http://www.daml.org/listarchive/joint-committee/1460.html`), but the most obvious solution is to extend the existing OWL Web Ontology Language XML Presentation Syntax [HEPS03], which can be straightforwardly modified to deal with SWRL. This has several advantages:

- arbitrary OWL classes (e.g., descriptions) can be used as predicates in rules;
- rules and ontology axioms can be freely mixed;
- the existing XSLT stylesheet[4] can easily be extended to provide a mapping to RDF graphs that extends the OWL RDF/XML exchange syntax (see Section 3.5).

In the first place, the ontology root element is extended so that ontologies can include rule axioms and variable declarations as well as OWL axioms, import statements etc. We then simply need to add the relevant syntax for variables and rules. (In this document we use the unspecified `owlr` namespace prefix. This prefix would have to be bound to some appropriate namespace name, either the OWL namespace name or some new namespace name.)

Variable declarations are statements about variables, indicating that the given URI is to be used as a variable, and (optionally) adding any annotations. For example:

```
<owlr:Variable owlr:name="x1" />,
```

states that the URI `x1` (in the current namespace) is to be treated as a variable.

Rule axioms are similar to OWL `SubClassOf` axioms, except they have `owlr:Rule` as their element name. Like `SubClassOf` and other axioms they may include annotations. Rule axioms have an antecedent (`owlr:antecedent`) component and a consequent

[4] `http://www.w3.org/TR/owl-xmlsyntax/owlxml2rdf.xsl`

(owlr:consequent) component. The antecedent and consequent of a rule are both lists of atoms and are read as the conjunction of the component atoms. Atoms can be formed from unary predicates (classes), binary predicates (properties), equalities or inequalities.

Class atoms consist of a description and either an individual name or a variable name, where the description in a class atom may be a class name, or may be a complex description using boolean combinations, restrictions, etc. For example,

```
<owlr:classAtom>
   <owlx:Class owlx:name="Person" />
   <owlr:Variable owlr:name="x1" />
</owlr:classAtom>
```

is a class atom using a class name (#Person), and

```
<owlr:classAtom>
   <owlx:IntersectionOf>
      <owlx:Class owlx:name="Person" />
      <owlx:ObjectRestriction
            owlx:property="hasParent">
         <owlx:someValuesFrom
            owlx:property="Physician" />
      </owlx:ObjectRestriction>
   </owlx:IntersectionOf>
   <owlr:Variable owlr:name="x2" />
</owlr:classAtom>
```

is a class atom using a complex description representing Persons having at least one parent who is a Physician.

Property atoms consist of a property name and two elements that can be individual names, variable names or data values (as OWL does not support complex property descriptions, a property atom takes only a property name). Note that in the case where the second element is an individual name the property must be an *individual-valued* Property, and in the case where the second element is a data value the property must be a *data-valued* Property. For example:

```
<owlr:individualPropertyAtom owlx:property="hasParent">
   <owlr:Variable owlr:name="x1" />
   <owlx:Individual owlx:name="John" />
</owlr:individualPropertyAtom>
```

is a property atom using an *individual-valued* Property (the second element is an individual), and

```
<owlr:datavaluedPropertyAtom owlr:property="grade">
  <owlr:Variable owlr:name="x1" />
  <owlx:DataValue
    rdf:datatype="&xsd;integer">4</owlx:DataValue>
</owlr:datavaluedPropertyAtom>
```

is a property atom using a *data-valued* Property datavalued property (the second element is a data value, in this case an integer).

Finally, same (different) individual atoms assert equality (inequality) between sets of individual and variable names. Note that (in)equalities can be asserted between arbitrary combinations of variable names and individual names. For example:

```
<owlr:sameIndividualAtom>
  <owlr:Variable owlr:name="x1" />
  <owlr:Variable owlr:name="x2" />
  <owlx:Individual owlx:name="Clinton" />
  <owlx:Individual owlx:name="Bill_Clinton" />
</owlr:sameIndividualAtom>
```

asserts that the variables $x1$, $x2$ and the individual names Clinton and Bill_Clinton all refer to the same individual.

The example rule (3.1) on page 15 can be written in the XML concrete syntax for rules as

```
<owlx:Rule>
  <owlr:antecedent>
    <owlr:individualPropertyAtom owlr:property="parent">
      <owlr:Variable owlr:name="a" />
      <owlr:Variable owlr:name="b" />
    </owlr:individualPropertyAtom>
    <owlr:individualPropertyAtom owlr:property="brother">
      <owlr:Variable owlr:name="b" />
      <owlr:Variable owlr:name="c" />
    </owlr:individualPropertyAtom>
  </owlr:antecedent>
  <owlr:consequent>
    <owlr:individualPropertyAtom owlr:property="uncle">
      <owlr:Variable owlr:name="a" />
      <owlr:Variable owlr:name="c" />
    </owlr:individualPropertyAtom>
  </owlr:consequent>
</owlr:Rule>
```

3.5 Mapping to RDF Graphs

It is widely assumed that the Semantic Web will be based on a hierarchy of (increasingly expressive) languages, with RDF/XML providing the syntactic and semantic foundation (see, e.g., [BL98]). One rather serious problem is that, unlike OWL, rules have variables, so treating them as a semantic extension of RDF is very difficult. It is, however, still possible to provide an RDF syntax for rules—it is just that the semantics of the resultant RDF graphs may not be an extension of the RDF Semantics [Hay03].

A mapping to RDF/XML is most easily created as an extension to the XSLT transformation for the OWL XML Presentation syntax.[5] This would introduce RDF classes for SWRL atoms and variables, and RDF properties to link atoms to their predicates (classes and properties) and arguments (variables, individuals or data values).[6] The example rule (3.1) on page 15 would be mapped into RDF as follows:

```
<owlr:Variable rdf:ID="a"/>
<owlr:Variable rdf:ID="b"/>
<owlr:Variable rdf:ID="c"/>
<owlr:Rule>
  <owlr:antecedent rdf:parseType="Collection">
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate rdf:resource="parent"/>
      <owlr:argument1 rdf:resource="#a" />
      <owlr:argument2 rdf:resource="#b" />
    </owlr:individualPropertyAtom>
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate rdf:resource="brother"/>
      <owlr:argument1 rdf:resource="#b" />
      <owlr:argument2 rdf:resource="#c" />
    </owlr:individualPropertyAtom>
  </owlr:antecedent>
  <owlr:consequent rdf:parseType="Collection">
    <owlr:individualPropertyAtom>
      <owlr:propertyPredicate rdf:resource="uncle"/>
      <owlr:argument1 rdf:resource="#a" />
      <owlr:argument2 rdf:resource="#c" />
    </owlr:individualPropertyAtom>
  </owlr:consequent>
</owlr:Rule>
```

Note that complex OWL classes (such as OWL restrictions) as well as class names can be used as the object of SWRL's classPredicate property.

[5] `http://www.w3.org/TR/owl-xmlsyntax/owlxml2rdf.xsl`

[6] The result is similar to the RDF syntax for representing disjunction and quantifiers proposed in [MD02].

3.6 Reasoning Support for SWRL

Although SWRL provides a fairly minimal rule extension to OWL, the consistency problem for SWRL ontologies is still undecidable (because it supports property compositions). This raises the question of how reasoning support for SWRL might be provided.

It seems likely, at least in the first instance, that many implementations will provide only partial support for SWRL. For this reason, users may want to restrict the form or expressiveness of the rules and/or axioms they employ either to fit within a tractable or decidable fragment of SWRL, or so that their SWRL ontologies can be handled by existing or interim implementations.

One possible restriction in the form of the rules is to limit antecedent and consequent classAtoms to be named classes, with OWL axioms being used to assert additional constraints on the instances of these classes (in the same document or in external OWL documents). Adhering to this format should make it easier to translate rules to or from existing (or future) rule systems, including Prolog, production rules (descended from OPS5), event-condition-action rules and SQL (where views, queries, and facts can all be seen as rules); it may also make it easier to extend existing rule based reasoners for OWL (such as Euler[7] or FOWL[8]) to handle SWRL ontologies. Further, such a restriction would maximise backwards compatibility with OWL-speaking systems that do not support SWRL. It should be pointed out, however, that there may be some incompatibility between the first order semantics of SWRL and the Herbrand model semantics of many rule based reasoners.

By further restricting the form of rules and DL axioms used in SWRL ontologies it would be possible to stay within DLP, a subset of the language that has been shown to be expressible in either OWL DL or declarative logic programs (LP) alone [GHVD03]. This would allow either OWL DL reasoners or LP reasoners to be used with such ontologies, although there may again be some incompatibility between the semantics of SWRL and those of LP reasoners.

Another obvious strategy would be to restrict the form of rules and DL axioms so that a "hybrid" system could be used to reason about the resulting ontology. This approach has been used, e.g., in the CLASSIC [PSMB+91] and CARIN systems [LR98], where sound and complete reasoning is made possible mainly by focusing on query answering, by restricting the DL axioms to languages that are *much* weaker than OWL, by restricting the use of DL terms in rules, and/or by giving a different semantic treatment to rules.

Finally, an alternative way to provide reasoning support for SWRL would be to extend the translation of OWL into TPTP[9] implemented in the Hoolet system,[10] and use a first order prover such as Vampire to reason with the resulting first order theory [RV02, TH03].

[7] `http://www.agfa.com/w3c/euler/`

[8] `http://fowl.sourceforge.net`

[9] A standard syntax used by many first order theorem provers—see `http://www.tptp.org`.

[10] `http://www.w3.org/2003/08/owl-systems/test-results-out`

This technique would have several advantages: no restrictions on the form of SWRL rules or axioms would be required; the use of a first order prover would ensure that all inferences were sound with respect to SWRL's first order semantics; and the use of the TPTP syntax would make it possible to use any one of a range of state of the art first order provers.

3.7 Summary

The main strengths of the proposal are its simplicity and its tight integration with the existing OWL language. As we have seen, SWRL extends OWL with the most basic kind of Horn rule (sweetened with a little "syntactic sugar"): predicates are limited to being OWL classes and properties (and so have a maximum arity of 2), there are no disjunctions or negations (of atoms), no built in predicates (such as arithmetic predicates), and no nonmonotonic features such as negation as failure or defaults. Moreover, rules are given a standard first order semantics. This facilitates the tight integration with OWL, with SWRL being defined as a syntactic and semantic extension of OWL DL.

While we believe that SWRL defines a natural and useful level in the hierarchy of Semantic Web languages, it is clear that some applications would benefit from further extensions in expressive power. In particular, the ability to express arithmetic relationships (cf. Chapter 5) between data values is important in many applications (e.g., to assert that persons whose income at least equals their expenditure are happy, while those whose expenditure exceeds their income are unhappy). It is not clear, however, if this would best be achieved by extending SWRL to include rules with built in arithmetic predicates, or by extending OWL Datatypes to include nary predicates [PH03].

Chapter 4

OWL-QL

Another key extension of OWL is a query language that provides a formalism for agents to query information stored in (possibly multiple) OWL *knowledge bases* (or simply *KB*), consisting of (possibly multiple) sets of OWL statements.

The Joint US/EU ad hoc Agent Markup Language Committee[1] has proposed an OWL query language called OWL-QL [FHH03], as a candidate standard language, which is a direct successor of the DAML Query Language (DQL) [FHe03], also released by the Joint US/EU ad hoc Agent Markup Language Committee. Both language specifications go beyond the aims of other current web query languages like XML Query [BCF+03], an XML [BPSM+04] query language, or RQL [KAC+02], an RDF [Bec04] query language, in that they support the use of inference and reasoning services for query answering.

The current Description Logic reasoners like RACER [HM01] or FaCT [Hor99] already offer some querying support, but often very limited and in their own fashion. Recently the query facilities of RACER have been extended and it is now possible to use variables in queries [HM01], which was until now only possible for languages like LOOM [MB87] for the price of incompleteness.

The OWL-QL specification suggests a reasoner independent and more general way for agents (clients) to query OWL knowledge bases on the Semantic Web. The specification is given on a structural level with no exact definition of the external syntax. By this it is easily adoptable for other knowledge representation formats, but on the semantic level OWL-QL is properly defined, due to the formal definition of the relationships among a query, a query answer and the knowledge base(s) provided by the specification (see [FHH03], page 10–11, Appendix Formal Relationship between a Query and a Query Answer).

[1] See `http://www.daml.org/committee/` for the members of the Joint Committee.

4.1 Patterns

To initiate a query-answering dialogue, a client sends a query to an OWL-QL server. The query necessarily includes a *query pattern* that is a collection of OWL statements where some URI references [IIETFTBL98] or literals are replaced by variables. The client also specifies for which variables the server has to provide a binding (*must-bind variables*), for which the server may provide a binding (*may-bind variables*) and for which variables no binding (*don't-bind variables*) should be returned. In this report, must-bind variables, may-bind variables and don't-bind variables are prefixed with "`?`", "$\sim$" and "`!`", respectively.

The client may also specify an answer KB pattern specifying which knowledge base(s) the server should use to answer the query. An *answer KB pattern* can be either a KB, a list of KB URI references or a variable (of the above three kinds); in the last case, the server is allowed to decide which KB(s) to use. The use of may-bind and don't-bind variables is one of the features that clearly distinguish OWL-QL from standard database query languages like SQL [Ins92] and from other web query languages like RQL [KAC+02] and XML Query [BCF+03].

Here is an example of a query pattern and an answer KB pattern.

```
queryPattern:  {(hasFather Bill ?f)}
answerKBPattern:  {http://owlqlExample/fathers.owl}
```

Figure 4.1: A query example

Assume that the KB referred to in the answer KB pattern includes the following OWL statements

```
SubClassOf(Person
           restriction(hasFather someValuesFrom(Person)))
Individual(Bill type(Person)),
```

which assure that every person has a father that is also a person and that Bill is a person.

It could then be inferred that Bill has a father, but we can't name him, so the OWL-QL server can't provide a binding and returns an empty answer collection. This is of course different if f is specified as a may-bind ($\sim$`f`) or don't-bind (`!f`) variable, in both cases an OWL-QL server should return one answer, but without a binding for $\sim$`f` resp. `!f`.

Assume now that the KB includes the additional statement that Mary has Joe as her father and a query with a must-bind variable for the child (`?c`). The type of the variable `f` for the father would change the answer set as follows:

```
queryPattern:  {(hasFather ?c ?f)}
```

If `f` is a must-bind variable (`?f`), a complete answer set contains only persons

whose father is known, in this example (hasFather Mary Joe) where Mary is a binding for `?c` and Joe is a binding for `?f`.

`queryPattern:` {`(hasFather ?c !f)`}
If `f` is a don't-bind variable (`!f`), a complete answer set contains all known persons since it is specified that all persons have a father, but without a binding for `!f`. In this example (hasFather Mary !f), (hasFather Joe !f) and (hasFather Bill !f) should be in the answer set.

`queryPattern:` {`(hasFather ?c ~f)`}
If `f` is a may-bind variable (`~f`), the complete and non-redundant answer set contains all known persons since it is specified that all persons have a father, but a binding for `~f` is only provided in case the father is known. In this example (hasFather Mary Joe), (hasFather Joe ~f) and (hasFather Bill ~f) should be in the answer set.

An optional query parameter allows the definition of a pattern that the server should use to return the answers. This *answer pattern* necessarily includes the format of all variables used in the query pattern. If no answer pattern is specified, a two item list whose first item is the querys must-bind variables list and whose second item is the querys may-bind variables list is used as the answer pattern. This is different to the DQL specification, where, for the case that no answer pattern was specified, the query pattern is used as the answer pattern.

Another option for a query is to include a *query premise* (a set of assumptions) to facilitate "if-then" queries, which can't be expressed otherwise since OWL does not support an "implies" logical connective. E.g. to ask a question like "If Bill is a person, then does Bill have a father?" the query premise part includes an OWL KB or a KB reference stating that Joe is a person and the query part is the same as in figure 4.1. The server will treat OWL statements in the query premise as a regular part of the answer KB and all answers must be entailed by this KB.

4.2 Query-Answering Dialogues

To initiate a query-answering dialogue the clients sends a query to an OWL-QL server. The server then returns an *answer bundle*, which includes a (possibly empty) answer set together with either a *termination token* to end the dialogue or a *process handle* to allow the continuation of the query-answering dialogue. A termination token is either *end* to indicate that the server can't for any reasons provide more answers or *none* to assert that no more answers are possible. If a server is unable to deal with a query, e.g. due to syntactical errors, a *rejected* termination token is sent in the answer. The specification also allows the definition of further termination token, e.g. to provide information about the rejection reasons.

Since an answer bundle can be very large and the computation can take a long time, the specification also allows to specify an *answer bundle size bound* that is an upper bound for the number of answers in an answer bundle. If the client specified an answer bundle size bound in the query, the server does not send more answers then allowed by the answer bundle size bound.

To continue a dialogue the client sends a *server continuation* request including the process handle and an answer bundle size bound for the next answer bundle. A server continuation must not necessarily be sent from the same client. The client can also pass the process handle to another client that then continues the query answering dialogue. If the server can't deliver any more answers for a server continuation request, it sends a termination token together with the probably empty answer set.

If the client does not want to continue the dialogue, the client can send a *server termination* request including the process handle. The server can use a received server termination request to possibly free resources. Figure 4.2 illustrates the query-answering-dialogue.

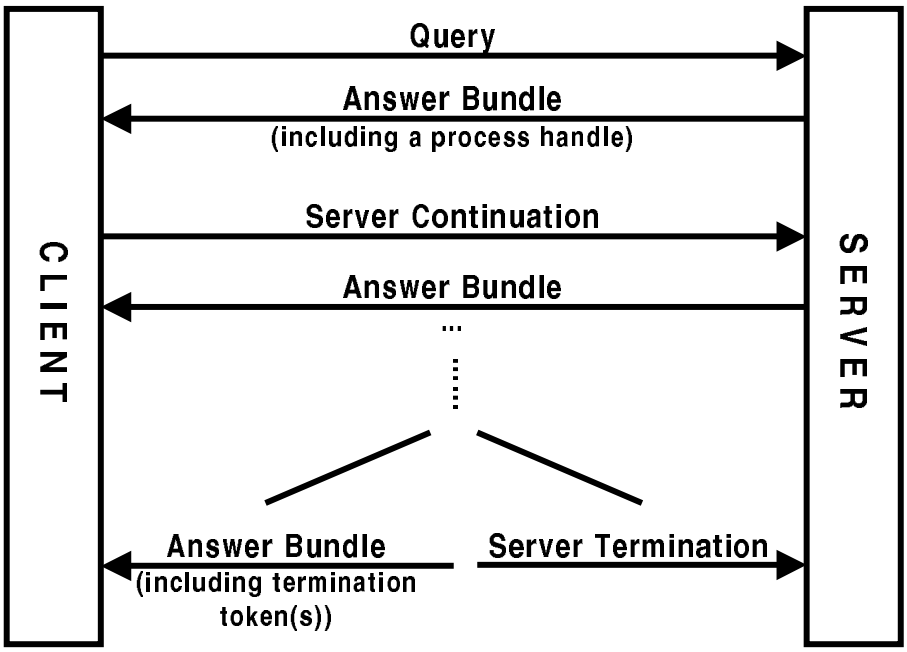

Figure 4.2: The query-answering dialogue

The specification provides some attributes for a server to promote the delivered quality of service or the so called *conformance level*. A server can guarantee to be *non-repeating*, so no answers with the same binding are delivered. The strictest level is called a *terse* server and only the most specific answers are delivered to the client. An answer is more general (subsumes another) if it only provides fewer bindings for may-bind variables or has less specific bindings for variables that occur only as values of minCardinality or

maxCardinality restrictions, e.g. if the KB is true for a binding of 4 for a maxCardinality variable, then it will also be true for a binding of 5, 6, ... Since this demand is very high for a server that produces the answers incrementally, a less restrictive conformance level is *serially terse*, where all delivered answers are more specific that previously delivered answers. Finally servers that guarantee to terminate with termination token *none* are called *complete*.

4.3 Summary

In general, OWL-QL provides a flexible framework in conducting a query-answering dialogue using knowledge represented in OWL. It allows the definition of additional parameters, delegation of queries to another server or the continuation of a query dialogue by other clients that know a valid process handle. If the client specifies an answer bundle size bound, the specification allows an OWL-QL server to compute all answers at once or to compute the answers incrementally, as long as the answer set returned to the client contains not more answers than specified by the answer bundle size bound. The specification also allows the definition of further termination token, e.g. to provide information about the rejection reasons.

The current version of OWL-QL, however, has the following limitations.

External Syntax The specification does not provide any exact syntax definition or a specification of how to communicate the supported conformance level to a client and also other mechanism like time-outs for a query are not specified. This is due to the focus on providing an abstract specification on a structural level and to allow the various syntactical preferences of the different web communities to fit the standard to their needs. An OWL-QL server therefore has to provide this information in a documentation or in an XML Schema [BM01] [TBMM01].

Semantics As the external syntax has not (yet) been specified, the formal semantics of OWL-QL is presented in a quite general way, and is only included as an appendix of the specification. In particular, the fact that the relationship between the OWL model-theoretic semantics and the OWL-QL semantics has not been specified is not very satisfactory.

Query classes The OWL-QL specification does not introduce the query classes that DQL provides. Since it is difficult for some reasoners to implement all of these requirements, DQL explicitly allows a partial implementation. A DQL server can restrict itself to special *query classes*, e.g. a server may only support queries that conform to a pattern like `?x rdf:type C`, where C is an DAML+OIL class expression, or `?x`

`daml:subClassOf ?y` and reject all other queries. The server is then said to apply to these query classes. Until now it is up to the implementer of an OWL-QL server to provide a documentation of supported query classes and how, if at all, this is communicated to a client. In a real agent-to-agent protocol, however, a client should be able to determine the supported query classes and this is one of the issues a future specification should address.

In short, for an implementer of an OWL-QL server, OWL-QL acts as a guide without a concrete external syntax, a formal relationship with the OWL model-theoretic semantics and proper means to communicate the supported query classes or the conformance level. Until now every implementation has to fill (some of) these gaps and to provide a detailed documentation of how these gaps have been filled.

Chapter 5

A Predicate Extension

Although OWL adds considerable expressive power to the Semantic Web, the OWL datatype formalism (or simply *OWL datatyping*) is much too weak for many applications. E.g., OWL datatyping does not provide a general framework for user-defined datatypes, such as XML Schema derived datatypes, nor does it support n-ary datatype predicates (such as the binary predicate $>$ for integers), not to mention user-defined datatype predicates (such as the binary predicate $>$ for non-negative integers).

OWL datatyping provides the uses of both datatypes and datatype expressions, where *OWL datatypes* are defined by external type systems and *OWL datatypes expressions* are constructed by the built-in OWL constructor oneOf. E.g., $\mathrm{xsd{:}integer}$ is a datatype defined by the XML Schema type system. Note that, however, the derived XML Schema datatype $>_{18}$ (by using the $\mathrm{xsd{:}minExclusive}$ facet) of $\mathrm{xsd{:}integer}$ is not an OWL datatype, because there is no standard access mechanism for derived XML Schema datatypes (simple types), i.e., there is no standard way to access an XML Schema datatype in an XML Schema document.[1] The only kind of OWL datatype expressions are called *enumerated datatypes*, which are constructed by explicitly specifying all the data values of the enumerated datatypes (using the oneOf constructor). E.g., the list $\{0,15,30,40\}$[2] is an enumerated datatype, which can be used as the range of datatype properties, such as $tennisGameScore$. It is easy to see that enumerated datatypes are not expressive enough to represent derived XML Schema datatypes such as $>_{18}$, which has infinite number of data values.

In this chapter, we first show how to extend OWL datatyping with predicates, and then present a proposal of a predicate extension of SWRL.

[1] URI references for resources in an XML Schema document can represent not only datatypes, but also entities, attributes and groups etc.

[2] Strictly speaking, enumerated datatypes are built using literals.

5.1 Datatype Groups: Extending OWL Datatyping with Predicates

This section describes an OWL compatible revision of the datatype group approach first presented in [PH03], in order to extend OWL datatyping with datatype predicates.

5.1.1 Predicates

Definition 5 (Datatype Predicate) *A* datatype predicate *(or simply* predicate*)* p *is characterised by an arity* $a(p)$*, and a predicate extension (or simply* extension*)* $E(p)$. ◇

Here are some examples of predicates:

1. $integer$ is a predicate with arity $a(integer) = 1$ and predicate extension $E(integer) = V(integer)$, where $V(integer)$ is the value space of $integer$. In general, datatypes can be seen as predicates with arity 1 and predicate extensions equal to their value spaces.
2. $>^{int}_{[18]}$ is a unary predicate, with $a(>^{int}_{[18]}) = 1$ and $E(>^{int}_{[18]}) = \{i \in E(integer) \mid i > 18\}$. We can use $>^{int}_{[18]}$ represent a derived XML Schema datatype derived from $\mathrm{xsd{:}integer}$, with 18 as the value of the minExclusive facet.
3. $=^{int}$ is a binary predicate with arity $a(=^{int}) = 2$ and extension $E(=^{int}) = \{\langle i_1, i_2\rangle \in E(integer)^2 \mid i_1 = i_2\}$.
4. sum is a predicate that does not have a fixed arity, where $E(sum) = \{\langle i_1, \ldots, i_n\rangle \in E(integer)^n \mid i_1 = i_2 + \cdots + i_n\}$ and $a(sum) \geq 3$.

In stating the semantics, we assume that datatype interpretations are relativised to a predicate map.

Definition 6 (Predicate Map) *We consider a predicate map* $\mathbf{M}_p$ *that is a partial mapping from predicate URI references to predicates.* ◇

Example 5 $\mathbf{M}_{p_1} = \{\langle \mathrm{xsd{:}string}, string\rangle, \langle \mathrm{xsd{:}integer}, integer\rangle, \langle \mathrm{owlx{:}integerEquality}, =^{int}\rangle, \langle \mathrm{owlx{:}integerLargerThanx\&n}, >^{int}_{[n]}\rangle\}$ *is a predicate map, where* $\mathrm{xsd{:}string}$*,* $\mathrm{xsd{:}integer}$*,* $\mathrm{owlx{:}integerEquality}$ *and* $\mathrm{owlx{:}integerLargerThanx\&n}$ *are predicate URI references,* $string$*,* $integer$ *and* $>^{int}_{[n]}$ *are unary predicates, and* $=^{int}$ *is a binary predicate. Note that, by '*$>^{int}_{[n]}$*', we mean there exist a predicate* $>^{int}_{[n]}$ *for each integer* n*, which is represented by the predicate URI* $\mathrm{owlx{:}integerLargerThanx\&n}$. ♢

Similar to supported and unsupported datatype URIs, we have supported and unsupported predicate URIs according to a predicate map.

Definition 7 (Supported and Unsupported Predicate URIs) *Given a predicate map* $\mathbf{M}_p$*, a predicate URI* u *is called a* supported predicate URI w.r.t. $\mathbf{M}_p$ *(or simply* supported predicate URI*), if there exists a predicate* p *s.t.* $\mathbf{M}_p(u) = p$ *(in this case,* p *is called a* supported predicate *w.r.t.* $\mathbf{M}_p$*); otherwise,* u *is called an* unsupported predicate URI w.r.t. $\mathbf{M}_p$ *(or simply* unsupported predicate URI*).* ◇

E.g., $\mathrm{owlx{:}integerEquality}$ is a supported predicate URI w.r.t. $\mathbf{M}_{p_1}$ presented in Example 5, while $\mathrm{owlx{:}integerInequality}$ is an unsupported predicate URI w.r.t. $\mathbf{M}_{p_1}$. Therefore, according to $\mathbf{M}_{p_1}$, we know neither the arity nor the extension of the predicate that $\mathrm{owlx{:}integerInequality}$ represents. Note that we make as few as assumptions as possible about unsupported predicates; e.g., we do not even assume that they have a fixed arity.

5.1.2 Datatype Groups

Informally speaking, a datatype group is a group of supported predicate URIs ('wrapped' around a set of base datatype URIs), which can potentially be divided into different sub-groups, so that predicates in each sub-group are about the base datatype of the sub-group. This allows us to make use of known decidability results about the satisfiability problems of predicate conjunctions of, e.g., the admissible/computable concrete domains presented in Section 2.4 of [Lut01]. Formally, a datatype group is defined as follows, and the sub-groups are defined in Definition 11.

Definition 8 (Datatype Group) *A* datatype group $\mathcal{G}$ *is a tuple (*$\mathbf{M}_p$,$\mathbf{D}_\mathcal{G}$,dom*), where* $\mathbf{M}_p$ *is the* predicate map *of* $\mathcal{G}$*,* $\mathbf{D}_\mathcal{G}$ *is the set of* base datatype *URI references of* $\mathcal{G}$*, and* dom *is the* declared domain function *of* $\mathcal{G}$*.*

We call $\Phi_\mathcal{G}$ *the set of supported predicate URI references of* $\mathcal{G}$*, i.e., for each* $u \in \Phi_\mathcal{G}$*,* $\mathbf{M}_p(u)$ *is defined; we require* $\mathbf{D}_\mathcal{G} \subseteq \Phi_\mathcal{G}$*. We assume that there exists a unary predicate URI reference* $\mathrm{owlx{:}DatatypeBottom} \notin \Phi_\mathcal{G}$*.*

The declared domain function dom *is a mapping s.t.* $\forall u \in \mathbf{D}_\mathcal{G}$*:* $\mathsf{dom}(u) = u$*, and* $\forall u \in \Phi_\mathcal{G}$*,* $\mathsf{dom}(u) \in (\mathbf{D}_\mathcal{G})^n$*, where* $n = a(\mathbf{M}_p(u))$*.* ◇

As we can see from the above definition, supported predicate URIs in $\mathbf{D}_\mathcal{G}$ are also treated as base datatype URIs, therefore they can be used in typed literals.[3] Supported predicate URIs relate to base datatypes URIs via the declared domain function dom, which also helps in defining the interpretation of the relativised negated predicate URIs in Definition 9.

Example 6 $\mathcal{G}_1 = (\mathbf{M}_{p_1}, \mathbf{D}_{\mathcal{G}_1}, \mathsf{dom}_1)$ *is a datatype group, where* $\mathbf{M}_{p_1}$ *is defined in Example 5,* $\mathbf{D}_{\mathcal{G}_1} = \{\mathrm{xsd{:}string}, \mathrm{xsd{:}integer}\}$*, and* $\mathsf{dom}_1 = \{\langle \mathrm{xsd{:}string}, \mathrm{xsd{:}string}\rangle, \langle \mathrm{xsd{:}integer}, \mathrm{xsd{:}integer}\rangle, \langle \mathrm{owlx{:}integerEquality}, (\mathrm{xsd{:}integer}, \mathrm{xsd{:}integer})\rangle, \langle \mathrm{owlx{:}}$*integerLargerThanx&n,* $\mathrm{xsd{:}integer}\rangle\}$.

[3] Typed literals are of the form "v"^^u, where v is a lexical form of a data value and u is a datatype URI.

According to $\mathbf{M}_{p1}$*, we have* $\Phi_{\mathcal{G}1} = \{\text{xsd:string}, \text{xsd:integer}, \text{owlx:integerEquality}, \text{owlx:integer}\textit{LargerThanx\&n}\}$. ♢

Definition 9 (Interpretation of Datatype Group) *A* datatype interpretation $\mathcal{I}_{\mathbf{D}}$ *of a datatype group* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_{\mathcal{G}}, \mathsf{dom})$ *is a pair* $(\Delta_{\mathbf{D}}, \cdot^{\mathbf{D}})$*, where* $\Delta_{\mathbf{D}}$ *(the datatype domain) is a non-empty set and* $\cdot^{\mathbf{D}}$ *is a datatype interpretation function, which has to satisfy the following conditions*

1. $\text{rdfs:Literal}^{\mathbf{D}} = \Delta_{\mathbf{D}}$*;*
2. *for each plain literal* l*,* $l^{\mathbf{D}} = l \in \mathbf{PL}$*, where* $\mathbf{PL}$ *is the value space for plain literals (i.e., the union of the set of Unicode strings and the set of pairs of Unicode strings and language tags);*
3. $\forall u \in \mathbf{D}_{\mathcal{G}}$*, let* $d = \mathbf{M}_p(u)$*:*
   (a) $u^{\mathbf{D}} = V(d) \subseteq \Delta_{\mathbf{D}}$*,*
   (b) *if* $v \in L(d)$*, then* $(\text{“}v\text{”}\hat{}\hat{}u)^{\mathbf{D}} = L2V(d)(v)$*,*
   (c) *if* $v \notin L(d)$*, then* $(\text{“}v\text{”}\hat{}\hat{}u)^{\mathbf{D}}$ *is not defined;*
4. *for any two* $u_1, u_2 \in \mathbf{D}_{\mathcal{G}}$*:* $u_1^{\mathbf{D}} \cap u_2^{\mathbf{D}} = \emptyset$*;*
5. $\mathbf{PL} \subseteq \Delta_{\mathbf{D}}$*, and* $\forall u \in \mathbf{D}_{\mathcal{G}}, u^{\mathbf{D}} \subseteq \Delta_{\mathbf{D}}$*;*
6. $\text{owlx:DatatypeBottom}^{\mathbf{D}} = \emptyset$*;*
7. $\forall u \in \Phi_{\mathcal{G}}$*,* $u^{\mathbf{D}} = E(\mathbf{M}_p(u))$*;*
8. $\forall u \in \Phi_{\mathcal{G}}$*,* $u^{\mathbf{D}} \subseteq (\mathsf{dom}(u))^{\mathbf{D}}$*, where* $(\mathsf{dom}(u))^{\mathbf{D}} = d_1^{\mathbf{D}} \times \cdots \times d_n^{\mathbf{D}}$ *for* $\mathsf{dom}(u) = (d_1, \ldots, d_n)$ *and* $a(\mathbf{M}_p(u)) = n$*.*
9. $\forall u \notin \Phi_{\mathcal{G}}$*,* $u^{\mathbf{D}} \subseteq \bigcup_{n \geq 1} (\Delta_{\mathbf{D}})^n$*, and* $\text{“}v\text{”}\hat{}\hat{}u \in \Delta_{\mathbf{D}}$*.*

Moreover, we extend $\cdot^{\mathbf{D}}$ *to (relativised) negated predicate URI references* $\overline{u}$ *as follows:*

$$(\overline{u})^{\mathbf{D}} = \begin{cases} \Delta_{\mathbf{D}} \setminus u^{\mathbf{D}} & \text{if } u \in \mathbf{D}_{\mathcal{G}} \\ (\mathsf{dom}(u))^{\mathbf{D}} \setminus u^{\mathbf{D}} & \text{if } u \in \Phi_{\mathcal{G}} \setminus \mathbf{D}_{\mathcal{G}} \\ \bigcup_{n \geq 1} (\Delta_{\mathbf{D}})^n \setminus u^{\mathbf{D}} & \text{if } u \notin \Phi_{\mathcal{G}}. \end{cases}$$

♢

Condition 4 requires the value spaces of the base datatype are disjoint, which is essential to dividing $\Phi_{\mathcal{G}}$ into sub-groups. Condition 5 states that the union of the value spaces of plain literals and base datatypes is a proper subset of the datatype domain, because a typed literal associated with an unsupported predicate can be interpreted as something outside

the above value spaces. Condition 6 states that $\mathrm{owlx{:}DatatypeBottom}$ is a negated predicate URI of $\mathrm{rdfs{:}Literal}$. Condition 7 and 8 ensure that the supported predicate URIs are interpreted as the extensions of the predicates they represent, and are subsets of the corresponding declared domains. Condition 9 ensures that unsupported predicate URIs are not restricted to any fixed arity, and that typed literals with unsupported predicates are interpreted as some member of the datatype domain.

Note that supported predicate URIs $u \in \Phi_{\mathcal{G}} \setminus \mathbf{D}_{\mathcal{G}}$ have relativised negations (to their declared domains). E.g., $\overline{\mathrm{owlx{:}integerLargerThanx\&18}}$, the negated predicate URI for $\mathrm{owlx{:}integerLargerThanx\&18}$, is interpreted as $V(integer)\backslash(\mathrm{owlx{:}integerLargerThan}x\&18)^{\mathbf{D}}$; therefore, its interpretation includes the integer 5, but not the string "Fred", no matter if there exist any other base datatypes in $\mathbf{D}_{\mathcal{G}}$.

Now we introduce the kind of basic reasoning mechanisms required in a datatype group.

Definition 10 (Predicate Conjunction) *Let* $\mathbf{V}$ *be a set of variables,* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_{\mathcal{G}}, \mathsf{dom})$ *a datatype group, we consider predicate conjunctions of* $\mathcal{G}$ *of the form*

$$\mathscr{C} = \bigwedge_{j=1}^{k} w_j(v_1^{(j)}, \ldots, v_{n_j}^{(j)}), \tag{5.1}$$

where the $v_i^{(j)}$ *are variables from* $\mathbf{V}$*,* w_j *are (possibly negated) predicate URI references of the form* u_j *or* $\overline{u_j}$*, and if* $u_j \in \Phi_{\mathcal{G}}, a(\mathbf{M}_p(u_j)) = n_j$*. A predicate conjunction* $\mathscr{C}$ *is called* satisfiable *iff there exists a function* δ *mapping the variables in* $\mathscr{C}$ *to data values in* $\Delta_{\mathbf{D}}$ *s.t.* $\langle \delta(v_1^{(j)}), \ldots, \delta(v_{n_j}^{(j)}) \rangle \in w_j^{\mathbf{D}}$ *for all* $1 \leq j \leq k$*. Such a function* δ *is called a* solution *for* $\mathscr{C}$*.* ◇

E.g., $\mathscr{C}_1 = \overline{\mathrm{owlx{:}integerLargerThanx\&38}}(v_1) \wedge \mathrm{owlx{:}integerLargerThanx\&12}(v_2) \wedge \mathrm{owlx{:}integerEquality}(v_1, v_2)$ is a predicate conjunction of $\mathcal{G}_1$ presented in Example 6 on page 31. The function $\delta = \{v_1 \mapsto 26, v_2 \mapsto 26\}$ is a solution of $\mathscr{C}_1$; therefore, $\mathscr{C}_1$ is satisfiable.

The predicate conjunction over a datatype group $\mathcal{G}$ can possibly be divided into independent sub-conjunctions of sub-groups of $\mathcal{G}$. Informally speaking, a sub-group includes a base datatype URI and the set of supported predicate URIs about the base datatype URI.

Definition 11 (Sub-Group) *Given a datatype group* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_{\mathcal{G}}, \mathsf{dom})$ *and a base datatype URI reference* $w \in \mathbf{D}_{\mathcal{G}}$*, the* sub-group of w in $\mathcal{G}$ *, abbreviated as* sub-group(w)*, is defined as:*

$$\textit{sub-group}(w) = \{u | u \in \Phi_{\mathcal{G}} \textit{ and } \mathsf{dom}(u) = \underbrace{(w, \ldots, w)}_{n \textit{ times}}\}$$

where $n = a(\mathbf{M}_p(u))$*.* ◇

Example 7 *The sub-group of* $\mathrm{xsd{:}integer}$ *in* $\mathcal{G}_1$ *presented in Example 6 on page 31 is* sub-group$(\mathrm{xsd{:}integer}) = \{\mathrm{xsd{:}integer}, \mathrm{owlx{:}integerEquality}, \mathrm{owlx{:}integerLargerTh}$-*anx&n*$\}$. *According to the above definition and condition 4 of Definition 9, the predicate conjunction over* sub-group$(\mathrm{xsd{:}integer})$ *and* sub-group$(\mathrm{xsd{:}string})$ *can be handled separately if there are no common variables; if there are common variables, there exist contradictions, due to the disjointness of* $V(integer)$ *and* $V(string)$. ♢

Since the datatype domain $\Delta_{\mathbf{D}}$ of a datatype group is not fixed, an admissible concrete domain can no longer be a conforming datatype group (cf. Lemma 4 in [PH03]). However, a sub-group of a datatype group is very close to a concrete domain; the following definition, accordingly, defines the *corresponding concrete domain* of a sub-group in a datatype group.

Definition 12 (Corresponding Concrete Domain) *Given a datatype group* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ *and a base datatype URI reference* $w \in \mathbf{D}_\mathcal{G}$*, let* $\mathbf{M}_p(w) = \mathcal{D}$*, the* corresponding concrete domain *of* sub-group(w) *is* $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$*, where* $\Delta_\mathcal{D} := V(\mathcal{D})$ *and* $\Phi_\mathcal{D} = \{\bot_\mathcal{D}\} \cup \{\mathbf{M}_p(u) | u \in$ sub-group$(w)\}$*, where* $\bot_\mathcal{D}$ *corresponds to* $\overline{w}$. ⋄

Example 8 *The corresponding concrete domain of* sub-group$(\mathrm{xsd{:}integer})$ *in* $\mathcal{G}_1$ *presented in Example 6 is* $(\Delta_{integer}, \Phi_{integer})$*, where* $\Delta_{integer} := V(integer)$ *and* $\Phi_{integer} = \{\bot_{integer}, integer, =^{int}, >^{int}_{[n]}\}$. *Note that the predicate* $\bot_{integer}$ *corresponds to* $\overline{\mathrm{xsd{:}integer}}$*, the negated form of* $\mathrm{xsd{:}integer}$. ♢

The benefit of introducing the corresponding concrete domain for a sub-group is that if the corresponding concrete domain is admissible, informally speaking, the sub-group is computable.

Lemma 13 *Given a datatype group* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_\mathcal{G}, \mathsf{dom})$ *and a base datatype URI reference* $w \in \mathbf{D}_\mathcal{G}$*, if the corresponding concrete domain of* w*,* $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$*, is admissible, then the satisfiability problem for finite predicate conjunctions* $\mathscr{C}_w$ *of the* sub-group(w) *is decidable.*

Proof: Direct consequence of Definition 12 and Definition 2.8 on page 28 of [Lut01]: (i) If $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$ is admissible, then $\Phi_\mathcal{D}$ is close under negation; hence $\forall u \in$ sub-group$(w) \setminus \{w\}$, there exists $u' \in$ sub-group(w), such that $\overline{u}^{\mathbf{D}} = u'^{\mathbf{D}}$. Therefore, predicate conjunctions over sub-group(w) can be equivalently transformed into predicate conjunctions of $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$. (ii) Predicate conjunctions over $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$ are decidable, if $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$ is admissible. ∎

Now we provide the conditions for comforming/computable datatype groups.

Definition 14 (Conforming Datatype Group) *A datatype group* $\mathcal{G}$ *is conforming iff*

1. *for any* $u \in \Phi_{\mathcal{G}} \setminus \mathbf{D}_{\mathcal{G}}$ *with* $a(\mathbf{M}_p(u)) = n \geq 2$*:* $\mathsf{dom}(u) = (\underbrace{w, \ldots, w}_{n \text{ times}})$ *for some* $w \in \mathbf{D}_{\mathcal{G}}$*, and*

2. *for any* $u \in \Phi_{\mathcal{G}} \setminus \mathbf{D}_{\mathcal{G}}$*: there exist* $u' \in \Phi_{\mathcal{G}} \setminus \mathbf{D}_{\mathcal{G}}$ *such that* $u'^{\mathbf{D}} = \overline{u}^{\mathbf{D}}$*, and*

3. *the satisfiability problems for finite predicate conjunctions of each sub-group of* $\mathcal{G}$ *is decidable, and*

4. *for each datatype* $u_i \in \mathbf{D}_{\mathcal{G}}$*, there exists* $w_i \in \Phi_{\mathcal{G}}$*, s.t.* $\mathbf{M}_p(w_i) = \neq_{u_i}$ *where* $\neq_{u_i}$ *is the binary inequality predicate for* $\mathbf{M}_p(u_i)$*.* $\diamond$

In the above definition, condition 1 ensure that $\Phi_{\mathcal{G}}$ can be completely divided into sub-groups. Condition 2 and 3 and all the sub-groups are computable. Condition 4 ensures that number restrictions can be handled.

Example 9 *$\mathcal{G}_1$ presented in Example 6 is not conforming, because it doesn't satisfy condition2 and 4 of the above definition. To make it conforming, we should extend* $\mathbf{M}_{p_1}$ *as follows:* $\mathbf{M}_{p_1} = \{\langle \text{xsd:string}, string\rangle, \langle \text{owlx:stringEquality}, =^{str}\rangle, \langle \text{owlx:stringInequality}, \neq^{str}\rangle, \langle \text{xsd:integer}, integer\rangle, \langle \text{owlx:integerEquality}, =^{int}\rangle, \langle \text{owlx:integerInequality}, \neq^{int}\rangle, \langle \text{owlx:integerLargerThanx\&n}, >^{int}_{[n]}\rangle, \langle \text{owlx:integerLessThanOrEqualx\&n}, \leq^{int}_{[n]}\rangle\}$. $\diamondsuit$

Lemma 15 *If* $\mathcal{G} = (\mathbf{M}_p, \mathbf{D}_{\mathcal{G}}, \mathsf{dom})$ *is a conforming datatype group, then the satisfiability problem for finite predicate conjunctions of* $\mathcal{G}$ *is decidable.*

Proof: Let the predicate conjunction be $\mathscr{C} = \mathscr{C}_{w_1} \wedge \cdots \wedge \mathscr{C}_{w_k} \wedge \mathscr{C}_U$,where $\mathbf{D}_{\mathcal{G}} = \{w_1, \ldots, w_k\}$ and $\mathscr{C}_{w_i}$ is the predicate conjunction for $\mathsf{sub\text{-}group}(w_i)$ and $\mathscr{C}_U$ the sub-conjunction of $\mathscr{C}$ where only unsupported predicate appear.

According to Definition 14, $\mathscr{C}_{w_1} \wedge \cdots \wedge \mathscr{C}_{w_k}$ is decidable. According to Definition 8, $\mathscr{C}_U$ is *unsatisfiable* iff there exist $u(v_1, \ldots, v_n)$ and $\overline{u}(v_1, \ldots, v_n)$ for some $u \notin \Phi_{\mathcal{G}}$ appear in $\mathscr{C}_U$; otherwise, $\mathscr{C}_U$ is *satisfiable*. Therefore, $\mathscr{C}$ is *satisfiable* iff both $\mathscr{C}_{w_1} \wedge \cdots \wedge \mathscr{C}_{w_k}$ and $\mathscr{C}_U$ are *satisfiable*; otherwise, $\mathscr{C}$ is *unsatisfiable*. ∎

5.1.3 Summary

When we extend OWL datatyping to predicates by datatype groups, we consider the similarities and differences between datatypes and predicates: on the one hand, datatypes can be seen as unary predicates; on the other hand, datatypes are characterised by their lexical spaces, value spaces and lexical-to-value mappings, while predicates are characterised by their arities and extensions. For datatypes, we are more concern about their members, i.e., data values; therefore, we could use datatype URI references in typed literals. Predicates

are more suitable to represent constraints about data values than datatypes in that they can represent not only unary but also n-ary constraints.

In a datatype group, predicates can be divided into some sub-groups, each of which is about a base datatype of the datatype group. The motivations of grouping come from the observation that the predicate conjunction problem of each (some) sub-group(s) is (are) decided by a datatype reasoner. More importantly, the decidability of the predicate conjunction problem of a datatype group depends of the decidability of the sub-problems of all its sub-groups.

Based on the datatype group approach, we propose OWL-E [PH04], which is a language extending OWL DL with datatype expression axioms, as well as the datatype group-based class constructors to allow the use of datatype expressions in class restrictions. The novelty of OWL-E is that it enhances OWL DL with much more datatype expressiveness and it is still decidable.

5.2 SWRL-P: Extending SWRL with Predicates

This section presents SWRL-P, an extension of SWRL 0.5 (Semantic Web Rule Language, cf. Chapter 3) with datatype predicates (or simply *predicates*), based on the OWL predicate extension presented in Section 5.1 on page 30. We will compare SWRL-P and SWRL 0.7 in Section 5.2.3.

SWRL-P extends the set of SWRL atoms to include predicate atoms (or *built-in atoms*);[4] both the abstract syntax and the model-theoretic semantics are extended accordingly. Predicate atoms are of the form $builtin(p, v_1, \ldots, v_n)$, where p is a predicate URI reference, and $v_1, \ldots, v_n$ are either literals or variables. Predicate atoms can be used in both the antecedent (body) and consequent (head).

5.2.1 Abstract Syntax

SWRL-P extends axioms to also allow predicate atoms, by adding the production:

atom ::= builtIn '(' dataPredicateID { d-object } ')'

Example 10 *We can define a business rule that one charges no shipping fees for orders (selected items only) over 50 dollars.*

```
Implies(
  Antecedent(priceInDollars(I-variable(x1) D-variable(t1)),
             SelectedItems(I-variable(x1)),
             builtIn(owlx:integerGreaterThan
```

[4] We call predicates *built-ins*, following SWRL 0.7, which is available at `http://www.daml.org/rules/proposal/`.

Atom	Condition on Interpretation
$\mathsf{C}(x)$	$S(x) \in EC(\mathsf{C})$
$P(x, y)$	$\langle S(x), S(y)\rangle \in ER(P)$
$Q(x, z)$	$\langle S(x), L(z)\rangle \in ER(Q)$
$u(z_1, \ldots, z_n)$	$\langle L(z_1), \ldots, L(z_n)\rangle \in EP(u)$
$sameAs(x, y)$	$S(x) = S(y)$
$differemtFrom(x, y)$	$S(x) \neq S(y)$

Table 5.1: Interpretation Conditions Table

D-variable(t1), "50"^^xsd:integer))
Consequent(*shippingFeeInDollars*(I-variable(x1) "0"^^xsd:integer))
)

In human readable syntax, this rule can be written as

$priceInDollars(?x1, ?t1) \wedge \mathsf{SelectedItems}(?x1) \wedge$ owlx:integerGreaterThan($?t1$, "50"^^xsd:integer)
$\rightarrow$ $shippingFeeInDollars(?x1,$ "0"^^xsd:integer)) $\diamondsuit$

5.2.2 Direct Model Theoretic Semantics

Given a datatype group $\mathcal{G}$, We extend an OWL interpretation to a tuple of the form

$$\mathcal{I}_p = \{\mathbf{R}, EC, ER, EP, L, S, \mathbf{LV}\}$$

where **R** is a set of resources, **LV** $\subseteq$ **R** is a set of literal values (the datatype domain of $\mathcal{G}$), EC is a mapping from class descriptions to subsets of **R**, ER is a mapping from property URIs to binary relations on R, EP is a mapping from supported predicate URIs $u \in \Phi_{\mathcal{G}}$ to the predicate extensions $E(\mathbf{M}_p(u))$ of the predicates they represent[5] and from unsupported predicate URIs $u \notin \Phi_{\mathcal{G}}$ to subsets of $\bigcup_{n\geq 1}(\mathbf{LV})^{\mathbf{n}}$, L is a mapping from typed literals to elements of **LV**, and S is a mapping from individual names to elements of EC(owl:Thing).

Given a datatype group $\mathcal{G}$ and an extended abstract OWL interpretation $\mathcal{I}_p$, a binding B($\mathcal{I}_p$) is an extended abstract OWL interpretation that extends $\mathcal{I}_p$ such that S maps i-variables to elements of EC(owl:Thing) and L maps d-variables to elements of **LV** respectively. An atom is satisfied by an interpretation $\mathcal{I}_p$ under the conditions given in the Interpretation Conditions Table 5.1, where C is an OWL DL class description, P is an OWL DL individualvalued property URI, Q is an OWL DL datavalued property URI, u is a predicate URI, x, y are variables or OWL individual URIs, and z, $z_1, \ldots, z_n$ are variables or typed literals.

A binding B($\mathcal{I}_p$) satisfies an antecedent A iff A is empty or B($\mathcal{I}_p$) satisfies every atom in A. A binding B($\mathcal{I}_p$) satisfies a consequent C iff C is not empty and B($\mathcal{I}_p$) satisfies every

[5]cf. Definition 5.

atom in C. A rule is satisfied by an interpretation $\mathcal{I}_p$ iff for every binding B such that B($\mathcal{I}_p$) satisfies the antecedent, B($\mathcal{I}_p$) also satisfies the consequent.

The semantic conditions relating to axioms and ontologies are unchanged. In particular, an interpretation satisfies an ontology iff it satisfies every axiom (including rules) and fact in the ontology; an ontology is consistent iff it is satisfied by at least one interpretation; an ontology O_2 is entailed by an ontology O_1 iff every interpretation that satisfies O_1 also satisfies O_2.

Example

Consider, for example, the "shipping fee" rule from Section 5.2.1. Assuming that $priceIn\ Dollars$ and $shippingFeeInDollars$ are datavaluedPropertyIDs, SeletedItems is a description, and owlx:integerGreaterThan is a predicate URI, then given an interpretation $\mathcal{I} = \langle R, EC, ER, EP, L, S, LV \rangle$, a binding $B(\mathcal{I})$ extends S to map the variable $?x_1$ to an element of $EC(\text{owl:Thing})$ and extends L to map the variable $?t_1$ to a data value in **LV**; we will use x_1 to denote the element and t_1 to denote the data value. The antecedent of the rule is satisfied by $B(\mathcal{I})$ iff $(x_1, t_1) \in ER(priceInDollars)$, $x_1 \in EC(\text{SeletedItems})$ and $(t_1, L2V(integer)(\text{“50”}\text{^^xsd:integer})) \in EP(\text{owlx:integerGreaterThan})$, where $L2V(integer)$ is the lexical-to-value mapping of $integer$. The consequent of the rule is satisfied by $B(\mathcal{I})$ iff $(x_1, L2V(integer)(\text{“0”}\text{^^xsd:integer})) \in ER(shippingFeeInDollars)$.

Thus the rule is satisfied by $\mathcal{I}$ iff for every binding $B(\mathcal{I})$ such that $(x_1, t_1) \in ER(price\ InDollars)$, $x_1 \in EC(\text{SeletedItems})$ and $(t_1, L2V(integer)(\text{“50”}\text{^^xsd:integer})) \in EP$ (owlx:integerGreaterThan), then it is also the case that $(x_1, L2V(integer)(\text{“0”}\text{^^xsd:integer}))$ $\in ER(shippingFeeInDollars)$, i.e.:

$$\begin{array}{l}\forall x_1 \in EC(\text{owl:Thing}), t_1 \in \mathbf{LV}. \\ \quad ((x_1, t_1) \in ER(priceInDollars) \wedge x_1 \in EC(\text{SeletedItems}) \wedge \\ \quad (t_1, L2V(integer)(\text{“50”}\text{^^xsd:integer})) \in EP(\text{owlx:integerGreaterThan})) \\ \rightarrow (x_1, L2V(integer)(\text{“50”}\text{^^xsd:integer})) \in ER(shippingFeeInDollars)\end{array}$$

5.2.3 SWRL-P vs. SWRL 0.7

In this section, we briefly compare the SWRL-P and SWRL 0.7.[6] SWRL-P follows the syntax of SWRL 0.7, except that SWRL-P allows the use of unsupported predicate URI references as dataPredicateIDs; in this sense, SWRL-P is closer to the OWL datatyping. SWRL 0.7 is based on a naive extension of OWL datatyping. It does not distinguish datatypes from predicates, such that it is not clear whether predicates or builtins can be used with typed literal or not in SWRL 0.7. Furthermore, it does not consider the semantics of negated predicate URIs.

[6] cf. `http://www.daml.org/rules/proposal/`.

Chapter 6

A Fuzzy Extension

The representation and management of uncertainty, imprecision and vague knowledge that exists in real life applications, has received a considerable attention in the AI community in an attempt to extend existing knowledge representation systems to deal with the imperfect nature of real world information. Furthermore, a lot of work have been carried out for the development of reasoning engines that can interpret imprecise knowledge. In this framework, experience in using DL in applications has shown that in many cases we would like to extend the representational and reasoning capabilities of them. For example, the use of DL in the context of multimedia points out the necessity of extending DL with capabilities which allow the treatment of the inherent imprecision in multimedia object representation, matching, detection and retrieval. In fact classical DL are insufficient for describing multimedia retrieval, detection and matching situations, as the situation is usually not only true or false.

One of the most widely used uncertainty theories, that has a very sound and complete mathematical structure, is the theory of fuzzy sets and fuzzy logic [Zad65]. Several ways of extending DL using the theory of fuzzy logic have been proposed in the literature. The preliminary idea, proposed in [Yen] [TM98] is to leave the DL syntax as it is and to use fuzzy logic for extending the interpretation and thus for defining the semantics. A fuzzy interpretation assigns fuzzy sets to concepts and roles. In this way, the interpretation of the Boolean operators and the quantifiers is extended from $\{0,1\}$ to the interval [0,1]. This idea, sufficiently covers the uncertainty introduced in several applications (like for example in video analysis, where the definition of objects is not vague, but the recognition of objects in the real, usually noisy, environment is fuzzy). However, Tresp and Monitor [TM98] also proposed an extension of the syntax by the so-called manipulators, which are unary operators that can be applied to concepts. Intuitively, the manipulators modify the membership degree function of the concepts they are applied to appropriately. Formally, the semantics of a manipulator is defined by a function that maps membership degree functions to membership degree functions. The manipulators considered in [TM98] are, however, of a restricted form. Moreover, the proposed extension of syntax increased the complexity of reasoning. Regarding the reasoning problems in fuzzy DL,

Yen [Yen] considered a crisp subsumption of fuzzy concepts. He described a structural subsumption algorithm for a rather small fuzzy DL, which is almost identical to the subsumption algorithm for the corresponding classical DL. In contrast, Tresp and Monitor are interested in determining fuzzy subsumption between fuzzy concepts (extension with a subsumption degree). In [Str01] and [[TM98], also ABoxes are considered, where the ABox assertions were a matter of degree.

Fuzzy logic provides different options for defining the semantics in the above extension. In [TM98], [Str01] the usual interpretation of conjunction as minimum, disjunction as maximum, negation as *(1-x)*, universal quantifier as *infimum*, and existential quantifier as *supremum* is considered. Both [Str01] and [TM98] contain complete algorithms providing reasoning in the respective fuzzy extension of *ALC*. Although, both algorithms are extensions of the usual tableaux-based algorithm for *ALC*, they differ considerably.

In this chapter, a fuzzy extension of OWL and SWRL is proposed. It is based on the idea of the fuzzification of the interpretation and does not change the syntax of the concept and role constructors. Providing the fuzzy interpretation, it changes the semantics of the above constructors and the interpretation of the rules. The structure of the chapter has as follows. In section 5.2 we give the mathematical background and notations used throughout the chapter. In section 5.3, a general framework for extending DL with the aid of fuzzy logic is given and the semantics of a wide set of concept constructors, role constructors and terminological and assertional axioms is given. In sections 5.4 and 5.5, the extension of fuzzy OWL and SWRL, respectively, is described in more detail, providing the new syntax (for assertion), the new semantics and some examples. Finally, in section 5.6, we describe a way to interpret fuzzy SWRL rules, with the aid of neurofuzzy inference systems.

6.1 Fuzzy set theory preliminaries

In this section we provide the reader with a brief description of the basics of fuzzy set theory and fuzzy logic.For a more complete and comprehensive presentation the authors suggest [KY95].

Let *X* be a finite crisp set with cardinality *m*, i.e $X = \{x_1, x_2, \ldots, x_m\}$ and let *A* be a fuzzy subset of *X*, with membership function $\mu_A(x)$, or simply *A(x)*, $x \in X$.

Height h(A), of *A* is the maximum membership grade of *A*, i.e.

$$h(A) = \sup_{x \in X} A(x)$$

We say that *A* is *normal* if and only (iff) $h(A) = 1$. If $h(A) < 1$, the fuzzy set *A* is said to be *subnormal*.

Support, Supp(A), of *A* is the crisp set which contains all the elements of *X* that have non-zero membership grades in *A*, i.e. $S(A) = \{x \in X \mid A(x) \neq 0\}$.

The *scalar cardinality* $\mid A \mid$, of *A* is defined as

$$\mid A \mid = \sum_{x \in X} A(x)$$

The *certainty subset,* $CS(A)$, of *A* is defined as the crisp set $CS(A) = \{x \in X \mid A(x) = 1\}$ and the *uncertainty subset,* $US(A)$, as the crisp set $US(A) = \{x \in X \mid 0 < A(x) < 1\}$. Obviously, it is $US(A) = S(A) - CS(A)$.

The *fuzzy powerset, $\mathcal{F}(X)$,* of the universe of discourse, X, is the set of all the fuzzy subsets of X. Let now $A, B \in \mathcal{F}(X)$. We say that *A* equals *B* iff $A(x) = B(x), \forall x \in X$. We say that *a* is a subset of *B* and denote $A \subseteq B$ iff *A(x)* $\leq$ *B(x),* $\forall x \in X$. If also $A \subseteq B$ and $A \neq B$, then *A* is a strict subset of *B.*

We will now give the basic theoretic-set operations (complement, intersection and union) defined on fuzzy sets.

The complement $\neg A$ of a fuzzy set *A* is given by $(\neg A)(x) = c(A(x))$ for any $x \in X$. The function *c* satisfies the following conditions in order to normally extend the nature of the standard logic complement:

Boundary conditions: $c(0) = 1$ *and* $c(1) = 0$

Monotonicity: $\forall a, b, \in [0, 1], a \leq b \Rightarrow c(a) \geq c(b)$

Continuity: c *continuous in* $[0, 1]$

Involution: $\forall a \in [0, 1]$ *it is* $c(c(a)) = a$

Several fuzzy complements have been defined in the literature. The standard complement is given by $(\neg A)(x) = 1 - A(x), \forall x \in X$. One example of parametric class of fuzzy complements is the *Sugeno class* defined by $c_\lambda(a) = \frac{1-a}{1+\lambda a}$, where $\lambda \in (-1, \infty)$.

The intersection of two fuzzy sets *A* and *B* is given by $(A \cap B)(x) = t[A(x), B(x)]$ where *t* is a triangular norm (t-norm). A t-norm is a function that satisfies the following conditions:

Boundary condition: $t(a, 1) = a$

Monotonicity: $\forall a, b, d \in [0, 1]$, *with* $b \leq d$ *is* $t(a, b) \leq t(a, d)$

Commutativity: $\forall a, b \in [0, 1]$ *is* $t(a, b) = t(b, a)$

Associativity: $\forall a, b, d \in [0, 1]$ *is* $t(a, t(b, d)) = t(t(a, b), d)$

Moreover, it is called *Archimedean* iff it is continuous and $t(a, a) < a$, $\forall a \in (0, 1)$. Obviously, the *only* continuous t-norm which is not Archimedean is the $min(a, b)$.

Examples of t-norm widely used in the literature are the following:

Standard intersection: $t(a, b) = min(a, b)$

Algebraic product: $t(a, b) = ab$

Bounded difference: $t(a, b) = max(0, a + b - 1)$

Hamacher's Function: $t(a,b) = \frac{ab}{r+(1-r)(a+b-ab)}$

The t-norm operation is also used for the definition of the cartesian product $A = A_1 \times A_2 \times \ldots \times A_n$ of *n* fuzzy subsets of *X* as a fuzzy subset of the cartesian product $X = X_1 \times X_2 \times \ldots \times X_n$ using:

$$A(x_1, x_2, \ldots, x_n) = t[A_1(x_1), A_2(x_2), \ldots, A_n(x_n)]$$

with $x_i \in X_i$, $i \in N_n$

The fuzzy union of two fuzzy sets is defined analogously to fuzzy intersection using a t-conorm *u*, a function that satisfies the following conditions.

Boundary condition: $u(a,0) = a$

Monotonicity: $\forall a, b, d \in [0,1]$, *with* $b \leq d$ *is* $u(a,b) \leq u(a,d)$

Commutativity: $\forall a, b \in [0,1]$ *is* $u(a,b) = u(b,a)$

Associativity: $\forall a, b, d \in [0,1]$ *is* $u(a, u(b,d)) = u(u(a,b),d)$

Examples of t-conorms are the following:

Standard union: $u(a,b) = max(a,b)$

Algebraic sum: $u(a,b) = a + b - ab$

Bounded sum: $u(a,b) = min(1, a+b)$

The operations of fuzzy complement, intersection and union extend classical logic into fuzzy logic. Every fuzzy powerset *F(X)* can be considered as a lattice in which the t-norm plays the role of the meet (infimum), while the t-conorm plays the role of the join (supremum). Generally speaking, given a t-norm, there is always a fuzzy complement and a fuzzy union such that the lattice is distributed and complemented under this triple and thus it is a *De Morgan algebra*.

Another important operation, used in fuzzy logic is the *fuzzy implication*, that gives a truth value to the predicate $A \Rightarrow B$ when the truth values of the predicates *A* and *B* are known. Actually, it is an extension of the standard implication since it represents clauses of the form "if A then B". A fuzzy implication is a function ω of the form $\omega : [0,1] \times [0,1] \rightarrow [0,1]$. In standard logic it is implemented using the formula $\omega(a,b) = \bar{a} \vee b$ or $\omega(a,b) = max\{x \in 0,1 \mid a \wedge x \leq b\}$. Extending this definition in fuzzy logic, operation $\omega_{u,c}(a,b) = u(c(a),b)$ is defined, where *u* and *c* is a fuzzy union and a fuzzy complement, respectively. Alternatively, fuzzy implication can be defined using

$$\omega_t(a,b) = \sup\{x \in [0,1] : t(a,x) \leq b\} \tag{6.1}$$

where *t* is a t-norm.

Let $X_1, X_2, \ldots, X_d$ be crisp sets. A *fuzzy relation*, $R : X_1 \times X_2 \times \ldots \times X_d \rightarrow [0,1]$ is defined as a fuzzy subset of the cartesian product $X_1 \times X_2 \times \ldots \times X_d$. The membership degree of each element vector $(x_1, x_2, \ldots, x_d) \in X_1 \times X_2 \times, \ldots, X_d$ in the fuzzy relation

R is the degree in which $x_1, x_2, \ldots, x_d$ are related in terms of *R*. The representation of fuzzy relations by matrices $R = [r_{ij}]$ is used in the case that the universe of discourses are finite.

The basic operations defined on fuzzy relations are the *inverse* and the *composition*. The inverse relation of $R(X, Y)$ is the fuzzy relation $R^{-1}(Y, X)$ with $R^{-1}(y, x) = R(x, y)$ for every $x \in X$ and $y \in Y$. The membership matrix that represents R^{-1} is the inverse matrix of *R*. The $\sup -t$ composition of two fuzzy relations $R_1 : X \times Y \rightarrow [0, 1]$ and $R_2 : Y \times Z \rightarrow [0, 1]$ is defined by

$$[R_1 o^t R_2](x, z) = \sup_{y \in Y} t[R_1(x, y), R_2(y, z)], \tag{6.2}$$

while the $\inf -\omega_t$ composition is defined by

$$[R_1 o^{\omega_t} R_2](x, z) = \inf_{y \in Y} \omega_t[R_1(x, y), R_2(y, z)] \tag{6.3}$$

where *t* is a t-norm.

All the properties of crisp relations are extended for fuzzy relations. We will now give the extensions for reflexive, symmetric and transitive relations. A fuzzy relation *R* is *reflexive* iff $R(x, x) = 1$ for all $x \in X$. Moreover, it is *symmetric* iff $R(x, y) = R(y, x)$ for all $x, y \in X$. It is also *sup-t transitive* iff

$$R(x, z) \geq \sup_{y \in Y} t[R(x, y), R(y, z)].$$

It has been proved (Klir 1995) that if a fuzzy relation *R* defined on X^2 with $|X| = n \geq 2$, is reflexive, then $R_T = R^{(n-1)}$, where R_T is the transitive closure of *R*.

6.2 Fuzzy Description Logics

In this section, we give the syntax and semantics of a fuzzy DL, using the fuzzy operators defined in the previous section. The fuzzy DL described here is based on the definition of the *fuzzy interpretation*. A fuzzy interpretation $\mathcal{I}$ consists of a non empty set $\Delta^{\mathcal{I}}$ and the mapping functions:

$$C^{\mathcal{I}} : \Delta^{\mathcal{I}} \longrightarrow [0, 1]$$

$$R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \longrightarrow [0, 1]$$

assigning fuzzy sets to concepts and roles, respectively. For example if $\alpha \in \Delta^{\mathcal{I}}$ then $A^{\mathcal{I}}(a)$ gives the degree that the object a belongs to the fuzzy concept A, i.e $A^{\mathcal{I}}(a) = 0.8$.

Tables 5.1, 5.2 and 5.3 summarise the syntax and the semantics of some constructors, role constructors and terminological and assertional axioms. The first column provides

the name of the constructor, the second its syntax and the third its semantics. Although the details on the definition of the semantics are beyond the scope of this chapter, the previous section gives all the necessary background needed for fully understand the underlying logic of this proposal.

Table 6.1: Concept Constructors

Name	Syntax	Semantics $(a \in \Delta^{\mathcal{I}})$
Top	$\top$	$\top^{\mathcal{I}}(a) = 1$
Bottom	$\bot$	$\bot^{\mathcal{I}}(a) = 0$
Fuzzy Intersection	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}}(a) = t(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))$
Fuzzy Union	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}}(a) = u(C^{\mathcal{I}}(a), D^{\mathcal{I}}(a))$
Fuzzy negation	$\neg C$	$(\neg C)^{\mathcal{I}}(a) = c(C^{\mathcal{I}}(a))$
Fuzzy Value Restriction	$\forall R.C$	$(\forall R.C)^{\mathcal{I}}(a) = \inf_{b \in \Delta^{\mathcal{I}}} w_t(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))$
Fuzzy existential quantifier	$\exists R.C$	$(\exists R.C)^{\mathcal{I}}(a) = \sup_{b \in \Delta^{\mathcal{I}}} t(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))$
Unqualified Number Restriction	$\geq nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid Supp[R^{\mathcal{I}}(a,b)] \mid \geq n\}$
	$\leq nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid Supp[R^{\mathcal{I}}(a,b)] \mid \leq n\}$
	$= nR$	$\{a \in \Delta^{\mathcal{I}} \mid \mid Supp[R^{\mathcal{I}}(a,b)] \mid = n\}$
Qualified Number Restriction	$\geq nR.C$	$\{a \in \Delta^{\mathcal{I}} \mid \mid Supp[t(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))] \mid \geq n\}$
	$\leq nR.C$	$\{a \in \Delta^{\mathcal{I}} \mid \mid Supp[t(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))] \mid \leq n\}$
	$= nR.C$	$\{a \in \Delta^{\mathcal{I}} \mid \mid Supp[t(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))] \mid = n\}$
Role Value Map	$R \subseteq S$	$(R \subseteq S)(a) = \inf_{b \in \Delta^{\mathcal{I}}} \omega_t(R^{\mathcal{I}}(a,b), S^{I}(a,b))$
	$R = S$	$(R = S)(a) = \inf_{b \in \Delta^{\mathcal{I}}} t(\omega_t(R^{\mathcal{I}}(a,b), S^{\mathcal{I}}(a,b)), \omega_t(S^{\mathcal{I}}(a,b), R^{\mathcal{I}}(a,b))$
Nominal	I	$h(\mathcal{I}) = 1 \, and \mid Supp(\mathcal{I}) \mid = 1$

In addition to terminology and world description, fuzzy rules can be used to express imprecise knowledge. In general, rules have the form:

$$a_1 \wedge a_2 \wedge \ldots \wedge a_n \Rightarrow b$$

where a_i ($i = 1, 2, \ldots, n$) and *b* are fuzzy predicates.

The semantics of the above fuzzy rule are given by:

$$b = t(a_1, a_2, \ldots a_n)$$

where *t* is a t-norm. Details on fuzzy rules are given in the next sections.

Let us now provide an important characterisation framework for fuzzy DLs.

Definition 5.1. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathcal{R})$ be a knowledge base, where $\mathcal{T}$ is the T-box, $\mathcal{A}$ is the A-box and $\mathcal{R}$ is a set of rules. We say that $\tilde{\mathcal{K}} = (\tilde{\mathcal{T}}, \tilde{\mathcal{A}}, \tilde{\mathcal{R}})$ is an assertional fuzzy*

Table 6.2: Role Constructors

Name	Syntax	Semantics
Universal role	$\cup$	$U^{\mathcal{I}}(a,b) = 1$
Fuzzy intersection	$R \sqcap S$	$(R \cap S)^{\mathcal{I}}(a,b) = t(R^{\mathcal{I}}(a,b), S^{\mathcal{I}}(a,b))$
Fuzzy Union	$C \sqcup D$	$(R \cup S)^{\mathcal{I}}(a,b) = u(R^{\mathcal{I}}(a,b), S^{\mathcal{I}}(a,b))$
(Fuzzy) Complement	$\neg R$	$(\neg R)^{\mathcal{I}}(a,b) = c(R^{\mathcal{I}}(a,b))$
(Fuzzy) inverse	R^-	$(R^-)^{\mathcal{I}}(a,b) = R^{\mathcal{I}}(b,a)$
Sup-t composition	$R \circ S$	$(R \circ^t S)^{\mathcal{I}}(a,b) = \sup_{a \in \Delta^{\mathcal{I}}} t(R^I(a,d), S^{\mathcal{I}}(d,b))$
inf-w_t composition		$(R \circ^{w_t} S)^{\mathcal{I}}(a,b) = \inf_{a \in \Delta^{\mathcal{I}}} w_t(R^{\mathcal{I}}(a,d), S^{\mathcal{I}}(d,b))$
t-Transitive closure	R^+	$(R^+)^{\mathcal{I}}(a,b) = R^{\mathcal{I}}(a,b) \circ^t \ldots \circ^t R^{\mathcal{I}}(a,b)\ (n-1\ times)$
Role Restriction	$R \mid_C$	$(R \mid_C)^{\mathcal{I}}(a,b) = t(R^{\mathcal{I}}(a,b), C^{\mathcal{I}}(b))$
Identity	$id(C)$	$(ID \mid_C)^{\mathcal{I}}(a,a) = 1$ if $a \in S(C^{\mathcal{I}})$
		$(ID \mid_C)^{\mathcal{I}}(a,a) = 0$ otherwise

Table 6.3: Terminological and Assertional Axioms

Name	Syntax	Semantics
Concept Inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}} (\forall a \in \Delta^{\mathcal{I}} \mid C^{\mathcal{I}}(a) \leq D^{\mathcal{I}}(a))$
Role Inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
		$(\forall (a,b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(a,b) \leq S^{\mathcal{I}}(a,b))$
Concept Equality	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
		$(\forall a \in \Delta^{\mathcal{I}} \mid C^{\mathcal{I}}(a) = D^{\mathcal{I}}(a))$
Role Equality	$R \equiv S$	$R^{\mathcal{I}} = S^{\mathcal{I}} (\forall (a,b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
		$\mid R^{\mathcal{I}}(a,b) = S^{\mathcal{I}}(a,b))$
Concept Assertion	$C(a)$	$(C(a))^{\mathcal{I}}(a) = C^{\mathcal{I}}(a) > 0$
Role Assertion	$R(a,b)$	$(R(a,b))^{\mathcal{I}}(a,b) = R^{\mathcal{I}}(a,b) > 0$

extension of $\mathcal{K}$ iff $\mathcal{T} \equiv \tilde{\mathcal{T}}$, $\mathcal{R} \equiv \tilde{\mathcal{R}}$ and $\mathcal{A} = S(\tilde{\mathcal{A}})$, where $Supp(\tilde{\mathcal{A}})$ contains the support of any assertion of the A-box. Let also $\mathcal{K}' = (\mathcal{T}', \mathcal{A}', \mathcal{R}')$ be the implicit knowledge of $\mathcal{K}$ that can be explicit with the aid of a reasoner. Then, if $\tilde{\mathcal{K}}' = (\tilde{\mathcal{T}}', \tilde{\mathcal{A}}', \tilde{\mathcal{R}}')$ is the implicit knowledge that can be extracted by a fuzzy reasoner, we say that the assertional fuzzy extension $\tilde{\mathcal{K}}$ is valid iff it is $\mathcal{T}' \equiv \tilde{\mathcal{T}}'$, $\mathcal{R}' \equiv \tilde{\mathcal{R}}'$ and $\mathcal{A}' = Supp(\tilde{\mathcal{A}}')$.

6.3 Fuzzy OWL

The concepts and the roles in classical OWL are interpreted as crisp sets, i.e an individual either belongs to the set or not. However, many real-life concepts are vague in the sense that they do not have precisely defined membership criteria. In fuzzy OWL an individual belongs to a degree of confidence to the set (membership). This means that, for example, the individual ”Peter” might belong to the degree of confidence of ”0.8” to the concept set ”TallPerson”. In classical OWL the example is represented as

$$Individual(Peter) = Type(TallPerson)$$

in fuzzy OWL the same example can be represented as

$$Individual(Peter) = [Type(TallPerson),\ 0.8]$$

Therefore, in order to extend the existed OWL syntax to support the fuzzy assertion, a membership value must be added in the individual OWL constructor. The syntax is as follows:

$$Individual(a) = [type(conceptID),\ membership]$$

where the membership $\in [0, 1]$

individual ::= ’Individual(’ [individualID] {annotation} {’type’ ’description’}
[membership]’)’
value ::= ’value(’ individualvaluedPropertyID individualID ’)’
| ’value(’ individualvaluedPropertyID individual ’)’
| ’value(’ datavaluedPropertyID dataLiteral ’)’

The value range of the membership value is [0,1].The membership value is omitted if the value is 1 since this value correspond to the classical assertion.

6.4 Fuzzy SWRL

The concepts and the roles in classical OWL are interpreted as crisp sets, i.e an individual either belongs to the set or not. However, many real-life concepts are vague in the sense that they do not have precisely defined membership criteria. In fuzzy OWL DL that we propose here an individual belongs to a degree of confidence to the set (membership). This means that, for example, the individual ”Peter” might belong to the degree of confidence of ”0.8” to the concept set ”TallPerson”. In classical OWL the example is represented as

Individual(Peter type(TallPerson))

in fuzzy OWL the same example can be represented as

Individual(Peter type(TallPerson) degree(0.8))

Therefore, in order to extend the existed OWL syntax to support the fuzzy assertion, a membership value must be added in the individual OWL constructor. The abstract syntax of fuzzy assertion is as follows:

individual ::= 'Individual(' [individualID] { annotation } { 'type(' type ')' }
{ value } { 'degree(' membership ')' } ')'

where the value range of membership is [0,1], while annotation, type and value are defined the same as in [**?**]. The membership value can be omitted if the value is 1 since this value correspond to the classical assertion.

A SWRL ontology contains a mixture of rules and other OWL DL constructs, including ontology annotations, axioms about classes and properties, and facts about OWL individuals, as well as the rules themselves.

A SWRL rule axiom consists of an antecedent (body) and a consequent (head), each of which consists of a set of atoms.

rule ::= 'Implies(' { annotation } antecedent consequent ')'
antecedent ::= 'Antecedent(' { atom } ')'
consequent ::= 'Consequent(' { atom } ')'

In fuzzy SWRL a weight representing the degree of importance of an atom is added. The abstract syntax of atom is modified as follows:

atom ::= description '(' i-object [weight]')'
| individualvaluedPropertyID '(' i-object i-object [weight]')'
| datavaluedPropertyID '(' i-object d-object [weight]')'
| sameAs '(' i-object i-object [weight]')'
| differentFrom '(' i-object i-object [weight]')'

where i-object and d-object are defined the same as in SWRL.

A rule now means that if the antecedent is activated to a degree of confidence (membership) $a \in [0, 1]$, and has a degree of importance (weight) $b \in [0, 1]$ then the consequent must also hold to a degree of confidence (membership) $c \in [0, 1]$ that can be computed from a and b with the aid of fuzzy operators. An empty antecedent is treated as trivially holding (true), and an empty consequent is treated as trivially not holding (false). Non-empty antecedents and consequents hold iff all of their constituent atoms hold. Atoms in fuzzy rules can be of the form C(x), P(x,y), Q(x,z), sameAs(x,y) or differentFrom(x,y), where C is an OWL DL. The output of a rule is a consequent equipped with a degree of confidence. The degree of importance of an antecedent atom, is omitted in case the value is 1 or 0 where we have the classical case of a SWRL rule.

Fuzzy Rules Interpretation

As previously defined, an abstract fuzzy OWL interpretation is a tuple of the form

$$\mathcal{I} = \langle R, EC_f, ER_f, L, S, LV \rangle,$$

where R is a set of resources, $LV \subseteq R$ is a set of literal values, EC is a mapping equipped with a degree of confidence denoting the fuzzy concept assertion, from classes and datatypes to subsets of R and LV respectively, ER is a mapping equipped with a degree of confidence denoting the fuzzy role assertion, from properties to binary relations on R, L is a mapping from typed literals to elements of LV, and S is a mapping from individual names to elements of $EC(\mathsf{owl:Thing})$ equipped with a degree of confidence as shown in the previous section.

Given an abstract OWL interpretation $\mathcal{I}$, a binding $B(\mathcal{I})$ is an abstract OWL interpretation that extends $\mathcal{I}$ such that S maps i-variables to elements of $EC(\mathsf{owl:Thing})$ and L maps d-variables to elements of LV respectively.

Atom	Condition on Interpretation
$C(x)$	$[EC(C)](S(x)) > 0$
$P(x, y)$	$[ER(P)](S(x), S(y)) > 0$
$Q(x, z)$	$[ER(Q)](S(x), L(z)) > 0$
sameAs(x, y)	$S(x) = S(y)$
differentFrom(x, y)	$S(x) \neq S(y)$

Table 6.4: Interpretation Conditions

Example

Consider the rule "If a person(*a*) has its eyebrows raised enough and his mouth open then is happy". Assuming that EyebrowsRaised, MoutOpen and Happy are ClassesIDs, then given an interpretation $\mathcal{I} = \langle R, EC_f,$
$ER_f, L, S, LV \rangle$, a binding $B(\mathcal{I})$ extends S to map the variable $?a$ to elements of $EC(\mathsf{owl:Thing})$; we will use a respectively to denote these elements. The antecedent of the rule is satisfied by $B(\mathcal{I})$ iff $(a) \in EC(\mathsf{EyebrowsRaised})$ with the degree of importance "0.8" and $(a) \in EC(\mathsf{OpenMouth})$ with the degree of importance "1". The consequent of the rule is satisfied by $B(\mathcal{I})$ iff $(a) \in EC(\mathsf{Happy})$. Thus the rule, as in classical case is satisfied by $\mathcal{I}$ iff for every binding $B(\mathcal{I})$ such that $(a) \in EC(\mathsf{EyebrowsRaised})$ and $(a) \in EC(\mathsf{MouthOpen})$, then it is also the case that $(a) \in EC(\mathsf{Happy})$, i.e.:

$$\begin{aligned}
&\forall a \in EC_f(\mathsf{owl:Thing}).\\
&\ t(t(EC_f(\mathsf{EyebrowsRaised})(a),\ 0.8), t(EC_f(\mathsf{MouthOpen})(a),\ 1))\\
&\ = (EC_f(\mathsf{Happy})(a),\ 1)
\end{aligned}$$

where *t* is a *t*-norm. The value 0.8 is the degree of importance(weight) of the antecedent atom, “RaiseEybrow”. This value corresponds to the degree that the eyebrows must be raised in order to detect that the individual(*a*) is happy. The degree of importance (weight) of the ”openMouth” can be omitted since it has the value 1. In order to compute this rule the assertion degree (membership) of the atoms is needed. For example, the assertion of the specific atoms might be:

Individual(a) = [Type(RaiseEyebrows), 0.9]
and
Individual(a) = [Type(OpenMouth), 0.5]

The difference between the degree of confidence and the degree of importance is that the first value (0.9) shows the degree that *a* belongs the set RaiseEyebrows, and the second value (0.8) shows how important is the antecedent atom RaiseEyebrow in order to detect the expression Happy.

The syntax of the rule has as follows:

Implies(
Antecedent($EyebrowsRaised$(I-variable(a) 0.8),
$MouthOpen$(I-variable(a) 1))
Consequent($Happy$(I-variable(a) 1))

In human readable syntax, this rule can be written as
$EyebrowsRaised(?a$ “0.8”) $\wedge$ MouthOpen($?a$ “1”)
$\rightarrow Happy(?a$ “1”)

In XML syntax the rule can be written as

```
<owlx:Rule>
    <owlr:antecedent>
    <owlr:ClassAtom>
      <owlr:Class owlr:name = ”RaiseEyebrows”>
        <owlr:Variable owlr:name=”a” />
      <owlx:weight owlx:datatype=”xsd;float”>0.8 />
    </ owlr:ClassAtom>
    < owlr:ClassAtom >
      <owlr:Class owlr:name=”MouthOpen” >
        < owlr:Variable owlr:name ”a” />
    </owlr:ClassAtom>
    </owlr:antecedent>
    <owlr:consequent>
    <owlr:ClassAtom>
      <owlr:Class owlr:name=”Happy” >
        <owlr:Variable owlr:name=”a” />
    </owlr:ClassAtom>
```

</owlr:consequent>
</owlr:Rule>

6.5 Reasoning in SWRL

The main difference between classical propositions and fuzzy propositions is in the range of their truth values. While each classical proposition is required to be either true or false, the truth or falsity of fuzzy propositions is a matter of degree. Assuming that truth and falsity is expressed by values 0 or 1, respectively, the degree of truth of each fuzzy proposition is expressed by a number in the unit interval [0,1]. There are various fuzzy propositions, which are classified into the following four types.

1. Unconditional and unqualified propositions
2. Unconditional and qualified propositions
3. Conditional and unqualified propositions
4. Conditional and qualified propositions

In this chapter we will discuss the interpretation of the third type, conditional and unqualified propositions.

Propositions *p* of the conditional and unqualified type are expressed by the canonical form

p: If *X* is *A*, $\rightarrow$ *Y* is *B*,

where *X*, *Y* are variables whose values are in the sets *X*, *Y*, respectively, and *A*, *B*, are fuzzy sets on *X*, *Y*, respectively. These propositions may also be viewed as propositions of the form

$\langle X, Y \rangle \in R$

where *R* is a fuzzy set on $X \times Y$ that is determined for each $x \in X$ and each $y \in Y$ by the formula

$$R(x, y) = \omega[A(x), B(y)], \tag{6.4}$$

where *R* expresses the relationship between the variables *X* and *Y* involved in the given fuzzy proposition. For each $x \in X$ and each $y \in Y$, the membership grade $R(x, y)$ represents the truth value of the proposition

p_{xy}: If *X=x*, Then *Y=y*

Now, the truth values of the propositions "$X = x$" and "$Y = y$" are expressed by the membership grades $A(x)$ and $B(y)$, respectively. Consequently, the truth value of the proposition p_{xy} , given by $R(x, y)$, involves a fuzzy implication in which $A(x)$ is the truth value of the antecedent and $B(y)$ is the truth value of the consequent.

Assume that *R* is a fuzzy relation on $X \times Y$ and A', B' are fuzzy sets on *X* and *Y*, respectively. Then if R and A' are given we can obtain B' by the equation

$$B'(y) = \sup_{x \in X} t[A'(x), R(x, y)] \tag{6.5}$$

for all $y \in Y$. This equation, which can also be written in the matrix from as

$$B' = A' \circ R, \tag{6.6}$$

is called the compositional rule of inference. This procedure is called the generalized fuzzy modus ponens.

The fuzzy relation employed in Eq.(6.5) is usually not given directly, but in some form. In the case that the relation is embedded in a single conditional fuzzy proposition, then is determined using the fuzzy implication operator, Eq.(6.4). A more general case, in which the relation emerges from several conditional fuzzy propositions, is as follows:

Rule 1: If X is A_1, Then Y is B_1
Rule 2: If X is A_2, Then Y is B_2
Rule 3: If X is A_3, Then Y is B_3
.....
Rule n: If X is A_n, Then Y is B_n

As previously described, any conditional (If-Then) fuzzy proposition can be expressed in terms of a fuzzy relation R between the two variables involved. One way to determine R is using the fuzzy implication, which operates on fuzzy sets involved in the fuzzy proposition. However, the problem of determining R for a given conditional fuzzy proposition can be detached from fuzzy implications and determine R using fuzzy relational equations.

As described, the equation to be solved for fuzzy modus ponens has the form

$$B = A \circ^t R, \tag{6.7}$$

where A and B are given fuzzy sets that represent, respectively, the IF-part and the THEN-part in the conditional fuzzy proposition involved and *t* is a *t*-norm.It will be proved in the following section that Eq.(6.7) is solvable for *R* if $A \circ^{\omega_t} B$ is a solution.

In the following section we present a complete algorithm for solving fuzzy relational equations for the interpretation of inference rules in the respective fuzzy extension of propositional logics. The proposed interpretation algorithm is realized using a hybrid neurofuzzy architecture.

Fuzzy systems are numerical model-free estimators. While neural networks encode sampled information in a parallel-distributed framework, fuzzy systems encode structured, empirical (heuristic) or linguistic knowledge in a similar numerical framework [KY95]. Although they can describe the operation of the system in natural language with the aid of human-like if-then rules, they do not provide the highly desired characteristics

of learning and adaptation. The use of neural networks in order to realize the key concepts of a fuzzy logic system enriches the system with the ability of learning and improves the susymbolic to symbolic mapping [CTL95].

The proposed neurofuzzy network is supported by an adaptation algorithm. This algorithm uses predefined input-output data to i) initiate and ii) adapt the weights of the fuzzy propositional rules. It is important to state that using the adaptation algorithm we are not altering the knowledge or generating new knowledge, but we refine the existed knowledge to achieve the optimal behaviour. Finally, using the adaptation algorithm we incorporate uncertainty by using degrees of confidence. The degree of confidence measures the belief of the existence of the specific concept or relation, since real-life applications involve uncertainty and fuzzy hypothesis.

Neurofuzzy Network

Let $y = [y_1, y_2, ..., y_m]$ denote a fuzzy set defined on the set of output predicates, the truth of which will be examined. Actually, each y_i represents the degree in which the *i*-th output fuzzy predicate is satisfied. The input of the proposed neurofuzzy network is a fuzzy set $x = [x_1, x_2, ..., x_n]$ defined on the set of the input predicates, with each x_i representing the degree in which the *i*-th input predicate is detected. The proposed network represents the association $f : X \rightarrow Y$ which is the knowledge of the system, in a neurofuzzy structure. After the evaluation of the input predicates, some output predicates represented in the knowledge of the system can be recognized with the aid of fuzzy systems' reasoning [KY95]. One of the widely used ways of constructing fuzzy inference systems is the method of approximate reasoning which can be implemented on the basis of compositional rule of inference [KY95]. The need for results with theoretical soundness lead to the representation of fuzzy inference systems on the basis of generalized sup-*t* norm compositions [GS99], [HP96].

The class of *t*-norms has been studied by many researchers [HP96], [CTL95]. Using the definition ω_t in Eq.(6.1) two additional operators $\hat{\omega}_t, \check{\omega}_t : [0,1] \times [0,1] \rightarrow [0,1]$, are defined by the following relations:

$$\hat{\omega}_t(a,b) = \begin{cases} 1 & a < b \\ a \hat{\otimes}^t b & a \geq b \end{cases} \tag{6.8}$$

$$\check{\omega}_t(a,b) = \begin{cases} a \check{\otimes}^t b & a \geq b \\ 1 & a < b \end{cases} \tag{6.9}$$

where $a \hat{\otimes}^t b = \sup\{x \in [0,1] : \ t(a,x) = b\}$, $a \check{\otimes}^t b = \inf\{x \in [0,1] : \ t(a,x) = b\}$.

With the aid of the above operators, compositions of fuzzy relations can be defined. These compositions are used in order to construct fuzzy relational equations and represent the rule-based symbolic knowledge with the aid of fuzzy inference [VT03]. Let *X, Z, Y* be three discrete crisp sets with cardinalities *n, l* and *m* respectively, and $A(X,Z)$, $B(Z,Y)$,

be two binary fuzzy relations. The definitions of sup-*t* and inf-$\breve{\omega}_t$ compositions are given in Eq.(6.2,6.3)

Let us now proceed to a more detailed description of the proposed neurofuzzy architecture Fig. 6.1. It consists of two layers of compositional neurons which are extensions of the conventional neurons [GS99]. While the operation of the conventional neuron is described by the equation:

$$y = a\left(\sum_{i=1}^{n} w_i x_i + \vartheta\right) \tag{6.10}$$

where a is non-linearity, ϑ is threshold and w_i are the weights, the operation of the sup-*t* compositional neuron is described by the equation:

$$y = a'\left\{\sup_{j\in N_n} t(x_i, w_i)\right\} \tag{6.11}$$

where *t* is a *t*-norm and is the following activation function

$$a'(z) = \begin{cases} 0 & x \in (-\infty, 0) \\ x & x \in [0, 1] \\ 1 & x \in (0, +\infty) \end{cases} \tag{6.12}$$

A second type of compositional neuron is constructed using the $\hat{\omega}_t$ operation. The neuron equation is given by:

$$y = a'\left\{\inf_{j\in N_n} \hat{\omega}_t(x_i, w_i)\right\} \tag{6.13}$$

The proposed architecture is a two-layer neural network of compositional neurons Fig. 6.1. The first layer consists of the inf-$\hat{\omega}_t$ neurons and the second layer consists of the sup-*t* neurons. The system takes as input, predicates, and gives to the output the recognized output predicates. The first layer computes the antecedents of the mapping rules, while the second implements the fuzzy reasoning using the fuzzy modus ponens schema.

The rules are used to initialize the neurofuzzy network (giving its initial structure and weights). During the learning process the number of neurons in the hidden layer and the weights of the two layers may change with the aid of a learning with the objective of the error minimization. The learning algorithm that supports the above network is applied in each layer independently. During the learning process, the weight matrices are adapted in order to approximate the solution of the fuzzy relational equation describing the association of the input with the output. Using a traditional minimization algorithm (for example the steepest descent), we cannot take advantage of the specific character of the problem. The algorithm that we use is based on a more sophisticated credit assignment that "blames" the neurons of the network using the knowledge about the topographic structure of the solution of the fuzzy relation equation [GS99]. After the learning process, the network keeps its transparent structure and the new knowledge represented in it can be extracted in the form of mapping If-Then rules.

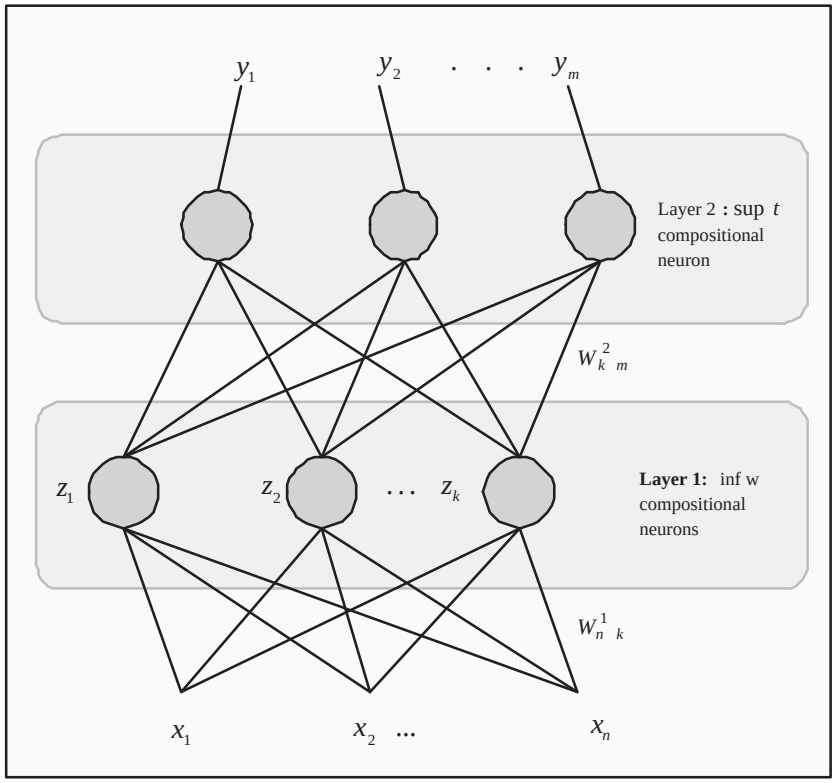

Figure 6.1: The Neurofuzzy Layers

Learning Operation

In the process of knowledge adaptation, the If-Then rules are inserted into the proposed neurofuzzy system. This refers to automatically transforming the structured knowledge provided by the knowledge base in order to perform the followings:

1. Define the required input predicates as *"input predicate(1), input predicate(2),..., input predicate(n)"*. The input predicates will define the set $X = \{x_1, x_2, ..., x_n\}$

2. Define the required output predicates as *"output predicate(1), output predicate(2),..., output predicate(n)"*. The output predicates will define the set $Y = \{y_1, y_2, ..., y_n\}$.

3. Insert the a priori knowledge given in If-Then rules of the form *"if input predicate(1) and input predicate(2) and then output predicate(5)"* into the neurofuzzy structural elements (the weights of the neurofuzzy system). The number of different antecedents (If parts of the rules) defines the set $Z = \{z_1, z_2, ..., z_n\}$. The predicates could be associated with confidence levels in order to produce the antecedents; this means that the antecedents could have the form (*input predicate(1), input predicate(2)*, 0.7, 0.9), with the 0.7 and 0.9 values corresponding to confidence levels.

The above degrees are used in order to define the weights $W_{ij}^1,\ i \in N_n,\ j \in N_l$ of the first layer. Furthermore, the consequences could also be associated with confidence levels, i.e. *"if input predicate(1) and input predicate(2) and then output predicate(5)"* with confidence 0.7". These values are used in order to define the weights $W_{ij}^2\ i \in N_l,\ j \in N_m$ of the second layer.

The knowledge refinement provided by the proposed neurofuzzy system will be now described. Let $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$ be the input and output, respectively, predicate sets and let also $R = \{r_1, r_2, ..., r_n\}$ be the set of rules describing the knowledge of the system. The set of antecedents of the rules is denoted by $Z = \{z_1, z_2, , z_n\}$ (see the structure of the neurofuzzy system given in Fig. 6.1). Suppose now that a set of input-output data $D = \{A_1, B_1),\ i \in N_q\}$, where $A_i \in F(X)$ and $B_i \in F(Y)$ ($F((*))$ is the set of fuzzy sets defined on $(*)$), is given sequentially and randomly to the system (some of them are allowed to reiterate before the first appearance of some others). The data sequence is described as $(A^{(q)}, B^{(q)})$, $q \in N$, where $(A^{(i)}, B^{(i)}) \in D$. The problem that arises is finding of the new weight matrices $W_{ij}^1,\ i \in N_n,\ j \in N_l$ and $W_{ij}^2,\ i \in N_l,\ j \in N_m$ for which the following error is minimised:

$$\varepsilon = \sum_{i \in N_q} \parallel B_i - y^i \parallel \tag{6.14}$$

where $y^I, i \in N_q$ is the output of the network when the input A_1 is given. The process of the minimisation of the above error is based on the resolution of the following fuzzy relational equations:

$$W^1 \circ^{\hat{\omega}_t} A = Z \tag{6.15}$$

$$Z \circ^t W^2 = B \tag{6.16}$$

where *t* is a continuous *t*-norm and Z is the set of antecedents fired when the input A is given to the network.

For the resolution of the above problem the adaptation process changes the weight matrices W^1 and W^2 in order to approximate a solution of the above fuzzy relational equations. During its operation the proposed network can generalize in a way that is inspired from the theory of fuzzy systems and the generalized modus ponens. Let us here describe the adaptation of the weights of the second layer (the adaptation of the first layer is similar). The proposed algorithm converges independently for each neuron. For simplicity and without loss of generality, let us consider only the single neuron case. The response of the neuron $f^{(k)}$ at time *k* is given by:

$$f^{(k)} = \sup_{i \in N_l} t\left(Z_i^{(k)}, w_i^{(k)}\right) \tag{6.17}$$

where $w_i^{(k)}$ are the weights of the neuron and $z_i^{(k)}$ the input, at time *k*. The desired output at time *k* is $B_i^{(k)}$. The algorithm has as following:

Initialize the weights as $w_i^{(0)}$, $i \in N_l$.

Process the input $z^{(k)}$ and the desired output $B^{(k)}$, compute the response of the network $f^{(k)}$ and update the weight accordingly (on-line variant of learning):

$$w_I^{(k+1)} = W_i^{(k)} + \Delta w_i^{(k)}$$

$$\Delta W_i^{(k)} = \eta l_s$$

$$l_s = \begin{cases} \eta_1 \left(\breve{\omega}_t \left(z_i^{(k)}, B^{(k)} \right) - w_i^{(k)} \right), & if\ w_i^{(k)} < \breve{\omega}_t \left(z_i^{(k)} \right) \\ \eta_2 \left(w_2^{(k)} - \hat{\omega}_t \left(z_i^{(k)}, b^{(k)} \right) \right), & if\ w_i^{(k)} > \hat{\omega}_t \left(z_i^{(k)}, B^{(k)} \right) \end{cases}$$

where η, η_1, η_2 are the learning rates. The adaptation is activated only if $|\varepsilon(B^{(k)}, y^{(k)}| > \varepsilon_c$, where ε is an error constant.

If the *t*-norm is Archimedean, then the learning signal is computed as:

$$l_s = \left(\hat{\omega}_t \left(z_i^{(k)}, B^{(k)} \right) - w_i^{(k)} \right),\ if\ z_i^{(k)} \geq b_i^{(k)}\ and\ z_i^{(k)} \neq 0,\ else\ l_s = 0$$

With the aid of the above learning process (and similar for the first layer, since the operator $\hat{\omega}_t$ is also used in order to solve the fuzzy relational equation of the first layer [3]), the network approximates the solutions of the fuzzy relational equations given above and thus minimize the error.

Chapter 7

A Context Extension

7.1 Introduction

In this chapter we show how the C-OWL language, an extension of OWL originally proposed in [BGvH$^+$03], allows to formalize spaces of heterogeneous ontologies related via a set of semantic mappings, also called contextualized ontologies. Starting from the proposal done in [BGvH$^+$03], the document contributes in the following ways:

1. We propose two alternative semantics for an OWL space (i.e., indexed set of ontologies specified in OWL). The two alternative semantics, called respectively *opaque semantics* and *transparent semantics* reflect two alternative interpretations for namespaces reference in OWL.
2. The semantics for C-OWL proposed in [BGvH$^+$03] is revised by taking into account the transparent and the opaque interpretation of OWL spaces introduced here.
3. We propose a concrete syntax of C-OWL based on RDF, and a mapping between the abstract C-OWL syntax and its concrete representation.
4. We finally show the relation between rules in SWRL and bridge rules, i.e., the rules that allow to express semantic mappings.

7.2 OWL overview

In this section we recall the main concepts about OWL that are relevant for the rest of this document. For the sake of readability, we slightly simplify the presentation of such concepts, without loosing the main properties.

According to [PSHH03b], an OWL ontology is a set of annotated *axioms* and *facts*, plus import references to other ontologies. OWL ontologies can be referenced by means

DisjointClasses(description$_1$ $\cdots$ description$_n$)	T(description$_i$) *owl:disjointWith* T(description$_j$) . *OR* T(description$_j$) *owl:disjointWith* T(description$_i$) . 1=i<j=n T(description$_i$) *owl:disjointWith* T(description$_j$) . [opt] 1=i$\neq$ j=n

Figure 7.1: Concrete Syntax of OWL-DL

of a URI (Uniform Resource Identifier). Ontologies can also have annotations that can be used to record authorship and other information associated with an ontology. Since annotation directives have no effect on the semantics of OWL ontologies in the abstract syntax, we ignore them. We concentrate on the OWL-DL fragment of OWL. This language is equivalent to the $\mathrm{SHOIQ}(\mathrm{D}^+)$ DL, i.e., $\mathrm{SHIQ}(\mathrm{D}^+)$- extended with an equivalent of the oneOf constructor. The proposed framework can be restricted or generalized to OWL-lite and OWL-full, respectively.

Let $\mathbb{C}$, $\mathbb{R}$ and $\mathbb{O}$ be the sets of strings that can used to denote concepts, roles and individuals respectively. The disjoint union of $\mathbb{C}$, $\mathbb{R}$ and $\mathbb{O}$ is denoted with $\mathbb{L}$.

Definition 16 (OWL Ontology) *An* OWL Ontology *(or simply an* ontology*)* O *is a pair* $\langle T, A\rangle$ *where* T *and* A *are a T-box and an A-box respectively in the* $\mathrm{SHOIQ}(\mathrm{D}^+)$ *description logic on* $\mathbb{L}$.

Definition 17 (OWL interpretation) *An* (abstract) OWL interpretation $\mathcal{I}$ *is a pair* $\left\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \right\rangle$, *where* $\Delta^{\mathcal{I}}$ *is an nonempty domain, and a mapping* $\cdot^{\mathcal{I}}$ *that assign to each concept name* $c \in \mathbb{C}$ *a subset of* $\Delta^{\mathcal{I}}$*, to each role name* $R \in \mathbb{R}$ *a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$*, and to each individual name* $o \in \mathbb{O}$ *an element of* $\Delta^{\mathcal{I}}$.

The concrete syntax of OWL-DL is consisting of a subset of RDF graphs [PSHH03b]. For example, Figure 7.1 shows a mapping from the abstract syntax to the RDF graph.

7.3 From single ontology to ontology space

Although ontology, from the epistemological point of view, is believed as an unique model for a domain, sometimes we have to face several ontologies related to the same (similar) object(s) in practical semantics web jobs. This intuition leads to the introduction of *OWL Space*.

Definition 18 (OWL space) *Let* I *be a set of indexes, standing for a set of URI's for ontologies. An* OWL space *is a family* $\{O_i\}_{i\in I}$*, such that every* O_i *is an ontology.*

Suppose that $C, D, E, F \in \mathbb{C}$ and $r, s \in \mathbb{R}$. The following are examples of concepts that can appear in O_i.

$$C,\ \ i{:}C,\ \ C \sqcap D,\ \ j{:}E,\ \ C \sqcap (j{:}E),\ \ \exists r.C \sqcup D,\ \ \exists (j{:}s).C \sqcup (j{:}F) \qquad (7.1)$$

Every expression occurring in O_i without an index is intended to be in the language defined by O_i, L_i. The expressions appearing in O_i with indexes j are supposed to be defined in O_j; therefore they appear in O_j without index or with the index j. Let's make this distinction more precise.

Definition 19 (Local language) *A* local concept, *w.r.t. i, is an element of $\mathbb{C}$ that appears in O_i either without indexes or with index equal to i.* Local roles *and* local individuals *are defined analogously. The set of local concepts, local roles, and local individuals w.r.t. i are denoted by $\mathbb{C}_i$, $\mathbb{R}_i$, and $\mathbb{O}_i$. The* local language *to i, $\mathbb{L}_i$, is the disjoint union of them.*

Local objects of a language L_i are also called *i-objects*. For notational convenience, in the following we always use the colon notation. Thus, for instance, local concepts $C \in \mathbb{C}_i$ of an ontology O_i are written as $i : C$. A *foreign concept*, or equivalently a *non local concept*, w.r.t. $i \in I$, is a concept that appears in O_i but is defined in some ontology O_j. Foreign concepts are referred with the notation $j : C$. An analogous definition can be given for roles and individuals.

Definition 20 (Foreign language) *A* j-foreign concept, *w.r.t. i (with $i \neq j$), is an element of $\mathbb{C}$ that appears in O_i with index equal to j.* j-foreign roles *and* j-foreign individuals *are defined analogously. The set of j-foreign concepts, j-foreign roles, and j-foreign individuals w.r.t. i are denoted by $\mathbb{C}_{ij}$, $\mathbb{R}_{ij}$, and $\mathbb{O}_{ij}$. The* j-foreign language *to i, $\mathbb{L}_{ij}$, is the disjoint union of them.*

Among the concepts described in (7.1), C and D are local concepts w.r.t. i and r is a local role (w.r.t. i), while E and F are j-foreign concepts and s is a j-foreign role.

7.4 Semantics for ontology spaces

Definition 21 (OWL-space interpretation) *An* OWL-space interpretation *for the OWL space $\{O_i\}_{i\in I}$, is a family $\mathcal{I} = \{\mathcal{I}_i\}_{i\in I}$, where each $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}, (.)^{\mathcal{I}_i} \rangle$ is an OWL interpretation on O_i*

Depending on how we interpret foreign languages in an ontology, we can have two different semantics for ontology spaces.

Opaque semantics Under this semantics any j-foreign term (concept, role, or individual) of an ontology O_i is assigned with an interpretation which is independent from the interpretation assigned to the corresponding local term in O_j.

Transparent semantics In this case the interpretation of a j-foreign term (concept, role, or individual) of an ontology O_i is assigned with an interpretation which is dependent on the interpretation assigned to the corresponding local term in O_j.

7.4.1 Opaque semantics

Definition 22 (Opaque model of an OWL space) *An OWL-space interpretation $\mathcal{I} = \{\mathcal{I}_i\}_{i\in I}$ is a* opaque model *of $\{O_i\}_{i\in I}$, if for each $i \in I$, $\mathcal{I}_i \models O_i$.*

Notice that, in opaque models it is possible that $(j:C)^{\mathcal{I}_i} \neq C^{\mathcal{I}_j}$.

Definition 23 (Ontology space entailment) *Let C and D be two concepts that appears in O_i of an OWL space $\{O_i\}_{i\in I}$. We say that $\{O_i\}_{i\in I} \models^{\bullet} i:(C \sqsubseteq D)$ if, for every opaque model $\mathfrak{I}$ of $\{O_i\}_{i\in I}$, $\mathcal{I}_i \models C \sqsubseteq D$.*

Notice that in the previous definition C and D can be foreign concepts, i.e., expressions of the form $j:X$ and $k:Y$. If we make this explicit we have that $\{O_i\}_{i\in I} \models^{\bullet} i:(j:X \sqsubseteq k:Y)$.

The expression

$$\{O_i\}_{i\in I} \models^{\bullet} i:(j:X \sqsubseteq k:Y)$$

can be read as: In the ontology O_i the j-foreign concept X is more specific than the k-foreign concept Y.

Proposition 1 *Global logical consequence is the disjoint union of each local logical consequence. Formally:*

$$O_i \models C \sqsubseteq D \quad \Longleftrightarrow \quad \{O_i\}_{i\in I} \models^{\bullet} i:(C \sqsubseteq D)$$

The proof of the above proposition is a straightforward application of the definition of entailment. Proposition 1 highlights the fact that, from the semantic viewpoint, the ontologies of an ontology space don't interfere one another. In other words, the property of a j-foreign concept that occurs in O_i, are stated in O_i itself, and can be completely different from the properties stated in O_j, or some other ontology.

The fact that $j:C$ that occurs in O_i denotes the same concept as C in O_j is left to the pragmatic use of the two ontologies.

This semantics is compliant with the official W3C semantic published in [PSHH03b].

Example 11 *Consider a OWL-space $\{O_1, O_2\}$, in which O_1 contains ($C \sqsubseteq 2:D$) and ($2:E \sqsubseteq F$), and O_2 contains $D \sqsubseteq E$ (Figure-7.2). Then under the opaque OWL semantics, we have $\{O_i\}_{i\in I} \not\models^{\bullet} C \sqsubseteq F$. Indeed the opaque model $\mathfrak{I} = \{\mathcal{I}_1, \mathcal{I}_2\}$ with*

$$\begin{array}{ll} (C)^{\mathcal{I}_1} = \{x\} & (D)^{\mathcal{I}_2} = \{a\} \\ (2:D)^{\mathcal{I}_1} = \{x, y\} & (E)^{\mathcal{I}_2} = \{a, b\} \\ (2:E)^{\mathcal{I}_1} = \{z\} & \\ (F)^{\mathcal{I}_1} = \{z, w\} & \end{array}$$

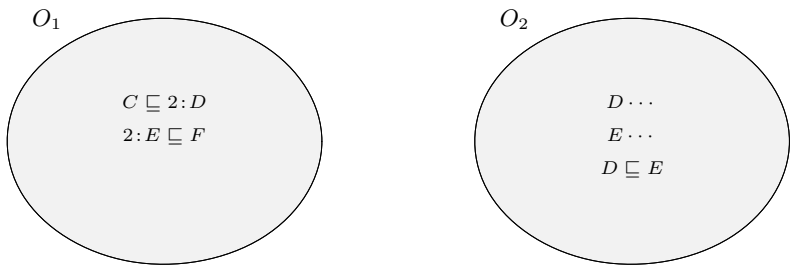

Figure 7.2: Ontology space $\{O_1, O_2\}$ of Example 11

7.4.2 Transparent semantics

In this subsection we present an alternative semantics based on the assumption that the meaning of a j-foreign concept (role, individual) depends from the definition of the concept in its ontology O_j. In this semantics foreign concepts, roles and individuals, can be used to refer to the *same* semantic object defined in a third ontology.

Definition 24 (Transparent model of an OWL space) *An OWL-space interpretation $\mathcal{I} = \{\mathcal{I}_i\}_{i\in I}$ is a* transparent model *of $\{O_i\}_{i\in I}$, if for each $i \in I$, $\mathcal{I}_i \models O_i$, and the following condition holds:*

COMPATIBILITY BETWEEN LOCAL INTERPRETATIONS

for any j-foreign concept $C \in \mathbb{C}_{ij}$, any j-foreign role $R \in \mathbb{R}_{ij}$, and j-foreign individual constant $a \in \mathbb{O}_{ij}$

1. $(j{:}C)^{\mathcal{I}_i} = (C)^{\mathcal{I}_j}$
2. $(j{:}R)^{\mathcal{I}_i} = (R)^{\mathcal{I}_j}$
3. $(j{:}a)^{\mathcal{I}_i} = (a)^{\mathcal{I}_j}$

Definition 25 (Ontology space transparent entailment) *Let C and D be two concepts that appears in O_i of an OWL space $\{O_i\}_{i\in I}$. We say that $\{O_i\}_{i\in I} \models^{\circ} i{:}(C \sqsubseteq D)$ if, for every transparent model $\mathfrak{I}$ of $\{O_i\}_{i\in I}$, $\mathcal{I}_i \models C \sqsubseteq D$.*

Proposition 2 *For any pair of j-foreign concepts C and D of O_i*

$$O_j \models C \sqsubseteq D \quad \Longrightarrow \quad \{O_i\}_{i\in I} \models i{:}(j{:}C \sqsubseteq j{:}D)$$

Example 12 *Consider a OWL-space of Example 11 (Figure-7.3). Then under the opaque OWL semantics, we have $\{O_i\}_{i\in I} \models^{\bullet} C \sqsubseteq F$. Notice that $\emptyset_1 \not\models C \sqsubseteq F$. This means*

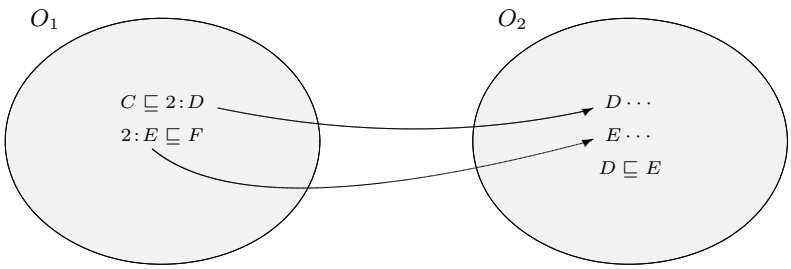

Figure 7.3: Transparent Model in Example-12: .

that the embedding of O_1 in the ontology space $\{O_1, O_2\}$ has the effect of adding new properties also to local concepts of O_1.

Proposition 3 *There exists a pair of j-foreign concepts C and D of O_i:*

$$\{O_i\}_{i\in I} \models i{:}(j{:}C \sqsubseteq j{:}D) \quad \not\Longrightarrow \quad O_j \models C \sqsubseteq D$$

Propositions 2 and 3 state that a j-foreign concept that occurs in O_i can be considered as a restriction of the corresponding local concept defined in O_j.

7.5 From ontology space to context space

In this section we will consider the notion of semantic mappings between ontologies. In [BGvH+03] we introduce the notion of Contextualized Ontology and of Context Space = ontology space + mappings. The basic notion towards the definition of context mappings are *bridge rules*.

Definition 26 (Bridge rules) *A* bridge rule from i to j is a statement of one of the four following forms,

$$i{:}x \xrightarrow{\sqsubseteq} j{:}y, \;\; i{:}x \xrightarrow{\sqsupseteq} j{:}y, \;\; i{:}x \xrightarrow{\equiv} j{:}y, \;\; i{:}x \xrightarrow{\perp} j{:}y, \;\; i{:}x \xrightarrow{*} j{:}y,$$

where x and y are either concepts, or individuals, or roles of the languages L_i and L_j respectively.

A mapping between two ontologies is a set of bridge rules between them.

Definition 27 (Mapping) *Given a OWL space* $\{O_i\}_{i\in I}$ *a* mapping M_{ij} *from* O_i *to* O_j *is a set of bridge rules from* O_i *to* O_j*, for some* $i, j \in I$.

Mappings are directional, i.e., M_{ij} is not the inverse of M_{ji}. A mapping M_{ij} might be empty. This represents the impossibility for O_j to interpret any i-foreign concept into some local concept. Dually M_{ij} might be a set of bridge rules of the form $i:x \xrightarrow{\equiv} j:y$ for any element x (concept, role, and individual) of O_i. This represents the operation of mapping all of O_i into an equivalent subset of O_j. If this subset is O_j itself then this becomes the contextual mapping version of the OWL import operation. However, notice that importing O_i into O_j is not the same as mapping O_i to O_j with M_{ij}. In both cases information goes from i to j. The difference is that, in the former case, O_j duplicates the information of i- foreign elements without any change, while, in the latter, O_j translates (via the mapping M_{ij}) the semantics of O_i into its internal (local) semantics.

Definition 28 (Context space) *A* context space *is a pair composed of an OWL space* $\{O_i\}_{i\in I}$ *and a family* $\{M_{ij}\}_{i,j\in I}$ *of mappings from* i *to* j*, for each pair* $i, j \in I$.

In the following we shorten $\{M_{ij}\}_{ij\in I}$ with $\{M_I\}$.

Definition 29 (Context) *Given a context space* $\langle\{O_i\}_{i\in I}, \{M_I\}\rangle$ *the* i*-th context* C_i *is equal to* $\langle O_i, \{M_{ji}\}_{j\neq i\in I}\rangle$*, i.e., the pair composed of the* i*-th ontology and the mapping entering in* O_i.

To give the semantics of context mappings we extend the definition of interpretation for ontology space with the notion of *domain relation*. A domain relation $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ states, for each element in $\Delta^{\mathcal{I}_i}$ to which element in $\Delta^{\mathcal{I}_j}$ it corresponds to. The semantics for bridge rules from i to j can then be given with respect to r_{ij}.

Definition 30 (Interpretation for context spaces) *An* interpretation for a context space $\langle\{O_i\}_{i\in I}, \{M_I\}\rangle$ *is a pair* $\langle\mathcal{I}, \{r_{ij}\}_{i\neq j\in I}\rangle$ *composed of an interpretation* $\{O_i\}_{i\in I}$ *of the ontology space* $\{O_i\}_{i\in I}$ *and a family of relation* $r_{ij} \in \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ *for each* $i \neq j \in I$. r_{ij} *is called the* domain relation *from* i *to* j.

Definition 31 (Satisfiability of bridge rules[1])

1. $\mathfrak{I} \models i:x \xrightarrow{\sqsubseteq} j:y$ *if* $r_{ij}(x^{\mathcal{I}_i}) \subseteq y^{\mathcal{I}_j}$;
2. $\mathfrak{I} \models i:x \xrightarrow{\sqsupseteq} j:y$ *if* $r_{ij}(x^{\mathcal{I}_i}) \supseteq y^{\mathcal{I}_j}$;
3. $\mathfrak{I} \models i:x \xrightarrow{\equiv} j:y$ *if* $r_{ij}(x^{\mathcal{I}_i}) = y^{\mathcal{I}_j}$;
4. $\mathfrak{I} \models i:x \xrightarrow{\perp} j:y$ *if* $r_{ij}(x^{\mathcal{I}_i}) \cap y^{\mathcal{I}_j} = \emptyset$;

[1] In this definition, to be more homogeneous, we consider the interpretations of individuals to be sets containing a single object rather than the object itself.

5. $\mathfrak{I} \models i:x \xrightarrow{*} j:y\ r_{ij}(x^{\mathcal{I}_i}) \cap y^{\mathcal{I}_j} \neq \emptyset$;

See [BGvH+03] for an intuitive explanation and a more detailed discussion about bridge rules.

Definition 32 (Opaque model for a context space) *An context space interpretation $\langle \mathcal{I}, \{r_{ij}\}_{i \neq j \in I} \rangle$ for $\langle \{O_i\}_{i \in I}, \{M_I\} \rangle$ is an* opaque model *for $\langle \{O_i\}_{i \in I}, \{M_I\} \rangle$, if $\mathcal{I}$ is an opaque model for $\{O_i\}_{i \in I}$ and all the bridge rules in $\{M_I\}$ are satisfied.*

The semantics for bridge rules is orthogonal w.r.t., the opaque/transparent semantics of local ontologies (OWL). Therefore we can have both transparent and opaque interpretation of context spaces.

Definition 33 (Transparent model for a context space) *An context space interpretation $\langle \mathcal{I}, \{r_{ij}\}_{i \neq j \in I} \rangle$ for $\langle \{O_i\}_{i \in I}, \{M_I\} \rangle$ is an* opaque model *for $\langle \{O_i\}_{i \in I}, \{M_I\} \rangle$, if $\mathcal{I}$ is a transparent model for $\{O_i\}_{i \in I}$ and all the bridge rules in $\{M_I\}$ are satisfied.*

7.6 C-OWL: Contextualized OWL

In this section we define an RDF-based syntax for such mappings. We introduce the semantics using an example, explain the different parts of the specification and define an RDF schema for the mapping representation.

The philosophy of C-OWL is to treat mappings as first class and to represent them independently from the ontologies they connect. There are a couple of advantages of this approach. From a syntactic point of view, the advantage is that we can define a language for specifying mappings independently from the OWL syntax specification. the resulting language will refer to elements of the OWL specification without extending it.

Figure 7.4 shows an example mapping of two ontologies about wines. In order to represent this mapping we have to capture the following aspects:

- a unique identifier for referring to the mapping
- a reference to the source ontology
- a reference to the target ontology
- a set of bridge rules relating classes from the two ontologies, each described by
  - (a reference to) the source concept
  - (a reference to) the target concept
  - the type of the bridge rule, which is one of $\equiv \sqsubseteq, \sqsupseteq, \perp, *$

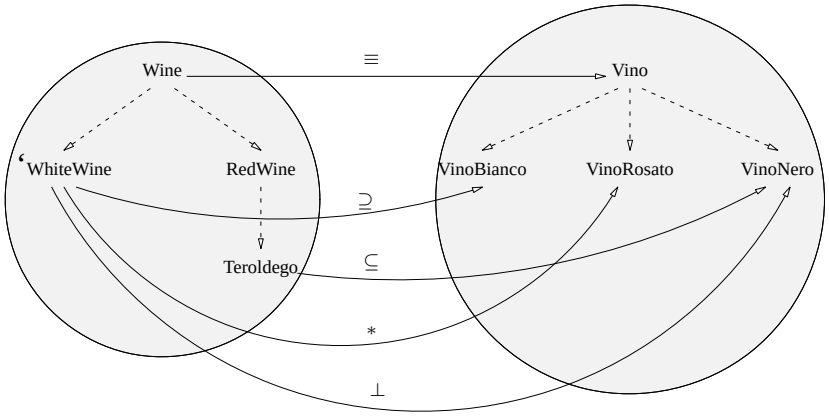

Figure 7.4: A C-OWL mapping from the ontology “wine” to the ontology “vino”.

Figure 7.5 shows an RDF-based representation of these elements. We use a resource of the type `cowl:Mapping` as a root element of the description. This resource is linked to two OWL models using the properties `sourceOntology` and `targetOntology`. The ontologies are represented by reference to their namespace. Further, the resource representing the overall mapping is linked to a number of resources through the `cowl:bridgeRule` property. These resources represent the individual rules in the mappings and can be of type `cowl:Equivalent`, `cowl:Into`, `cowl:Onto`, `cowl:Incompatible` or `cowl:Compatible` each representing one of the types mentioned above. Each of the resources representing a bridge rule is linked to an OWL class from the target ontology through the `cowl:source` and to a class from the target ontology by the `cowl:target` property. The classes can be represented by a reference to the corresponding resource in the ontology definition but it can also be a complex OWL class definition that uses elements from the respective ontology. In this way we can represent complex mappings that go beyond semantic relations between class names. We have defined an RDF schema for the mapping representation. This schema is shown in Figure 7.6.

7.7 Reasoning in C-OWL

Reasoning in C-OWL with bridge rules $\xrightarrow{\sqsubseteq}$, $\xrightarrow{\sqsupseteq}$, $\xrightarrow{\equiv}$, and $\xrightarrow{\perp}$ on concepts is decidable. No investigation has been done on the decidability with bridge rules on roles and individuals and on the bridge rule $\xrightarrow{*}$. [ST04] describes a sound and complete distributed tableaux algorithm for Distributed Description Logics (DDL) [BS03] which is the logic of C-OWL. The same paper describes also a first prototypical implementation, of reasoning in acyclic distributed T-Boxes, i.e., T-boxes connected by bridge rules which do not form a cycle.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:cowl="http://www.cowl.org/"
    xmlns:owl="http://www.w3.org/2002/07/owl#">
    <cowl:Mapping rdf:ID="myMapping">
        <rdfs:comment>Example Mapping for Web Semantics Journal Paper</rdfs:comment>
        <cowl:sourceOntology>
            <owl:Ontology rdf:about="http://www.example.org/wine.owl"/>
        </cowl:sourceOntology>
        <cowl:targetOntology>
            <owl:Ontology rdf:about="http://www.example.org/vino.owl"/>
        </cowl:targetOntology>
        <cowl:bridgeRule>
            <cowl:Equivalent>
                <cowl:source>
                    <owl:Class rdf:about="http://www.example.org/wine.owl#wine"/>
                </cowl:source>
                <cowl:target>
                    <owl:Class rdf:about="http://www.example.org/vino.owl#vino"/>
                </cowl:target>
            </cowl:Equivalent>
        </cowl:bridgeRule>
        <cowl:bridgeRule>
            <cowl:Onto>
                <cowl:source>
                    <owl:Class rdf:about="http://www.example.org/wine.owl#RedWine"/>
                </cowl:source>
                <cowl:target>
                    <owl:Class rdf:about="http://www.example.org/vino.owl#VinoRosso"/>
                </cowl:target>
            </cowl:Onto>
        </cowl:bridgeRule>
        <cowl:bridgeRule>
            <cowl:Into>
                <cowl:source>
                    <owl:Class rdf:about="http://www.example.org/wine.owl#Teroldego"/>
                </cowl:source>
                <cowl:target>
                    <owl:Class rdf:about="http://www.example.org/vino.owl#VinoRosso"/>
                </cowl:target>
            </cowl:Into>
        </cowl:bridgeRule>
        <cowl:bridgeRule>
            <cowl:Compatible>
                <cowl:source>
                    <owl:Class rdf:about="http://www.example.org/wine.owl#WhiteWine"/>
                </cowl:source>
                <cowl:target>
                    <owl:Class rdf:about="http://www.example.org/vino.owl#Passito"/>
                </cowl:target>
            </cowl:Compatible>
        </cowl:bridgeRule>
```

```
        <cowl:bridgeRule>
            <cowl:Incompatible>
                <cowl:source>
                    <owl:Class rdf:about="http://www.example.org/wine.owl#WhiteWine"/>
                </cowl:source>
                <cowl:target>
                    <owl:Class rdf:about="http://www.example.org/vino.owl#VinoNero"/>
                </cowl:target>
            </cowl:Incompatible>
        </cowl:bridgeRule>
    </cowl:Mapping>
</rdf:RDF>
```

Figure 7.5: Specification of the Mappings from figure 7.4

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xmlns:owl ="http://www.w3.org/2002/07/owl#">
    <rdfs:Class rdf:about="Mapping"/>
    <rdfs:Class rdf:about="Correspondence"/>
    <rdfs:Class rdf:about="Equivalence">
        <rdfs:subClassOf rdf:resource="#Correspondence"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="Onto">
        <rdfs:subClassOf rdf:resource="#Correspondence"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="Into">
        <rdfs:subClassOf rdf:resource="#Correspondence"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="Compatible">
        <rdfs:subClassOf rdf:resource="#Correspondence"/>
    </rdfs:Class>
    <rdfs:Class rdf:about="Incompatible">
        <rdfs:subClassOf rdf:resource="#Correspondence"/>
    </rdfs:Class>
    <rdf:Property rdf:about="sourceOntology">
        <rdfs:domain rdf:resource="#Mapping"/>
        <rdfs:range rdf:resource="owl:Ontology"/>
    </rdf:Property>
    <rdf:Property rdf:about="targetOntology">
        <rdfs:domain rdf:resource="#Mapping"/>
        <rdfs:range rdf:resource="owl:Ontology"/>
    </rdf:Property>
    <rdf:Property rdf:about="bridgeRule">
        <rdfs:domain rdf:resource="#Mapping"/>
        <rdfs:range rdf:resource="#Correspondence"/>
    </rdf:Property>
    <rdf:Property rdf:about="source">
        <rdfs:domain rdf:resource="#Correspondence"/>
        <rdfs:range rdf:resource="owl:Class"/>
    </rdf:Property>
    <rdf:Property rdf:about="target">
        <rdfs:domain rdf:resource="#Correspondence"/>
        <rdfs:range rdf:resource="owl:Class"/>
    </rdf:Property>
</rdf:RDF>
```

Figure 7.6: RDF schema defining the Extensions to OWL

$cowl:Equivalence$(ontologyID$_1$.description$_1$, ontologyID$_2$.description$_2$)	T(ontologyID$_1$.description$_1$) *owl:equivalentClass* T(ontologyID$_2$.description$_2$) .
$cowl:Into$(ontologyID$_1$.description$_1$, ontologyID$_2$.description$_2$)	T(ontologyID$_1$.description$_1$) *rdfs:subClassOf* T(ontologyID$_2$.description$_2$) .
$cowl:Onto$(ontologyID$_1$.description$_1$, ontologyID$_2$.description$_2$)	T(ontologyID$_2$.description$_2$) *rdfs:subClassOf* T(ontologyID$_1$.description$_1$) .
$cowl:Compatible$(ontologyID$_1$.description$_1$, ontologyID$_2$.description$_2$)	
$c-owl:Incompatible$(ontologyID$_1$.description$_1$, ontologyID$_2$.description$_2$)	T(ontologyID$_1$.description$_1$) *owl:disjointWith* T(ontologyID$_2$.description$_2$) . *OR* T(ontologyID$_2$.description$_2$) *owl:disjointWith* T(ontologyID$_1$.description$_1$) .

Figure 7.7: Concrete Syntax of C-OWL bridge rules

7.8 C-OWL and SWRL

In this section we discuss the connections of C-OWL and SWRL. A first observation is that one can consider the opportunity to extend C-OWL by allowing local rules, i.e., rules within an ontology. This leads us to the definition of C-SWRL, which is the contextualized version of SWRL. Syntax and semantics for C-SWRL, can be obtained with by trivially combine the semantics of SWRL and the one of C-OWL.

A second issue concerns the formal relation between rules and bridge rules. I.e., can bridge rules be considered as rules of a global ontology? In the following we will not provide a definitive answer to this question, we only raise some point and show some possible interrelations mappings and rules.

In the following we present three possible reformulations of a context space in a global theory. These reformulation rewrite ontologies and bridge rules in a suitable set of axioms of a global language in which the global theory is expressed

First Order (F) reformulation The global theory is a first order many sorted theory, with sorts $\{D_i\}_{i\in I}$. It's interpreted is a single first order many sorted interpretation, where each sort D_i represents the local interpretation domain of the i-th ontology. Bridge rules can be reformulated as SWRL rules where the domain relation R_{ij} is explicitly mentioned.

Epistemic (E) reformulation The global theory is a first order many sorted modal theory, with sorts $\{D_i\}_{i\in I}$ and a unique KD45 modality operator $\Box$. It's interpreted as a KD45-structure with constant models, As in the previous case each sort D_i represents the local interpretation domain of the i-th ontology. Bridge rules can be reformulated as epistemic rules where the domain relation R_{ij} is explicitly mentioned.

Distributed (D) reformulation The global theory is a first order many sorted modal theory, with sorts $\{D_i\}_{i\in I}$ and a set $\{\Box_i\}_{i\in I}$ of KD45-modal operators. It's interpreted is a multi modal KD45-structure with constant models, As in the previous case each sort D_i represents the local interpretation domain of the i-th ontology. Facts about the i-th ontology are represented using the modal operator $\Box_i$.

In the following we propose F-, E-, and D-reformulation of all the bridge rules but the $\xrightarrow{\equiv}$ bridge rule, as it can be considered as the conjunction of the $\xrightarrow{\sqsubseteq}$ and $\xrightarrow{\sqsupseteq}$ bridge rules.

In these transformations we adopt the following conventions. For every C which is an atomic or a complex concept, $C(x)$ denotes its first order transformation with the free variable x. Furthermore, every variable that occurs in the transformation of a concept of the i-th ontology is intended to be of the sort D_i. The variables x, y occurring in $R_{ij}(x, y)$ are intended to be of the sort D_i and D_j, respectively.

T-box axioms	$i:A \sqsubseteq B$
F-reformulation	$\forall x(A_i(x) \rightarrow B_i(x))$
E-reformulation	$\Box\forall x(A_i(x) \rightarrow B_i(x))$
D-reformulation	$\Box_i\forall x(A(x) \rightarrow B(x))$

A-box axioms	$i:A(a)$
F-reformulation	$A_i(a)$
E-reformulation	$\Box A_i(a)$
D-reformulation	$\Box_i A(a)$

bridge rule	$i:C \xrightarrow{\sqsubseteq} j:D$
F-reformulation	$\forall xy(C_i(x) \wedge R_{ij}(x, y) \rightarrow D_j(y))$
E-reformulation	$\forall xy(\Box C_i(x) \wedge R_{ij}(x, y) \rightarrow \Box D_j(y)))$
D-reformulation	$\forall xy(\Box_i C(x) \wedge R_{ij}(x, y) \rightarrow \Box_j D(y)))$

bridge rule	$i:C \xrightarrow{\sqsupseteq} j:D$
F-reformulation	$\forall y(D_j(y) \rightarrow \exists x(R_{ij}(x, y) \wedge C_i(x)))$
E-reformulation	$\forall y(\Box D_j(y) \rightarrow \exists x(R_{ij}(x, y) \wedge \Box C_i(x)))$
D-reformulation	$\forall y(\Box_j D(y) \rightarrow \exists x(R_{ij}(x, y) \wedge \Box_i C(x)))$

bridge rule	$i:C \xrightarrow{\perp} j:D$
F-reformulation	$\forall xy(C_i(x) \wedge R_{ij}(x, y) \rightarrow \neg D_j(y))$
E-reformulation	$\forall xy(\Box C_i(x) \wedge R_{ij}(x, y) \rightarrow \neg\Box D_j(y)))$
D-reformulation	$\forall xy(\Box_i C(x) \wedge R_{ij}(x, y) \rightarrow \neg\Box_j D(y)))$

bridge rule	$i:C \xrightarrow{*} j:D$
F-reformulation	$\exists xy(C_i(x) \wedge R_{ij}(x,y) \wedge D_j(y))$
E-reformulation	$\exists xy(\Box C_i(x) \wedge R_{ij}(x,y) \wedge \Box D_j(y))$
D-reformulation	$\exists xy(\Box_i C(x) \wedge R_{ij}(x,y) \wedge \Box_j D(y))$

Monotonicity The global theory is an extension of the theories

$$G_i \models G(i:\phi) \Longrightarrow G \models G(i:\phi)$$

Directionality If no bridge rules enter in $\mathcal{T}_i$ then T_i is not affected by the other ontologies. Formally if $BR_{ij} = \emptyset$ for any $j \neq i$, then

$$G_i \models G(i:\phi) \Longleftarrow G \models G(i:\phi)$$

Local inconsistency The fact that the i0th ontology is inconsistent does not imply that the other are also inconsistent. Formally for $i \neq j$

$$G_i \models G(i:\top \sqsubseteq \bot) \not\Longrightarrow G \models G(j:\top \sqsubseteq \bot)$$

Assertional incompleteness Local A-Boxes can be incomplete. Formally:

$$G \not\models G(i:A(a)) \vee G(i:\neg A(a))$$

The fact that F-, E-, and D-Reformulation enjoy the properties listed above is an open issue. In the following table we report the result already established.

	CS	**F**	**E**	**D**
Mon	x	Y	x	x
A-Incomp	x	N	x	x
Dir	x	N		x
Loc-Inc	x	N		x

Chapter 8

Conclusion

In this report, we investigate the problem of combining ontologies with rule and query languages and provide an unified framework in which the existing (and future) proposals of integrating different sorts of rule based language with the OWL DL Web ontology language can be compared. Theorem 4 on page 11 show that under certain restrictions, the logical implication problem is equivalent in the three (i.e., the axiom-based, the DL-Log and the autoepistemic) approaches described in Chapter 2.

The SWRL[1] proposal is a special case of the axiom-based approach.[2] I.e., SWRL extends OWL DL with safe DL rule axioms that disallow negated atomic role atoms. For the axiom-based approach, we understand that:

1. Logical implication in an axiom-based rule extended knowledge base remains undecidable even in the case of atomic negation-free safe DL rules with a DL having just the universal role constructor $\forall R.\mathsf{C}$. Therefore, logical implication in SWRL is obviously undecidable.

2. Theorem 1 on page 6 provides serval decidable sub-languages of SWRL as well as their complexity results.

Independently from these limitations, several extensions of SWRL (or of some fragment of it) are explored in this report, in order to meet various user requirements in the Semantic Web applications. In particular, we have presented the predicate extension[3] and fuzzy extension,[4] where the former one allows the use of datatype predicates (based on a datatype group) as atoms and the latter one allows the use of weight in the atoms to represent the degree of confidence that the atoms are true.

[1] cf. Chapter 3.
[2] cf. Section 2.2 on page 6.
[3] cf. Chapter 5.
[4] cf. Chapter 6.

Currently, we are working on the specification of the OWL-Log rule-extended knowledge base language in the DL-Log approach. OWL-Log is based on the various dialects of OWL (OWL-Lite and OWL-DL), and a syntax based on the interoperation between OWL and RuleML is planned. Finally, we are also extending the unifying logical framework to include more sub-cases, and to clarify better the similarities and the differences of the various approaches.

Bibliography

[BCF+03] Scott Boag, Don Chamberlin, Mary F. Fernndez, Daniela Florescu, Jonathan Robie, and Jrme Simon (editors). XQuery 1.0: An XML Query Language. URL, November 12 2003. `http://www.w3.org/TR/2003/WD-xquery-20031112/`, W3C Working Draft.

[Bec04] Dave Beckett. RDF/XML Syntax Specification. URL, February 10 2004. W3C Recommendation, `http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/`.

[BGvH+03] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Sencond Internatinal Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 164–179. Springer Verlag, 2003.

[BL98] Tim Berners-Lee. Semantic web roadmap, 1998. Available at `http://www.w3.org/DesignIssues/Semantic`.

[BM01] Paul V. Biron and Ashok Malhotra. XML Schema Part 2: Datatypes. URL, May 2 2001. `http://www.w3.org/TR/xmlschema-2/`.

[BM02] Jean-Francois Baget and Marie-Laure Mugnier. Extensions of simple conceptual graphs: the complexity of rules and constraints. *Journal of Artificial Intelligence research (JAIR)*, 16:425–465, 2002.

[BPS94] Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research*, 1:277–308, 1994.

[BPSM+04] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Franois Yergeau (editors). Extensible Markup Language (XML) 1.0 (Third Edition). URL, February 04 2004. `http://www.w3.org/TR/2004/REC-xml-20040204`, W3C Recommendation.

[BS03] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003. Editor in Chief S. Spaccapietra. LNCS 2800, Springer Verlag.

[BvHH+04] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, Lynn Andrea Stein; Mike Dean, and Guus Schreiber (editors). OWL Web Ontology Language Reference. Technical report, W3C, February 10 2004. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`.

[CDGL00] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[CDGL+04] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. What to ask to a peer: Ontology-based query reformulation. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, 2004.

[CR03] Diego Calvanese and Riccardo Rosati. Answering recursive queries under keys and foreign keys is undecidable. In *Proc. of the 10th Int. Workshop on Knowledge Representation meets Databases (KRDB 2003)*, pages 3–14. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-79/, 2003.

[CTL95] C.S. Lee C.-T. Lin. *Neural fuzzy Systems: A neuro-fuzzy synergism to intelligent systems*. Prentice-Hall, 1995.

[DLN+98] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. An epistemic operator for description logics. *Artificial Intelligence*, 100(1-2):225–274, April 1998.

[DLNS98] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. $\mathcal{AL}$-log: integrating datalog and description logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[ELST04] Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. In *Proc. of the International Conference of Knowledge Representation and Reasoning (KR'04)*, 2004.

[FHe03] Richard Fikes, Patrick Hayes, and Ian Horrocks (editors). DAML Query Language (DQL) Abstract Specification; Joint United States/European Union ad hoc Agent Markup Language Committee; DARPA Agent Markup Language (DAML) Program, April 1 2003. `http://www.daml.org/2003/04/dql/`.

[FHH03] Richard Fikes, Patrick Hayes, and Ian Horrocks. OWL-QL - A Language for Deductive Query Answering on the Semantic Web. Technical

report, Knowledge Systems Laboratory, Stanford University, Stanford, CA, KSL-03-14, 2003.

[FKLS03] Enrico Franconi, Gabriel Kuper, A. Lopatenko, and L. Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *International VLDB Workshop On Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'03)*, 2003.

[GHVD03] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, pages 48–57. ACM, 2003.

[GKWZ03] D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *many-Dimensional Modal Logics: Theory and Applications*. Elsevier, 2003.

[GS98] Chiara Ghidini and Luciano Serafini. Distributed first order logics. In Franz Baader and Klaus Ulrich Schulz, editors, *Frontiers of Combining Systems 2*, Berlin, 1998. Research Studies Press.

[GS99] S. G. Tzafestas G.B. Stamou. Fuzzy relation equations and fuzzy inference systems: an inside approach. *IEEE Trans. on Systems, Man and Cybernetics*, 99:694–702, 1999.

[Hay03] Patrick Hayes. RDF model theory. W3C Working Draft, 10 October 2003. Available at `http://www.w3.org/TR/rdf-mt/`.

[HEPS03] Masahiro Hori, Jérôme Euzenat, and Peter F. Patel-Schneider. OWL web ontology language XML presentation syntax. W3C Note, 11 June 2003. Available at `http://www.w3.org/TR/owl-xmlsyntax/`.

[HM01] Volker Haarslev and Ralf Möller. RACER System Description. In Rajeev Gor, Alexander Leitsch, and Tobias Nipkow, editors, *Automated Reasoning: First International Joint Conference, IJCAR 2001 Siena, Italy, June 18-23, 2001, Proceedings*, volume 2083 / 2001 of *Lecture Notes in Computer Science*, pages 701–705. Springer-Verlag Heidelberg, 2001.

[HMS04] U. Hustadt, B. Motik, and U. Sattler. Reasoning for description logics around SHIQ in a resolution framework. Technical Report 3-8-04/04, FZI, 2004.

[Hor99] Ian Horrocks. FaCT and iFaCT. In Patrick Lambrix, Alexander Borgida, Maurizio Lenzerini, Ralf Möller, and Peter Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99), Linkping, Sweden, July 30 - August 1, 1999*, volume 22, pages 133–135. CEUR Workshop Proceedings, 1999.

[HP96] K. Hirota and W. Pedrycz. Solving fuzzy relational equations through logical filtering. *Fuzzy Sets and Systems*, 99:179–186, 1996.

[HPS04] Ian Horrocks and Peter F. Patel-Schneider. A proposal for an owl rules language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, 2004.

[HT02] Ian Horrocks and Sergio Tessaris. Querying the semantic web: A formal approach. In *Proc. International Semantic Web Conference 2002 (ISWC-02)*, pages 177–191, 2002.

[IIETFTBL98] L. Masinter (editors) IETF (Internet Engineering Task Force); T. Berners-Lee, R. Fielding. RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax. URL, August 1998.

[Ins92] American National Standards Institute. Information Technology — Database Languages — SQL: ISO/IEC 9075:1992, 1992. New York.

[KAC+02] Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis, and Michel Scholl. RQL: A Declarative Query Language for RDF. In *Proceedings of the eleventh international conference on World Wide Web*, pages 592–603, Honolulu, Hawaii, USA, May 7–11 2002. ACM Press, New York, USA. ISBN:1-58113-449-5.

[KC03] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Working Draft, 10 October 2003. Available at `http://www.w3.org/TR/rdf-concepts/`.

[KY95] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, 1995.

[Llo87] John W. Lloyd. *Foundations of logic programming (second, extended edition)*. Springer series in symbolic computation. Springer-Verlag, New York, 1987.

[LR98] Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[Lut01] Carsten Lutz. *The Complexity of Reasoning with Concrete Domains*. PhD thesis, Teaching and Research Area for Theoretical Computer Science, RWTH Aachen, 2001.

[Mar99] Maarten Marx. Complexity of products of modal logics. *J. Log. Comput.*, 9(2):197–214, 1999.

[MB87] Robert MacGregor and Raymond Bates. The LOOM knowledge representation language. Technical Report ISI/RS-87-188, The University of Southern California, Information Sciences Institute, 1987.

[MD02] Drew V. McDermott and Dejing Dou. Representing disjunction and quantifiers in rdf. In *Proc. of the 2002 International Semantic Web Conference (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 250–263. Springer, 2002.

[PH03] Jeff Pan and Ian Horrocks. Web ontology reasoning with datatype groups. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 47–63. Springer, 2003.

[PH04] Jeff Z. Pan and Ian Horrocks. Owl-e: Extending owl dl with datatype expressions. Technical report, Information Management Group, Computer Science Department, The University of Manchester, April 2004.

[PSHH03a] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Candidate Recommendation, 18 August 2003. Available at `http://www.w3.org/TR/owl-semantics/`.

[PSHH03b] P.F. Patel-Schneider, P. Hayes, and I. Horrocks. Web Ontology Language (OWL) Abstract Syntax and Semantics. Technical report, W3C, `www.w3.org/TR/owl-semantics/`, February 2003.

[PSMB+91] Peter F. Patel-Schneider, Deborah L. McGuiness, Ronald J. Brachman, Lori Alperin Resnick, and Alexander Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bull.*, 2(3):108–113, 1991.

[Rei92] Raymond Reiter. What should a database know? *Journal of Logic Programming*, 14(2,3), 1992.

[Ros99] Riccardo Rosati. Towards expressive KR systems integrating datalog and description logics: a preliminary report. In *Proc. of the 1999 International Description Logics workshop (DL'99)*, pages 160–164, 1999.

[RV02] A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[ST04] L. Serafini and A. Tamilin. Local tableaux for reasoning in distributed description logics. In V. Haarslev and R. Möller, editors, *Proceedings of the 2004 Intl. Workshop on Description Logics +(DL2004)*, pages 100–109. CEUR-WS, 2004.

[Str01] U. Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence*, 14:137–166, 2001.

[Swi04] Terrance Swift. Deduction in ontologies via ASP. In *Proc. of LPNMR 2004*, pages 275–288, 2004.

[TBMM01] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. XML Schema Part 1: Structures. URL, May 2 2001. `http://www.w3.org/TR/xmlschema-1/`.

[TH03] Dmitry Tsarkov and Ian Horrocks. DL reasoner vs. first-order prover. In *Proc. of the 2003 Description Logic Workshop (DL 2003)*, volume 81 of *CEUR (`http://ceur-ws.org/`)*, pages 152–159, 2003.

[THG02] Sergio Tessaris, Ian Horrocks, and Graham Gough. Evaluating a modular abox algorithm. In *Proc. of the International Conference of Knowledge Representation and Reasoning (KR'02)*, pages 227–238, 2002.

[TM98] C. Tresp and R. Monitor. A description logic for vague knowledge. In *In proc of the 13th European Conf. on Artificial Intelligence (ECAI-98)*, 1998.

[VT03] S. Kollias V. Tzouvaras, G. Stamou. Knowledge refinement using fuzzy compositional neural networks. 2003.

[Yen] J. Yen. Generalising term subsumption languages to fuzzy logic. In *In Proc of the 12th Int. Joint Conf on Artificial Intelligence (IJCAI-91*.

[Zad65] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.