D2.1.2.2.v2 Report on realizing practical approximate and distributed reasoning for ontologies

Pascal Hitzler (Universität Karlsruhe) (coordinator)
Peter Dolog (Aalborg University), Perry Groot (University of Nijmegen)
Michel Klein (Vrije Universiteit Amsterdam), Malgorzata Mochol (FU Berlin)
Lyndon Nixon (FU Berlin), Linda Peelen (Academic Medical Center, Amsterdam)
Sebastian Rudolph (Universität Karlsruhe)
Stefan Schlobach (Vrije Universiteit Amsterdam)
Heiner Stuckenschmidt (Universität Mannheim)
Denny Vrandecic (Universität Karlsruhe), Holger Wache (Vrije Universiteit Amsterdam)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.1.2.2.v2 (WP2.1)
We report on the progress we have made in KnowledgeWeb on the topic of scalable ontology reasoning. This deliverable contains contributions which advance the state of the art on a broad front, covering query approximation, ABox reasoning and TBox reasoning. It also covers approximation for uncertainty handling and for multi-perspective reasoning.

Document Identifier	KWEB/2007/D2.1.2.2.v2
Project	KWEB EU-IST-2004-507482
Version	1.0
Date	January 24, 2007
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

University of Aberdeen (UNIABDN)
Kings College
AB24 3FX Aberdeen
United Kingdom
Contact person: Jeff Pan
E-mail address: jpan@csd.abdn.ac.uk

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
École Polytechnique Fédérale de Lausanne
France Telecom
Free University of Bozen-Bolzano
Freie Universität Berlin
Institut National de Recherche en Informatique et en Automatique
Learning Lab Lower Saxony
National University of Ireland Galway
The Open University
Universidad Politécnica de Madrid
University of Innsbruck
University of Karlsruhe
University of Liverpool
University of Manchester
University of Sheffield
University of Trento
Vrije Universiteit Amsterdam
Vrije Universiteit Brussel
University of Aberdeen

Changes

Version	Date	Author	Changes
0.1	2006-10-12	Pascal Hitzler	creation
0.2	2006-10-20	Pascal Hitzler	draft inputs added
0.3	2006-12-18	Pascal Hitzler	all final chapters integated
0.4	2006-12-20	Pascal Hitzler	submission to quality review
1.0	2007-01-24	Pascal Hitzler	final version

Executive Summary

With the advent of the Semantic Web, ontology reasoning faces two complementary challenges. On the one hand, the demand rises for ontologies which employ the highly expressive features of modern ontology languages such as OWL. On the other hand, the enourmous quantitiy of application data available urgently calls for scalable reasoning algorithms.

It is a fact, however, that expressivity and scalability of reasoning generally do not go well together: Expressive logics usually scale badly, while scalable algorithms perform shallow reasoning only. This dichotomy is well-known in artificial intelligence, and elsewhere.

One of the methods to achieve scalable reasoning which we put forward in Work Package 2.1. is to use approximate reasoning techniques. This essentially means that we trade correctness of reasoning for speed, however care must be taken such that pracically useful algorithms evolve from the endeavour.

In this deliverable we report on the progress we have made in KnowledgeWeb on this topic. It contains contributions which advance the state of the art on a broad front, covering query approximation, ABox reasoning and TBox reasoning. It also covers approximation for uncertainty handling and for multi-perspective reasoning.

The contributions are compiled from a respectable number of papers by KnowledgeWeb partners which have been published at international conferences and workshops.

Contents

Chapter 1

Introduction

Knowledge representation and reasoning on the Semantic Web is done by means of ontologies [65]. While the quest for suitable ontology languages is still ongoing, the Web Ontology Language OWL [75] has been established as a core standard. It comes in three flavours, as OWL Full, OWL DL and OWL Lite, where OWL Full contains OWL DL, which in turn contains OWL Lite. The latter two coincide semantically with certain description logics [2] and can thus basically be considered fragments of first-order predicate logic. The most prominent of the fragments is undoubtedly OWL DL, which is also what we will mainly be dealing with in this deliverable.

Description Logics – and thus OWL DL – allow to describe knowledge in terms of complex relationships between so-called *concepts* (or *classes*), which in turn can be interpreted extensionally as sets of elements of some domain of interpretation. The relationships between the concepts describe intensional knowledge about the concepts. It is furthermore possible to assign so-called *individuals* to concepts. In OWL DL, individuals can be URIs, which allows to describe complex knowledge about entities found on the web. By means of automated reasoning, OWL DL facilitates the access to knowledge which is implicit in the complex relationships described by an ontology. It thus surpasses the limited applicability of pure metadata by specifying precisely what the encoded knowledge entails.

This usage of OWL DL reasoning, however, comes with an important drawback: The sophisticated reasoning services accessible through OWL DL are of high worst-case computational complexity, as even the time complexity of OWL Lite is exponential. In practice, OWL DL reasoning does not behave quite as badly. But due to the sheer size of the world wide web, and the large amounts of data which are relevant even for restricted applications, OWL DL reasoning does not yet scale to the extent needed for many practical settings.

This scalability problem is not due to a flaw in the design of OWL, as high computational complexity is inherent in expressive knowledge representation and reasoning tasks. Quite to the contrary, the design of OWL, and in particular of OWL DL, is based in part

on the explicit consideration of complexity issues. In particular, OWL DL is a decidable language, which due to common experience indicates that implementations are possible which are sufficiently efficient for practice. The fact that usage of OWL is spreading widely, and that it is also lately being taken up by industry is indicative of the practical relevance of OWL, despite its high theoretical complexity.

Nevertheless, OWL DL inherently still carries the problem of achieving scalable system behaviour, due to the high computational complexities of the reasoning problems one tries to solve. It is indeed entirely unrealistic to expect the development of reasoning algorithms which feature linear or even sublinear worst-case runtime performance. Most likely, it is also unrealistic to expect such runtime performance in terms of average-case data. Research needs to investigate different directions in order to tackle the ontology reasoning scalability problem.

The approach which we put forward in this deliverable is to use approximate reasoning, which trades correctness for time, but in a controlled and well-understood way. It is by no means the only possible or promising approach, but it appears to be suitable for advancing performance, at least for application scenarios where absolute correctness of reasoning is not required, e.g. when the recepient of the result of the computation is a human who can filter out the suitable respronses by common sense.

The contributions to this endeavour which we report on in the sequel advance the state of the art on a broad front, covering query approximation, ABox reasoning and TBox reasoning. It also covers approximation for uncertainty handling and for multi-perspective reasoning.

Chapter 2 focuses on queries, and in particular on query rewriting in order to achieve suitable approximate reasoning behaviour. It reports on two different methods with corresponding evaluations which show the added value of the approach.

Chapter 3 deals with approximate ABox reasoning, more precisely with instance retrieval. It basically contains a positive evaluation of the Screech system which was proposed in Deliverable D2.1.2.2v1.

Chapter 4 deals with TBox reasoning. In particular, a notion of approximate subsumption is proposed.

Chapter 5 reports on an application of approximate reasoning techniques to context sensitivity, called multi-perspective reasoning.

Chapter 6 reports on concept approximation in the context of vague knowledge, using the theory of rough sets.

Chapter 7 contains some brief conclusions.

The contributions in this deliverable are compiled from a respectable number of papers by KnowledgeWeb partners which have been published at international conferences and workshops:

- Part of Section 2.2 was published at the Workshop on Scalable Semantic Web Knowledge Base Systems 2005 [76].
- Section 2.4 is a summary of submissions to the FQAS 2006 conference [24] and the SEBIZ 2006 workshop [51]. The application to job portals was done in cooperation with WP1.1.
- Chapter 3 is an evaluation of a system reported on in Deliverable 2.1.2.2v2, published at ISWC 2005 [39].
- Chapter 5 appeared at ESWC 2006 [70].
- Chapter 6 is a full paper at IJCAI-07 [64].

We would also like to notice that dissemination of some of the results of this deliverable took place as a course at the European Summer School on Logic, Language, and Information, ESSLLI 2006, entitled *Approximate Reasoning for the Semantic Web* and conducted by Frank van Harmelen, Pascal Hitzler, and Holger Wache.

Since we report on seperate efforts on the same general theme, we decided to present them in the form of relatively self-contained chapters, such that the reader can focus on one topic at a time.

D2.1.2.2v2 was originally conceived as a direct continuation of D2.1.2.2v2. However, in the course of the year between the two versions, the contributing Knowledge Web partners made significant new contributions to the topic, as reported herein. It was thus decided to make the v2-deliverable independent of the v1-version, in order to have the space to report on the recent advances. Please note that distributedness has already been covered in version v1 of the deliverable.

Chapter 2

Query Rewriting Techniques for Approximation and Robustness

2.1 Motivation

by HOLGER WACHE

A central issue in the Semantic Web research community is the expressivity of its underlying language and the complexity of the reasoning services it supports. There is a direct correspondence between the current Semantic Web ontology language OWL and Description Logic (DL).[1] Research in DL has lead to sophisticated DL reasoners [40, 35, 37] that can be used to reason with OWL ontologies on the Semantic Web. Considering T-Box reasoning, current state of the art techniques seem capable of dealing with real world ontologies [41, 36]. However, besides T-Box reasoning, an important application domain of ontologies is A-Box reasoning, i.e., reasoning and retrieving the individuals in an ontology. Experiments have shown that state of the art DL reasoners break down for A-Box reasoning when the number of instances becomes large [43].

On the other hand users are often not able to formulate queries correctly which results in user dissatisfaction and frustration. This is even more the case for semantic web systems based on RDF for the following reasons:

- The data accessed often comes from different sources. The internal structure of these sources is not always known.
- The data is semi structured. Sources do not have to describe all aspects of the information resources.
- There is no fixed integrated schema. Each source can have its own schema, sources may make partial use of different available schemas.

[1] More precisely two of the three species of OWL.

With the increasing popularity of RDF as a representation language in domains such as medicine [71] or e-leaning [23] this problem becomes more pressing. If RDF query languages are to be used in a large scale we have to make sure that people will be able to formulate meaningful queries. If this is not the case, we have to find ways to still provide the user with the intended results.

Both problems can be tackled with one fundamental technique: *query rewriting*. If in the original query some complex parts are substituted by other simpler terms one can get simpler queries for improved instance retrieval in A-Boxes. In deliverable D.2.1.2 "Methods for approximate reasoning" it was shown how a sequence of simpler queries can introduce anytime behavior in order to approximate the reasoning process and theoretically improve the performance. But with replacing some parts in a query one can also modify a query to a revised version which reflects more the intended results of the user. So this general technique allows to study the benefits but also the key success factors for a successful approximated and robust query answering.

In the following two sections we discuss the results in applying query rewriting for approximating instance retrieval and robust query answering.

2.2 Scalable Instance Retrieval

by HOLGER WACHE, PERRY GROOT & HEINER STUCKENSCHMIDT

The underlying idea of approximating instance retrieval these techniques is to replace certain inference problems by simpler problems such that either the soundness or the completeness, but not both, of the solutions is preserved. The solutions to the simpler problems are approximate solutions to the original problem.

The contribution of this work is in comparing the performance of two approximate reasoning methods proposed in the literature applied to the real world task of answering conjunctive queries over DL Knowledge Bases. For this, we used the Instance Store [43], a state of the art system developed to scale-up instance retrieval for ontologies with a large number of instances, and extended it with two approximation techniques. The Gene Ontology is used as benchmark data set to evaluate the performance of the approximation techniques.

The section is organized as follows. Section 2.2.1 defines briefly the problem of instance retrieval in the context of Description Logics, which is restricted to conjunctive queries. Section 2.3 gives a brief overview of two approximation methods and describes how they can be applied to the problem of instance retrieval. Section 2.3.1 gives the results of experiments with the two approximation methods applied to instance retrieval using the Gene Ontology. Section 2.3.3 concludes our work.

2.2.1 Instance Retrieval Queries

In this section we focus on the following instance retrieval problem:

Definition 1 (Instance retrieval w.r.t. some query) *Given an A-Box $\mathcal{A}$ and a query Q, i.e., a concept expression, find all individuals a such that a is an instance of Q, i.e., $\{a \,|\, \forall a \in \mathcal{A}, a : Q\}$.*

Often, an analogy is made between databases (DBs) and DL KBs. The schema of a DB corresponds to the T-Box and the DB instances correspond to the A-Box. However, A-Boxes have a very different semantics. This makes query answering in a DL setting often much more complex than query answering in a DB. Given the expressivity of DLs, retrieving instances to a query cannot simply be reduced to model checking as in the database framework because there is no single minimal model for a query. Knowledge Bases may contain nondeterminism and/or incompleteness. Therefore, deductive reasoning is needed when answering a query in a DL setting.

A-Box query languages have been quite weak for earlier DL systems. Usually they supported very simple A-Box queries like instantiation (is individual i an instance of concept C, i.e., $i : C$), realisation (what are the most specific concepts i is an instance of), and retrieval (which individuals are instances of concept C).

In [42] an approach for answering conjunctive queries over arbitrary DL KBs is given based on the translation of the query into an equivalent concept expression, i.e., by *rolling up* the query.

Definition 2 (Boolean Conjunctive Query) *A* Boolean conjunctive query Q *is of the form $q_1 \wedge \cdots \wedge q_n$, where $q_1, \ldots, q_n$ are query terms of the form $x : C$ or $\langle x, y \rangle : R$, where C is a concept, R is a role, and x, y are either individual names or variables.*

The approach makes use of the fact that binary relations in a conjunctive query can be translated into an existential restriction such that logical consequence is preserved. Standard DL inference methods can then be used to classify the concept expression the query is translated into as well as retrieve the instances that belong to it. The method of [42] enables us to use an expressive query language for arbitrary expressive DL KBs.

Because binary relations in a conjunctive query can be translated into an existential restriction such that logical consequence is preserved, standard DL inference methods can then be used to classify the concept expression the query is translated into as well as retrieve the instances that belong to it. [42] enables us to use an expressive query language for arbitrary expressive DL KBs.

DL reasoning is hard, especially in the case of instance retrieval when the number of instances grows very large. To speed up the overall cost of instance retrieval, one can address the number and cost of checking whether a single instance belongs to a query.

Instance Store [43] is developed to speed up instance retrieval by replacing costly instantiation checks $a : Q$ with database retrieval. However, Instance Store can not replace all DL reasoning steps using database retrieval. In some situations DL instantiation checks must still be performed. An analysis of the Instance Store revealed a drastic breakdown in performance in these situations, which hampers its goal to scale-up reasoning to ontologies with a large number of instances. At the moment Instance Store only supports role-free A-Boxes, i.e., relationships between instances in the A-Box are not allowed, but this was sufficient for our purpose.

2.3 Approximation Techniques for Instance Retrieval

All methods propose to approximate an instantiation test using a sequence of tests $C_1, \ldots, C_n$. Assuming that less complex tests can be answered in less time, instance checking can then be speeded up. However, both methods differ in their strategy for selecting the sequence of expressions C_i to be checked successively. In general, [68] argues that the order should balance two factors:

1. The *smoothness* of the approximation. In particular, the next test C_{i+1} should lead to the next best approximation.

2. The potential contribution of the extension of C_{i+1} to the *time complexity* of the tests to be done by the system.

Experiments reported for example in deliverable D2.1.2 "Methods for approximate reasoning" indicate that the order in which parts are replaced is crucial for the performance of the reasoning. Therefore an appropriate strategy, i.e. an excellent selection function, is needed for determining which part is replaced when. In the following we briefly review the techniques of [63] and [68] that can be used to approximate instance retrieval in DL. Further we adopt [68] in order to implement a strategy which bases on an estimation of the needed reasoning time.

Approximating Description Logic Satisfiability

In DLs, satisfiability checking can be seen as the most basic task as many reasoning services can be restated into satisfiability checks [2]. In [63] a technique has been developed to approximate satisfiability checks. Concept expressions are approximated by two sequences $C_1, \ldots, C_n$ of simpler concept expressions, obtained by syntactic manipulations, which can be used to determine the satisfiability of the original concept expression.

For every subconcept D, [63] defines the *depth* of D to be 'the number of universal quantifiers occurring in C and having D in its scope'. The scope of $\forall R.\phi$ is ϕ which can be any concept term containing D. A sequence of weaker (stronger) approximated

concepts can be defined, denoted by $C_i^\top$ ($C_i^\bot$), by replacing every *existentially quantified* subconcept, i.e., $\exists R.\phi$ where ϕ is any concept term, of depth greater or equal than i by $\top$ ($\bot$). Concept expressions are assumed to be in negated normal form (NNF) before approximating them.

Theorem 1 ([63]) *For each i, if $C_i^\top$ is unsatisfiable then $C_j^\top$ is unsatisfiable for all $j \geq i$, hence C is unsatisfiable. For each i, if $C_i^\bot$ is satisfiable then $C_j^\bot$ is satisfiable for all $j \geq i$, hence C is satisfiable.*

The sequences $C^\top$ and $C^\bot$ can be used to gradually approximate the satisfiability of a concept expression. [63] only replaces subconcepts $D \equiv \exists R.C$ as the worst case complexity depends on the nesting of existential and universal quantifiers. Theorem 1 leads to the following for $C^\bot$-approximation:

$$\begin{array}{llcll} (I \sqsubseteq Q)_i^\bot & \text{is not satisfiable} & \Leftrightarrow & (I \sqcap \neg Q)_i^\bot & \text{is satisfiable} \quad \Rightarrow \\ (I \sqcap \neg Q) & \text{is satisfiable} & \Leftrightarrow & (I \sqsubseteq Q) & \text{is not satisfiable} \end{array}$$

Therefore, we are only able to reduce complexity when approximated subsumption tests are not satisfiable. When an approximated subsumption test $(I \sqsubseteq Q)_i^\bot$ is satisfiable, nothing can be concluded and the approximation continues to level $i+1$ until no more approximation is applicable, i.e., the original concept term is obtained. Analogously, from Theorem 1 one obtains that when $(I \sqsubseteq Q)_i^\top$ is satisfiable this implies that $(I \sqsubseteq Q)$ is satisfiable. When $(I \sqsubseteq Q)_i^\top$ is not satisfiable nothing can be deduced and the approximation continues to level $i+1$.

Approximating Conjunctive Queries

In [68] a method is introduced for approximating conjunctive queries. The method computes a sequence $Q^1, \ldots, Q^n$ of queries such that: (1) $i < j \Rightarrow Q^i \sqsupseteq Q^j$ and (2) $Q^n \equiv Q$. The first property ensures that the quality of the results of the queries doesn't decrease. The second property ensures that the last query computed returns the desired exact result.

The proposed method can easily be adapted for instantiation checks. The computed sequence $Q^1, \ldots, Q^n$ is used to generate the sequence $C_1^\Delta, \ldots, C_n^\Delta$ with $C_i^\Delta = a : Q^i$. Assuming that less complex queries can be answered in less time, instantiation checks can then be speeded up using the following implication:

$$(I \not\sqsubseteq Q') \wedge (Q \sqsubseteq Q') \Rightarrow I \not\sqsubseteq Q$$

In [68] the sequence of subsuming queries $Q^1, \ldots, Q^n$ is constructed by stepwise adding a conjunct (of the original query) starting with the universal query.

A problem that remains to be solved in this approach is a strategy for selecting the sequence of queries to be checked successively. This problem boils down to ordering

the conjuncts of the query which should balance the two factors 'smoothness' and 'time complexity'.

As described in [68] the smoothness of the approximation can be guaranteed by analyzing the dependencies between variables in the query. After translating the conjunctive query to a DL expression, these dependencies are reflected in the nesting of subexpressions. As the removal of conjuncts from a concept expression is equivalent to substitution by $\top$, this nesting provides us with a selection strategy to determine a sequence of approximations S_i where all subexpressions at depth greater or equal than i are replaced by $\top$. Hence, this method is somewhat similar to $C^{\top}$-approximation except that it is restricted to the conjunctive query, i.e., the instance description is not approximated, and it can replace any conjunct in the query with $\top$, not only existentially quantified conjuncts.

Typically, however, queries often have a very flat structure. For example, all queries used in our experiments with the Gene Ontology are of depth one. This means that S_0 is the query $\top$ whereas S_1 is already the original query. To avoid this bad approximation scheme, next we propose an improved strategy.

To overcome the flatness of queries typically found in ontologies, we propose a strategy that also provides an order for subexpressions at the same level of depth. A possible ordering is the expected time contribution of a conjunct to the costs of the subsumption test. As measuring the actual time is practically infeasible, a heuristic is proposed.

For this purpose, we unfold the conjuncts using the definitions of the concepts from the ontology occurring in the conjunct. In order to determine a suitable measure of complexity for expressions, we consider the standard proof procedure for DLs. Most existing DL reasoners are based on tableau methods, which determine the satisfiability of a concept expression by constructing a constraint system based on the structure of the expression. As the costs of checking the satisfiability of an expression depends on the size of the constraint system, we can use this size as a measure of complexity. As determining the exact size of the constraint system requires to run the tableau method, heuristics are used for estimating the size. Based on this estimated size, we determine the order in which conjuncts at the same level of depths are considered.

In the following, we propose a method for estimating the size of the tableau for expressions in $\mathcal{ALC}$ that will be used in the experiments. The tableau rules [2] provide us with quite a good idea about an estimation of the maximal size of the tableau in the worst case. For this purpose, we define a function Φ that assigns a natural number representing the estimated size of the corresponding constraint system to an arbitrary $\mathcal{ALC}$ expression

in the following way:

$$\begin{aligned}
\Phi(A) &= 1 \\
\Phi(\neg A) &= 0 \\
\Phi(C \sqcap D) &= 2 + \Phi(C) + \Phi(D) \\
\Phi(C \sqcup D) &= \phi + 2 + \Phi(C) + \Phi(D) \text{ where } \phi \text{ is the current value of } \Phi(C \sqcup D) \\
\Phi(\exists\, R.C) &= 2 + \Phi(C) \\
\Phi(\forall\, R.C) &= n + n \cdot \Phi(C) \text{ where } n \text{ is the number of existential quantifiers in } \forall\, R.C
\end{aligned}$$

A **and** $\neg A$**:** Atomic concepts are added as a single constraint. Negated concepts are not added as they are merely used to check the existence of a contradiction.

$C \sqcap D$**:** Two new constraints are added. The expressions in these constraints have to be evaluated recursively, therefore, we also have to estimate the number of constraints that will be generated by C and D.

$C \sqcup D$**:** Two new constraints are added and each of the constraints has to be evaluated recursively, however, we have to deal with two separate constraint systems from this point on. The number of constraints in the system at this point has to be doubled. For an estimation we add the current estimation value.

$(\exists\, R.C)$**:** Two new constraints are added, one for the relation and one restricting the object in the relation to C Object y has to be evaluated recursively.

$(\forall\, R.C)$**:** A new constraint has to be added for every existing constraint xRy in the constraint system S and each one has to be evaluated recursively. As we do not know how many of these statements are or will be in S, we use the overall number of existential quantifiers in the expression that can lead to the addition of these constraints as an upper bound.

The value Φ can now be computed for each conjunct in the query and be used as a basis for determining the order in which conjuncts at the same level of nesting are processed.

Generalizing the Approximation of Conjunctive Queries

The approach of previous section can easily be generalized. Also the approach of Cadoli and Schaerf [63] can be realized with the help of some selection function Φ_{CS}. Then Φ_{CS} returns for each conjunct the appropriate number which induce a substitution in the exact same order as the original strategy would propose.

But several other strategies are possible. Instead of estimating the time which is needed to reason with some conjuncts (see previous section) one can estimate how effective a conjunct would be for the reducing the complexity in the search tree. I.e. how effective this conjunct would be in reducing the number of returned answers. Obviously those conjuncts should be preferred which eliminate most incorrect elements from the answer set. This techniques is well-known for database optimization where the number of entries in tables is considered when a join between these tables has to be computed.

However besides the particular strategy the question may raise how big the potential for approximation might be in general. If we assume that we always have the best strategy Φ_{opt} then we can estimate the success factor of the rewriting technique itself independent from the particular Φ. This investigation is motivated by our first experiments where the first results show some performance improvements, but only with a limited value (less than 20 percent). However in these cases the selection seemed to be optimal can not be further improved. This might introduce the suspect that in general the potential for performance improvements with rewriting techniques might be limited - independent how good your selection function might be. The results of this investigation is reported in the next section.

2.3.1 Experimental Evaluation

In this section experimental results are shown of the approaches described in the previous sections. The main question focused on in the experiments is *if*, and if yes, in *what way* does approximation reduce the complexity of the retrieval task. We focus on the number of operations needed and the overall computation time used. The goal of our approximation approach is to replace costly reasoning operations by a (small) number of cheaper approximate reasoning operations. The approximation methods used are sound and complete. Therefore, the suitability of the approximation methods depend solely on the time gained (or lost) when classical operations are replaced by a number of approximate ones.

Our experiments were made with the Gene ontology and Instance Store [43]. The focus of our experiments are those queries where Instance Store cannot replace all DL reasoning with database retrieval, but must still check the instantiations of some instances. These instantiation checks were found to be a bottleneck in the scalability of this approach. We originally started with 17 queries (with $Q1$ to $Q6$ user formulated queries and queries $Q7$ to $Q17$ artificial), but discarded the queries that didn't require instantiation checks from further experiments.

The results of the experiments are shown in Table 2.1 on page 30, which is divided into five columns with each column reporting the number of subsumption testsperformed. The first column reports results for the experiment without any approximation, the second column with $C^{\top}$-approximation, the third column with $C^{\bot}$-approximation, the fourth column with C^{Δ}-approximation, and the fifth column with optimal C^{opt}-approximation. Each column is further divided into smaller rows and columns. The rows represent the level of the approximation used, where *no* denotes without approximation, and Li denotes the level of the approximation approach. The subcolumns show the number of subsumption tests that resulted in true or false.[2] This distinction is important, because Section 2.3 tells us that only when a $C^{\top}$-approximated subsumption succeeds, or a $C^{\bot}$- or C^{Δ}-approximated subsumption test fails we obtain a reduction in complexity.

[2]We will use the shorthand 'true subsumption test' and 'false subsumption test' to indicate these two distinct results.

	normal	$C^{\top}$	$C^{\bot}$	C^{Δ}	C^{opt}	
Q2	175	348	299	547	360	(-105.71%)
Q8	5373	8383	7753	9912	4811	(10.46%)
Q12	61410	93100	85764	56478	57495	(6.38%)
Q14	4372	6837	6017	7391	3449	(21.11%)
Q15	61560	90847	83714	114162	49956	(18.85%)
Q17	113289	158218	144689	93074	57647	(49.12%)

Figure 2.1: Time needed for Subsumption tests (in milliseconds)

2.3.2 Discussion

Let us first focus on the question *if* the approximation methods can lead to any reduction in complexity. Table 2.1 on page 30 shows that $C^{\top}$- and $C^{\bot}$-approximation cannot reduce the number of normal subsumption tests. C^{Δ} is able to reduce, except for $Q15$, all false subsumption tests to 0. But obviously only C^{opt} is able to reduce the number of all false subsumption tests.

The first column in Table 2.1 shows that much more false subsumption tests are needed than true subsumption tests. This indicates that $C^{\top}$-approximation is wrong in this approach as it can only be used to lower the complexity of true subsumption tests, which is negligible when compared to false subsumption tests. This may explain its bad approximating behavior, however, $C^{\bot}$ also performs badly, which does approximate false subsumption tests. Closer analysis shows that *term collapsing* [34], i.e., the substitution of terms by $\top$ or $\bot$ results in the query becoming equivalent to $\top$ or $\bot$, is the reason for this. An analysis of $C^{\bot}$ shows that this occurs in *all cases*.

Apart from looking at *if* an approximation method can successfully reduce the number of normal subsumption tests, we must also consider the cost for obtaining the reduction, i.e., in *what way* are the normal subsumption tests reduced. For example, approximating $Q8$ changes $607 = 10 + 597$ normal subsumption tests into 10 normal subsumption tests, 607 C_1^{Δ} subsumption tests, and 607 C_0^{Δ} subsumption tests. Thus, the number of subsumption tests may increase, but the complexity of most tests will be lower than normal. Note however, that some computations seem unnecessary as nothing can be deduced from them, e.g., the 607 C_0^{Δ} tests. Obviously, in this approach unnecessary subsumption tests should be minimized. Several cases can be observed in the experiments with C^{Δ}-approximation. Either no subsumption test is unnecessary ($Q12, Q17$), some subsumption tests are unnecessary ($Q2, Q8, Q14$), or all subsumption tests are unnecessary ($Q15$).

For the optimal approximation C^{opt} one can see that no subsumption test is unnecessary for all queries. All subsumption tests are performed in that way that all (necessary) false subsumption are performed as early as possible. So C^{opt} shows the best possible approximation when replacing terms with top $\top$.

This distinction seems to influence the overall time needed when approximating a query. Table 2.1 reports the overall time in milliseconds needed for each query. For comparison $C^{\top}$ and $C^{\bot}$ are also reported. For queries having unnecessary subsumption tests, approximation always leads to more computation time. In those cases, reducing the complexity of subsumption tests do not weigh up to the costs of additional (unnecessary) subsumption tests. For queries having no unnecessary subsumption tests, approximation does save time when compared to the normal case.

However for the optimal approximation C^{opt} there is some time reduction for nearly each query. This time reduction could be expected because a lot of normal false subsumption tests are replaced by cheaper false subsumption test. The additional true subsumption tests do not damage this balance. However with respect to the number of subsumption tests performed the ammount of time reduction is disappointing. In most cases less than 20 percent (percentage number in brackets in the last column of Table 2.1) with respect to the normal case could be saved. If we consider the worst case complexity of DL reasoners then this reduction is only a negligible improvement which may have no effect in practice.

Another observation of Table 2.1 (on page 30) is that false subsumption tests for C^{Δ} only occur at *one level*. It seems that the conjunct that is added to the approximated conjunctive query on which the false subsumption tests occur is crucial in determining the outcome.The role of conjunct in a subsumption test is still unclear. More research is needed if this conjunct (or a group of conjuncts) can be identified in advance to speed up approximation.

2.3.3 Conclusions

Instance retrieval is one of the most important inferences in the Semantic Web. In order to make methods more scalable for ontologies with a large set of instances we investigated two approximation methods and evaluated them on a benchmark set. Both methods use a similar idea, i.e., removing parts of an expression to make it simpler to speed up retrieval. However, the method of [63] shows bad approximating behaviour because the selection and substitution of subconcepts is too restrictive. The method of [68] was extended with a heuristic for subconcept selection and shows some potential for speeding up instance retrieval. More research is needed to improve the heuristic and to determine if the approximation method can be used to speed up instance retrieval. However, even the optimal approximation shows that only a limited ammount of performance improvement can be achieved. It has to be checked if this behaviour is also comprehensible for other data sets.

2.4 Robust Query Processing

by PETER DOLOG, MALGORZATA MOCHOL, LYNDON NIXON, HEINER STUCKENSCHMIDT & HOLGER WACHE

2.4.1 Motivation

Research in Cooperative Query answering is triggered by the observation that users are often not able to correctly formulate queries to databases that return the intended result. Cooperative query processing supports the user by automatically modifying the query in order to better fit the real intention of the user. Based on the assumed kind of mismatch between the users intention and the formulated query there are different techniques used. Gaasterland et al [27] provide a unifying view on different relaxation techniques. Here we described an approach for *relaxing conjunctive queries* over description logic knowledge bases by removing conjuncts from the query in a particular order (see 2.2 and [68, 76]). In particular, we

- describe a framework for information access that provides query relaxation trogh query rewriting in order to provide robust, personalized access to heterogeneous RDF data,
- propose an implementation of the framework in terms of conditional rewriting rules for RDF query patterns, and
- discuss the application of the framework in the context of an existing e-learning system and job portals.

2.4.2 Rewriting RDF Queries

We propose a simple but expressive approach for query rewriting to solve the problem of over-constraint queries. This rewriting relaxes the over-constraint query based on rules and in order defined by some conditions. This has an advantage that we start with the strongest possible query that is supposed to return the “best” answers satisfying most of the conditions. If the returned result set is either empty or contains unsatisfactory results, the query is modified either by replacing or deleting parts of the query, or in other words relaxed. The relaxation should be a continuous step by step, (semi-)automatic process, to provide a user with possibility to interrupt further relaxations. Before we investigate concrete relaxation strategies in the context of our example domain, we first give a general definition of the framework for re-writing an RDF query.

The RDF data model foresees sets of statements which are in the form of triples [38]. In [24] Dolog et.al. proposed a rule-based query rewriting framework for RDF queries independent of a particular query language which we summarize here. The framework is

based on the notion of triple patterns (RDF statements that may contain variables) as the basic element of an RDF query and represents RDF queries in terms of three sets:

- triple patterns that must be matched by the result (mandatory patterns)
- triple patterns that may be matched by the results (optional triple patterns).
- Conditions in terms of constraints on the possible assignment of variables in the query patterns.

More precisely Dolog et.al. define a (generic) RDF query as a triple of these three sets.

Definition 3 *RDF Query*
Let $\mathcal{T}$ be a set of terms, $\mathcal{V}$ a set of variables, $\mathcal{RN}$ a set of relation names, and $\mathcal{PN}$ a set of predicate names. The set of possible triple patterns $\mathcal{TR}$ is defined as $\mathcal{TR} \subseteq (\mathcal{T} \cup \mathcal{V}) \times (\mathcal{RN} \cup \mathcal{V}) \times (\mathcal{T} \cup \mathcal{V})$. A query Q is defined as the tuple $\langle M_Q, O_Q P_Q \rangle$ with $M_Q, O_Q \in \mathcal{TR}$ and $P_Q \subseteq \mathcal{P}$ where M_Q is the set of mandatory pattern (patterns that have to be matched by the result, O_Q is a set of optional pattern (patterns that contribute to the result but do not necessarily have to match the result) and $\mathcal{P}$ is the set of predicates with name $\mathcal{PN}$, defined over $\mathcal{T}$, and $\mathcal{V}$.

A result set of such a RDF query is set of substitutions. Formally a substitution τ is a list of pairs (X_i, T_i) where each pair tells which variable X_i has to be replaced by $T_i \in \mathcal{T} \cup \mathcal{V}$. A ground substitution replaces variables X_i by a term and not by another variable, i.e. $T_i \in \mathcal{T}$ for all i. The (ground) substitution τ replaces variables in M_Q and O_Q with appropriate terms. If $\tau(M_Q)$ is equal to some ground triples then the substitution is valid. All valid ground substitutions for M_Q plus existing ground substitutions for O_Q constitute answers to the query. Additionally the predicates P_Q restrict these substitutions because only those bindings are valid answers where the predicates, i.e. $\tau(P_Q)$, are also satisfied. The predicates additionally constraint the selection of appropriate triples.

Re-writings of such queries are described by transformation rules $Q \xrightarrow{R} Q^R$ where Q the original and Q^R the rewritten query generated by using R. Rewriting rules consists of three parts:

- A matching pattern represented by a RDF query in the sense of the description above
- A replacement pattern also represented by an RDF query in the sense of the description above
- A set of conditions in terms of special predicates that restrict the applicability of the rule by restricting possible assignments of variables in the matching and the replacement pattern.

Based on the abstract definition of an RDF query, we can now define the notion of a rewriting rule. We define rewriting in terms of rewriting rules that take parts of a query, in particular triple patterns and conditions, as input (PA) and replace them by different elements (RE).

Definition 4 *Rewriting Rule*
A rewriting rule R is a 3-tuple $\langle PA, RE, CN\rangle$ where PA and RE are RDF queries according to Definition 3 and CN is a set of predicates.

For conditions the same constructs as for queries are used where the possible results are also constrained by predicates. Patterns and replacements formally have the same structure like queries. They also consist of a set of triples and predicates. But patterns normally do not address complete queries but only a subpart of a query. Normally the subpart addresses some triples as well as some predicates in the query. In order to write more generic rewriting rules the pattern must be instantiated which is done by an substitution.

Definition 5 *Pattern Matching*
A pattern PA of a rewriting rule R is applicable to a query $Q = \langle M_Q, O_Q, P_Q\rangle$ if there are subsets $M'_Q \subseteq M_Q$, $O'_Q \subseteq O_Q$ and $P'_Q \subseteq P_Q$ and a substitution θ with $\langle M'_Q, O'_Q, P'_Q\rangle = \theta(PA)$.

In contrast to term rewriting systems [5] the definition of a query as two sets of triples and predicates differentiate the pattern matching. The identification of the right subpart of the query for the pattern match is simplified because of the use of sets. Only a subset of both sets has to be determined which must be syntactically equal to the instantiated pattern. Please note that due to set semantics, the triples and predicates in the pattern may be distributed over the query.

A re-writing is now performed in the following way: If the matching pattern matches a given query Q in the sense that the mandatory and optional patterns as well as the conditions of the matching pattern are subsets of the corresponding parts of Q then these subsets are removed from Q and replaced by the corresponding parts of the replacement pattern. The application of R is only allowed if the predicates in the conditions of R are satisfied for some variable values in the matching and the replacement pattern.

Definition 6 *Query Rewriting*
If a rewriting rule $R = \langle PA, RE, CN\rangle$

- *matches a query $Q = \langle M_Q, O_Q, P_Q\rangle$ with subsets $M'_Q \subseteq M_Q$, $O'_Q \subseteq O_Q$ and $P'_Q \subseteq P_Q$ substitution θ and*
- *$\theta(CN)$ is satisfied*

then the rewritten query $Q^R = \langle M_Q^R, O_Q^R, P_Q^R \rangle$ *can be constructed with* $M_Q^R = (M_Q \setminus M'_Q) \cup \theta(M_{RE})$, $O_Q^R = (O_Q \setminus O'_Q) \cup \theta(O_{RE})$ *and* $P_Q^R = (P_Q \setminus P'_Q) \cup \theta(P_{RE})$ *with* $RE = \langle M_{RE}, O_{RE}, P_{RE} \rangle$.

The former definition clarifies formally how to generate a rewritten query Q^R from Q with the help of R, i.e. $Q \xrightarrow{R} Q^R$. We denote with $Q^{\mathcal{R}}$ all queries which can generated from Q with all rules $R \in \mathcal{R}$. On each query from $Q^{\mathcal{R}}$ the rewriting rules can be applied again. We denote with $Q^{\mathcal{R}*}$ all queries which can be generated by application of the rules in $\mathcal{R}$ successively.

2.4.3 Application to E-learning

One application area on robust querying is in the domain of open learning repositories where learning resources (courses, exercises, modules, etc.) are annotated with RDF metadata to allow users to find suitable material for his or her learning goal.

The formal apparatus introduced in the previous section provides us with a general mechanism for relaxing RDF queries based on certain patterns and conditions. We have developed a specialized rule language that implements this mechanism which we will discuss in this section. In order to successfully use this language to relax over-constraint queries, we need a strategy for successively applying relaxations in such a way that we find answers that match the interests of the user as closely as possible and implement it in terms of query rewriting rules.

The main problem with query relaxation in general is the fact that it is almost impossible to find generic relaxation strategies that work well across different applications. A good strategy rather depends on many factors including the nature of the information and the goals of the user. A solution for this problem is to employ explicit knowledge to drive the relaxation of a query. Corresponding to the factors that influence the usefulness of a strategy, there are two sources of knowledge we use for relaxation:

- Domain and Application knowledge;
- Knowledge about the user and user preferences.

The former represents a domain knowledge about dependencies between predicates and ordering according to their importance for queries within a domain. The second type of knowledge concerns the interests of the users and his profile. For example, to correctly determine the content of a resource in e-learning domain, we should first look into the subject, then the title, and finally the description in its metadata. This information can be used to constrain rewriting rules for the subject of the target resource in the title of the resource and the rule that looks for it in the description, in a way that the rewriting for title is performed before the rewriting for the description. Therefore, the order of importance

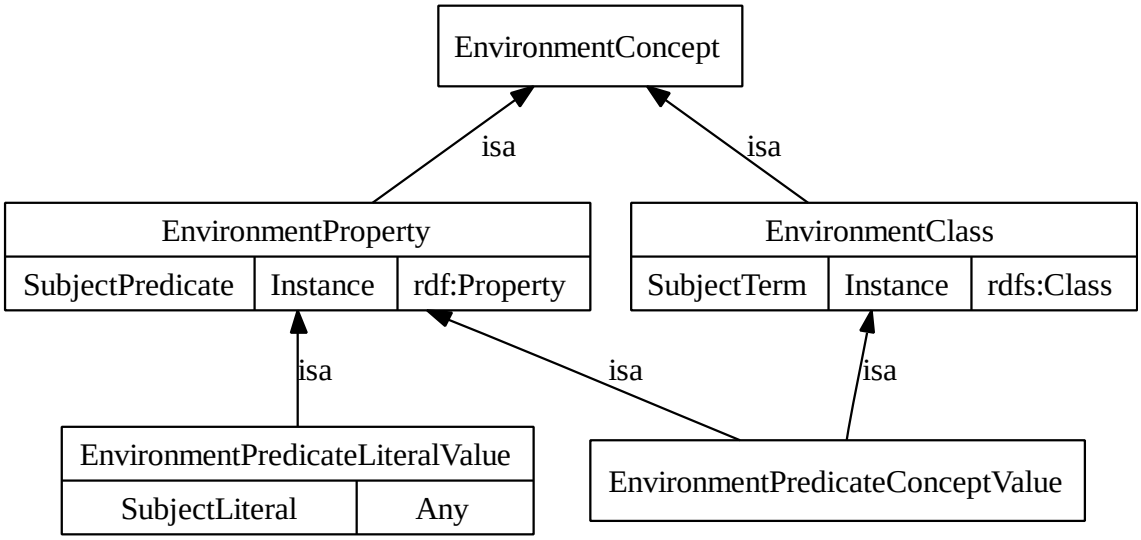


Figure 2.2: A Schema for Generic Environment

between predicates in a domain serves as an order in which the rewriting rules should be applied. We will show later how this approach can be implemented using our rewriting approach. Another example relates to the structuring of the domain. If for example the language of a learning resource is not mentioned in the metadata, we can for example look at other learning resources that are part of the same course.

The second type of knowledge concerns the interests of the users. These interests are hard to determine automatically as they are influenced by many factors. A common approach is to use an explicit model of user preferences in terms of a user model. In the context of e-learning, this user model contains information about topics of user's interest, previous knowledge and preferences with respect to the type and format of learning resources, and so on.

Environment and Preferences

In order to include knowledge about the domain of interest and the preferences of the user into the query relaxation process, we have designed a general scheme for representing relevant knowledge independent of a concrete application. This general scheme exploits the meta-modeling capabilities of RDF to define aspects of the world we can take into account in the rewriting process (compare fig. 2.2).

The schema follows an idea, that each environment can be generated according to an application domain schema used by the application. Rather than directly representing domain knowledge or user preferences it provides metaclasses that can be instantiated by existing representation schemes for information resources such as Learning Object Metadata (LOM) [54] as well as metadata schemas like the Dublin Core standard [19], and taxonomies and ontologies used for predicate values in the information resource schemas

such as ACM computing classification system [56].

Domain Knowledge and Relaxation

In general the rewriting is a very powerful approach in order to manipulate the overconstrained query. With replacing parts of a query we can realize five types of actions:

- *Making Triples/Predicates optional* — this provides a query which considers a situation that some of the triples/predicates do not have to appear in metadata. A query then gives also results where particular predicate relaxed to an optional predicate does not occur;
- *Replacing Value* — this provides a query where particular predicate value is replaced with another value. Taxonomies may be used to provide siblings, more general terms, and so on;
- *Replacing quantifiers and operators* — this provides a query where quantifiers (like forall and exists) are replaced for each other. It also includes operators replacement like AND for OR, equal for a range, and so on;
- *Replacing Triples/Predicate* — this provides a query where particular triple resp. predicate in restrictions is replaced by another triple resp. predicate. A domain knowledge is employed for this purposes. For example, if a subject query is not satisfied, it may be replaced by title query with similarity measures;
- *Deleting Triple/Predicate* — this provides a query where particular predicate is deleted from a query completely.

As such, these operations are independent of the application domain and the user preference. A connection to the knowledge described above is made through the elements of the query that are affected by the corresponding operation. In most cases, we can identify a certain property that is affected. For example, in the learning environment a user searches a resource with a specific subject. Figure 2.3 shows a related RDF query.

But if there is no resource with that subject then we would like to relax the query that the subject term can also appear in the title of a resource description. This strategy can be derived from an environment preference stating that the ”subject” relation has the highest priority as it can be assumed to most precisely reflect the content of a resource followed by the ”title” and finally the ”description” relation. In the environment preference model, this order is described in terms of the hasImportanceOver relation. For the actual relaxation process each of these relations is implemented by a rewriting rule. The fist rewriting rule for that relaxation is specified in Figure 2.4.

Obviously, `PATTERN` defines the pattern, the `REPLACE-BY` the replacement, and `WITH` the conditions for the rewriting. The pattern contains one triple and one predicate.

```
SELECT * FROM
  {Resource} subject {Subject},
  {Resource} title {Title},
  {Resource} description {Description},
  {Resource} language {Language},
  {Resource} requires {} subject {Prerequisite},
  {User} hasPerformance {Performance},
  {Performance} learning_competency {Competence}
WHERE
  Subject Like "inference engines",
  Prerequisite = Competence,
  Language = de,
  User = user42
```

Figure 2.3: Query extended with user preferences

The triple {Resource} subject {Subject} looks for any resource Resource with subject Subject. The predicate Subject Like Value ^^xsd:string constrains the variable Subject to the user's term (Value of type string), i.e. the subject the user is looking for. If a query contains such an triple and such an predicate then the rewriting rule is applicable.

The replacement part of the rule defines how the matched triple and predicate has to be replaced. The triple is extended by the second triple {Resouce} title {Title}. The second triple now allows to refer to the title of a resource. The first triple about the subject of the resource is not removed because there may be some other triples or predicates in the query which may refer to the subject of the resource. But the predicate is no longer needed and is completely substituted by the predicate TMPTitle Like NEWTitle, which now try to constrain the title of the resource instead of the subject. The variable NEWTitle is determined in the conditions. With the build-in function concat the value is prefixed and finished with a star which means that the title must only contain the subject the user is looking for.

The rewriting rule from figure 2.4 can be applied to the query in Figure 2.3. The result is shown in Figure 2.5. Note that now the query refer twice to the title of a resource. The second reference was introduced by the rewriting.

User Preferences and Relaxation

Another kind of relaxation is the rewriting the overconstrained query according to the knowledge about the user. In the learning scenario the user might prefer learning resources in German but Dutch may also be okay. This knowledge is used to refine the query, i.e.

```
PATTERN
  {Resource} subject {Subject}
  WHERE
  Subject Like Value^^xsd:string,
REPLACE-BY
  {Resource} subject {Subject},
  {Resource} title {TMPTitle}
  WHERE
  TMPTitle Like NEWTitle
WITH
  NEWTitle = concat("*",concat(Value,"*"))
```

Figure 2.4: Simple rewriting rule

looking for resources in German. However if there is no resources in German then the query can be relaxed according to user's second preference.

As described in [24], our environment preference model allows user to specify an importance order between predicates. In contract to the domain preferences mentioned in the last section that can be specified inside the application, these preferences can be different for each user. As a consequence we have to provide an interface where each user can specify his or her personal preferences that can then be stored in the user profile. Using our general environment model, these preferences can be used in the same way as domain preferences once they have been entered by the user. In particular, the `hasImportanceOver` relation then defines conditions which are satisfied just when particular predicate is on its turn to rewrite it.

Conditions for User-constrained Relaxation

Conditions play a crucial part for rewriting queries according to user's preferences. They can control when particular rewriting rule can applied.

Formally, conditions can be predicates where variables which are used in the pattern and the replacement are set together with some built-in functions for manipulating strings or numbers. The condition in the simple rewriting rule of Figure 2.4 is such an example; the used predicate is equality. But a condition can also be a query which should return at least one result in order to be satisfied and to bind variables to the values returned by the query. In this case, a query behaves like a normal predicate. But only the first result will be used during the rewriting; further results will be ignored. We use such queries to refer to users profile and user preferences.

An example is given in Figure 2.6. The rule try to relax user's first language preference to his second reference as stored in his profile. The condition of that rewriting rule starts from the root of his profile (`User`) to find his two preferences `Preference` and

```
SELECT * FROM
  {Resource} subject {Subject},
  {Resource} title {TMPTitle},

  {Resource} title {Title},
  {Resource} description {Description},
  {Resource} language {Language},
  {Resource} requires {} subject {Prerequisite},
  {User} hasPerformance {} learning_competency {Competence}
WHERE
  TMPTitle Like "*inference engines*",
  Prerequisite = Competence,
  Language = de,
  User = user42
```

Figure 2.5: Relaxed query extended with user preferences

`Preference2` whereas the first preference is preferred to the second preference (relation `hasImportanceOver`). Both preferences refer to an environment item of type `EnvironmentPredi-`
`cateConceptValue` with `subjectTerm language`. Their values are addressed in the pattern resp. replacement. The value of the first preference is replaced by the value of the second preference. The above rewriting rule can easily be generalized to a rewriting for any value preference which the user related with the relation `hasImportanceOver`. The condition then would be that the environment items of such preferences must not refer to the subject predicate `language` but only to the same predicate (which is represented as a variable). So a variable instead of the literal `language` in the condition yields into a general rewriting rule for value preferences.

2.4.4 Application in the job portal

In a Semantic Web-based recruitment scenario the data exchange between employers, applicants and job portals is based on a set of vocabularies which provide shared terms to describe occupations, industrial sectors and job skills [50].

In semantic annotated job portals each job request and opening is annotated with an RDF description which is a set of triples. A query over these job openings is formulated as triple patterns and a set of conditions that restrict the possible variables bindings in the patterns. Each triple pattern represents a set of triples. The corresponding abstract definition of a query focuses on the essential features of queries over RDF.

When implementing a (Semantic Web) job portal the requirements of the system depend on the meaningful use cases which are derived by the industrial project partner from

```
PATTERN
  {Resource} language {Language}
  WHERE
  Language = L,
  User = UserID
REPLACE-BY
  {Resource} language {Language}
  WHERE
  Language = L2,
  User = UserID
WITH
  SELECT * FROM
   {User} hasPreference {Preference},
   [{User} hasPreference {Preference2}],
   {Preference} hasImportanceOver {Preference2},

   {Preference} EnvironmentItem {Item},
   {Item} type {EnvironmentPredicateConceptValue},
   {Item} subjectPredicate {language},
   {Item} subjectTerm {L},

   {Preference2} EnvironmentItem {Item2},
   {Item2} type {EnvironmentPredicateConceptValue},
   {Item2} subjectPredicate {language},
   {Item2} subjectTerm {L2}
  WHERE
   true
```

Figure 2.6: Rewriting rule for language preferences

its day to day business practices within the HR-domain. To clarify the still outstanding problems we will briefly present one of such use cases which (at first view) seems to be quite simple. However if we look closer and try to represent the data in an ontology or satisfy the requirements in the semantic portal we will meet some difficulties which at the same time show the complexity of such “simple” queries.
We are looking for a person which:

1. has an degree in computer science
2. wants to work in software consulting and development,
3. is an expert in C, Java, PHP, UML, .Net and WindowsNT,
4. has worked for at least 5 years in an industrial and 5 year in a research project,

5. should have experience as project or team manager,
6. should be not older then 25

Looking for such a person requires from the system to translate this free text description into an instance retrieval problem. The query must be translated into a concept expression. The retrieval process will return all job seekers which belong to that concept expression, i.e. satisfying all the requirement in the concept expression. The following OWL[3] expression shows the concept expression for some person who has experience in some of (the intersectionOf property) the OWL classes C, Java, PHP or UML [4] .

```
<owl:Class rdf:ID="Query">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Person"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Class rdf:about="C"/>
            <owl:Class rdf:about="Java"/>
            <owl:Class rdf:about="PHP"/>
            <owl:Class rdf:about="UML"/>
          </owl:intersectionOf>
        </owl:Class>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasExperience"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

In the following we give some examples for the rewriting rules which use the aforementioned example as a basis.

A very simple rewriting rule takes into account the required skill e.g. Java. It relax the some requirements in the experiences, i.e. instead of `JAVA` the `PureObjectOrient-`

[3]OWL is an extension of RDF allowing for more expressive features than RDF like number restrictions etc.

[4]Originally we modelled these as nominals (enumerations like Week =Monday, Tuesday, ...). Nominals are instances and classes at the same time. However current DL systems have problems with nominals therefore we use classes in the current approach.

edLanguages or even the ObjectOrientedLanguages could be possible weakenings of the original query:[5]

```
PATTERN
   <owl:Class rdf:about="Java"/>
REPLACE-BY
   <owl:Class rdf:about="PureObjectOrientedLanguages"/>
WITH
   true.
```

This means that whenever anywhere the term representing the Java language in a query appears then it can be replaced by a more general term representing pure object oriented languages, of which Java is one.

Making use of the predicates we can generalize previous rewriting rules and generate generic rules that are guided by information from the ontology. The predicate subsumed for example is satisfied when X is more specific than Y. With the following rewriting rule we are able to consider the knowledge in the ontology.

```
PATTERN
   <owl:Class rdf:about="X"/>
REPLACE-BY
   <owl:Class rdf:about="Y"/>
WITH
   subsumed(X,Y).
```

In the same way some number restrictions can be applied. In our example the requirement that a person has experiences in a five year industrial project is encoded with the help of the (artificial) class FiveYearsOrMore. This class represents all Numbers representing years which are larger or equal to five. This class can be replaced by the class TwoYearsOrMore which obviously is more general (weaker) then the former. Furthermore we can restrict the replacement in that way that we only allow this for the restriction on property hasDuration. The corresponding rewriting rule look like:

```
PATTERN
   <owl:Restriction>
       <owl:onProperty rdf:resource="#hasDuration"/>
           <owl:someValuesFrom>
               <owl:Class rdf:ID="FiveYearsOrMore"/>
           </owl:someValuesFrom>
       </owl:Restriction>
REPLACE-BY
```

[5] For the sake of readability the examples are simplified.

```
   <owl:Restriction>
       <owl:onProperty rdf:resource="#hasDuration"/>
           <owl:someValuesFrom>
               <owl:Class rdf:ID="TwoYearsOrMore"/>
           </owl:someValuesFrom>
       </owl:Restriction>
WITH
   true.
```

2.4.5 Ordering different rewriting rules

The main problem of the re-writing approach to query relaxation is the definition of an appropriate control structure to determine in which order the individual rewriting rules are applied to general new queries. In other words how to explore $Q^{\mathcal{R}*}$. Different strategies can be applied to deal with the situation where multiple re-writings of a given query are possible. Example is a Divide and Conquer strategy: The best results of each possible combinations of re-writings is returned. In the current version of the system we have implemented a simple version with similarities to skylining [45, 47] which is well-known in database query relaxation.

In particular, we interpret the problem of finding relaxed queries as a classical search problem with small adaptations. The search space is defined by the set $Q^{\mathcal{R}*}$ of all possible queries. Each application of a rewriting rule R on a query Q is a possible action denoted as $Q \xrightarrow{R} Q^R$. A query represents a goal state in the search space if it does have answers. In the current implementation we use breadth-first search for exploring this search space. Different from classical search, however, the method does not stop when a goal state is reached; each goal state has to be determined because each goal state represent one sequence of successful rewriting when answers are provided. However a goal state need not to be explored further (i.e. no further re-writings must be applied to a goal state) but each search branch has to be closed by a goal state (or by a predefined depth). Breadth-first search ensures that each goal state represents the best solution to the relaxation problem with respect to a certain combination of re-writings. The goal states form a "skyline" for the rewriting problem and each of them is returned to the user together with the query answers.

The second difference to classical search is that we do not allow the same rule to be applied more than once with the same parameters in each branch of the search tree. The only kind of rules that can in principle be applied twice are rules that add something to the query (Rules that delete or replace parts of the query disable themselves by removing parts of the query they need to match against). Applying the same rule that extend the query twice leads to an unwanted duplication of conditions in the query that do not change the query result, but only increase the complexity of query answering.

2.4.6 Related Work

Query relaxation have been studied in the context of cooperative query answering where information systems explicitly attempt to cooperate with their users [27]. Relaxation is a form of generalization where the scope of a query is extended through rewriting so that more information can be gathered in the answers. A query rewriting approach in deductive databases with a help of specialized clauses is presented in [28]; if the head appears in a query then the head is replaced by the body of the specialized clause. A variant is presented in [29, 30] where users preferences are directly annotated to the logical atoms.

The work presented here shares the principle of query relaxation but proposes a framework for the semantic web meta data. The semantic web environment differs from the databases in its less strict way of annotating the resources and heterogeneity. Therefore, more dimensions of knowledge about where to find the missing information have to be considered in a relaxation framework. Furthermore, our approach is more expressive due to the fact that the pattern of a rewriting rule can refer to more than one item [28] and use the more intuitive matching than unification.

Our approach also relates to term rewriting systems (see e.g. [5]) and graph transformation (see e.g. [61]) Our approach assumes the RDF queries where the set semantics of triple patterns and the conditions simplify considerably the rewriting process.

Furthermore, query relaxation approaches based on query rewriting (and many term rewriting approaches) proposed so far lack on conditionals. In our approach, we are able to define conditions which guard rewriting that it can take place just when the conditions are met. This allows us control the rewriting process. Moreover, with the help of conditions in the rewriting rules we are able to incorporate user profiles and user preferences which is separated from the data itself; annotating the RDF data of each distributed resource directly with all user preferences as proposed by [28] is not applicable for the semantic web.

Preference models have been studied in the fields of databases and artificial intelligence. A foundation on preference models in database systems has been given in [44]. A model for numerical and lexicographical preferences is given. An algebra which defines modification operators for such preferences is given as well. The preference model we have defined considers the partial order between several preferences similarly. We also allow for the numerical preferences ratings which is stored directly from a position of slider at the user interface. Contrary, we distinguish the predicate or schema preferences from the value preferences and their order. This allows us to order the relaxation steps in a way given by the order of preferences.

A query relaxation approach for discovering web services matching user goals has been proposed in [6]. They define preference model as a domain ontology, similarly to our approach. The approach differs in an algorithm for computing relaxation order. The main difference is that the preference model does not deal with ordering between predicates and values separately. It assumes them as bound together. Furthermore, it does not

consider any ordering relations between the predicates. The order of relaxation is then computed according to combinations provided as levels. The levels are computed according to siblings in preference ontologies which are considered in several query predicates. As oppose, in our approach we consider order between the predicates and values separately. This gives us more informed strategy for computing order of relaxation steps.

Our environment and preference ontology relates to the work on CP-NETs [13] and TCP-NETs [14] in artificial intelligence. The formalism allows to specify importance over variables and values as in our approach. This means that the model for environment preferences using semantic web formalism is transformable to the TCP-NETs. This allows to employ a reasoning about ceteris paribus used in preference models based on the TCP-NETs. In our approach, we have used the preference model for query approximation based event-condition-action and term rewriting system.

2.4.7 Conclusions and Further Work

In this chapter, we have proposed a framework for query relaxation to provide personalized information access to resources on the semantic web. The framework is based on the event-condition-action (ECA) paradigm where events are matching patterns, conditions are based on ordering between concepts of common sense domain knowledge and user preferences, and actions are the replacements for relaxing a query. The relaxation is based on the term rewriting principles enhanced with conditions provided by the ECA paradigm. This integration is a contribution to the term rewriting domain. The relaxation is controlled by conditions from domain and user preference ontology. The order is given by importance of predicates and values in the ontology for environment preferences, user profile, and common sense domain knowledge. This makes the approach very well suitable for the access to metadata on the semantic web as the domain knowledge helps to overcome the fact of heterogeneity and differences in how the metadata are authored on the semantic web.

In our further work, we would like to concentrate on ordering of the different rewriting possibilities and the algorithms for determining termination of relaxation. We have considered several strategies in this chapter but it requires further studies to give a recommendation how to decide among them. We also would like to experiment with different user preference models and how they contribute to the relaxation process. Last but not least, user preference elicitation methods and techniques needs to be studied to get as accurate user preferences as possible to support personalized access to information on the semantic web.

		normal		$C^\top$			$C^\bot$			C^Δ			C^{opt}		
		true	false		true	false		true	false		true	false		true	false
Q2	no	9	11	$L0$	0	19	$L0$	19	0	$L0$	20	0	$L0$	9	11
				no	9	11	no	9	11	$L1$	20	0	$L1$	9	0
										$L2$	9	11	$L2$	9	0
										no	9	0	no	9	0
Q8	no	10	597	$L0$	0	606	$L0$	606	0	$L0$	607	0	$L0$	10	597
				no	10	597	no	10	597	$L1$	10	597	$L1$	10	0
										no	10	0	no	10	0
Q12	no	15	7856	$L0$	0	7871	$L0$	7871	0	$L0$	15	7856	$L0$	15	7856
				no	15	7856	no	15	7856	no	15	0	no	15	0
Q14	no	5	403	$L0$	0	407	$L0$	407	0	$L0$	408	0	$L0$	5	403
				no	5	403	no	5	403	$L1$	5	403	$L1$	5	0
										$L2$	5	0	$L2$	5	0
										no	5	0	no	5	0
Q15	no	46	6647	$L0$	0	6693	$L0$	6693	0	$L0$	6693	0	$L0$	46	6647
				no	46	6647	no	46	6647	no	46	6647	no	46	0
Q17	no	1	7872	$L0$	0	7873	$L0$	7873	0	$L0$	1	7872	$L0$	1	7872
				no	1	7872	no	1	7872	no	1	0	no	1	0

Table 2.1: Performed Subsumption Tests

Chapter 3

Approximate ABox-Reasoning with Screech

by PASCAL HITZLER, SEBASTIAN RUDOLPH & DENNY VRANDECIC

We report on evaluations of the Screech approximate reasoning system which was introduced in Deliverable D2.1.2.2v1. Screech performs complete ABox reasoning and trades soundness for time. It is based on the transformation algorithms underlying the KAON2 OWL reasoner.[1]

In this chapter, we will briefly review the Screech approach, starting with a birds' eyes perspective on KAON2. We omit details and discussions which have already been discussed in Deliverable D2.1.2.2v1. A brief example is followed by the original contribution, namely a performance evaluation covering a range of different ontologies.

3.1 The KAON2 Transformation Algorithms

Reasoning with KAON2 is based on special-purpose algorithms which have been designed for dealing with large ABoxes. They are detailed in [52] and we present a birds' eyes perspective here. The underlying rationale of the algorithms is that algorithms for deductive databases have proven to be efficient in dealing with large numbers of facts. The KAON2 approach utilises this by transforming OWL DL ontologies to disjunctive datalog, and by the subsequent application of the mentioned and established algorithms for dealing with disjunctive datalog.

A birds' eyes perspective on the KAON2 approach is depicted in Figure 3.1. KAON2 can handle $\mathcal{SHIQ}(\mathbf{D})$ description logic ontologies, which corresponds roughly to OWL DL without nominals. The TBox, together with a query are processed by the transformation algorithm which is detailed further below and which returns a disjunctive datalog program. This, together with an ABox, is then fed into a disjunctive datalog reasoner

[1] `http://kaon2.semanticweb.org`

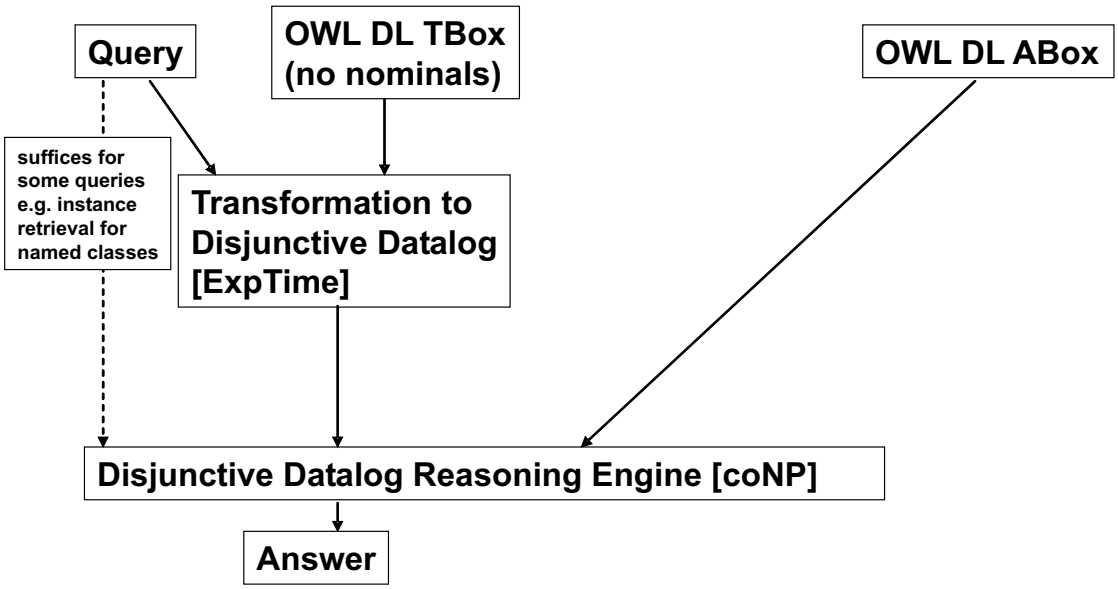


Figure 3.1: KAON2 approach to reasoning

which eventually returns an answer to the query. In some cases, e.g. when querying for instances of named classes, the query does not need to be fed into the transformation algorithm but instead needs to be taken into account only by the datalog reaoner.

The transformation algorithm is depicted in Figure 3.2. It accepts a $\mathcal{SHIQ}$ (or $\mathcal{SHIQ}(\mathbf{D})$) TBox and returns a disjunctive datalog program. Note that the returned program is in general not logically equivalent to the input TBox; the exact relationship is given below in Theorem 2.

The steps of the algorithm can roughly be described as follows. (1) Transitivity axioms for roles S are replaced by adding axioms of the form $\forall R.C \sqsubseteq \forall S.\forall S.C$ whenever $S \sqsubseteq R$. This is a standard method for eliminating transitivity axioms, and the resulting knowledge base is satisfiable if and only if the original knowledge base is. This ensures that the translation can be used to solve typical $\mathcal{SHIQ}$ reasoning problems by reducing them to unsatisfiability of a $\mathcal{SHIQ}$ knowledge base.

Employing the fact that $\mathcal{SHIQ}$ can be regarded as a subset of first-order logic, step (2) uses standard algorithms to transform the knowledge base into conjunctive normal form. This involves eliminating existential quantifiers by *Skolemization*, and thus function symbols must be introduced into the knowledge base.

Next, in step (3), the obtained set of clauses is partially *saturated* by adding logical consequences. This is the crucial step of the algorithm where one has to compute enough consequences to allow for a reduction to function-free Datalog. Since the computational complexity is EXPTIME for $\mathcal{SHIQ}$ but only NP for disjunctive Datalog, it should not come as a surprise that this transformation step can be exponential in the size of the input.

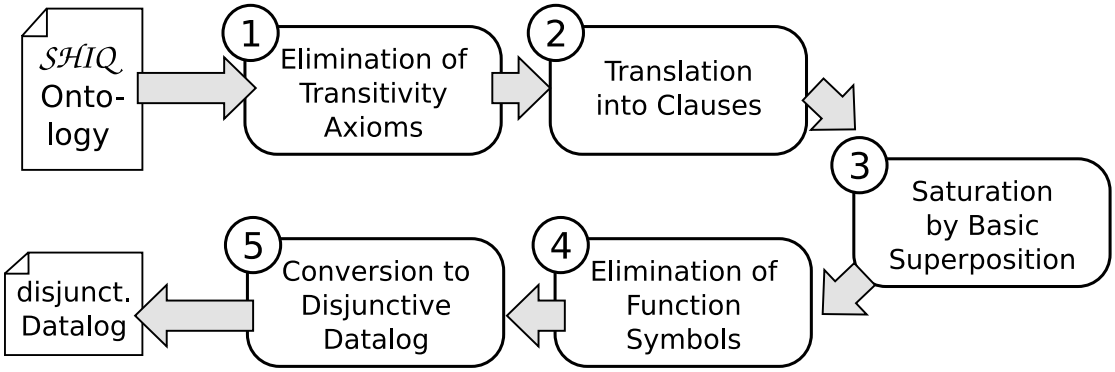


Figure 3.2: Algorithm for reducing $\mathcal{SHIQ}$ to disjunctive Datalog.

The details of this step are rather sophisticated and we refer to [52] for details and proofs.

Now function symbols can safely be eliminated in step (4). To ensure that this process still preserves satisfiability of the knowledge base, one has to add a linear number of auxiliary axioms. Finally, it is easy to syntactically transform the resulting set of clauses into a disjunctive Datalog program in step (5).

Due to the transformations in steps (1) and (2), the output of the algorithm is in general not logically equivalent to the input. Since all major reasoning tasks for $\mathcal{SHIQ}$ can be reduced to satisfiability checking, it is sufficient for the transformation to preserve satisfiability, as shown in the following theorem from [52].

Theorem 2 *Let K be a $\mathcal{SHIQ}(\mathbf{D})$ TBox and $D(K)$ be the datalog output of the KAON2 transformation algorithm on input K. Then the following claims hold.*

- *K is unsatisfiable if and only if $D(K)$ is unsatisfiable.*
- *$K \models \alpha$ if and only if $D(K) \models \alpha$, where α is of the form $A(a)$ or $R(a, b)$, for A a named concept and R a role.*
- *$K \models C(a)$ for a nonatomic concept C if and only if, for Q a new atomic concept, $D(K \cup \{C \sqsubseteq Q\}) \models Q(a)$.*

A performance evaluation of KAON2 is reported in [53]. It shows that KAON2 is indeed superior over other reasoners in most cases where size of the ABox dominates compared to the size of the TBox.

3.2 The Screech Approach

The approach which we report here is based on the fact that data complexity is polynomial for non-disjunctive datalog, and we utilise the KAON2 algorithms. But rather than doing

(expensive) exact reasoning over the resulting disjunctive datalog knowledge base, we do approximate reasoning by treating disjunctive rules as if they were non-disjunctive ones. The resulting reasoning procedure is complete, but may be unsound in cases. Its data complexity is polynomial.

More precisely, we use a modified notion of *split program* [62] in order to deal with the disjunctive datalog. Given a rule

$$H_1 \vee \cdots \vee H_m \leftarrow A_1, \ldots, A_k,$$

as an output of the KAON2 transformation algorithm, the *derived split rules* are defined as:

$$H_1 \leftarrow A_1, \ldots, A_k \qquad \ldots \qquad H_m \leftarrow A_1, \ldots, A_k.$$

For a given disjunctive program P its *split program* P' is defined as the collection of all split rules derived from rules in P. It can be easily shown that for instance retrieval tasks, the result obtained by using the split program instead of the original one is complete but may be unsound. This is even the case if all integrity constraints, i.e. rules of the form

$$\leftarrow B_1, \ldots, B_n$$

are removed from the split program.

In order to be able to deal with all of OWL DL including nominals, which are currently not supported by KAON2, we can also add a preprocessing step to get rid of nominals, i.e. we need to compile $\mathcal{SHOIN}(\mathbf{D})$ ontologies to $\mathcal{SHIQ}(\mathbf{D})$. We can do this by *Language Weakening* as follows: For every occurrence of $\{o_1, \ldots, o_n\}$, where $n \in \mathbb{N}$ and the o_i are abstract or concrete individuals, replace $\{o_1, \ldots, o_n\}$ by some new concept name D, and add ABox assertions $D(o_1), \ldots, D(o_n)$ to the knowledge base. Note that the transformation just given does in general not yield a logically equivalent knowledge base, so some information is lost in the process. Putting all the pieces together, we propose the following subsequent steps for approximate ABox reasoning for OWL DL.

1. Apply Language Weakening as just mentioned in order to obtain a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base.
2. Apply transformations as in Section 3.1 in order to obtain a negation-free disjunctive datalog program.
3. Obtain the split program as described above.
4. Do reasoning with the split program, e.g. using the KAON2 datalog reasoning engine.

The first two steps can be considered to be preprocessing steps for setting up the intensional part of the database. ABox reasoning is then done in the last step. The resulting approach has the following theoretical features:

- It is complete with respect to OWL DL semantics.
- Data complexity is polynomial.

The approach is spelled out in detail in [39].

The SCREECH System

A preliminary implementation of our approach is available as theSCREECH OWL approximate reasoner.[2] It is part of the KAON2 OWLtools.[3] KAON2[4] is the KArlsruhe ONtology framework, which includes a fast OWL reasoner based on the transformation algorithms mentioned in Section 3.1, as already mentioned, and also includes many other features helpful to work with ontologies. Among the KAON2 OWL tools, `deo` performs the language weakening step described in the previous Section in order to obtain a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base. We then convert an OWL ontology into a disjunctive datalog program, e.g. by using the `dlpconvert` KAON2 OWL tool with the `-x` switch. SCREECH then accesses the results of the translation through the KAON2 API, creates the corresponding split programs and serializes them as Horn logic programs in Edinburgh Prolog syntax. The result can be fed to any Prolog interpreter — or other logic programming engine —, which in turn can be used to perform ABox reasoning and inferencing over the knowledge base. For completeness, we need to mention that in general support for concrete domains and other features like integrity constraints is not necessarily implemented in off-the-shelf logic programming systems. In these cases, concrete domains etc. cannot be used. The KAON2 OWL tool `ded`,[3] for example, performs a language weakening step by removing all concrete domains, and may come in handy in such situations.

3.3 An Example

We demonstrate our approach by means of a simple OWL DL ontology. It contains only a class hierarchy and an ABox, and no roles, but this will suffice to display the main issues.

The ontology is shown in Figure 3.3, and its intended meaning is self-explanatory. Note that the fourth line,

$$\text{beneluxian} \equiv \text{luxembourgian} \sqcup \text{dutch} \sqcup \text{belgian},$$

[2] http://logic.aifb.uni-karlsruhe.de/screech
[3] http://www.aifb.uni-karlsruhe.de/WBS/dvr/owltools
[4] http://kaon2.semanticweb.org

$$\begin{aligned}
\text{serbian} \sqcup \text{croatian} &\sqsubseteq \text{european} \\
\text{eucitizen} &\sqsubseteq \text{european} \\
\text{german} \sqcup \text{french} \sqcup \text{beneluxian} &\sqsubseteq \text{eucitizen} \\
\text{beneluxian} &\equiv \text{luxembourgian} \sqcup \text{dutch} \sqcup \text{belgian}
\end{aligned}$$

serbian(ljiljana)	serbian(nenad)	german(pascal)	french(julien)
croatian(boris)	german(markus)	german(stephan)	croatian(denny)
indian(sudhir)	belgian(saartje)	german(rudi)	german(york)

Figure 3.3: Example ontology

translates into the four clauses

$$\begin{aligned}
\text{luxembourgian}(x) \vee \text{dutch}(x) \vee \text{belgian}(x) &\leftarrow \text{beneluxian}(x), \qquad (3.1)\\
\text{beneluxian}(x) &\leftarrow \text{luxembourgian}(x), \\
\text{beneluxian}(x) &\leftarrow \text{dutch}, \\
\text{and} \qquad \text{beneluxian}(x) &\leftarrow \text{belgian}(x).
\end{aligned}$$

Thus, our approach changes the ontology by treating the disjunctions in line (3.1) as conjunctions. This change affects the soundness of the reasoning procedure. However, most of the ABox consequences which can be derived by approximate SLD-resolution are still correct. Indeed, there are only two derivable facts which do not follow from the knowledge base by classical reasoning, namely

$$\text{dutch}(\text{saartje}) \qquad \text{and} \qquad \text{luxemburgian}(\text{saartje}).$$

All other derivable facts are correct.

SCREECH translates the ontology from Figure 3.3 into the Prolog program listed in Figure 3.4. As standard implementations of SLD-resolution do not use fair selection functions and also use depth-first search for higher efficiency, they may sometimes fail to produce answers because they run into infinite branches of the search tree. This occurs, for example, when using SWI-Prolog[5]. A reordering of the clauses may improve the results, but does not solve the problem entirely. More satisfactory performance can be obtained by using SLD-resolution with tabling, as implemented e.g. in the XSB Prolog system[6]. In this case, all desired consequences can be derived.

[5] http://www.swi-prolog.org/
[6] http://xsb.sourceforge.net

```
serbian(ljiljana).  serbian(nenad).    german(pascal).
french(julien).     croatian(boris).   german(markus).
german(stephan).    croatian(denny).   indian(sudhir).
belgian(saartje).   german(rudi).      german(york).

european(X) :- serbian(X).    european(X) :- croatian(X).
european(X) :- eucitizen(X).  eucitizen(X):- german(X).
eucitizen(X):- french(X).     eucitizen(X):- beneluxian(X).

beneluxian(X) :- luxembourgian(X).
beneluxian(X) :- dutch(X).
beneluxian(X) :- belgian(X).

        dutch(X):- beneluxian(X).
luxembourgian(X):- beneluxian(X).
      belgian(X):- beneluxian(X).
```

Figure 3.4: Example SCREECH output

3.4 Experiments and Evaluation

An approximate reasoning procedure needs to be evaluated on real data from practical applications. Handcrafted examples are of only limited use as the applicability of approximate methods depends on the structure inherent in the experimental data.

So we evaluated some popular publicly available ontologies. In some reasons we had to cautiously modify them in order to enable KAON to perform reasoning tasks on them. For all our experiments, we used a T40p IBM Thinkpad with 512MB of RAM, with the Java 2 Runtime Environment, Standard Edition (build 1.5.0_09-b03).

GALEN

For our first evaluation we have performed experiments with the OWL DL version of the GALEN Upper Ontology,[7] as it appears to be sufficiently natural and realistic. As it is a TBox ontology only, we populated GALEN's 175 classes randomly with 500 individuals.[8] GALEN does not contain nominals or concrete domains. GALEN has 673 axioms (the population added another 500). The TBox translation to disjunctive datalog took about 1400 ms, after which we obtained 1449 disjunctive datalog rules 54 of which contained disjunctions. After splitting the "proper" disjunctive rules, we arrived at 1554 Horn rules.

[7] http://www.cs.man.ac.uk/∼rector/ontologies/simple-top-bio/

[8] Using the `pop` KAON2 OWL tool.

Time (DD)	Time (SPLIT)	Instances	Class Name
10050 ms	4281 ms	154/154	Biological_object
8942 ms	5468 ms	9/9	Specified_set
8320 ms	5899 ms	9/13	Multiple
9454 ms	4997 ms	16/16	Probe_structural_part_of_heart
8838 ms	5978 ms	4/4	Human_red_blood_cell_mature
9183 ms	4301 ms	24/58	Biological_object_that...

Table 3.1: Performance comparison for instance retrieval using disjunctive datalog (DD) vs. the corresponding split program (SPLIT), on the KAON2 datalog engine. *Instances* indicates the number of instances retrieved using DD versus SPLIT, e.g. class *Multiple* contained 9 individuals, while the split program allowed to retrieve 13 (i.e. the 9 correct individuals plus 4 incorrect ones). The full name of the class in the last row is Biological_object_that_has_left_right_symmetry.

We then randomly selected classes and queried for their extension using the KAON2 datalog engine, both for processing the disjunctive datalog program and for the split program. Some of the typical results are listed in Table 3.1, which indicates a significant speed-up of about 37% on average, while the vast majority of the retrieved answers is correct. In a complete run we queried for the extensions of all 175 GALEN classes, resulting in a total number of 5652 classifications performed by SCREECH, of which 5187 (i.e. 91.8%) were correct. For 138 out of 175 classes the extension computed by SCREECH was correct. The average time saved when computing the extension was 38.0% over all 175 classes.

DOLCE

DOLCE (a Descriptive Ontology for Linguistic and Cognitive Engineering) is a foundational ontology, developed by the Laboratory for Applied Ontology in the Institute of Cognitive Sciences and Technology of the Italian National Research Council. In full, it exceeds the reasoning capabilities of current reasoners, hence we used a fraction for our experiments, namely the factorized OWL build 397 containing the modules DOLCE-Lite, ExtDnS, Modal, and Common with a total amount of 1302 axioms. Moreover we had to eliminate the transitivity axioms. Since DOLCE is a pure TBox-Ontology, we randomly populated it with 500 individuals to be able to carry out instance retrieval.

The conversion into disjunctive datalog yielded 1797 rules of which 47 are disjunctive. Splitting the program took about 460ms resulting in 1670 Horn rules.[9] Table 3.2 shows the results of some exemplary runs. From all retrieved instances, 62.10% were correct while the amount of saved time was 29,1%. Moreover, 93 of the 123 named class extensions

[9]The reduced number of rules compared with the disjunctive datalog program is due to the removed integrity constraints.

Time (DD)	Time (SPLIT)	Instances	Class Name
8390 ms	4813 ms	341/432	endurant
8672 ms	6046 ms	28/28	cognitive-modal-description
8640 ms	5460 ms	15/15	commitment
8953 ms	6656 ms	6/11	situation
8688 ms	6063 ms	41/168	physical-object

Table 3.2: Performance comparison for some named classes of the DOLCE ontology.

Time (DD)	Time (SPLIT)	Instances	Class Name
2353 ms	1022 ms	53/53	Wine
2143 ms	1372 ms	25/25	DryRedWine
2083 ms	1527 ms	18/24	WhiteNonSweetWine
2118 ms	1397 ms	23/23	CaliforniaWine
2038 ms	1337 ms	4/5	SauvignonBlanc
2233 ms	1077 ms	43/43	Winery

Table 3.3: Performance comparison for some named classes of the WINE ontology.

were retrieved correctly.

WINE

The next ontology we tested was the WINE ontology[10]. It is a well-known ontology containing a classification of wines. Moreover, it is one of the rare ontologies with both an ABox and a nontrivial TBox. However, it also contains nominals, which the KAON algorithm cannot deal with. The best way to nevertheless work with this data is to apply a sound but an incomplete approximation: one replaces each enumerated concept $\{i_1, ..., i_n\}$ with a new concept O, and adds assertions $O(i_k)$. The ontology obtained this way contains 218 axioms, including functionality, disjunctions, and existential quantifiers. The corresponding ABox contains 247 concept membership axioms and 246 role membership axioms.

The translation procedure into disjunctive datalog produced altogether 559 rules, among them 26 disjunctive ones. The conversion into a split program took about 530 ms and yielded a total amount of 575 Horn rules.

We carried out an instance retrieval for every named class. From 780 overall retrieved instances, 747 were correct, yielding a precision of 95,8%. Furthermore, 131 of the 140 computed class extensions were correct. The average time saved was 34,5%.

[10] `http://www.schemaweb.info/schema/SchemaDetails.aspx?id=62`

Time (DD)	Time (SPLIT)	Instances	Class Name
1172 ms	901 ms	5369/5369	Client
1231 ms	130 ms	659/659	Classic
1046 ms	160 ms	682/682	Loan
1262 ms	1417 ms	4500/4500	Individual
942 ms	702 ms	2645/2645	Woman

Table 3.4: Performance comparison for some named classes of the SEMINTEC ontology.

SEMINTEC

In a last investigation, we considered two ontologies, the translation of which turned out to not contain proper disjunctive rules. Nevertheless, removing integrity constraints is supposed to result in improving runtime behaviour (while in this case even preserving soundness).

So, the last ontology we considered was created in the SEMINTEC project[11] at the university of Poznan and is concerned with financial services. Its TBox contains 222 axioms of comparably simple structure, apart from some functionality constraints which require equality reasoning.

The TBox translation generated 221 rules (all being Horn), removing the integrity constraints left 105 of them.

Table 3.4 shows the acquired data for some named classes. Again, we retrieved the extensions of all 59 named classes of the ontology. As expected, all of the 25592 retrieved instances were correct. Moreover, we observed a drastic speedup of 67.3%, i.e. the reduced version was three times faster than the original disjunctive datalog program. Still, there were some cases, where the SPLIT program took significantly longer.

3.5 Conclusion

Concluding, we can state that in all considered cases, we could obtain a significant speed-up for the instance retrieval task, although the KAON2 datalog engine is not optimized for Horn programs, but rather tuned to efficient performance on definite disjunctive datalog.

Looking at our data, it seems to be not too illusory an assumption, that a considerable amount of ontologies is – at least nearly – Horn. In those cases, the application of the presented technique seems feasible.

However, we are aware that this quantitative approach is not in all cases suitable to access the practical usefulness of our approach. Thus further investigation and empirical

[11] `http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm`

long-term evaluation in the intended field of application will be needed to reliably estimate its applicability.

Chapter 4

Approximate Subsumption

by HEINER STUCKENSCHMIDT

Description Logics are becoming more and more popular as a formalism for representing and reasoning about conceptual knowledge in different areas such as databases and semantic web technologies. In particular, subsumption reasoning for expressive ontologies has been used to compute matches between conceptual description in the context of different real world tasks including information integration [69], product and service matching [48] and data retrieval [7]. In practical situations, however, it often turns out that logical reasoning is inadequate in many cases, because it does not leave any room for *partial matches*.

Recently, there are some efforts that try to address this problem by combining description logics with numerical techniques for uncertain reasoning, in particular with techniques for probabilistic [32] and fuzzy reasoning [67]. These approaches are able to compute partial matches by assigning an assessment of the degree of matching to the subsumption relation. This degree of matching normally is a real number between zero and one and therefore allows an absolute ordering of the solutions Although, in principle this is a solution to the problem of computing the best partial match but defining an interpreting numerical assessments of uncertainty is a difficult problem. Further, the reduction to a single numerical assessment of the mismatch does not allow different users to discriminate between different kinds of mismatches.

In this paper, we propose a notion of approximate subsumption that supports the computation of partial matches between complex concept expressions without relying on a single number to represent the degree of mismatch. Instead, *the approach describes the degree of matching in terms of a subset of the aspects of the request that are met by the solution*. This approach allows the user to decide whether to accept a partial match based on whether important aspects are missed or not. In order to implement this approach we borrow from the area of approximate deduction. In particular, we extend the notion of S-Interpretations of propositional logic proposed in [63] to description logics and use the result notion of a non-standard interpretation of OWL concept expressions to define an approximate subsumption operator that computes subsumption with respect to a particular

subset of the vocabulary used.

Our approach is similar to the notion of approximate entailment for description logics proposed in [16]. Our work extends this work different ways:

- Previous work was restricted to rather inexpressive description logics, in particular $\mathcal{ALC}$. We extend this to expressive description logics. In particular, our approach includes qualified number restrictions.
- We provide a more natural way of approximating concept descriptions based on the set of atomic concepts.
- While the work of Cadoli and Schaerf relied on a rather complicated formalization in terms of the Herbrand Universe of the first-order translation of description logics, we provide a direct model theoretic semantics for approximate subsumption and show that is has the required properties[1]

The paper is organized as follows. Section 2 summarizes the notion of an S-Interpretation and explains the idea of extending the notion to concept expressions. Section 3 recalls some basic notions of descriptions logics. In section 4, we formally define a non-standard semantics for concept expressions and show that it can be used as upper and lower approximation of a concept. The approximate subsumption operator is defined in section 5 and an algorithm for deciding approximate subsumption is given. In section 5, we give an example of the use of approximate subsumption for computing partial matches. We summarize with a discussion of the approach and future work.

4.1 Approximation based on Sub-Vocabularies

In propositional logic, the vocabulary of a formula consists of a set of propositional letters. A classical interpretation I assigns to each letter either the value *true* or *false* such that for all propositional letters p one of the following holds:

$$\begin{aligned} I(p \wedge \neg p) &= false \\ I(p \vee \neg p) &= true \end{aligned} \tag{4.1}$$

Checking satisfiability of a formula relies on showing that there is no assignment of truth values that satisfies this condition and makes the whole formula true. A possible way for approximating satisfiability testing for propositional logic is now to restrict the condition above to a subset of the propositional letters. This subset is denoted as S and the

[1] Proofs for the theorems are omitted due to lack of space. A technical report including full proofs for all theorems is available on request.

corresponding interpretation is called an S-interpretation of the formula [63]. Depending on how the letters not in S are treated, an S-Interpretation is sound or complete with respect to the classical interpretation. One kind of non-standard interpretation called S-3 Interpretation assigns both, a letter and its negation to *true*.

$$I(p \wedge \neg p) = true, p \notin S \tag{4.2}$$

The resulting calculus is sound, but incomplete. The counterpart of S-3 interpretation are S-1 Interpretations that assign *false* to both a letters and their negation if the letters are not in the set S.

$$I(p \vee \neg p) = false, p \notin S \tag{4.3}$$

S-1 Interpretations define a complete but unsound calculus for propositional logic. In both cases, the advantage of the approach is that we can decide which parts of the problem to approximate by selecting an appropriate set of letters S. Therefore the approach provides a potential solution to the problem of partial matching described above.

The idea of our approach is now to apply the underlying idea of S-Interpretations to Description Logics. In fact, Cadoli and Schaerf do propose an extension of S-Interpretations to Description logics, but they define S not in terms of a subset of the vocabulary, but in terms of the structure of the concept expression [16]. In [34] it has been shown that this way of applying S-Interpretations to description logic ontologies does not produce satisfying results on real data. We therefore propose an alternative way of defining S-Interpretations for description logics which is closer to the notion of S-Interpretations in propositional logic. The idea is to interpret description logics as an extension of propositional logic, where class names correspond to propositional letters[2]. As for propositional logic, we select a subset of the class names that is interpreted in the classical way and approximate class names not in this set. In particular, a classical interpretation ($\Delta^{\mathcal{I}}$, $\mathcal{I}$) of class names requires that a concept name and its negation form a disjoint partition of the domain:

$$\begin{aligned} C^{\mathcal{I}} \cap (\neg C)^{\mathcal{I}} &= \emptyset \\ C^{\mathcal{I}} \cup (\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \end{aligned} \tag{4.4}$$

We can now define approximations for description logics by relaxing these requirements for a subset of the concept names. The corresponding S-3 and S-1 Interpretations are very similar to the ones for propositional logic. For S-3 Interpretations we have:

$$C^{\mathcal{I}} \cap (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}}, C \notin S \tag{4.5}$$

[2] In fact, a description logic that just contains the Boolean operators is equivalent to propositional logic.

As a consequence, the concept name C cannot cause a clash in a tableaux proof and therefore, constraints that force a certain value to be of type C will be ignored in a subsumption proof. The resulting subsumption operator is sound, but incomplete. For S-1 Interpretations, we have

$$C^{\mathcal{I}} \cup (\neg C)^{\mathcal{I}} = \emptyset, C \notin S \tag{4.6}$$

which means that both C and $\neg C$ have to be mapped to the empty set. In a tableaux proof, all attempts to construct a model that involves a variable of type C will fail. The corresponding subsumption operator is complete, but unsound with respect to classical subsumption.

4.2 Preliminaries

We consider concept expressions in the $\mathcal{SHIQ}$ description logic. Concept expressions are defined recursively as follows: Let $\mathcal{V}$ be a set of concept names and R a set of role names. Further let there be a set $R^+ \subseteq R$ of transitive roles (i.e. for each $r \in R^+$ we have $r(x,y) \wedge r(y,z) \Rightarrow r(x,z)$). If now R^- denotes the inverse of a role (i.e. $r(x,y) \Rightarrow r^-(y,x)$) then we define the set of roles as $R \cup \{r^-|r \in R\}$. Every atomic concept name A is a concept expression. If C and D are concept expressions, and r is a binary relation then $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists\, r.C$, $\forall\, r.C$, $\leq\, n\, r.C$ and $\geq\, n\, r.C$ are also concept expressions.

The semantics of $\mathcal{SHIQ}$ is defined in terms of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$ where $.^{\mathcal{I}}$ is a function that maps every concept on a subset of $\Delta^{\mathcal{I}}$ and every role on a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that for all concepts C and D and for roles r where #M denotes the cardinality of M and $(r^{\mathcal{I}})^+$ the transitive closure of $r^{\mathcal{I}}$ we have:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$
- $r^{\mathcal{I}} = (r^{\mathcal{I}})^+$ for $r \in R^+$ and $r^- = \{(y,x)|(x,y) \in r^{\mathcal{I}}\}$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - C^{\mathcal{I}}$, $(C \sqcap D) = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ and $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\forall r.C)^{\mathcal{I}} = \{x|\forall y.(x,y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
- $(\exists r.C)^{\mathcal{I}} = \{x|\exists y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
- $(\geq n\, r.C)^{\mathcal{I}} = \{x|\#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$
- $(\leq n\, r.C)^{\mathcal{I}} = \{x|\#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$

An interpretation satisfies a terminology $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all general concept inclusions $C \sqsubseteq D$ in $\mathcal{T}$ and $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ for all role inclusion axioms $r \sqsubseteq s$ in $\mathcal{T}$. In this case we call $\mathcal{I}$ a model for $\mathcal{T}$. A concept D subsumes a concept C in $\mathcal{T}$ if $C \sqsubseteq D$ holds for all models of $\mathcal{T}$.

4.3 A Non-Standard Semantics

A limited vocabulary is a set $S \subseteq \mathcal{V}$. Our aim is to define approximate reasoning in Description Logics based on such a subset of the vocabulary. For this purpose, we propose a non-standard interpretation concept expressions. Such an interpretation is defined as a pair of interpretations $\mathcal{I}_S = (\mathcal{I}_S^+, \mathcal{I}_S^-)$ We call $\mathcal{I}_S^+$ - the counterpart of an S-3 interpretation - an upper approximation and $\mathcal{I}_S^-$ - the counterpart of an S1 interpretation - a lower approximation with respect to S.

Definition 7 (Lower Approximation) *A lower approximation of a concept expression in negation normal form is a non standard interpretation $(\Delta^{\mathcal{I}}, \mathcal{I}_S^-)$ such that:*

$$A^{\mathcal{I}_S^-} = \begin{cases} A^{\mathcal{I}} & A \in S \\ \emptyset & otherwise \end{cases} \tag{4.7}$$

$$(\neg C)^{\mathcal{I}_S^-} = \Delta^{\mathcal{I}} - C^{\mathcal{I}_S^+} \tag{4.8}$$

$$(C \sqcap D)^{\mathcal{I}_S^-} = C^{\mathcal{I}_S^-} \cap D^{\mathcal{I}_S^-} \tag{4.9}$$

$$(C \sqcup D)^{\mathcal{I}_S^-} = C^{\mathcal{I}_S^-} \cup D^{\mathcal{I}_S^-} \tag{4.10}$$

$$(\geq n\, r.C)^{\mathcal{I}_S^-} = \{x | \#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^-}\} \geq n\}$$

$$(\leq n\, r.C)^{\mathcal{I}_S^-} = \{x | \#\{y | (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^+}\} \leq n\}$$

where $(\Delta^{\mathcal{I}}, \mathcal{I}_S^+)$ is an upper approximation as defined in definition 8

Definition 8 (Upper Approximation) *An upper approximation of a concept expression in negation normal form is a non standard interpretation $(\Delta^{\mathcal{I}}, \mathcal{I}_S^+)$ such that:*

$$A^{\mathcal{I}_S^+} = \begin{cases} A^{\mathcal{I}} & A \in S \\ \Delta^{\mathcal{I}} & otherwise \end{cases} \tag{4.11}$$

$$(\neg C)^{\mathcal{I}_S^+} = \Delta^{\mathcal{I}} - C^{\mathcal{I}_S^-} \tag{4.12}$$

$$(C \sqcap D)^{\mathcal{I}_S^+} = C^{\mathcal{I}_S^+} \cap D^{\mathcal{I}_S^+} \tag{4.13}$$

$$(C \sqcup D)^{\mathcal{I}_S^+} = C^{\mathcal{I}_S^+} \cup D^{\mathcal{I}_S^+} \tag{4.14}$$

$$(\geq n\, r.C)^{\mathcal{I}_S^+} = \{x | \#\{y.(x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^+}\} \geq n\}$$

$$(\leq n\, r.C)^{\mathcal{I}_S^+} = \{x | \#\{y | (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}_S^-}\} \leq n\}$$

where $(\Delta^{\mathcal{I}}, \mathcal{I}_S^-)$ is a lower approximation as defined in definition 7

Note that existential and universal quantifiers can be translated into the following qualified number restrictions: $(\forall r.C) \equiv (\leq 0 r.\neg C)$, $(\exists r.C) \equiv (\geq 1 r.C)$. In the following, we will show that these interpretations meet our requirements for approximate reasoning about expressive description logics both technically and intuitively.

4.3.1 Negation Normal Form

As mentioned above, negation normal forms play an important role withe respect to tableaux-based algorithms for description logics. The ability to compute the negation normal form as a basis for a tableaux proof relies on the equivalent transformation rules. We can show that the re-writing rules used for computing the negation normal form of arbitrary concept expressions also hold under the non-standard interpretations. This implies that we can translate every concept expression into its negation normal form without changing the non-standard interpretation. This result is formalized in the following theorem.

Theorem 3 (Negation Normal Form) *For every concept expression C there is an expression $nnf(C)$ in negation normal form (negation only applies to concept names) such that $C^{\mathcal{I}_S^-} = nnf(C)^{\mathcal{I}_S^-}$ and $C^{\mathcal{I}_S^+} = nnf(C)^{\mathcal{I}_S^+}$.*

Based on the negation normal form, we can define a simplified version of the semantics with respect to negation. Instead of the general definition of negation, we can use a special rule for negation with respect to negation of atomic concept names. In particular, for a concept expression in negation normal form equation 4.8 can be replaced by the following equation:

$$(\neg A)^{\mathcal{I}_S^-} = \begin{cases} A^{\mathcal{I}} & A \in S \\ \emptyset & otherwise \end{cases} \tag{4.15}$$

Analogously, equation 4.12 can be replaced by the following equation:

$$(\neg A)^{\mathcal{I}_S^+} = \begin{cases} A^{\mathcal{I}} & A \in S \\ \Delta^{\mathcal{I}} & otherwise \end{cases} \tag{4.16}$$

These definitions bring us back to equation 4.4 in the motivation. The idea of our work was to relax the equivalences in this equation for concept names not in the set S. Applying the simplified definitions above leads us directly to equations 4.5 and 4.6

$$C^{\mathcal{I}_S^+} \cap (\neg C)^{\mathcal{I}_S^+} = \Delta^{\mathcal{I}} \cap \Delta^{\mathcal{I}} = \Delta^{\mathcal{I}}, \text{for } C \notin S \tag{4.17}$$

$$C^{\mathcal{I}_S^-} \cup (\neg C)^{\mathcal{I}_S^+} = \emptyset \cup \emptyset = \emptyset, \text{for } C \notin S \tag{4.18}$$

This means that the non-standard interpretation presented in this section is actual a generalization of the notion of vocabulary-based approximation for propositional logic.

4.3.2 Formal Properties

This non-standard interpretation is defined in such a way, that it makes concept expressions strictly more general for upper and strictly more specific for lower approximations.

This property which we call monotonicity is important in order to be able to guarantee formal properties of approximation methods defined based on this interpretation. Therefore the following theorem describes a central property of approximation in description logics.

Theorem 4 (Monotonicity) *Given a non-standard interpretation as defined above, the following equation holds for all concept expressions C:*

$$C^{\mathcal{I}_S^-} \subseteq C^{\mathcal{I}} \subseteq C^{\mathcal{I}_S^+} \tag{4.19}$$

We can generalize the theorem by observing that the standard interpretation is an extreme case of the non-standard interpretation with $S = \mathcal{V}$. In particular, the general version of monotonicity says that for upper approximations removing concept names from the set S will make concepts expressions strictly more general. Conversely, for lower approximations concept expressions become less general when we remove concepts from the set S. The corresponding general property is defined in the following theorem:

Theorem 5 (Generalized Monotonicity) *Given a non-standard interpretation as defined above and two sub-vocabularies S_1 and S_2 with $S_1 \subseteq S_2$, the following equations hold for all concept expressions C:*

$$C^{\mathcal{I}_{S_1}^-} \supseteq C^{\mathcal{I}_{S_2}^-} \qquad C^{\mathcal{I}_{S_1}^+} \subseteq C^{\mathcal{I}_{S_2}^+} \tag{4.20}$$

The generalized monotonicity property is interesting, because it allows us to successively compute more precise upper and lower approximations of a concept by addition concept names to the set S. This is convenient in cases where users provide a preference order over concept names indicating the relative importance of different aspects of a concept. In this case, use the preference relation provided by the user to determine a sequence of approximations to be used in the matching process.

4.4 An Approximate Subsumption Operator

Up to now, we have only considered interpretations as such. As our aim is to develop approximate notions of subsumption as a basis for approximate matching, we now have to define the notion of approximate subsumption based on the non-standard interpretation defined above. It turns out, that this can be done in a straightforward way using the standard definition of the subsumption operator as:

$$\mathcal{I} \models C \sqsubseteq D \Leftrightarrow (C \sqcap \neg D)^{\mathcal{I}} = \emptyset$$

the idea is now to use this definition and replace the standard interpretation $\mathcal{I}$ by a the lower approximation $\mathcal{I}_S^-$ with respect to a certain sub-vocabulary S. Based on the choice of S, this defines different subsumption operators with certain formal properties that will be discussed in the following.

Definition 9 (Approximate Subsumption) *Let $S \subseteq \mathcal{V}$ be a subset of the concept names and $(\Delta^{\mathcal{I}}, \mathcal{I}_S^-)$ a lower approximation, then the corresponding approximate subsumption relation $\underset{S}{\sqsubseteq}$ is defined as follows*

$$\mathcal{I} \models (C \underset{S}{\sqsubseteq} D) \;\Leftrightarrow\; (C \sqcap \neg D)^{\mathcal{I}_S^-} = \emptyset \tag{4.21}$$

We say that C is subsumed by D with respect to sub-vocabulary S.

The monotonicity of the non-standard interpretation has an impact on the formal properties of the approximate subsumption operator. In particular, we can establish a relation between the subset of the vocabulary considered and the strength of the subsumption operator. The more concepts we exclude from the set S the weaker the subsumption operator as well as the matches we can compute get. This implies that if we can prove subsumption with respect to a particular set S the subsumption relation also holds for with respect to all subsets of S. Conversely, if we fail to prove subsumption with respect to a set S, we can be sure that the subsumption relation does also not hold with respect to any superset of S. These properties are stated formally in the following theorem.

Theorem 6 (Properties of Approximate Subsumption) *Let f be a lower approximation, then the following equation holds:*

$$\left(C \underset{S_2}{\sqsubseteq} D\right) \Rightarrow \left(C \underset{S_1}{\sqsubseteq} D\right) \text{ for } S_1 \subseteq S_2 \tag{4.22}$$

$$\left(C \underset{S_1}{\not\sqsubseteq} D\right) \Rightarrow \left(C \underset{S_2}{\not\sqsubseteq} D\right) \text{ for } S_1 \subseteq S_2 \tag{4.23}$$

These properties allow us to develop approximation strategies by successively selecting smaller subsets of concepts to be considered for matching and trying to compute the corresponding subsumption relation until we succeed.

4.4.1 Computing Approximate Subsumption

A nice feature of our approach is that it can actually be implemented using existing description logic reasoners and simply performing syntactic modifications on the input concept. In particular, in oder to check whether a statement $C \underset{S}{\sqsubseteq} D$ holds, we take the expression $(C \sqcap \neg D)$, transform it into negation normal form and replace all occurrences of all

Algorithm 1 approximate

Require: A subsumption statement $C \sqsubseteq D$
Require: A set S of concept names
  MODE := -1
  TEST := $(C \sqsubseteq \neg D)$
  Transform TEST into negation normal form.
  APPROX := rewrite(TEST, S, MODE)
  if APPROX is satisfiable **then**
    RESULT := false
  else
    RESULT := true
  end if
  return RESULT

negated and non-negated atomic concepts that are not in S by either $\bot$ or $\top$, depending on whether an upper or a lower approximation is used on the corresponding subexpression. This process is described more formally in algorithms 1 and 2.

The correctness of the re-writing approach is established by theorem 7.

Theorem 7 (Correctness of approximation) *Let C and D be concept expressions as defined above, then the following equations hold for all C, D:*

$$approximate(C \sqsubseteq D, S) = true \Leftrightarrow \mathcal{I} \models C \underset{S}{\sqsubseteq} D$$
$$approximate(C \sqsubseteq D, S) = false \Leftrightarrow \mathcal{I} \models C \underset{S}{\not\sqsubseteq} D$$

4.4.2 Maximal Subsumption Vocabularies

The notion of approximate subsumption actually describes a whole family of subsumption operators, one for each subset of the vocabulary. This fact the use of approximate subsumption complicated, because we often do not know in advance what a good choice for S is in a given situation. Identifying those S for which $C \underset{S}{\sqsubseteq} D$ holds is a complex problem in itself. In particular, all possible sets S form a combinatorial search space. Searching this space is equivalent to enumerating all subsets of the vocabulary. While in many cases the vocabulary with be rather small, the problem is inherently intractable in the general case. In order to be able to efficiently find the sets we are interested in, we have to find ways to prune the search space. The fact that subsumption between two concepts with respect to a certain sub-vocabulary implies subsumption between the same concepts with respect to all of the subsets of S help us to guide the search and prune large parts of the search space. In particular, it means that we are mostly interested in the maximal sets S for which $C \underset{S}{\sqsubseteq} D$ holds.

Algorithm 2 rewrite

Require: A concept expression C in negation normal form
Require: A set S of concept names
Require: A value $mode \in \{-1, 1\}$

if $C = (\leq\ n\, r.D)$ **then**
  C := $(\geq\ n\, r.\text{rewrite}(D, S, -mode))$
else if $C = (\geq\ n\, r.D)$ **then**
  C := $(\geq\ n\, r.\text{rewrite}(D, S, mode))$
else if $C = (D \sqcup E)$ **then**
  C := $(\text{rewrite}(D, S, mode) \sqcup \text{rewrite}(E, S, mode))$
else if $C = (D \sqcap E)$ **then**
  C := $(\text{rewrite}(D, S, mode) \sqcap \text{rewrite}(E, S, mode))$
else if $C = (\neg D) \vee C = D$ **then**
  if $D \notin S$ **then**
    if $mode = 1$ **then**
      $C := \top$
    else
      $C := \bot$
    end if
  end if
end if
return C

Computing the maximal sets S for which $C \underset{S}{\sqsubseteq} D$ holds can be done by successively enumerating subsets of $\mathcal{V}$ of decreasing size. Starting with subsets of size $|\mathcal{V}|$ we determine those sets S for which $C \underset{S}{\sqsubseteq} D$ and add them to the result. Before actually trying to prove the subsumption, we check whether the current set is a subset of any element in the result set already. After this has been done for all subsets of a certain size we recursively call the procedure for subsets that have one element less.

Besides the actual subsumption test, the critical part of this algorithm is the generation of the subsets S and the test whether S is a subset of one of the previous results. There are efficient ways of implementing the corresponding operations based on bit-vector encodings of subsets that can be used to solve this problem.

4.5 An Example

We illustrate this problem using a small example taken from chapter 4 of [69]. The problem addressed is the integration of different land-use classification schemes (ATKIS and CORINE) for supporting the automatic update of official registry records with satellite image data. As an example, we take the land-use class 'Mixed-Forest' from the ATKIS

catalogue which is defined as a region that has a vegetation composed of coniferous and broad-leaved plants that are all trees or shrubs. The corresponding concept expression is the following[3].

$$\begin{aligned} Mixed-Forest \equiv\ & Region \sqcap \\ & \geq 1\ vegetation.Magnoliophyta \sqcap \\ & \geq 1\ vegetation.Coniferophyta \sqcap \\ & \leq 0\ vegetation.\neg(trees \sqcup shrubs) \end{aligned}$$

When matching this description with the CORINE classification using standard subsumption, the most specific CORINE concept that subsumes Mixed Forest is the concept Vegetation Area which is defined by the presence of some vegetation:

$$Vegetation \equiv Region \sqcap (\geq 1\ vegetation.\top)$$

This result is somewhat disappointing as there are more specific classes in CORINE that we would have expected to also match. In particular, there is a concept 'Forest' that would qualify as the correct solution to the integration problem from a commonsense point of view. Looking at the definition of the Forest Concept in CORINE reveals that the problem is caused by the fact, that the definition does not mention shrubs as a possible form of vegetation of forest areas

$$Forests \equiv Vegetation \sqcap (\leq 0\ vegetation.(\neg trees))$$

.Using the algorithm given above, we can show that $Mixed-Forest \underset{\mathcal{V}-\{shrubs\}}{\sqsubseteq} Forests$ holds. This also shows the main advantage of our approach over existing proposals for partial matching: the result of our method clearly states what aspects of the two concepts did not match; in or case the presence of shrubs. The importance of this feature becomes clear when we look at the other concepts in the CORINE classification that are candidates for a match with the concept of mixed forest. In particular, there is a concept that specifies herbaceous and shrub areas. This concept is defined as follows:

$$\begin{aligned} Herbaceous \equiv\ & Vegetation \sqcap \\ & (\leq 0\ vegetation.\neg(shrubs \sqcup herbs)) \end{aligned} \tag{4.24}$$

If we apply our partial matching approach with respect to this concept expression, we can find out that $Mixed-Forest \underset{\mathcal{V}-\{trees\}}{\sqsubseteq} Herbaceous$ In order to determine the best match, we can ask the user which concept she considers to be more important with respect to determining a match for mixed forest. In this case most users will decide that the concept tree is more significant with respect to matching mixed forest than shrub and therefore should not be excluded from S and therefore lead the system to prefer the 'natural' match between mixed forest and forest.

[3]We already transformed existential and universal quantifiers into qualified number restrictions

4.6 Related Work

Previous work on approximate reasoning based on subsets of the vocabulary in different kinds of logic is reported in [16] and [63] as well as [17, 18]. Approaches for approximating concept expressions by weakening the logical language rather than the vocabulary are reported in [15, 46]. An approach for partial matching in description logics that is more similar to ours is reported in [21]. In [20] the approach is generalized to abductive matchmaking in description logics.

4.7 Discussion

We presented an approach for computing approximate subsumption between concept expressions in $\mathcal{SHIQ}$ based on a subset of the vocabulary used in the expressions. The approach solves some of the problems of classical reasoning in description logics, in particular, the inability to accept imperfect matches between concepts without having to leave the realms of formal logic. As a side-effect, the subset of the vocabulary also provides us with a qualitative characterization of the mismatch between the expressions, which is clearly an advantage over numerical approaches for dealing with imperfect matches.

A problem of the approach is its complexity. As we have seen in section 5, computing the maximal subsumption vocabulary in the worst case requires up to 2^n subsumption tests, where n is the number of elements in the vocabulary. In many practical cases, we will be able to prune large parts of the search space, but the worst time complexity remains extremely high. Another observation is that with the set S getting smaller, the actual subsumption tests become cheaper, because existing optimization techniques can be applied to simplify the rewritten concept expression. Another promising idea is to try to reuse parts the proof for the original subsumption problem for the approximated ones. In particular, instead of building up a new tableaux each time, we can examine the leaves of the existing tableaux and check whether the current subset would cause a clash in the corresponding branch. Such an analysis could also be used to more efficiently search for the maximal subsumption vocabulary by collecting the open branches and deriving the subset of the vocabulary that needs to be set to $\bot$ in order to close all branches.

Chapter 5

Applying Approximate Reasoning for Multiple Perspectives

by HEINER STUCKENSCHMIDT

Originally, ontologies where meant as a task-neutral description of a certain domain of interest that can be reused for different purposes. This idea is also at the heart of the semantic web vision, where the ontology-based description of Information is supposed to make it possible to use the information for different purposes and in different contexts. In practice, however, it has turned out that the re-use of ontologies for different tasks and purposes causes problems [73]. The reason for this is that ontologies are often not really designed independent of the task at hand. The development is rather driven by the special needs of a particular system or task. In general the context of use has an impact on the way concepts are defined to support certain functionalities. As some aspects of a domain that are important for one application do not matter for another one and vice versa, an ontology does not represent the features needed for a particular application. In this case, there is little hope for direct reuse. Another potential problem, that we will address in this paper is that an ontology contains too many aspects of a domain. This can become a problem, because in introduces unnecessary complexity and can even lead to unwanted conclusions, because the ontology introduces unwanted distinctions between classes that should be treated in the same way in the current application context. We argue that in order to solve this problem, we have to find ways to enable the representation of different viewpoints on the same ontology, that better reflects the actual needs of the application at hand.

Related Work The concept if having different viewpoints on the same model is a well established concept in the area of software engineering [26]. In order to extend this to semantic web systems, the concept of a viewpoint has to be extended to the semantic models used in the system. There has been some work on specifying viewpoints on RDF data, mainly inspired by the concept of views in database systems. The idea is to define rules for extracting an possibly restructuring parts of a basic RDF model to better reflect the

needs of a certain user or application. Different techniques have been proposed including the use of named queries [49], the definition of a view in terms of graph traversal operations [55] and the use of integrity constraints for ensuring the consistency of a viewpoint [74]. In this paper, we focus on ontologies represented in description logics, in particular OWL-DL. In the context of description logics, the classical notion of views can only be used in a restricted way as relevant inference problems related to views have been shown to be undecidable [8].

An alternative approach to viewpoints in description logics has been proposed based on the concept of contextual reasoning . Here, each viewpoint is represented in terms of a separate model with a local interpretation [33]. Relations between different viewpoints are represented by context mappings that constrain the local interpretations. Based on these basic principles of contextual reasoning, approaches for representing and linking different viewpoints on the same domain have been developed for description logics [10] and for OWL resulting the C-OWL language [12]. These approaches, however have a slightly different goal as they mainly aim at providing means for integrating different existing models. Our interest is to develop methods that allows us to extract a certain viewpoint from an existing model that best fits the requirements of an application.

An approach that is very similar to this idea is the work of Arara and others [60, 1]. They propose the use of modal description logics for encoding multiple viewpoints in the same ontology by indexing parts of the definitions with the contexts they are supposed to hold in. A drawback of their approach is that they require an extension of the representation language and its semantics to deal with multiple perspectives. In contrast to the contextual approaches mentioned above there currently is no reasoning support for this formalism.

Contributions and Motivating Example In this paper, we propose an approach for multi-viewpoint reasoning that do not require an extension to the OWL-DL language. The approach is based on the idea of approximate logical reasoning and uses an approximate subsumption operator that can be tuned to only use a certain part of the definitions in the ontology. In particular, we address the problem of efficient computing concept hierarchies that represent a certain viewpoint on a domain in terms of ignoring a certain subset of the vocabulary used in concept expressions.

To clarify this idea we consider the family ontology shown in figure 5.1. The ontology classifies persons into different concepts according to certain criteria including gender and the presence of children.

The silent assumption underlying this ontology is that all of the criteria used in the definitions are actually relevant for the application. In particular, the assumption is that it is important to distinguish between male and female persons (man vs. woman) and between people with and without children (woman vs. mother).

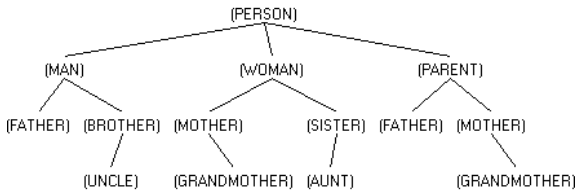


Figure 5.1: The Example Ontology

We can imagine applications that would benefit from an ontology of people, but in which only some of the distinguishing aspects are important. An example would be a system for processing salary information in the German public sector. In such a system it makes sense to distinguish between people with and without children, because the existence of children entitles to special benefits. The distinction between genders in completely irrelevant in this context and even prohibited by laws guaranteeing gender equality. Other applications e.g. related to private pension funds the gender is relevant as there are different regulations with respect to the age in which male and female persons can retire. In this application the existence of children is not important.

The paper is structured as follows. In section 2 first briefly introduce Description Logics as a basis for representing ontologies including some modeling examples from our example ontology and review the notion of approximate deduction in description logics proposed by Cadoli and Schaerf [63]. Section 3 introduces our notion of a viewpoint and its definition in terms of an approximate subsumption operator. In section 4 we discuss some axiomatic properties of the approximate subsumption operators and discuss their use for implementing basic reasoning services relevant for multi-viewpoint reasoning. The paper concludes with a discussion of the approach.

5.1 The Description Logics $\mathcal{SIN}$

The basic modeling elements in Description Logics are concepts (classes of objects), roles (binary relations between objects) and individuals (named objects). Based on these modeling elements, Description Logics contain operators for specifying so-called concept expressions that can be used to specify necessary and sufficient conditions for membership in the concept they describe. Basic reasoning tasks associated with these kinds of logics are checking whether an expression is satisfiable (whether it is possible that an object satisfies the membership condition) and deciding subsumption between two concepts (deciding whether a concept expression implies another one). We now look at these issues on a more formal level.

Let $\mathcal{C}$ be a set of concept names and R a set of role names. Further let there be a set $R^+ \subseteq R$ of transitive roles (i.e. for each $r \in R^+$ we have $r(x,y) \wedge r(y,z) \Rightarrow r(x,z)$). If

now R^- denotes the inverse of a role (i.e. $r(x,y) \Rightarrow r^-(y,x)$) then we define the set of roles as $R \cup \{r^- | r \in R\}$. A role is called a simple role if it is not transitive. The set of concepts (or concept expressions) in $\mathcal{SIN}$ is the smallest set such that:

- $\top$ and $\bot$ are concept expressions
- every concept name A is a concept expression
- if C and D are concept expressions, r is a role, s is a simple role and n is a non-negative integer, then $\neg C$, $C \sqcap D$, $C \sqcup D$,$\forall r.C$, $\exists r.C$, $\geq n\, r$ and $\leq n\, r$ are concept expressions.

A general concept inclusion axioms is an expression $C \sqsubseteq D$ where C and D are concepts, the equivalence axiom $C \equiv D$ is defined as $C \sqsubseteq D \wedge D \sqsubseteq C$ A terminology is a set of general concept inclusion and role inclusion axioms. In the following, we only consider axioms of the form $A \sqsubseteq C$ and $A \equiv C$ where A is an atomic concept name. Further, we assume that all concepts C are in negation normal form (negation only applies to atomic concept names). Note that for every concept can deterministically be transformed into an equivalent concept in negation normal form. Thus this assumption does not impose any restriction on the approach.

This logic covers a significant part of the OWL-DL Language. We exclude the following language elements, because their behavior in connection with the approximation approach presented below needs more investigation:

- Role Hierarchies: It is not clear how to deal with the situation where we want to consider a certain role but not its super-roles.
- Qualified Number restrictions: The use of qualified number restrictions make it hard to predict the effect of restricting reasoning to a sub-vocabulary, because ignoring the type restriction to C in the expression $(\geq n\, r.C)$ makes the overall expression more general whereas ignoring C in $(\leq n\, r.C)$ makes the expression more specific.
- Nominals: The current mechanism for implementing multi-viewpoint reasoning is based on concept and role names and does not cover objects as part of the signature of an ontology.
- General Concept Inclusion Axioms: The presence of general inclusion axioms makes it hard to determine the impact of ignoring parts of the vocabulary on the interpretation of a certain concept.

Examples In the following we illustrate the use of description logic for defining and reasoning about concepts in our example ontology from figure 5.1. In particular, we look at the definitions of concepts related to motherhood. In our ontology the concept mother is defined as an intersection of the concepts Woman and Parent stating that each mothers is both, a woman and a parent.

$$\texttt{Mother} \equiv \texttt{Woman} \sqcap \texttt{Parent}$$

These two concepts in turn are defined as special cases of the person concept using the relations `has-gender` and `has-child`. In particular, these relations are used to claim that each woman must have the gender female and that each parent must have a person as a child.

$$\texttt{Woman} \equiv \texttt{Person} \sqcap \exists \texttt{has} - \texttt{gender}.\texttt{Female}$$

$$\texttt{Parent} \equiv \texttt{Person} \sqcap \exists \texttt{has} - \texttt{child}.\texttt{Person}$$

Finally, the concept of a grandmother is defined by chaining the has-child relation to state that every instance of this class is a Woman with a child that has a child itself which is a Person.

$$\texttt{Grandmother} \equiv \texttt{Woman} \sqcap \exists \texttt{has} - \texttt{child}.(\exists \texttt{has} - \texttt{child}.\texttt{Person})$$

Description Logics are equivalent to a fragment of first order logic. Corresponding semantics preserving translation rules from Description logic expressions are given in [2, 11, 72]. Subsumption between concepts ($C \sqsubseteq D$) can be decided based on this semantics. In particular one concept subsumes another if the first order representation of D is implied by the first order representation of C. This way, we can for example find out that `Grandmother` is a subclass of `Mother`.

5.2 Reasoning with Limited Vocabularies

The idea of reasoning with limited vocabularies has been used in the area of approximate reasoning in order to improve efficiency of propositional inference. Cadoli and Schaerf propose a notion of approximate entailment that allows errors on parts of the vocabulary – in their case propositional letters [63]. We adopt the general idea of allowing errors on parts of the vocabulary and generalize this idea to the case where the vocabulary does not consist of propositional letters, but of concepts and relations. Cadoli and Schaerf also present an extension of their approach to description logics, but in this work the sub-vocabulary does not correspond to concept and role names but rather to the position of

a subexpression [16]. As our aim is to provide a mechanism for "switching on and off" certain concept and relation names, we have to find a different way of defining inference with respect to a certain sub-vocabulary.

5.2.1 Vocabulary-Limited Subsumption

The basic idea of our approach to inference with limited vocabularies is that terminologies define restrictions on the interpretation of certain concepts. Based on these restrictions, we can decide whether one concept is subsumed by another one. In the case of a limited vocabulary, we only want to consider such restrictions that relate to a certain sub-vocabulary under consideration and ignore other restriction. If we want to implement this idea, the basic problem is to identify and eliminate those restrictions that are not related to the sub-vocabulary under consideration. Here we have to distinguish two kinds of restrictions:

1. the interpretation of a concept can be restricted by claiming that instances of this concept belong to a set defined by a Boolean expression over other concept names.
2. the interpretation of a concept can be restricted by claiming that instances of the concept are related to other object with certain properties via a certain relation.

We can deal with the first kind of restriction in the same way as with propositional logic. Therefore we adopt the approach of Cadoli an Schaerf who replace concepts that are not in the relevant sub-vocabulary as well as their negations by true. For the case of Description logics this means that we replace concepts and their negations by $\top$, thus disabling the restriction imposed by them.

The second kind of restrictions can be dealt with by just ignoring those restrictions that are related to a certain relation. This includes the restrictions on the related objects. More specifically, we can disable these kind of restrictions by replacing subexpressions that contain a relations $r \notin V$ – in particular subexpressions of the form $(\exists r.C), (\forall r.C), (\geq n\, r)$ and $(\leq n\, r)$ – by $\top$.

Definition 10 (Approximation) *Let $\mathcal{V} = \mathcal{C} \cup \mathcal{R}$ be the vocabulary (the set of all concept and role names) of an ontology. Let further $V \subseteq \mathcal{V}$ be a subset of $\mathcal{V}$ and X a concept expression in negation normal form, then the approximation of a concept expression X $approx_V(X)$ is defined by:*

- *Replacing every concept name $c \in \mathcal{V} - V$ that occurs in X and its negation by $\top$*
- *Replacing every subexpression of X that directly contains a slot name $r \in \mathcal{V} - V$ and its negation by $\top$*

The restriction of terminologies to axioms that only have atomic concept names on the left hand side allows us to apply the approximation defined above to complete terminologies in a straightforward way by replacing the right hand sides of the axioms in a terminology by their approximation. Further, we remove the definitions of concepts not in V as they are irrelevant. the corresponding definition of an approximated terminology is the following:

Definition 11 (Approximated Terminology) *Then we define the approximation of a terminology $\mathcal{T}$ with respect to sub-vocabulary V as*

$$\begin{aligned}\mathcal{T}_V &= \{A \sqsubseteq approx_V(C) | A \in V, (A \sqsubseteq C) \in \mathcal{T}\} \cup \\ &\quad \{A \equiv approx_V(C) | A \in V, (A \equiv C) \in \mathcal{T}\}\end{aligned}$$

The approximated terminology T_V represents the original model while ignoring the influence of the concepts and relations not in V. Consequently, if we can derive a subsumption statement $C \sqsubseteq D$ from this terminology, we can interpret this as subsumption with respect to the sub-vocabulary V.

Definition 12 (Subsumption wrt a sub-vocabulary) *Let $\mathcal{T}$ be a terminology with sub-vocabulary $V \subseteq \mathcal{V}$, let further $C, D \in V$ be concept names in V, then we define the notion of subsumption with respect to sub-vocabulary V as:*

$$\mathcal{T} \models C \underset{V}{\sqsubseteq} D \Leftrightarrow_{def} \mathcal{T}_V \models C \sqsubseteq D$$

In this case, we say that C is subsumed by another concept D with respect to sub-vocabulary V

The definition leaves us with a family of subsumption operators, one for each subset of the vocabulary. Below we illustrate the use of the operator with respect to the motivating example.

Example 1: Gender We can now apply the notion of subsumption with respect to a sub-vocabulary to our example ontology and exclude certain aspects from the definitions. The first is the case where the target application does not care about the gender of a person. We treat this case by replacing the classical notion of subsumption by subsumption with respect to the vocabulary $\mathcal{V} - \{\texttt{has} - \texttt{gender}\}$. We implement this by replacing subexpressions that directly contain the slot has-gender by $\top$. The result of this operation on the example definitions from above are:

$$\texttt{Woman} \equiv \texttt{Person} \sqcap \top$$

$$\texttt{Parent} \equiv \texttt{Person} \sqcap \exists \texttt{has} - \texttt{child}.\texttt{Person}$$

$$\texttt{Mother} \equiv \top \sqcap \texttt{Parent}$$

$$\texttt{Grandmother} \equiv \texttt{Mother} \sqcap \exists \texttt{has} - \texttt{child}.(\exists \texttt{has} - \texttt{child}.\texttt{Person})$$

As a consequence of this operation, there are a number of changes in the inferable concept hierarchy. In particular, the concept `Mother` becomes equivalent to `Person` with respect to the sub-vocabulary $\mathcal{V} - \{\texttt{has} - \texttt{gender}\}$. The same happens with respect to the concept `Man` which also becomes equivalent to the other two concepts with respect to $\mathcal{V} - \{\texttt{has} - \texttt{gender}\}$. This means that the ontology does not make a distinction between male and female persons any more which is exactly what we wanted to achieve.

Example 2: Children In the same way, we can implement our second motivating example where we do not want to distinguish between persons with and without children. For this case, we use subsumption with respect to sub-vocabulary $\mathcal{V} - \{\texttt{has} - \texttt{child}\}$. Replacing the corresponding subexpressions in our example by $\top$ leads to the following definitions:

$$\texttt{Woman} \equiv \texttt{Person} \sqcap \exists \texttt{has} - \texttt{gender}.\texttt{Female}$$

$$\texttt{Parent} \equiv \texttt{Person} \sqcap \top$$

$$\texttt{Mother} \equiv \texttt{Woman} \sqcap \texttt{Parent}$$

$$\texttt{Grandmother} \equiv \texttt{Mother} \sqcap \top$$

In this case, we see that the concept `Parent` becomes equivalent to `Person` with respect to subvocabulary $\mathcal{V} - \{\texttt{has} - \texttt{child}\}$. This, in turn makes `Mother` and `Grandmother` equivalent to `Woman`. As we can see, using this weaker notion of subsumption a whole branch of the hierarchy that used to describe different kinds of female parents collapses into a single concept with different names. With respect to our application that does not care about children, this is a wanted effect as we do not want to distinguish between different female persons on the basis of whether they have children or not.

5.2.2 Defining Viewpoints

As sketched in the motivation, each approximate subsumption operator defines a certain viewpoint on an ontology. In particular, it defines which aspects of a domain are relevant from the current point of view. If we chose the sub-vocabulary such that it does not contain the slot `has-gender` then we state that the corresponding aspect is not of interest for the particular viewpoint implemented by the subsumption operator. This basically means that we actually define a viewpoint in terms of a relevant part of the vocabulary.

The corresponding subsumption operator serves as a tool for implementing this viewpoint. Based on this idea we define a viewpoint on an ontology as the set of subsumption relations that hold with respect to a certain sub-vocabulary.

Definition 13 (Viewpoint) *Let $V \subseteq \mathcal{V}$ a sub-vocabulary, then the viewpoint induced by sub-vocabulary V ($\mathcal{P}_V$) is defines as:*

$$\mathcal{P}_V = \{C \sqsubseteq D | C \underset{V}{\sqsubseteq} D\}$$

Example 1: Gender If we apply the above definition of a viewpoint on our example, we get a modified concept hierarchy, that reflects the corresponding viewpoint on the domain. For the case of the sub-vocabulary $\mathcal{V} - \{\texttt{has} - \texttt{gender}\}$ we get the hierarchy shown in figure 5.2.

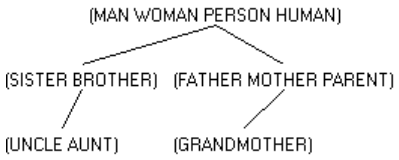


Figure 5.2: The Hierarchy if we ignore the gender

If we compare this hierarchy with the original one shown in figure 5.1, we see that all distinctions that were based on the gender of a person have disappeared from the hierarchy. Now there is a single concept containing men, women, persons and humans a single class containing mothers, fathers and parents as well as a single concept containing brothers and sisters and a single class containing uncles and aunts.

Example: Children A similar observation can be made when looking at the viewpoint defined by the the sub-vocabulary $\mathcal{V} - \{\texttt{has} - \texttt{child}\}$. The concept hierarchy representing this viewpoint is shown in figure 5.3.

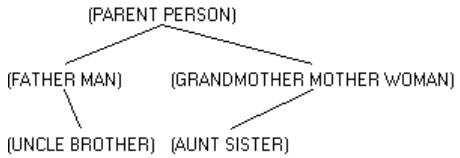


Figure 5.3: The Example Ontology if we ignore children

Again, comparing this hierarchy to the original ontology shows that all distinctions that were based on the excluded property have disappeared from the hierarchy. In particular, the root of the hierarchy is now a concept that contains all people and all parents which are now indistinguishable. As in the previous example, this phenomenon occurs across the hierarchy as we now have a single class for women, mothers and grandmothers, a single class for men and fathers as well as a single class for brothers and uncles as well as for sisters and aunts.

5.3 Multi-Perspective Reasoning

The notion of subsumption with respect to a sub-vocabulary comes with new forms of reasoning. We can not longer only ask whether one concept subsumes another, but also whether it does with respect to a certain sub-vocabulary or ask for sub-vocabularies in which a concept subsumes another one. In the following, we first discuss some general properties of the subsumption operator introduced above that defines the relation between subsumption and sub-vocabularies. We then show how we can use the formal properties to efficiently compute viewpoints using sets of maximal vocabularies that ensure subsumption between a pair of concepts.

The reasoning tasks we have to consider in the context of multi-viewpoint representations are the same as for standard OWL ontologies. As in OWL, computing subsumption between two concept expressions is one of the basic reasoning tasks many other tasks such as classification and instance retrieval can be reduced to.

What makes reasoning in our framework different from standard reasoning is the fact that we have to deal with many different subsumption operators. In order to reduce the complexity of the task, we can refer to the axiomatic properties shown above and use the implications between subsumption statements to improve reasoning. If we know for example that C is subsumed by D with respect to the complete vocabulary, we do not have to check whether C subsumes D in any sub-vocabulary, as this is necessarily always the case.

We can use the same insight to support the computation of a viewpoint. The idea is that in order to compute the viewpoint with respect to a sub-vocabulary V, we do not really have to check whether for each pair of concepts whether subsumption holds with respect to V. It is sufficient if we know that subsumption holds with respect to a larger sub-vocabulary $V' \supseteq V$. It is not directly evident why this helps to reduce reasoning effort as normally computing subsumption with respect to a larger vocabulary is more costly. We can make use of this property, however, if we know the maximal sub-vocabulary V for which $C \underset{V}{\sqsubseteq} D$ holds. In this case, we just have to test whether the current sub-vocabulary is a subset of the maximal vocabulary in order to decide conditional subsumption.

Definition 14 (Maximal Subsumption Vocabulary) *Let C and D be concept expressions. A sub-vocabulary $V \subseteq \mathcal{V}$ is called a maximal Subsumption Vocabulary for C and D if*

1. $C \underset{V}{\sqsubseteq} D$
2. *there is no $V' \supset V$ such that $C \underset{V'}{\sqsubseteq} D$*

Unfortunately, there is not always a unique maximal sub-vocabulary with the required properties. If we look at the following example, we see that C is subsumed by D with respect to $V = \{Q\}$ as $approx_{\{Q\}}(C) = approx_{\{Q\}}(D) = Q$ and that C is subsumed by D with respect to $V = \{R\}$, because in this case we have $approx_{\{R\}}(C) = approx_{\{R\}}(D) = \top \sqcap \exists R.\top$. At the same time, C is not subsumed by D with respect to $V = \{Q, R\}$ as we can easily see.

$$D \equiv Q \sqcap \exists R.Q \tag{5.1}$$
$$C \equiv Q \sqcap \exists R.(\neg Q) \tag{5.2}$$

Nevertheless, maximal sub-vocabularies, even though there may be more than one are important with respect to efficient reasoning about viewpoints. In particular, we can store a list of all maximal sub-vocabularies with reach pair of concepts, which suffices to test whether a given viewpoint is defined by a sub-vocabulary of one of the maximal ones stored. In this case, we know that C is subsumed by D in the current viewpoint.

This means that computing the set of maximal subsumption vocabularies for each pair of concepts is the primal reasoning task in the context of multi-viewpoint reasoning. In the following we provide a first algorithm for computing maximal subsumption vocabularies as a basis for more advanced reasoning tasks.

The algorithms computes for every pair C,D of concepts the set $MSV(C, D)$ of maximal Subsumption Vocabularies for C and D. This is done on the basis of a partial ordering of possible sub-vocabularies where the complete vocabulary is the first element in the order and sub-vocabularies are ordered by their cardinality. The algorithm now tests for each vocabulary if C is subsumed by D with respect to this vocabulary starting with the largest one. If this is the case, the vocabulary is added to $MSV(C, D)$ and all subsets of the vocabularies are removed from the order as they do not satisfy the second condition of definition 14. The result is a complete set of maximal subsumption vocabularies for each pair of concepts that can be used for efficiently checking the subsumption with respect to a certain sub-vocabulary. In particular, we can use the result of the algorithm to compute Viewpoints without actually computing subsumption. The corresponding algorithm is given below.

Algorithm 3 Maximal Subsumption Vocabulary (MSV)

Require: A set $\mathcal{C}$ of Concept Expressions over Vocabulary $\mathcal{V}$
Require: An ordering $(V_0, V_1, \cdots, V_m)$ on the subsets of $\mathcal{V}$ such that $V_0 = \mathcal{V}$ and $i < j \Rightarrow |V_i| > |V_j|$
  for all $\{(C, D)|C, D \in \mathcal{C}\}$ **do**
    $MSV(C, D) := \emptyset$
    $Cand(C, D) := (V_0, V_1, \cdots, V_m)$
    for all $V \in Cand(C, D)$ **do**
      if $approx_V(C) \sqsubseteq approx_V(D)$ **then**
        $MSV(C, D) := MSV(C, D) \cup \{V\}$
        $Cand(C, D) := Cand(C, D) - \{V'|V' \subset V\}$
      end if
    end for
  end for

Algorithm 4 Viewpoint

Require: A set $\mathcal{C}$ of Concept Expressions over Vocabulary $\mathcal{V}$
Require: A subvocabulary $V \subseteq \mathcal{V}$
  $\mathcal{P}_V := \emptyset$
  for all $\{(C, D)|C, D \in \mathcal{C}\}$ **do**
    if $\exists V' \in MSV(C, D) : V \subset V'$ **then**
      $\mathcal{P}_V := \mathcal{P}_V \cup \{C \subseteq D\}$
    end if
  end for

The computation can further be optimized by using special index structures that already contain all subsets of $MSV(C, D)$. In this case, a viewpoint can be computed in linear time with respect to the number of concept pairs (quadratic with respect to the number of concepts). This means that based on a centralized generated index structure different applications can efficiently access their personal viewpoint of the model.

5.4 Discussion

In this paper, we proposed a model for representing and reasoning with multiple viewpoints in description logic ontologies. Our goal was to support the reuse of existing ontologies by applications that consider different aspects of the domain to be relevant. We have shown how we can deal with the case where a new application only considers a subset of the aspects encoded in the ontology relevant using an approximate subsumption operator that only takes a subset of the vocabulary into account.

If we really want to support the reuse of ontologies, we also have to take cases into account, where the aspects relevant to the new application are not a strict subset of the aspects covered by the ontology. In this case, the new aspects have to be integrated into the ontology. Currently this is often not done on the original ontology, because there is a danger of producing unwanted inconsistencies and to destroy existing subsumption relationships. Instead, a new ontology is created and customized to the needs of the new context. We think that the framework for multiple-viewpoints in ontologies can also help in this situation as it makes it possible to extend the original ontology with new aspects while still keeping it intact for its previous applications. The previous applications can just use the viewpoint that corresponds to the vocabulary that existed before the extension.

This possibility to keep one ontology and extend it for different purposes brings us closer to the idea of an ontology as a conceptualization that us actually shared between different applications. The use of viewpoints makes it possible to sign up for a common ontology without being forced to a viewpoint taken by other applications. This increases the chances of reducing the fragmentation of ontology development where a new ontology is created for every new application. The hope is, that the number of ontologies about a certain domain can be reduced to a number of models that represent completely non-compatible views on a domain while applications that have a different but compatible view on the domain use different viewpoints on the same ontology which evolves with every new application that introduces new aspects into the ontology.

From a theoretical point of view, the notion of approximate subsumption is a very interesting one. In this work, we chose a very specific definition and implementation of subsumption with respect to a sub-vocabulary. The definition was directly motivated by the aim to define different viewpoints on the same ontology. In future work we will aim at investigating approximate subsumption based on limited vocabularies in a more general

setting. In particular, we will investigate a model-theoretic characterization of approximate subsumption in terms of weaker and stronger approximations (the work presented here is a special form of weaker approximation).

Chapter 6

Concept Approximation using Rough Sets

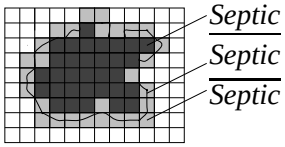
by STEFAN SCHLOBACH, MICHEL KLEIN, AND LINDA PEELEN

6.1 Introduction

Many existing knowledge modeling techniques are best suited for modeling crisp knowledge. In practice, however, it is not always possible to make clear-cut distinctions. A modeler frequently has to account for borderline cases. Approaches that do take such uncertainty or vagueness into account often do this via some kind of weighting mechanism or an approach based on fuzzy sets. A drawback of these approaches is that uncertainty is introduced in the model, which often has the consequence that no crisp answers can be given to queries on the model. This chapter introduces a complementary mechanism that allows for modelling of vague knowledge by crisp specification of approximations of a concept.

Medicine is a typical domain where concepts cannot always be described in a crisp manner. E.g., the definition of a disease is not always clear-cut, especially if a single marker is lacking that distinguishes a patient with a disease from a patient without the disease. This is common in psychiatry and in diseases in which the underlying pathology of the disease is unclear. An example of the latter is *sepsis*. Rough Description Logics (Rough DL) provides us with the possibility to describe such diseases for which a crisp definition is lacking.

Rough DL extends classical Description Logic ([2]) by two modal-like operators, called the lower and upper approximations. In the spirit of Rough Set theory [58], two concepts approximate an underspecified, vague, concept as particular sub- and super-concepts, describing *which elements* are **definitely**, respectively **possibly**, elements of the concept. The following picture illustrates the general idea:



Each square denotes a set of domain elements, which cannot further be discerned by any available criterion. Then, the circled line denotes the set of septic patients, i.e., the vague concept which we are incapable to formally define. If we capture this lack of criteria to discern between two objects as a indiscernibility relation dis$^{\sim}$, we can formally define the upper approximation as the set of patients that are indiscernible from at least one septic patient.

$$\overline{\text{Septic}} \equiv \{pat_1 \mid \exists\, pat_2\text{: dis}^{\sim}(pat_1,pat_2) \,\&\, pat_2 \in \text{Septic}\}.$$

Similarly, we can define the lower approximation as the set of patients containing all, and only those patients, for which it is known that all indiscernible patients must be septic.

$$\underline{\text{Septic}} \equiv \{pat_1 \mid \forall\, pat_2\text{: dis}^{\sim}(pat_1,pat_2) \rightarrow pat_2 \in \text{Septic}\}$$

In our picture, the upper approximation is depicted as the union of the dark squares (the lower approximation), and the gray squares, the boundary. This semantics can be transferred to Rough DL approximations in a straightforward way: the patients in the concept $\underline{\textit{Septic}}$ are the **definitely** septic patients, those that are unmistakably septic, the concept $\overline{\textit{Septic}}$ models the **possibly** septic patients, as opposed to the white squares, which model **definitely not** septic patients. These approximations are to be defined in a crisp way.

Technically, Rough DL are very simple languages, as they can be simulated with traditional DL without added expressiveness. This means that reasoning can be performed by translation, and subsequent use of a common DL reasoner. We consider it a big advantage of our approach that we can use an optimised DL reasoner without having to develop new ad-hoc decision procedures and implementations. In other words, our Rough DL's are strictly speaking not more expressive than traditional DL's, but the notions that we introduce are useful modeling devices for specific types of knowledge (namely non-crisp concepts).

Our current research was motivated by a recent study of the definitions for sepsis used in clinical trials. Before a medical treatment can be used in daily clinical practice, its effect and impact on the patient have to be investigated in a clinical trial. When several trials have been performed it is interesting to compare the results of those trials. Unfortunately, the nine different trials that were investigated in [59] showed too much variation in their definitions of severe sepsis patients to enable a fair comparison of trial results.

We show how to use Rough DL to formalise and compare sepsis definitions used in different trials. Describing sepsis through approximations enforces powerful semantic consequences. Rough DL turns out to be an appropriate logical representation language

to model vague concepts and provide crisp answers to queries, and can thereby assist in the *validation* of existing and, ultimately, the *construction* of new trials.

The remainder of the chapter is structured as follows. First, we introduce our use-case, the medical condition sepsis. In Section 6.3, Rough DL is defined as an extension to standard DL for modeling vague knowledge. We give some logical consequences of the semantics of the extension, and explain how reasoning can be done by reducing Rough DL to standard DL reasoning. In Section 6.4, we use Rough DL to model definitions of severe sepsis used in different clinical trials. Based on real patient data we evaluate the design of the trials.

6.2 Sepsis: a condition with a vague definition

Severe sepsis is our example for vague information throughout the chapter. Therefore, we will briefly provide some medical background. Sepsis is a disease in which the immune system of the patient overreacts to an infection. Due to this reaction the patient becomes severely ill, which easily results in organ failure and eventually death. The cause and underlying cellular pathways of this disease are unclear, which hinders the precise characterization of the sepsis patient. Therefore, a *consensus definition* of sepsis was established in 1992 to define several stages of sepsis [9]. This definition does not provide a precise definition of sepsis, but gives the criteria for which there was a consensus that they should at least hold for a patient with severe sepsis. In this chapter we focus on the patients with *severe sepsis*, but for brevity we will refer to these patients as *septic*. The consensus statement defines patients with severe sepsis as 'patients having a confirmed infection with at least two out of four Systemic Inflammatory Response Syndrome (SIRS) criteria:

- temperature $>38°C$ OR temperature $<36°C$
- respiratory rate >20 breaths/min OR $PaCO_2<32$ mmHg
- heart rate >90 beats/minute
- leucocyte count $<4{,}000$ mm^3 OR $>12{,}000$ mm^3

and organ dysfunction, hypoperfusion, or hypotension. From now on we refer to these criteria as the *Bone* criteria.

Patients who have this combination of symptoms may have sepsis, however, this is not necessarily the case. We refer to these patients as being *possibly septic*. On the other hand, we can define a group of patients that are septic for sure, namely those who fulfill the Bone criteria and have severe multiple organ failure. We will refer to these patients as the *definitely* septic patients and define them as fulfilling the *strict* criteria: the Bone criteria plus at least three of the following symptoms of organ failure:

- pH $\leq$ 7.30
- thrombocyte count $<$ 80,000 mm^3
- urine output $<$ 0.5 ml/kg body weight/hour (provided the patient is not on chronic dialysis),
- $PaO_2/FiO_2 \leq 250$, and
- systolic blood pressure $<$90 $_{mmHg}$ OR vaso-active medication.

6.3 Rough DL for vague knowledge

We now present a conservative extension of Description Logics (DLs), i.e. an extension which improves the modeling capacities without changing the expressive power of the language. More concretely, we will introduce two modal-like operators $(\underline{\cdot})$ and $(\overline{\cdot})$ for lower and upper approximations to describe elements which either belong definitively or possibly to the concepts under its scope. These operators introduce a notion of approximation without effectively increasing the expressiveness of the language. Thus, we get extra modeling facilities for free, without having to develop new calculi, and without paying an extra price in computational complexity.

6.3.1 Description Logics

Description Logics (DL) are a well-studied family of set-description languages which usually come with (some or all) Boolean operators and limited quantification, and which can be extended with additional functionality in a modular way. This way properties on relations (such as symmetry, transitivity or inclusion hierarchies), number restrictions, or even some form of data-types (Concrete Domains) are often included. Description Logics have a well-defined model-theoretic semantics, and the last two decades the computational properties of a wide variety of DLs has been studied.

Formally, we introduce the DL $\mathcal{ALC}$, which is sufficient to model our case-study. The general definition of approximations, however, will be independent of any particular DL. $\mathcal{ALC}$ is a simple DL with conjunction $C \sqcap D$, disjunction $C \sqcup D$, negation $\neg C$ and universal $\forall r.C$ and existential quantification $\exists r.C$. The semantics is given as follows:

Definition 15 *Let $\mathcal{I} = (U, \cdot^{\mathcal{I}})$ be an interpretation, where U is a universe, and $\cdot^{\mathcal{I}}$ a function mapping concept names to subsets and role names to relations over U. It extends to the Boolean operators as usual and to the quantifier as follows:*

- $(\exists R.C)^{\mathcal{I}} = \{i \in U \mid \exists j \in U : (i,j) \in R^{\mathcal{I}} \;\&\; j \in C^{\mathcal{I}}\}$
- $(\forall R.C)^{\mathcal{I}} = \{i \in U \mid \forall j \in U : (i,j) \in R^{\mathcal{I}} \rightarrow j \in C^{\mathcal{I}}\}$

In a terminology $\mathcal{T}$ (called *TBox*) the interpretations of concepts can be restricted to the *models* of $\mathcal{T}$ by *axioms* of the form $C \sqsubseteq D$ or $C \doteq D$. Based on this model-theoretic semantics, concepts can be checked for *unsatisfiability*: whether they are necessarily interpreted as the empty set. Another useful semantic implication is *subsumption* of two concepts C and D (a subset relation of $C^{\mathcal{I}}$ and $D^{\mathcal{I}}$ w.r.t. all models $\mathcal{I}$ of $\mathcal{T}$) denoted by $\mathcal{T} \models C \sqsubseteq D$.

A *knowledge base* $\Sigma = (\mathcal{T}, \mathcal{A})$ extends a TBox $\mathcal{T}$ with an assertional component (usually called *ABox*) $\mathcal{A}$, which is a set of assertions $i : C$ and $R(i, j)$ for individual names i, j, a relation R and a concept C. The semantics is a straightforward extention of the previous definition: an interpretation $\mathcal{I}$ is a model for a assertions $i : C$ and $R(i, j)$ if, and only, $i^{\mathcal{I}} \in C^{\mathcal{I}}$ and $R^{\mathcal{I}}(i^{\mathcal{I}}, j^{\mathcal{I}})$. Then, a knowledge base is *consistent*, if there is a model for both its TBox and ABox.

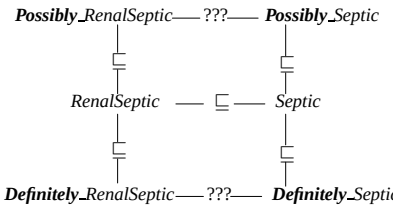
6.3.2 Rough Description Logics

Description Logics are suitable for modeling crisp knowledge but are often too rigid for approximate information. For example, no explicit mechanism is in place when a definition is not commonly agreed upon, or when exceptions need to be captured. The sepsis use-case provides an example for such vaguely defined classes, for which *no agreed upon criteria* exist to determine whether a patient is indeed septic or not.

The basic idea is rather straightforward: even though we fail to formally define the class of septic patients, we can approximate it by giving an upper and a lower bound. The *upper approximation* of the set of septic patients is formed by the set of patients that fulfill the Bone criteria, i.e. the **possibly** septic patients. Orthogonally, the *lower approximation* of the set of septic patients is the set of patients that are **definitely** septic, i.e. the patients that fulfill the strict criteria.

Traditionally, in DLs this is modeled using *primitive* definitions, i.e. axioms of the form $C \sqsubseteq D$, where C is restricted by D without being fully defined. The relation between the concept *Septic* and its approximations is in the pure DL modeling just ***Definitely*** $\_$*Septic* $\sqsubseteq$ *Septic* $\sqsubseteq$ ***Possibly*** $\_$*Septic*.

Rough DL: Approximations, Syntax and Semantics Modelling vague concepts with the traditional approach has its limits when the vague concept of *Septic* patients needs to be defined. Let us consider a special type of sepsis where the renal system fails. In DL terms, the relation between renal sepsis and sepsis would be modeled by an axiom *RenalSeptic* $\sqsubseteq$ *Septic*. Again, renal sepsis is not definable in a crisp way, but there could be an approximation describing patients which have possibly renal sepsis. Now, the question arises whether possibly renal septic patients should be possibly septic, i.e. whether ***Possibly*** $\_$*RenalSeptic* $\sqsubseteq$ ***Possibly*** $\_$*Septic* or not. In traditional DL it is possible to have all typical properties of the renal sepsis, but not the typical properties of a sepsis. What is missing is automatic inheritance of the approximations in a monotonic way.



In our motivating picture there should be subsumption relations at the "???" positions, i.e. that ***Definitely*** *RenalSeptic* $\sqsubseteq$ ***Definitely*** *Septic* and ***Possibly*** *RenalSeptic* $\sqsubseteq$ ***Possibly*** *Septic* should be a logical consequence of the knowledge base. In this sense, DL is inappropriate to model vague information, as there is a stronger semantic relations underlying the approximations of a concept. With Rough Description Logics (Rough DL), which we are about to introduce, we attempt to close this gap in a conceptually simple way.

Before providing formal semantics it is worth pointing out that approximations have very distinct properties. The upper approximation is the set of patients with a strong indication that they might be septic. Formally, this means that for every patient pat_1 in ***Possibly*** *Septic*, there must be at least one septic patient pat_2, for which there are no criteria to explain why pat_2 differs from pat_1, i.e. pat_1 is indiscernible from pat_2.

Rough DL is not restricted to a particular DL, and will be defined for an arbitrary Description Logic $\mathcal{DL}$.

Definition 16 *The language $\mathcal{RDL}$ of Rough DL is the smallest set of concepts containing $\mathcal{DL}$, and for every concept $C \in \mathcal{RDL}$ also the* upper approximation $\overline{C} \in \mathcal{RDL}$ *and the* lower approximation $\underline{C} \in \mathcal{RDL}$.

The notions of rough T- and ABox, as well as rough knowledge base extend the usual notions in the expected way.

Definition 17 *Let a rough interpretation be a triple $\mathcal{I} = (U, R^{\sim}, \cdot^{\mathcal{I}})$, where U is a universe, $\cdot^{\mathcal{I}}$ an interpretation function, and $R^{\sim}$ an equivalence relation over U. The function $\cdot^{\mathcal{I}}$ maps $\mathcal{RDL}$ concepts to subsets and role names to relations over the domain U. It extends to the new constructs as follows:*

- $(\overline{C})^{\mathcal{I}} = \{i \in U \mid \exists j \in U : (i,j) \in R^{\sim} \;\&\; j \in C^{\mathcal{I}}\}$
- $(\underline{C})^{\mathcal{I}} = \{i \in U \mid \forall j \in U : (i,j) \in R^{\sim} \rightarrow j \in C^{\mathcal{I}}\}$

Intuitively, the upper approximation of a concept C covers the elements of a domain with the *typical properties* of C, whereas the lower approximation contains the *prototypical* elements of C.

What did we gain? Even if it is impossible to formally define a concept, such as *Septic*, we can often specify the approximations. In our use-case, the upper approximation can

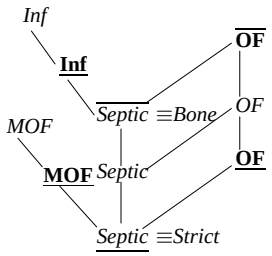
be defined using the "Bone criteria", the lower approximation, using the set of "Strict criteria" described in Section 2. In Rough DL we now model vague knowledge in a precise way; with explicit formal semantics.

Some logical consequences of the semantics Consider a simplistic Rough DL terminology, which models sepsis by its approximations. Concretely, having an infection is a certain property of **possibly septic** patients , i.e. the upper approximation $\overline{Septic}$ is a subconcept of *Inf*. Also, septic patients must have an organ failure (OF) in at least one organ system. Furthermore, **definitely septic patients** must have multiple organ failure. This gives the following terminology: $\mathcal{T} = \{\overline{Septic} \doteq$ Bone, $\underline{Septic} \doteq$ Strict, $\overline{Septic} \sqsubseteq$Inf, Septic$\sqsubseteq$ OF, $\underline{Septic} \sqsubseteq$MOF$\}$ With the implicit semantics of $\overline{\mathcal{RDL}}$ there are logical consequences, some of which we will discuss in more detail.

- *Possibly septic patients must be definitely infected.* In logical terms, we have $\mathcal{T} \models \overline{Septic} \sqsubseteq \underline{Inf}$. Why is this the case? Take a patient pat with all the typical properties of sepsis, including an infection. Assume that he has an atypical infection, i.e., that there is a similar patient pat_2 without an infection. But pat being typically septic means that there must be a septic patient pat_3 similar to pat, to which pat_2 is also similar because of transitivity. Then pat_2 is similar to a septic patient, and must belong to the upper approximation $\overline{Septic}$. By this he must have all the typical properties of sepsis, including an infection, which is a contradiction.

- *Possibly septic patients must have possible organ failure.* Formally, we can conclude that $\mathcal{T} \models \overline{Septic} \sqsubseteq \overline{OF}$. This means that if we know that organ failure is part of the proper definition of sepsis, patients that are possibly septic must at least have some condition that resembles organ failure. A similar result holds for the lower approximation $\underline{Septic}$ and multiple organ failure.

  The following figure shows the taxonomy of axioms based on the subsumption hierarchy w.r.t. the rough DL semantics, where the relations with the boldly printed concepts are implicitly derived.



  There are more examples of the intrinsic semantics of Rough DLs, which do not show in the previous figure.

- *There are no definitely non-typical sepsis patients.* Suppose that we define non-typical sepsis patients (*NTS*) as those septic patients which are not definitely septic, i.e., patients for which a similar patient exists which is not diagnosed as septic. Formally, we add an axiom $NTS \sqsubseteq Septic \sqcap \neg\underline{Sepsis}$ to $\mathcal{T}$ to get a new TBox $\mathcal{T}'$. Rough DL semantics implies that there can be no definitely non-typical septic patients, i.e. that $\mathcal{T}' \models \underline{NTS} = \bot$.
- *Definitively septic or definitively not septic.* Suppose that for a new trial only patients are selected which are definitively only diagnosed as either definitively septic, or definitively not septic, i.e., $\forall\, diag.(\underline{Septic} \sqcup \neg\underline{Septic})$. Then, every patient who **is** diagnosed as possibly septic, $\exists diag.\overline{Septic}$, must possibly have been diagnosed as definitively septic (or $\overline{\exists\, diag.\underline{Septic}}$).
- Finally, it is a simple consequence of the semantics that approximations of approximations are equivalent to the approximations themselves, e.g., that $\underline{\underline{Septic}} \equiv \underline{Septic}$.

Reasoning with Rough DLs One of the main advantages of our newly introduced modeling mechanism is that reasoning almost comes for free. As opposed to most other mechanisms to deal with vague knowledge in DL, reasoning with approximations can be reduced to standard DL reasoning, by translating rough concepts into pure DL concepts with a special reflexive, transitive and symmetric role.

Let C be a rough concept. We define a translation function $(\cdot)^t : \mathcal{RDL} \rightarrow \mathcal{DL}$ for concepts with $A^t = A$ for atomic concepts A, and $(\overline{C})^t = \exists r^{\sim}.C$, and $(\underline{C})^t = \forall r^{\sim}.C$ for $C \in \mathcal{RDL}$ where $r^{\sim}$ is a new role symbol, and where the translation function is inductively applied on subconcepts for all other constructs. This definition can be extended to axioms $(C \sqsubseteq D)^t = C^t \sqsubseteq D^t$ and TBoxes $\mathcal{T} = \{ax_1, \ldots, ax_n\}$ as follows: $\mathcal{T}^t = \{refl(r^{\sim}), sym(r^{\sim}), trans(r^{\sim}), ax_1^t, \ldots, ax_n^t\}$.

For any DL $\mathcal{DL}$ with universal and existential quantification, and symmetric, transitive and reflexive roles, there is no increase in expressive power, i.e. Rough DL can be simulated in (almost) standard DL.

Proposition 1 *Let $\mathcal{RDL}$ be the rough extension of a Description Logic $\mathcal{DL}$, $\mathcal{T}$ an $\mathcal{RDL}$ TBox, and $(\cdot)^t$ the above given translation. An $\mathcal{RDL}$ concept C is satisfiable in a rough interpretation w.r.t. $\mathcal{T}$ iff C is $\mathcal{DL}$-satisfiable w.r.t. $\mathcal{T}^t$. Formally: $\mathcal{T} \models C = \bot$ iff $\mathcal{T}^t \models C^t = \bot$.*

The proof is by contradiction: assume that $\mathcal{T} \models C = \bot$ in Rough DL, but that there is a DL model $\mathcal{I} = (U, (\cdot)^{\mathcal{I}})$ of $\mathcal{T}^t$ such that $(C^t)^{\mathcal{I}} \neq \varnothing$. It follows from the construction of the translation function $(\cdot)^t$ that $\mathcal{I}' = (U, r^{\sim}, (\cdot)^{\mathcal{I}'})$ is a model for $\mathcal{T}$, and that $C^{\mathcal{I}'} \neq \varnothing$, which is a contradiction. The other direction is similar.

As with usual DLs, one can reduce other reasoning services, such as subsumption, to satisfiability (and finally to ABox consistency) in the presence of negation. Rough DL are

no different. As the translation is linear, the complexity of reasoning in Rough DL is the same as of reasoning in its carrier DL with quantifiers, symmetry and transitivity.

6.4 Modeling Clinical trials with Rough DL

Clinical trials use *entry criteria* to select patients for the study. The choice of these criteria is an important step in clinical trial design: to be able to compare the results of the trial with those of other trials and to assess the generalizability of the results to daily clinical practice, the entry criteria have to be compatible with definitions used in comparable trials and the agreed standard definitions of disease. This is obviously complicated when no crisp disease definition exists.

In the case of severe sepsis, nine recent randomized clinical trials all used different entry criteria to select patients with severe sepsis [59]. Seven out of the nine investigated trials used a structure similar to the original consensus definition for severe sepsis: confirmed infection plus SIRS criteria plus organ failure. However, the number of required SIRS criteria varied between the trials and some trials used a slight modification of the original SIRS criteria. Furthermore, the specification of organ failure and the required number of failing organ systems differed.

One way to investigate the differences in entry criteria is to compare the definitions used in the trials with the approximations of the medical condition. In our study, we use the concepts *Strict* and *Bone* as approximations of sepsis and compare them to the entry criteria used in the nine trials. There are four interesting situations. Are there patients that are

1. in one of the trials but not in *Bone*?
2. in all trials but not in *Strict*?
3. in *Bone* but not in one of the trials?
4. in *Strict* but not in all trials?

The existence of such patients would signal a discrepancy between the trial definitions and the interpretation of sepsis, pointing to potential flaws in the set-up of the trials. With $\mathcal{RDL}$ a validation of these flaws comes for free as it allows the user to model their assumptions about the inherent vagueness of the definitions in a precise way. We will now describe how we used $\mathcal{RDL}$ to perform such an investigation.

In order to use $\mathcal{RDL}$ for patient selection we first translated the definition for each trial into a DL formula. We did the same for the *Bone* definition and the *Strict* definition of sepsis, thus building a TBox with 11 definitions for septic patients. In addition we have translated a dataset from the Dutch National Intensive Care Evaluation (NICE) registry containing information on 71,929 patients into an ABox, using the terminology from the

TBox. With the selection criteria for the different trials and the translated data, we used a DL-reasoner (Racer [36]) to select the patients that would be eligible for the different trials (thereby mimicking the patient selection process). The following table shows the numbers of patients of 4 of the 9 trials:

Definition	# patients		Definition	# patients
BONE-sepsis	5633		Lexipafant-sepsis	1607
Strict	982		OPTIMIST-sepsis	5088
UnionOfTrials	6203		PROWESS-sepsis	6201
IntersectionOfTrials	534		2SPLA2I-sepsis	4002

To answer the aforementioned questions, we define $\overline{\textit{Sepsis}} \equiv \textit{Bone}$ and $\underline{\textit{Sepsis}} \equiv \textit{Strict}$ in our $\mathcal{RDL}$ terminology, as those are the most widely accepted upper and lower approximations of *Sepsis*. Additionally, we can model the relation of the trials to the concept *Sepsis* explicitly. Although the 9 different trials widely cover different ways of describing possibly septic patients, it might be conceivable that there are patients outside the scope of all of these trials. However, one could assume that the 9 trials cover the *most typical* of *all possible* sepsis patients. Because $\mathcal{RDL}$ provides formal representations for the intuitions 'most typical' and 'all possible', we can model this assumption in a formal way. Namely, the union of all trials is equivalent to the lower approximation (i.e. the typical cases) of the upper approximation (i.e. all possible cases) of *Sepsis*. Similarly, we can model the assumption that the intersection of all trials covers the *most typical* patients that are *definitively septic*. This is done by defining the intersection of all trial concepts to be a lower approximation (i.e. the most typical cases) of the lower approximation (i.e. the definitively septic patients) of the concept *Sepsis*.

Given our experimental setup it is easy to show that there are serious flaws in the trial selection. It is a consequence of the semantics of $\mathcal{RDL}$ that an approximation itself can not be approximated. This implies that $\textit{Bone} \equiv \textit{UnionOfTrials}$ and $\textit{Strict} \equiv \textit{IntersectionOfTrials}$. This resulted in inconsistency of the definitions with respect to the trial data.

Using our infrastructure one can now perform a more detailed *data-based validation* to detect the source of the logical contradiction. For example, we queried for patients with queries like $\neg\textit{Bone} \sqcap \textit{trial-X}$ to look for violations of the upper approximation and queries like $\textit{Strict} \sqcap \neg\textit{trial-X}$ for violations of the lower approximation. In this way, we found 141 patients in *PROWESS-sepsis* and 6 patients in *Lexipafant-sepsis* that do not fulfill the *Bone* criteria.

Finally, we can use purely *terminological reasoning* to analyse the trial criteria. For example, classifying all definitions brought to light that none of the concepts describing the trials is subsumed by *Bone*. This is an interesting result when compared to the data-based validation. Although for 7 of the trial definitions we did not find any patient that violated the upper approximation, such patients can exist in principle. Similarly, with respect to the lower approximation, we found that only 4 of the trial definitions subsumed *Strict*.

Advantage over standard DL Trial validation using a standard DL infrastructure without the rough extension is already a significant improvement over the current situation, in which patient selection is procedurally performed as a sequence of database queries. Using standard DL we can check violations, as discussed above, with A-box reasoning over the data set and the terminology, or purely terminologically, as suggested in the previous two paragraphs (which are not necessarily restricted to $\mathcal{RDL}$).

Modeling the definitions in $\mathcal{RDL}$ gives an additional improvement: the validation against the criteria is done automatically. There is a way of achieving the same validation with pure DL, which we is much less elegant, though. Here, one would sequentially check the validation criteria 1 to 4 introduced above, i.e. by checking satisfiability of the concept $\neg$*Bone* $\sqcap$ *trial-X* for all trials. However, this amounts to a procedural verification of the assumptions about the vague definitions about *Sepsis*, which is error-prone, and tedious.

Moreover, our $\mathcal{RDL}$ model excludes the invalid definitions automatically. For example, the set of patients in *Lexipafant-sepsis* $\sqcap$ $\neg$*Bone* is empty per definition. To achieve the same result in a pure DL TBox one has to model the relation between trials and *Bone* explicitly, e.g. by asserting *UnionOfTrials* $\equiv$ *Bone*. But this is an incorrect oversimplification of the relation between the trials and the approximation of *Sepsis* as opposed to the much more accurate $\mathcal{RDL}$ formalisation.

6.5 Related and future work

The work described in this chapter covers a wide variety of topics that have been studied extensively in the literature. This means that there are a plethora of similar approaches, to which we will briefly refer.

- *Rough DL versus Modal DL.* From a technical perspective, Rough DL is a *fusion* of DL with modal **S5**.[1] Most attempts to introduce modal operators into DL focus on unions or produces, which usually requires more complex, mostly Kripke-based, semantics (e.g., [3]) and new decision procedures. Our modalities range over the domain itself rather than over varying domains, which makes them easier to handle, e.g., decidability and complexity results come for free, and we can apply existing reasoners.

- *Rough DL versus Fuzzy DL.* Fuzzy DLs have recently got increasing attention, particularly starting with the work of Straccia [66]. Vagueness of concepts is expressed as a degree of membership. Rough DL advocate a simpler and *qualitative* approach, which is appropriate for some domains, such as the medical. In our case study, e.g.,

[1] Fusion means, that the different operators of the two languages apply on different sets of roles, and don't interfere [4]. This makes fusions behave better than their more complex relatives unions or products [31].

there is no way of quantifying membership of the class *Septic*, but well-defined upper and lower approximations. Note, for example, that the Bone criteria are define in a crisp, non-fuzzy way.

- *Rough DL versus Rough Sets.* The connection between Rough Set theory and modal logic is well-established [25], and there have been previous attempts to introduce concept languages to model approximations [57]. Orlowska's work, which is closed to our own, is restricted to propositional logic and is, to the best of our knowledge, neither implemented nor practically applied and evaluated. An interesting orthogonal approach in [22] where concepts are defined as pairs of approximations. However, their semantics is non-standard, and approximate concepts cannot as easily be integrated in standard ontology languages as with Rough DL.
- *Rough DL versus Defaults.* Rough DL can also be useful to model defaults as one can use lower approximations to capture exceptions in an intuitive way. Simply speaking, the lower approximation then contains the *typical* subset of the elements of a concept. Further discussion of this idea is out of the scope of this chapter.

Extension and alternative definitions The restriction of the semantics to equivalence relations goes back to Pawlak's work [58]. To model vague concepts, one might also study approximation operators based on tolerance relations (reflective and symmetric). Also one could think of sets of equivalence classes according to different similarity relations. An interesting extension to graded rough modalities, as suggested in [77] is easily integrated into Rough DL, as they can be translated into number restrictions.

Before extending the language, the more pressing issue of efficiency of the reasoning has to be solved. So is Racer, our current DL reasoner, not optimised for reasoning with equivalence classes, which makes reasoning sometimes inefficient.

A different path for future research is the explicit integration of equivalence relations into Rough DL ABoxes. Often, data can be classified into indiscernible clusters. In a first step, Rough DL can be a suitable query language, but it is also conceivable to learn Rough DL concepts from the explicit definitions of the instances of particular concepts.

6.6 Conclusions

Rough DL, the extension to standard DLs, allows for precise modeling of vague knowledge. Modeling vague knowledge is a common need in realistic domains, e.g. in medicine. An advantage of modeling concept approximations in a qualitative way is that queries to the model give crisp answers. We have shown that reasoning in $\mathcal{RDL}$ can be reduced to standard DL satisfiability, which gives us access to reasoning infrastructure.

In our evaluation of medical trials about sepsis patients we have shown that modeling vague knowledge can help to answer important questions in the design of clinical trials.

The validation of trials based on their formal definitions is an improvement over the usual data-based validation. When the validation declaratively is done using Rough DL, the logical consequences of the semantics immediately reveals inconsistencies in the trial definitions, whereas several successive queries are necessary to do the same with standard DLs. Finally, we claim that Rough DL can be very useful when building new trials with vaguely defined medical conditions, as they enforce better models for the selection of patients.

Chapter 7

Conclusions

When KnowledgeWeb was originally conceived, approximate reasoning for achieving scalability of ontology reasoning was hardly more than an idea which appeared to be feasible. While KnowledgeWeb still has a full year to go, Deliverable D2.1.2.2v2 already shows that the approximate reasoning idea holds up to its promises. The reported evaluations show that added value can be achieved, and that a reasonable trade-off between loss of correctness and time saved is possible. It is also noticeable that the results presented in this deliverable have been published at prestigeous and highly visible conferences such as IJCAI, ISWC and ESWC, and so it is fair to say that approximate reasoning methods have made their way into the repertoire of the semantic web community.

We thus observe that the topic has gained a momentum which will no doubt carry it beyond the duration of KnowledgeWeb, and will spawn further investigations in the future. In particular, it remains to further refine the foundational approaches developed in Work Package 2.1 and to integrate them in standard reasoning solutions – and to further test them within real use cases.

Bibliography

[1] Ahmed Arara and Djamal Benslimane. Towards formal ontologies requirements with multiple perspectives. In *Proceedings of the 6th International Conference on Flexible Querz Answering Systems*, pages 150–160, 2004.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P F. Patel-Schneider. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[3] F. Baader and A. Laux. Terminological logics with modal operators. In *Proc. of IJCAI*, pages 808–814, 1995.

[4] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics and abstract description systems. *JAIR*, 16:1–58, 2002.

[5] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.

[6] Wolf-Tilo Balke and Matthias Wagner. Through different eyes: assessing multiple conceptual views for querying web services. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW (Alternate Track Papers & Posters)*, pages 196–205. ACM, 2004.

[7] Sean Bechhofer, Ian Horrocks, and Daniele Turi. The owl instance store: System description. In *Proceedings CADE-20*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[8] Catriel Beeri, Alon Y. Levy, and Marie-Christine Rousset. Rewriting queries using views in description logics. In *Proceedings of The 16th Symposium on Principles of Database Systems*, pages 99–108, 1997.

[9] Bone, R.C. Definitions for sepsis and organ failure and guidelines for the use of innovative therapies in sepsis. *Crit Care Med*, 20(6):864–874, 1992.

[10] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003.

[11] Alexander Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.

[12] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Second International Semantic Web Conference ISWC'03*, volume 2870 of *LNCS*, pages 164–179. Springer, 2003.

[13] Craig Boutilier, Ronen I. Brafman, Holger H. Hoos, and David Poole. Reasoning with conditional ceteris paribus preference statements. In Kathryn B. Laskey and Henri Prade, editors, *UAI*, pages 71–80. Morgan Kaufmann, 1999.

[14] Ronen I. Brafman, Carmel Domshlak, Solomon E. Shimony, and Yael Silver. Tcp-nets for preferences over sets. In *WS on Advances in Preference Handling*, 2005.

[15] Sebastian Brandt, Ralf Kuesters, and Anni-Yasmin Turhan. Approximation and difference in description logics. In *Proceedings of the International Conference on Knowledge Representation and Reasoning KR-02*, pages 203–214, 2002.

[16] Marco Cadoli and Marco Schaerf. Approximation in concept description languages. In *Proceedings of the International Conference on Knowledge Representation and Reasoning*, pages 330–341, 1992.

[17] Mukesh Dalal. Efficient propositional constraint propagation. In *Proceedings of AAAI-92*, pages 409–414, 1992.

[18] Mukesh Dalal. Semantics of an anytime family of reasoners. In *Proceedings of ECAI-96*, pages 360–364, 1996.

[19] The dublin core metadata initiative. `http://dublincore.org/`.

[20] T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Abductive matchmaking using description logics. In *Proceedings of IJCAI-03*, pages 337–342, 2003.

[21] Tommaso Di Noia, Francesco Donini Eugenio Di Sciascio, and Marina Mongiello. A system for principled matchmaking in an electronic marketplace. In *Proceedings of WWW-03*, pages 321–330, 2003.

[22] P. Doherty, M. Grabowski, W. Lukaszewicz, and A. Szalas. Towards a framework for approximate ontologies. *Fundam. Inf.*, 57:147–165, 2003.

[23] Peter Dolog, Nicola Henze, Wolfgang Nejdl, and Michael Sintek. Personalization in distributed e-learning environments. In *Proc. of WWW2004 — The Thirteen International World Wide Web Conference*, New Yourk, May 2004. ACM Press.

[24] Peter Dolog, Heiner Stuckenschmidt, and Holger Wache. Robust query processing for personalized information access on the semantic web. In Henrik Legind Larsen, Gabriella Pasi, Daniel Ortiz Arroyo, Troels Andreasen, and Henning Christiansen,

editors, *7th International Conference on Flexible Query Answering Systems (FQAS 2006)*, volume 4027 of *Lecture Notes in Computer Science (LNCS)*, pages 343–355, Milan, Italy, June 2006. Springer.

[25] I. Düntsch. A logic for rough sets. *Theoretical Computer Science*, 179(1-2):427–436, 1997.

[26] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: a framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering*, 2:31–57, 1992.

[27] Terry Gaasterland, Parke Godfrey, and Jack Minker. An overview of cooperative answering. *Journal of Intelligent Information Systems*, 1(2):123–157, 1992.

[28] Terry Gaasterland, Parke Godfrey, and Jack Minker. Relaxation as a platform for cooperative answering. *Journal of Intelligent Information Systems*, 1(3/4):293–321, 1992.

[29] Terry Gaasterland and Jorge Lobo. Qualified answers that reflect user needs and preferences. In *Proceedings of 20th International Conference on Very Large Data Bases (VLDB94)*, pages 309–320, 1994.

[30] Terry Gaasterland and Jorge Lobo. Qualifying answers according to user needs and preferences. *Fundamenta Informaticae*, 32(2):121–137, 1997.

[31] D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyatschev. *Many-Dimensional Modal Logic: Theory and Applications*. Elsevier, 2003.

[32] Rosalba Giugno and Thomas Lukasiewicz. P-shoq(d): A probabilistic extension of shoq(d) for probabilistic ontologies in the semantic web. In *Proceedings of JELIA'02*, 2002.

[33] F. Giunchiglia and C. Ghidini. Local models semantics, or contextual reasoning = locality + compatibility. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 282–289. Morgan Kaufmann, 1998.

[34] Perry Groot, Heiner Stuckenschmidt, and Holger Wache. Approximating description logic classification for semantic web reasoning. In *Proceedings of the 2nd European Semantic Web Conference*, Heraklion, Crete, 2005.

[35] V. Haarslev and R. Möller. RACE system description. In *Proceedings of the 1999 DL Workshop*, CEUR Electronic Workshop Proceedings, pages 130–132, 1999.

[36] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *IJCAI'2001*, pages 161–168, 2001.

[37] V. Haarslev and R. Möller. RACER system description. In *IJCAR'2001*, volume 2083 of *LNAI*, pages 701–705. Springer, 2001.

[38] Pat Hayes. Rdf semantics. Recommendation, W3C, 2004.

[39] Pascal Hitzler and Denny Vrandecic. Resolution-based approximate reasoning for OWL DL. In Y. Gil et al., editor, *Proceedings of the 4th International Semantic Web Conference, Galway, Ireland, November 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 383–397. Springer, Berlin, 2005.

[40] I. Horrocks. The FaCT System. In *TABLEAUX'98*, volume 1397 of *LNAI*, pages 307–312. Springer, 1998.

[41] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *KR'98*, pages 636–647. Morgan Kaufmann, 1998.

[42] I. Horrocks and S. Tessaris. A Conjunctive Query Language for Description Logic Aboxes. In *AAAI*, pages 399–404, 2000.

[43] Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The Instance Store: DL reasoning with large numbers of individuals. In *Proceedings of the International Workshop on Description Logics, DL2004, Whistler, Canada*, pages 31–40, 2004.

[44] Werner Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.

[45] Werner Kießling and Gerhard Köstler. Preference sql - design, implementation, experiences. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 990–1001. Morgan Kaufmann, 2002.

[46] Ralf Kuesters and Ralf Molitor. Approximating most specific concepts in description logics with existential restrictions. *AI Communications*, 15(1):47–59, 2002.

[47] M. Lacroix and Pierre Lavency. Preferences; putting more knowledge into queries. In Peter M. Stocker, William Kent, and Peter Hammersley, editors, *VLDB'87, Proceedings of 13th International Conference on Very Large Data Bases, September 1-4, 1987, Brighton, England*, pages 217–225. Morgan Kaufmann, 1987.

[48] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. *International Journal of Electronic Commerce*, 8(4):39 – 60, 2004.

[49] A. Magkanaraki, V. Tannen, V. Christophides, and D. Plexousakis. Viewing the semantic web through rvl lenses. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):359–375, 2004.

[50] M. Mochol and E. Paslaru Bontas. Practical Guidelines for Building Semantic eRecruitment Applications. In *International Conference on Knowledge Management, Special Track: Advanced Semantic Technologies (AST' 06)*, 2006.

[51] Malgorzata Mochol, Holger Wache, and Lyndon Nixon. Improving the recruitment process through ontology-based querying. In *First International Workshop on Applications and Business Aspects of the Semantic Web (SEBIZ 2006)*, 2006. to appear.

[52] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Karlsruhe, 2006.

[53] Boris Motik and Ulrike Sattler. A comparison of reasoning techniques for querying large description logic aboxes. In *Proceedings of the 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2006), Phnom Penh, Cambodia, November, 2006*, 2006. To appear.

[54] M. Nilsson. Ims metadata rdf binding guide. http://kmr.nada.kth.se/el/ims/metadata.html, May 2001.

[55] N.F. Noy and M.A. Musen. Specifying ontology views by traversal. In *Proceedings of the Third International Conference on the Semantic Web (ISWC-2004)*, 2004.

[56] Assosiation of Computing machinery. The acm computer classification system. http://www.acm.org/class/1998/, 2002.

[57] E. Orlowska. Logical aspects of learning concepts. *Int. J. of Approx. Reasoning*, 2:349–364, 1988.

[58] Z. Pawlak. Rough sets. *Int. J. of Computer and Information Sciences*, 11:341–356, 1982.

[59] L. Peelen, N.F. De Keizer, N. Peek, E. De Jonge, R.J Bosman, and G.J. Scheffer. Influence of entry criteria on mortality risk and number of eligible patients in recent studies on severe sepsis. *Crit Care Med*, 33(10):2178–2183, 2005.

[60] Rami Rifaieh, Ahmed Arara, and Aïcha-Nabila Benharkat. Muro: A multi-representation ontology as a foundation of enterprise information systems. In *Proceedings of the 4th International Conference on Computer and Information Technology*, pages 292–301, 2004.

[61] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997.

[62] C. Sakama and K. Inoue. An alternative approach to the semantics of disjunctive logic programs and deductive databases. *Journal of Automated Reasoning*, 13:145–172, 1994.

[63] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74:249–310, 1995.

[64] Stefan Schlobach, Michel Klein, and Linda Peelen. Description logics with approximate definitions: Precise modeling of vague concepts. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 07*, Hyderabad, India, January 6–12 2007.

[65] Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer Verlag, Heidelberg, 2004.

[66] Umberto Straccia. Reasoning with fuzzy description logics. *J. of AI Research*, 14:137–166, 2001.

[67] Umberto Straccia. Towards a fuzzy description logic for the semantic web (preliminary report). In *Proceedings of ESWC-05*, pages 167–181, 2005.

[68] H. Stuckenschmidt and F. van Harmelen. Approximating terminological queries. In *Proceedings of the Fifth International Conference on Flexible Query Answering Systems FQAS 2002*, Lecture Notes in Artificial Intelligence, Copenhagen, Denmark, 2002. Springer Verlag.

[69] H. Stuckenschmidt and F. van Harmelen. *Information Sharing on the Semantic Web*. Advanced Information Processing. Springer Verlag, Berlin, Heidelberg, 2004. to appear.

[70] Heiner Stuckenschmidt. Towards multi-viewpoint reasoning in OWL ontologies. In *Proceedings ESWC2006, Budva, Montenegro*, 2006.

[71] Heiner Stuckenschmidt, Anita de Waard, Ravinder Bhogal, Christiaan Fluit, Arjohn Kampman, Jan van Buel, Erik van Mulligen, Jeen Broekstra, Ian Crowlesmith, Frank van Harmelen, and Tony Scerri. Exploring large document repositories with rdf technology - the dope project. *IEEE Intelligent Systems*, 2004. to appear.

[72] Dmitry Tsarkov, Alexandre Riazanov, Sean Bechhofer, and Ian Horrocks. Using vampire to reason with owl. In *Proceedings of the International Semantic Web Conference*, pages 471–485, 2004.

[73] Andre Valente, Thomas Russ, Robert MacGregor, and William Swartout. Building and (re)using an ontology of air campaign planning. *IEEE Intelligent Systems*, 14(1):27 – 36, 1999.

[74] R. Volz, D. Oberle, and R. Studer. Views for light-weight web ontologies. In *Proceedings of the ACM Symposium on Applied Computing SAC 2003*, 2003.

[75] W3C. Web ontology language (OWL). www.w3.org/2004/OWL/, 2004.

[76] Holger Wache, Perry Groot, and Heiner Stuckenschmidt. Scalable instance retrieval for the semantic web by approximation. In Mike Dean, Yuanbo Guo, Woochun Jun, Roland Kaschek, Shonali Krishnaswamy, Zhengxiang Pan, and Quan Z. Sheng, editors, *Proceedings of Web Information Systems Engineering - WISE 2005 Workshops; workshop Scalable Semantic Web Knowledge Base Systems*, volume 3807 of *Lecture Notes in Computer Science*, pages 245–254. Springer, 2005.

[77] Y.Y. Yao and T.Y. Lin. Generalization of rough sets using modal logics. *Intelligent Automation and Soft Computing*, 2(2):103–120, 1996.