knowledgeweb
realizing the semantic web

D2.1.2 Methods for Approximate Reasoning

Perry Groot (Vrije Universiteit Amsterdam)
Pascal Hitzler (Universität Karlsruhe)
Ian Horrocks (University of Manchester)
Boris Motik (FZI Karlsruhe)
Jeff Z. Pan (University of Manchester)
Heiner Stuckenschmidt (Vrije Universiteit Amsterdam)
Daniele Turi (University of Manchester)
Holger Wache (Vrije Universiteit Amsterdam)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.1.2 (WP2.1)

This deliverable shows examples about approximating symbolic inference engines in a Semantic Web environment. Approaches of language weakening, knowledge compilation, and approximated deduction are presented. The last one is evaluated in practical applications with mixed results.
Keyword list: state-of-the-art, scalability, approximation, modularisation, distribution, symbolic reasoning

Document Identifier	KWEB/2004/D2.1.2/v1.2
Project	KWEB EU-IST-2004-507482
Version	v1.2
Date	January 30^{th}, 2005
State	draft
Distribution	public

Knowledge Web Consortium

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Work package participants

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

Centre for Research and Technology Hellas
École Polytechnique Fédérale de Lausanne
Free University of Bozen-Bolzano
Institut National de Recherche en Informatique et en Automatique
Learning Lab Lower Saxony
Universidad Politécnica de Madrid
University of Karlsruhe
University of Manchester
University of Sheffield
University of Trento
Vrije Universiteit Amsterdam
Vrije Universiteit Brussel

Changes

Version	Date	Author	Changes
0.0	06.10.04	Holger Wache	creation
0.1	02.11.04	Pascal Hitzler	input chapter 3
0.2	10.11.04	Holger Wache	input chapter 2
0.4	13.11.04	Jeff Pan	input chapter 4
0.5	20.12.04	Pascal Hitzler	revised chapter 3
0.6	21.12.04	Holger Wache	Introduction and revised chapter 2
1.0	22.12.04	Holger Wache	Abstract and Executive Summary
1.1	10.01.05	Holger Wache	Implementing comments from WP leader
1.2	30.01.05	Holger Wache	Implementing comments from Quality assors

Executive Summary

In general approximating Semantic Web inferences can be achieved by language weakening, knowledge compilation, and approximate deduction. This deliverable discusses three approaches of approximating which all fall into this single coherent framework and are representative examples for the three classes.

An example for language weakening is InstanceStore which only allows instances without any relations to other instances. But the instances can be asserted to concepts which have relations to other concepts. The ability of representing relations is shifted from the instances to the concepts. This restriction seems to be reasonable in practical applications and allows to fall back on database technology which is known for efficient and scalable inferences.

For approximated deduction only a few approaches are available for the inferences in the context of Semantic Web. A promising approach is proposed by Cadoli and Schaerf who also apply their method for description logic reasoning. In a simplified view their method replaces the $\exists$-quantifier in concept expressions with $\top$ resp. $\bot$. Their replacement leads to simplified concept expressions with hopefully faster reasoning. However, an evaluation of their method in practical scenarios of the Semantic Web shows mixed results. First, the proposed method is not applicable for all ontologies but only for ontologies with a reasonable amount of $\exists$-quantifier in their concept definitions. Second, only few successful applications of the approximation method can be observed. Both points indicate that the wrong constructor is replaced. Third, the few successful approximations come with high amounts of unnecessary reasoning which ruins the benefit of approximated reasoning. It seems that the last point is the consequence of the replacement with $\top$ or $\bot$ which "over-simplifies" the concept expressions to meaningless statements. Summarising, in practical situations the proposed method seems to approximate the wrong constructor and seems to replace the constructor with the wrong term. However, more adapted approximation methods which prevent the "over-simplification" have the potential for better results.

OWL ontologies which are based on description logics (i.e., the species OWL-DL and OWL Lite) can also be translated to disjunctive Datalog programs. During the translation the implicit knowledge can be derived and directly added to the translated knowledge base. Furthermore the inference engine for disjunctive Datalog programs can be replaced by an approximated SLD-Resolution which ignores the disjunctions in the heads of the clauses. This reasoning is not sound but complete. The approach is representative for a combination of several approximation methods, i.e., knowledge compilation and approximate deduction.

Contents

Chapter 1

Introduction and Motivation

by PERRY GROOT, HEINER STUCKENSCHMIDT & HOLGER WACHE

A strength of the current proposals for the foundational languages of the Semantic Web is that they are all based on formal logic. This makes it possible to formally reason about information and derive implicit knowledge. However, this reliance on logics is not only a strength but also a weakness. Traditionally, logic has always aimed at modelling idealized forms of reasoning under idealized circumstances. Clearly, this is not what is required under the practical circumstances of the Semantic Web. Instead, the following are all needed:

- reasoning under time-pressure
- reasoning with other limited resources besides time
- reasoning that is not 'perfect' but instead 'good enough' for given tasks under given circumstances

It is tempting to conclude that symbolic, formal logic fails on all these counts, and to abandon that paradigm. Our aim is to keep the advantages of formal logic in terms of definitional rigour and reasoning possibilities, but at the same time address the needs of the Semantic Web.

Research in the past few years has developed methods with the above properties while staying within the framework of symbolic, formal logic. However, many of those previously developed methods have never been been considered in the context of the Semantic Web. Some of them have only been considered for some very simple underlying description languages [Schaerf and Cadoli, 1995]. As the languages proposed for modelling ontologies in the Semantic Web are becoming more and more complex, it is an open question whether those approximation methods are able to meet the practical demands of the Semantic Web. In this deliverable, we look at approximation methods for Description Logics (DLs), which are closely related to some of the currently proposed Semantic Web

languages, e.g., OWL. To be more precise, during the whole deliverable we consider only OWL DL and in some way OWL Lite but not OWL Full.

1.1 Approximation Approaches

A typical architecture for a KR system based on DLs can be sketched as in Figure 1.1 [Baader *et al.*, 2003b], which contains three components that can be approximated to obtain a simplified system that is more robust and more scalable. These components are: (1) the underlying description language, (2) the knowledge base, and (3) the reasoner. The knowledge base itself comprises two components (TBox and ABox), which can also be approximated as a whole or separately. Some general approximation techniques that can be applied to one or more of these components are the following:

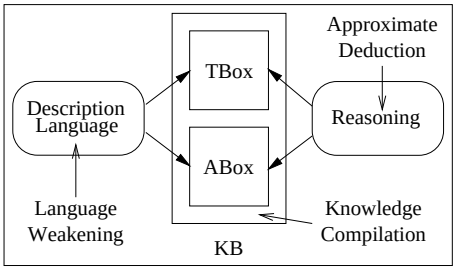

Figure 1.1: Architecture of a KR system based on Description Logic together with possible approximation approaches.

Language Weakening: The idea of language weakening is based on the well-known trade-off between the expressiveness and the reasoning complexity of a logical language. By weakening the logical language in which a theory is encoded, we are able to trade the completeness of reasoning against run-time. For example, [Borgida and Etherington, 1989] shows how hierarchical knowledge bases can be used to reason approximately with disjunctive information. The logic that underlies OWL Full for example is known to be intractable, reasoners can use a slightly weaker logic (e.g. OWL Lite) that still allows to compute some consequences. This idea can be further extended by starting with a very simple language and iterating over logics of increasing strength supplementing previously derived facts.

Knowledge Compilation: In order to avoid complexity at run-time, knowledge compilation aims at pre-processing the ontology off-line such that on-line reasoning becomes faster. For example, this can be achieved by explicating hidden knowledge. Derived facts are added to the original theory as axioms, avoiding the need to deduce them again. In the case of ontological reasoning, implicit subsumption and membership relations are good candidates for compilation. For example, implicit subsumption relations in an OWL ontology could be identified using a DL reasoner, the resulting more complete hierarchy could be encoded e.g. in RDF schema and used by systems that do not have the ability to perform complex reasoning. This example can be considered to be a transformation of the DL language. When one transforms an ontology into a less expressive DL language [Baader *et al.*, 2000, Brandt *et al.*, 2002], this often results in an approximation of the original ontology.

Approximate Deduction: Instead of modifying the logical language, approximations can also be achieved by weakening the notion of logical consequence [Schaerf and Cadoli, 1995, McAllester, 1990]. The approximated consequences are usually characterized as sound but incomplete, or complete but unsound. Only [Schaerf and Cadoli, 1995] have made some effort in the context of DLs.

Note that there is not always a clear classification of one method to the three categories defined above.

In the following chapters we discuss some approximation methods in more detail. Chapter 2 gives an example of approximate deduction. A concrete and already proposed approximation method — the S-1- and S-3-approximation by Cadoli and Schaerf [Schaerf and Cadoli, 1995] — is investigated in practical applications. The approach in Chapter 3 is based on language weakening. The expressiveness of the ABox is reduced in order to fall back on database technology. A combination of several approximation methods is illustrated in Chapter 4. First the knowledge base is compiled into disjunctive Datalog programs explicating the implicit knowledge directly. Then the standard resolution principle is replaced by an approximated variant. Therefore this approach can be seen as a combination of language weakening and approximate deduction.

Chapter 2

S-1- and S-3-Approximation

by PERRY GROOT, HEINER STUCKENSCHMIDT & HOLGER WACHE

The elements of a DL are concept expressions and determining their satisfiability is the most basic task. The most of the other reasoning services (e.g., subsumption, classification, instance retrieval) can be restated in terms of satisfiability checking [Baader *et al.*, 2003b]. With approximation in DLs, we mean determining the satisfiability of a concept expression through some other means than computing the satisfiability of the concept expression itself. This use of approximation differs with other work on approximating DLs [Baader *et al.*, 2000, Brandt *et al.*, 2002] in which a concept expression is translated to another concept expression, defined in a second typically less expressive DL.

In our approach (originally proposed by Cadoli and Schaerf in [Schaerf and Cadoli, 1995]), in a DL only other, somehow 'related', concept expressions can be used that are in some way 'simpler' when determining their satisfiability. For example, a concept expression can be related to another concept expression through its subsumption relation, and a concept expression can be made simpler by either forgetting some of its sub concepts or by replacing some of its sub concepts with simpler concepts. In particular, there are two ways that a concept expression C can be approximated by a related simpler concept expression D. Either the concept expression C is approximated by a weaker concept expression D (i.e., less specific, $C \sqsubseteq D$) or by a stronger concept expression D (i.e., more specific, $D \sqsubseteq C$). When $C \sqsubseteq D$, unsatisfiability of D implies unsatisfiability of C. When $D \sqsubseteq C$, satisfiability of D implies satisfiability of C. Note that this is similar to set theory. For two sets C, D, when $C \subseteq D$ holds, emptiness of D implies emptyness of C, and when $D \subseteq C$ holds, non-emptiness of D implies non-emptyness of C.

In [Schaerf and Cadoli, 1995] Cadoli and Schaerf propose a syntactic manipulation of concept expressions that simplifies the task of checking their satisfiability. The method generates two sequences of approximations, one sequence containing weaker concepts and one sequence containing stronger concepts. The sequences of approximations are obtained by substituting a substring D in a concept expression C by a simpler concept.

More precisely, for every substring D they define the depth of D to be "the number of universal quantifiers occurring in C and having D in its scope" [Schaerf and Cadoli, 1995]. The scope of $\forall R.\phi$ is ϕ which can be any concept term containing D. Using the definition of depth a sequence of weaker approximated concepts can be defined, denoted by $C_i^\top$, by replacing every existentially quantified sub concept[1] of depth greater or equal than i by $\top$. Analogously, a sequence of stronger approximated concepts can be defined, denoted by $C_i^\bot$, by replacing every existentially quantified sub concept of depth greater or equal than i by $\bot$. Please note that before replacement the concept expression is transformed into the negated normal form which allows negation only for concept names. These definitions lead to the following result:

Theorem 1 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.

These definitions are illustrated by the following concept expression taken from the Wine ontology[2]

$$\texttt{Merlot} \equiv \texttt{Wine} \sqcap \ \leq 1\,\text{madeFromGrape}.\top \sqcap \exists\,\text{madeFromGrape}.\{MerlotGrape\},$$

which states that a Merlot wine is a wine that is made from the Merlot grape and no other grape. This concept expression contains no $\forall$-quantifiers. Therefore the depth of the only existentially quantified sub concept '$\exists\,\text{madeFromGrape}.\{MerlotGrape\}$' is 0. Substituting either $\top$ or $\bot$ leads to the following approximations for level 0:

$$\texttt{Merlot}_0^\top \equiv \texttt{Wine} \sqcap (\leq 1\,\text{madeFromGrape}.\top) \sqcap \top,$$

$$\texttt{Merlot}_0^\bot \equiv \texttt{Wine} \sqcap (\leq 1\,\text{madeFromGrape}.\top) \sqcap \bot.$$

No sub concepts of level 1 appear in the concept expression for `Merlot`. Therefore, $\texttt{Merlot}_1^\top$ and $\texttt{Merlot}_1^\bot$ are equivalent to `Merlot`. The nesting of existential and universal quantifiers is an important measure of the complexity of satisfiability checking when considered from a worst case complexity perspective [Donini *et al.*, 1992]. These are motivations for Cadoli and Schaef to make their specific substitution choices. Furthermore, they are able to show a relation between $C_i^\top$- and $C_i^\bot$-approximation and their multi-valued logic based on S-1- and S-3-interpretations [Schaerf and Cadoli, 1995]. Therefore, properties obtained for S-1- and S-3-approximation also hold for $C_i^\top$- and $C_i^\bot$-approximation. These properties include the following: (1) Semantically well founded, i.e., there is a relation with a logic that can be used to give meaning to approximate answers; (2) Computationally attractive, i.e., approximate answers are cheaper to compute than the original problem; (3) Duality, i.e., both sound but incomplete and complete but unsound approximations can be constructed; (4) Improvable, i.e., approximate answers

[1] i.e., $\exists R.\phi$ where ϕ is any concept term

[2] A wine and food ontology which forms part of the OWL test suite [OWL, a].

can be improved while reusing previous computations; (5) Flexible, i.e., the method can be applied to various problem domains. These properties were identified by Cadoli and Schaerf to be necessary for any approximation method.

Although the proposed method by Cadoli and Schaerf [Schaerf and Cadoli, 1995] satisfies the needs of the Semantic Web identified in Section 1 in theory, little is known about the applicability of their method to practical problem solving. Few results have been obtained for S-1- and S-3-approximation when applied to propositional logic [Groot *et al.*, 2004, ten Teije and van Harmelen, 1997, ten Teije and van Harmelen, 1996], but no results are currently known to the authors when their proposed method is applied to DLs. Current work focuses on empirical validation of their proposed method. Furthermore, DLs have changed considerably in the last decade. Cadoli and Schaerf proposed their method for approximating the language $\mathcal{ALE}$ (they also give an extension for $\mathcal{ALC}$), but $\mathcal{ALE}$ has a much weaker expressivity then the languages that are currently proposed for ontology modelling on the Semantic Web such as OWL. The applicability of their method to a more expressive language like OWL is an open question. Current work takes the method of Cadoli and Schaerf as a basis and focuses on extending it to more expressive DLs.

2.1 Approximating Classification

The problem of classification is to arrange a complex concept expression into the subsumption hierarchy of a given TBox. We choose this task for two reasons. First, the worst-case complexity of classification algorithm even for expressive representation languages like OWL-DL is known to be worse. Efficient alternatives have only been proposed e.g. for subsets of DLs [Grosof *et al.*, 2003].

Second, classification is a very important part of many other reasoning services and applications. For example, classification is used to generate the subsumption hierarchy of the concept descriptions in an ontology. Furthermore, classification is used in the task of retrieving instances. From a theoretical point of view, checking whether an instance i is member of a concept Q can be done by proving the unsatisfiability of $\neg Q(i)$. Doing this for all existing instances, however, is intractable. Therefore, most DL systems use a process that reduces the number of instance checks. It is assumed that the ontology is classified and all instances are assigned to the most specific concept they belong to. Instance retrieval is then done by first classifying the query concept Q in the subsumption hierarchy and then selecting the instances of all successors of Q and of all direct predecessors of Q that pass the membership test in Q. We conclude that there is a lot of potential for approximating the classification task.

In the following, we first describe the process of classification in DL systems. Afterwards we explain how the approximation technique introduced in Section 2 can be used to approximate (part of) this problem.

Algorithm 1 classification

Require: A classified concept hierarchy with root $Root$
Require: A query concept Q
  VISITED := $\varnothing$
  RESULT := $\varnothing$
  GOALS := $\{\top\}$
  while Goals $\neq \varnothing$ **do**
    C $\in$ Goals *where* {*direct parents of* C} $\subseteq$ Visited
    GOALS := Goals $\setminus$ {C}
    VISITED := Visited $\cup$ {C}
    if subsumed-by(Q,C) **then**
      GOALS := Goals $\cup$ {*direct children of* C}
      RESULT := (Result $\cup$ {C}) $\setminus$ {*all ancestors of* C}
    end if
  end while
  if |Result| = 1 $\wedge$ subsumed-by(C,Q) **then**
    EQUAL := 'yes'
  else
    EQUAL := 'no'
  end if
  return Equal, Result

For classifying a concept expression Q into the concept hierarchy (Algorithm 1) a number of subsumption tests are required for comparing the query concept with other concepts C_i in the hierarchy. As the classification hierarchy is assumed to be known, the number of subsumption tests can be reduced by starting at the highest level of the hierarchy and to move down to the children of a concept only if the subsumption test is positive. The most specific concepts w.r.t. the subsumption hierarchy which passed the subsumption test are collected for the results. At the end of the algorithm, we check if the result is subsumed by Q as this implies that both are equal.

Algorithm 1 contains more than one step that can be approximated. For example, the subsumption tests, represented by `subsumed-by(X,Y)` in the algorithm, can be approximated using the method of Cadoli and Schaerf. The necessary subsumption tests $Q \sqsubseteq C$ can be reformulated to test the unsatisfiability of $Q \sqcap \neg C$ (Algorithm

Algorithm 2 subsumption

Require: A complex concept expression C
Require: A Query Q
  CURRENT := $Q \sqcap \neg C$
  RESULT := unsatisfiable(Current)
  return Result

Algorithm 3 approx-$C^{\top}$-subsumption

Require: A complex concept expression C
Require: A Query Q
  I := 0
  repeat
    CURRENT := $(Q \sqcap \neg C)_I^{\top}$
    RESULT := unsatisfiable(Current)
    if Result = 'true' **then**
      break
    end if
    I := I+1
  until Current = C
  return Result

Algorithm 4 approx-$C^{\perp}$-subsumption

Require: A complex concept expression C
Require: A Query Q
  I := 0
  repeat
    CURRENT := $(Q \sqcap \neg C)_I^{\perp}$
    RESULT := unsatisfiable(Current)
    if Result = 'false' **then**
      break
    end if
    I := I+1
  until Current = C
  return Result

2). The idea is to replace standard subsumption checks by a series of approximate checks of increasing exactness. In particular, we use weaker approximations $C_i^{\top}$ for the `approx-`$C^{\top}$`-subsumption` algorithm (see Algorithm 3) and stronger approximations $C_i^{\perp}$ for the `approx-`$C^{\perp}$`-subsumption` algorithm (see Algorithm 4). Using Theorem 1 we can conclude that we are done whenever the test succeeds. If the test does not succeed, we move to a more precise approximation (by increasing I). This is repeated until we reach the original concept expression and perform the exact subsumption test. Algorithm 5 integrates both approximations in one procedure. The approximate versions, i.e., `approx-`$C^{\top}$`-subsumption`, `approx-`$C^{\perp}$`-subsumption`, and `approx-`$C_I^{\perp}$`-`$C_I^{\top}$`-subsumption` will replace the method `subsumed-by` in Algorithm 1.

While these approximate versions can in principle be applied to all occurrences of subsumption tests, we restricted the use of approximations to the first part of the algorithm where the query concept is classified into the hierarchy.

Algorithm 5 approx-$C_I^{\perp}$-$C_I^{\top}$-subsumption

Require: A complex concept expression C
Require: A Query Q
  I := 0
  repeat
    CURRENT := $(Q \sqcap \neg C)_I^{\perp}$
    RESULT := unsatisfiable(Current)
    if Result = 'false' **then**
      break
    end if
    CURRENT := $(Q \sqcap \neg C)_I^{\top}$
    RESULT := unsatisfiable(Current)
    if Result = 'true' **then**
      break
    end if
    I := I+1
  until Current = C
  return Result

Each DL reasoner (e.g., Fact [Horrocks, 1998a], Racer [Haarslev and Möller, 1999, Haarslev and Möller, 2001b]) implements the classification functinality internally. In order to obtain comparable statements about approximate classification, independently from the implementation of a particular DL reasoner, which may use highly optimized heuristics, we implement our own and independent classification method. The classification procedure was built on top of an arbitrary DL reasoner according to Algorithm 1. The satisfiability tests are propagated to the DL reasoner through the DIG interface [Bechhofer *et al.*, 2003] as depicted in Figure 2.1.

2.2 Experiments

The main question focused on in the experiments is which form of approximation, i.e., $C_i^{\top}$, $C_i^{\perp}$, or their combination, can be used to reduce the complexity of reasoning tasks. The focus of the experiments will not be on the overall computation time, but on the number of operations needed. The goal of approximation is to replace costly reasoning operations by a (small) number of cheaper approximate reasoning operations. The suitability of the method of Cadoli and Schaerf therefore depends on the number of classical reasoning operations that can be replaced by their approximate counterparts without changing the result of the computation.

In the experiments queries are generated automatically. The system randomly selects a number of concept descriptions from the loaded ontology. These definitions are used as

queries and are reclassified into the subsumption hierarchy. Note that the queries are first randomly selected, then they are used in the experiments with all forms of approximation.

The first experiments were made with the TAMBIS ontology in which we (re)classified 16 concept definitions.[3] Only the approximation method originally suggested by Cadoli and Schaerf [Schaerf and Cadoli, 1995] for $\mathcal{ALE}$ (described in Section 2) was used.

The results of the first experiments are shown in Table 2.1, which is divided into four columns. Each column reports the number of subsumption tests when using a certain form of approximation. The first column reports results for the experiment with normal classification (i.e., without approximation), the second column for $C_i^{\perp}$-approximation, the third column for $C_i^{\top}$-approximation, and the fourth column for a combination of $C_i^{\perp}$- and $C_i^{\top}$-approximation.

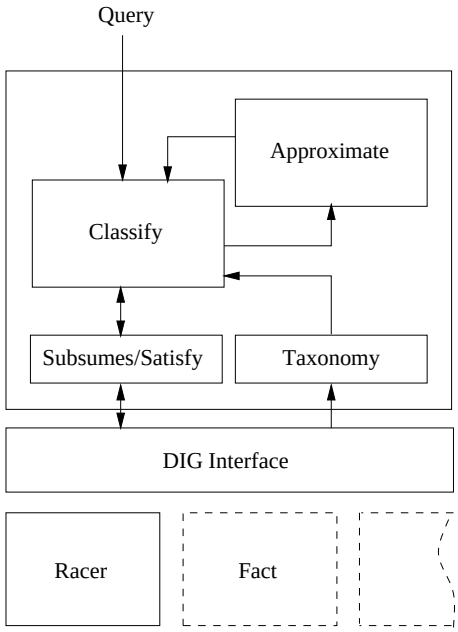

Figure 2.1: Architecture of the experimental setup.

Each column of Table 2.1 is divided into a number of smaller rows and columns. The rows represent the level of the approximation used, where N denotes normal subsumption testing, i.e., without approximation. The columns represent whether the subsumption test resulted in true or false.[4] This distinction is important, because Theorem 1 tells us that only one of those two results will immediately lead to a reduction in complexity, while for the other result approximation has to continue at the next level. This continues until no more approximation steps can be done.

	normal			$C_i^{\perp}$			$C_i^{\top}$			$C_i^{\perp}$&$C_i^{\top}$		
		true	false		true	false		true	false		true	false
Tambis (16)				$C_0^{\perp}$	157	32	$C_0^{\top}$	8	181	$C_0^{\perp}$	157	32
				$C_1^{\perp}$	0	0	$C_1^{\top}$	0	0	$C_0^{\top}$	8	149
	N	24	279	N	24	247	N	16	279	N	16	247

Table 2.1: Subsumption tests for the reclassification of 16 concepts in TAMBIS.

The first column shows that for the reclassification of 16 concepts in the TAMBIS ontology, 24 true subsumption tests and 279 false subsumption tests were needed.

[3] A biochemistry ontology developed in the TAMBIS project [Baker *et al.*, 1998].

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

The second column shows that $C_i^{\perp}$-approximation leads to a change in normal subsumption tests. Compared to the normal case, the number of false subsumption tests are reduced from 279 to 247. However, the 24 true subsumption tests are not reduced. Note that 32 (279 - 247) false subsumption tests are replaced by 157 true $C_0^{\perp}$-subsumption tests and 32 false $C_0^{\perp}$-subsumption tests.[5]

The third column shows that $C_i^{\top}$-approximation also leads to a change in normal subsumption tests, but quite different when compared to $C_i^{\perp}$-approximation. With $C_i^{\top}$-approximation we reduce the true subsumption tests from 24 to 16. However, the 279 false subsumption tests are not reduced. Note that 8 (24 - 16) true subsumption tests are replaced by 8 true $C_0^{\top}$-subsumption tests and 181 false $C_0^{\top}$-subsumption tests. Analogously to $C_i^{\perp}$-approximation, no $C_i^{\top}$-approximation was used when this would not lead to a change in the subsumption expression.

The fourth column shows the combination of $C_i^{\perp}$- and $C_i^{\top}$-approximation by using the approximation sequence $C_0^{\perp}, C_0^{\top}, C_1^{\perp}, C_1^{\top}, ..., C_{n-1}^{\perp}, C_{n-1}^{\top}, normal$. This combination leads to a reduction of normal subsumption tests, which is the combination of the reductions found when using $C_i^{\perp}$- or $C_i^{\top}$-approximation by itself. The true subsumption tests are reduced from 24 to 16 and the false subsumption tests are reduced from 279 to 247. Note that the reduction of 8 (24 - 16) true subsumption tests and 32 (279 - 247) are now replaced by 157 true $C_0^{\perp}$-subsumption tests, 32 false $C_0^{\perp}$-subsumption tests, 8 true $C_0^{\top}$-subsumption tests, and 149 false $C_0^{\top}$-subsumption tests.

2.2.1 Analysis of $C_i^{\perp}$-/$C_i^{\top}$-Approximation

The approximation of concept classification in the TAMBIS ontology using the method of Cadoli and Schaerf reveals at least four points of interest. First, Table 2.1 shows that using $C_i^{\perp}$-approximation can only lead to a reduction of the false subsumption tests and $C_i^{\top}$-approximation can only lead to a reduction of the true subsumption tests. These results could be expected as they follow from Theorem 1 and are reflected by Algoritm 3, 4, and 5. Using Theorem 1 we have the following reasoning steps for $C_i^{\perp}$-approximation:

$$\begin{aligned} \texttt{Query} \not\sqsubseteq \texttt{Concept} &\Leftrightarrow (\texttt{Query} \sqcap \neg\texttt{Concept}) \text{ is satisfiable} \\ &\Leftarrow (\texttt{Query} \sqcap \neg\texttt{Concept})_i^{\perp} \text{ is satisfiable}. \end{aligned}$$

Hence, when $(\texttt{Query} \sqcap \neg\texttt{Concept})_i^{\perp}$ *is not satisfiable*, nothing can be concluded and approximation cannot lead to any gain.

[5] Note that the numbers do not add up. The reason for this is that approximation is not used when there would not be any change in the subsumption expression, i.e., when $C_i^{\perp} = C$ the DL reasoner is not called and no subsumption check for $C_i^{\perp}$ is performed.

Using Theorem 1 we have the following reasoning steps for $C_i^\top$-approximation (cf. Algorithm 3):

$$\begin{aligned}\texttt{Query} \sqsubseteq \texttt{Concept} &\Leftrightarrow (\texttt{Query} \sqcap \neg\texttt{Concept}) \text{ is not satisfiable} \\ &\Leftarrow (\texttt{Query} \sqcap \neg\texttt{Concept})_i^\top \text{ is not satisfiable.}\end{aligned}$$

Hence, when $(\texttt{Query} \sqcap \neg\texttt{Concept})_i^\top$ *is satisfiable*, nothing can be concluded and approximation cannot lead to any gain.

Second, no true approximations are used on a level higher than zero. This is a direct consequence of the TAMBIS ontology containing no nested concept definitions. Further on, we show this to be the case for most ontologies found in practice.

Third, both $C_i^\bot$- and $C_i^\top$-approximation are not applied in all subsumption tests that are theoretically possible. With normal classification 303 (24 + 279) subsumption tests are needed. However, with $C_i^\bot$-approximation in only 189 (157 + 32) cases was approximation actually used. In the remaining 114 (303 - 189) cases approximation had no effect on the concept definitions, i.e., $C_i^\bot = C$, and no test was therefore performed. Hence, in 38% of the subsumption tests, approximation was not used. Similar observations hold for $C_i^\top$-approximation. This observation indicates that $C_i^\bot$-/$C_i^\top$-approximation is not very useful (at least for the TAMBIS ontology) for approximating classification in an ontology.

Fourth, apart from the successful reduction of normal subsumption tests, we must also consider the cost for obtaining the reduction. For example, with $C_i^\bot$-approximation we obtained a reduction in 32 false subsumption tests, i.e., 32 normal false subsumption tests could be replaced by 32 cheaper false $C_0^\bot$-subsumption tests, however it also cost an extra 157 true $C_0^\bot$-subsumption tests that did not lead to any reduction. As nothing can be deduced from these 157 true $C_0^\bot$-subsumption tests, these computations are wasted and reduce the gain obtained with the 32 reduced false subsumption tests considerably. Obviously, these unnecessary true $C_0^\bot$-subsumption tests should be minimized. No final verdict can be made however, because it all depends on the computation time needed to compute the normal subsumption tests and $C_0^\bot$-subsumption tests, but 157 seems rather high. Similar observations hold for $C_i^\top$-approximation.

Analysing the high amount of unnecessary subsumption tests, we discovered a phenomenon, which we call *term collapsing*. We illustrate term collapsing through an example taken from the Wine ontology. Suppose that during a classification the subsumption test `Query` $\sqsubseteq$ `WhiteNonSweetWine` is generated. The definition for `WhiteNonSweetWine` is:

$$\texttt{Wine} \sqcap \exists \text{hasColor}.\{White\} \sqcap \forall \text{hasSugar}.\{OffDry, Dry\}.$$

The subsumption query is first transformed into a satisfiability test, i.e., `Query` $\sqsubseteq$ `WhiteNonSweetWine` $\Leftrightarrow$ `Query` $\sqcap \neg$`WhiteNonSweetWine` *is unsatisfiable*, because $C_i^\bot$-/$C_i^\top$-approximation is defined in terms of satisfiability checking.

The definition of $\neg$`WhiteNonSweetWine` is

$$\begin{aligned} &\equiv \neg(\texttt{Wine} \sqcap \exists \text{hasColor}.\{White\} \sqcap \forall \text{hasSugar}.\{OffDry, Dry\}) \\ &\equiv \neg\texttt{Wine} \sqcup \forall \text{hasColor}.\neg\{White\} \sqcup \exists \text{hasSugar}.\neg\{OffDry, Dry\}. \end{aligned}$$

and therefore the approximation $(\neg\texttt{WhiteNonSweetWine})_0^\top$ is

$$\begin{aligned} &\equiv (\neg\texttt{Wine} \sqcup \forall \text{hasColor}.\neg\{White\} \sqcup \exists \text{hasSugar}.\neg\{OffDry, Dry\})_0^\top \\ &\equiv (\neg\texttt{Wine})_0^\top \sqcup (\forall \text{hasColor}.\neg\{White\})_0^\top \sqcup (\exists \text{hasSugar}.\neg\{OffDry, Dry\})_0^\top \\ &\equiv \neg\texttt{Wine} \sqcup \forall \text{hasColor}.\neg\{White\} \sqcup \top \\ &\equiv \top. \end{aligned}$$

Therefore, approximating the expression `Query` $\sqcap$ $\neg$`WhiteNonSweetWine` results in checking unsatisfiability of $\texttt{Query}_0^\top$, i.e., $(\texttt{Query} \sqcap \neg\texttt{WhiteNonSweetWine})_0^\top \Leftrightarrow \texttt{Query}_0^\top \sqcap \top \Leftrightarrow \texttt{Query}_0^\top$ *is unsatisfiable*. This test most likely fails, because in a consistent ontology `Query` will be satisfiable and as `Query` is more specific than $\texttt{Query}_0^\top$, the latter will be satisfiable.

Analogously, applying $C_i^\bot$-approximation may result in a collapse of the `Query` to $\bot$. This occurs whenever `Query` contains a conjunction with at least one $\exists$-quantifier. In this case, the entire subsumption test is collapsed into checking the satisfiability of $\bot$. As $\bot$ can never be satisfied, this results in an unnecessary subsumption test.

For the TAMBIS ontology we counted the numbers of occurrences of term collapsing in approximated concept expressions. With $C_i^\top$-approximation 65 terms out of 181 collapsed. In other words, 35.9% of the approximated false subsumption tests are obviously not needed and should be avoided. With $C_i^\bot$-approximation it becomes more drastic: 157 terms out of 157 collapsed. If we can avoid term collapsing 100% of the approximated true subsumption tests can be reduced leading to a real improvement in this case. With the combination of $C_i^\bot$- and $C_i^\top$-approximation 190 terms out of 306 collapsed or 62.1% of the approximated subsumption tests are obviously not necessary.

Summarizing, using the proposed approximation method by Cadoli and Schaerf [Schaerf and Cadoli, 1995] on classification queries in the TAMBIS ontology led to many collapsing terms. Furthermore, in only a few cases are expensive subsumption tests replaced by cheaper approximated subsumption test. These results indicate that their approximation method does not fit in practical situations. A different approximation method may provide better approximation.

2.2.2 Further experiments

Although practical results of $C_i^\bot$-/$C_i^\top$-approximation are somewhat disappointing for the TAMBIS ontology, similar experiments were made with other ontologies. Table 2.2 summarizes the results of $C_i^\bot$-/$C_i^\top$-approximation applied to the reclassification of 10 concepts in five other ontologies.

		normal		$C_i^{\perp}$		$C_i^{\top}$		$C_i^{\perp}\&C_i^{\top}$	
		true	false	true	false	true	false	true	false
Dolce (10)	$C_0^{\perp}$	-	-	0	0	-	-	0	0
	$C_0^{\top}$	-	-	-	-	0	0	0	0
	N	10	113	10	113	10	113	10	113
Galen (10)	$C_0^{\perp}$	-	-	0	0	-	-	0	0
	$C_0^{\top}$	-	-	-	-	0	0	0	0
	N	10	12190	10	12190	10	12190	10	12190
Monet (10)	$C_0^{\perp}$	-	-	0	0	-	-	0	0
	$C_0^{\top}$	-	-	-	-	0	0	0	0
	N	20	656	20	656	20	656	20	656
MadCow (10)	$C_0^{\perp}$	-	-	145	0	-	-	145	0
	$C_0^{\top}$	-	-	-	-	5	140	5	140
	N	66	152	66	152	61	152	61	152
Wine (10)	$C_0^{\perp}$	-	-	228	1	-	-	228	1
	$C_0^{\top}$	-	-	-	-	6	223	6	222
	N	33	252	33	251	27	252	27	251

Table 2.2: Number of subsumption tests for reclassification in five ontologies.

For the first three ontologies of Table 2.2, the DOLCE[6], Galen[7], and Monet ontology[8], $C_i^{\perp}$- or $C_i^{\top}$-approximation has no effect. In these three ontologies, $C_i^{\perp}$-/$C_i^{\top}$-approximation does not change any concept expression and therefore no reduction in normal subsumption tests can be obtained. An analysis of these three ontologies shows that the ontologies use some roles and/or attributes, but the $\exists$- and/or $\forall$-quantifiers are very rarely used. For example, the Monet ontology contains 2037 concepts, 34 roles, and 10 attributes. The $\exists$-constructor is only used in 13 definitions (0.64% of all concept definitions). The $\forall$-constructor is only used in 11 cases (0.54% of all concept definitions). Therefore in the ten queries, which are randomly selected, none of the checked concept definitions contains any quantifiers. The $C_i^{\perp}$-/$C_i^{\top}$-approximation seems to be useless for those ontologies.

The next two ontologies of Table 2.2, MadCow[9] and Wine, are somewhat artificial because they are developed for demonstrating the expressive power of DLs rather than for being used in practice. $C_i^{\perp}$-/$C_i^{\top}$-approximation was applied to classification in both ontologies, but this leads to almost no reduction of normal subsumption tests. In the Madcow ontology only 5 true subsumption tests are reduced and in the Wine ontology only 7 subsumption tests are reduced (6 true subsumption tests + 1 false subsumption test).

[6] An ontology for linguistic and cognitive engineering [Masolo *et al.*, 20003].
[7] A medical terminology developed in the Galen project [Rector *et al.*, 1993].
[8] An ontology for mathematical web services [Caprotti *et al.*, 2004].
[9] Ontology about mad cows, part of the OWL Reasoning Examples [OWL, b].

Many more subsumption tests are not reduced. In many cases approximating subsumption tests led to term collapsing and useless subsumption tests.

2.3 Conclusions

We argued that the idea of approximate logical reasoning matches the requirements of the Semantic Web in terms of robustness against errors and the ability to cope with limited resources better than conventional reasoning methods. At the same time, approximate logical inference avoids the problems of many numerical approaches for approximate reasoning that lack a proper interpretation of the numeric value assigned to statements and the problem of acquiring these numbers. We tested a concrete method for approximate logical reasoning in DLs against these claims by applying it to the classification problem on a number of real ontologies. In particular, we used the method to replace subsumption tests by a series of presumably easier tests. We investigated the use of weaker approximations to approximate negative and stronger approximations to approximate positive tests. We showed that in principle both approximations can contribute to a replacement of subsumption tests (compare Section 2.2).

The main result is that the use of the approximation method for DLs proposed by Cadoli and Schaerf is problematic for two reasons:

- A problematic side effect of using the approximation method is the collapsing of concept expression that produce meaningless results. This happens either when terms of a disjunction are replaced by $\top$ or terms of a conjuction are replaced by $\bot$. The former case appears when using the weaker approximation $C_i^{\top}$ on a concept that contains a universal quantifier at the top level of the definition. The latter is an effect of using the stronger approximation $C_i^{\bot}$ for a query concept with existential quantifiers at the top level of the definition. This feature of the approach is quite problematic as it excludes an important class of query concepts from the method, namely translations of conjunctive queries which are mostly translated using nested existential quantifications.

- The experiments showed that only in some cases was the method able to replace subsumption tests. In the other cases like DOLCE, Galen, and Monet no test could be substituted. This observation can be explained by the fact that the approximation method only works on nested expressions that are existentially quantified. Many existing ontologies, however, do not contain concept expressions with nested expressions. The average ontology on the Semantic Web rather uses quite simple concept expressions that, if at all, are of depth one. The approximation method by Cadoli and Schaerf was designed based on theoretical considerations and the approximation strategy was chosen in such a way that it reduces the worst case complexity of the subsumption problem and it does not take practical considerations like the nature of definitions that are likely to be found in ontologies into account.

We conclude that the use of this specific way of approximating subsumption is often not suited for Semantic Web reasoning. Nevertheless, we believe in the general idea of approximate logical reasoning. The goal is to find an approximation strategy that takes the specifics of ontologies into account. A particular problem with the current approach is the reliance on nested definitions. A straightforward way to modify this approach is to find alternative strategies for selecting subexpressions that are to be replaced by $\top$ or $\bot$, or other simpler sub concepts. A good candidate, that will be explored in future work, can use domain knowledge to determine the subset of the vocabulary to be replaced. We could for example first exclude very specific terms and then gradually add more specific ones. This and other options for approximating Semantic Web reasoning will be studied in future research.

Chapter 3

Approximation in ABox Reasoning

by IAN HORROCKS, DANIELE TURI & JEFF Z. PAN

The W3C recommendation OWL is a recently emerged standard for expressing ontologies in the Semantic Web. One of the main features of OWL is that there is a direct correspondence between (two of the three "species" of) OWL and Description Logics (DLs) [Horrocks and Patel-Schneider, 2003].

Unfortunately, while existing techniques for *TBox* reasoning (i.e., reasoning about concepts) seem able to cope with real world ontologies [Horrocks, 1998b, Haarslev and Möller, 2001a], it is not clear if existing techniques for *ABox* reasoning (i.e., reasoning about individuals) will be able to cope with realistic sets of instance data. This difficulty arises not so much from the computational complexity of ABox reasoning, but from the fact that the number of individuals (e.g., annotations) might be extremely large.

In this section, we describe an approach to ABox reasoning that deals with *role-free* ABoxes, i.e., ABoxes that do not contain any axioms asserting role relationships between pairs of individuals. The result, which we call an *Instance Store*, is a system that can deal with very large ABoxes, and is able to provide sound and complete answers to instance retrieval queries (i.e., computing all the instances of a given query concept) over such ABoxes.

Although this approximation may seem a rather severe restriction, the functionality provided by the Instance Store is precisely what is required by many applications, and in particular by applications where ontology based terms are used to describe/annotate and retrieve large numbers of objects. Examples include the use of ontology based vocabulary to describe documents in "publish and subscribe" applications [Uschold *et al.*, 2003], to annotate data in bioinformatics applications [GO,] and to annotate web resources such as web pages [Dill *et al.*, 2003] or web service descriptions [Li and Horrocks, 2003] in Semantic Web applications.

3.1 Instance Store

An ABox $\mathcal{A}$ is role-free if it contains only axioms of the form $x : C$. We can assume without loss of generality that there is exactly one such axiom for each individual as $x : C \sqcup \neg C$ holds in all interpretations, and two axioms $x : C$ and $x : D$ are equivalent to a single axiom $x : (C \sqcap D)$. It is well known that for a role-free ABox, instantiation can be reduced to TBox subsumption [Hollunder, 1996, Tessaris, 2001]; i.e., if $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, and $\mathcal{A}$ is role-free, then $\mathcal{K} \models x : D$ iff $x : C \in \mathcal{A}$ and $\mathcal{T} \models C \sqsubseteq D$. Similarly, if $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and $\mathcal{A}$ is a role-free ABox, then the instances of a concept D could be retrieved simply by testing for each individual x in $\mathcal{A}$ if $\mathcal{K} \models x : D$. This would, however, clearly be very inefficient if $\mathcal{A}$ contained a large number of individuals.

An alternative approach is to add a new axiom $C_x \sqsubseteq D$ to $\mathcal{T}$ for each axiom $x : D$ in $\mathcal{A}$, where C_x is a new atomic concept; we will call such concepts *pseudo-individuals*. Classifying the resulting TBox is equivalent to performing a complete realisation of the ABox: the most specific atomic concepts that an individual x is an instance of are the most specific atomic concepts that subsume C_x and that are not themselves pseudo-individuals. Moreover, the instances of a concept D can be retrieved by computing the set of pseudo-individuals that are subsumed by D.

The problem with this latter approach is that the number of pseudo-individuals added to the TBox is equal to the number of individuals in the ABox, and if this number is very large, then TBox reasoning may become inefficient or even break down completely (e.g., due to resource limits). The basic idea behind the Instance Store is to overcome this problem by using a DL reasoner to classify the TBox and a database to store the ABox, with the database also being used to store a complete realisation of the ABox, i.e., for each individual x, the concepts that x realises (the most specific atomic concepts that x instantiates). The realisation of each individual is computed using the DL (TBox) reasoner when an axiom of the form $x : C$ is added to the Instance Store ABox.

A retrieval query to the Instance Store (i.e., computing the set of individuals that instantiate a query concept) can be answered using a combination of database queries and TBox reasoning. Given an Instance Store containing a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ and a query concept Q, the instances of Q can be computed using the following steps:

1. use the DL reasoner to compute $\mathbf{C}$, the set of most specific atomic concepts in $\mathcal{T}$ that subsume Q, and $\mathbf{D}$, the set of all atomic concepts in $\mathcal{T}$ that are subsumed by Q;
2. use the database to compute A_Q, the set of individuals in $\mathcal{A}$ that realise *some* concept in $\mathbf{D}$, and A_C, the set of individuals in $\mathcal{A}$ that realise *every* concept in $\mathbf{C}$;
3. use the DL reasoner to compute A'_Q, the set of individuals $x \in A_C$ such that $x : B$ is an axiom in $\mathcal{A}$ and B is subsumed by Q;
4. return the answer $A_Q \cup A'_Q$.

Theorem 2 The above procedure is sound and complete for retrieval, i.e., given a concept Q, it returns all and only individuals in $\mathcal{A}$ that are instances of Q.

Proof: For soundness, if x is in $A_Q \cup A'_Q$, then it must be an instance of some concept $C \sqsubseteq Q$ s.t. $C \in \mathbf{D}$, or $x : B$ is an axiom in $\mathcal{A}$ and B is subsumed by Q. In either case, x is an instance of Q.

For completeness, suppose that x is an instance of Q and x is not in the answer. As x is an instance of Q, there must be an axiom $x : B$ in $\mathcal{A}$ such that B is subsumed by Q. Moreover, x is an instance of every concept that subsumes Q, and in particular of every concept in $\mathbf{C}$, but does not realise any concept in $\mathbf{D}$, so it must realise every concept in $\mathbf{C}$. This means that x is in the answer, contradicting our initial assumption. ∎

Note that if Q is equivalent to an atomic concept X, then $\{X\} \subseteq \mathbf{C} \subseteq \mathbf{D}$, and the answer A_Q can be returned without computing A'_Q.

3.2 An Optimised Instance Store

In practice, several refinements to the above procedure are used to improve the performance of the Instance Store. In the first place, as it is potentially costly, we should try to minimise the DL reasoning required in order to compute realisations (when instance axioms are added to the ABox) and to check if individuals in A_C are instances of the query concept (when answering a query).

One way to (possibly) reduce the need for DL reasoning is to avoid repeating computations for "equivalent" individuals, e.g., individuals x_1, x_2 where $x_1 : C_1$ and $x_2 : C_2$ are ABox axioms, and C_1 is equivalent to C_2 (concepts C and D are equivalent, written $C \equiv D$, iff $C \sqsubseteq D$ and $D \sqsubseteq C$). As checking for semantic equivalence between two concepts would require DL reasoning (which we are trying to avoid), the optimised Instance Store only checks for syntactic equality using a database lookup.[1] Individuals are grouped into equivalence sets, where each individual in the set is asserted to be an instance of a syntactically identical concept, and only one representative of the set is added to the Instance Store ABox as an instance of the relevant concept. When answering queries, each individual in the answer is replaced by its equivalence set.

Similarly, we can avoid repeated computations of sub and super-concepts for the same concept (e.g., when repeating a query) by caching the results of such computations in the database.

Finally, the number and complexity of database queries also has a significant impact on the performance of the Instance Store. In particular, the computation of A_Q can be

[1] The chances of detecting equivalence via syntactic checks could be increased by transforming concepts into a syntactic normal form, as is done by optimised DL reasoners [Horrocks, 2003], but this additional refinement has not yet been implemented in the Instance Store.

costly as $\mathbf{D}$ (the set of concepts subsumed by the query concept Q) may be very large. One way to reduce this complexity is to store not only the most specific concepts instantiated by each individual, but to store *every* concept instantiated by each individual. As most concept hierarchies are relatively shallow, this does not increase the storage requirement too much, and it greatly simplifies the computation of A_Q: it is only necessary to compute the (normally) much smaller set $\mathbf{D}'$ of most general concepts subsumed by Q, and to query the database for individuals that instantiate some member of $\mathbf{D}'$. On the other hand, the computation of A_C is slightly more complicated as A_Q must be subtracted from the set of individuals that instantiate every concept in $\mathbf{C}$. Empirically, however, the saving when computing A_Q seems to far outweigh the extra cost of computing A_C.

3.3 Implementation

We have implemented the Instance Store using a component based architecture that is able to exploit existing DL reasoners and databases. The core component is a Java application that implements an API and, for test purposes, a simple user interface. The Instance Store connects to a DL reasoner via the DIG interface [Bechhofer, 2003], and can therefore use one of several DIG compliant reasoners, including FaCT [Horrocks, 1998b] and RACER [Haarslev and Möller, 2001c]. It also connects to a DB via standard interfaces, and has been tested with HSQL[2], MySQL[3] and Oracle[4].

```
initialise(Reasoner reasoner, Database db, TBox t)
assert(Individual i, Description D)
remove(Individual i)
retrieve(Description Q): Set⟨Individual⟩
```

Figure 3.1: Instance Store basic functionality

The basic functionality of the Instance Store is illustrated by Figure 3.1. The four basic operations are `initialise`, which loads a TBox into the DL reasoner, classifies the TBox and establishes a connection to the database; `assert`, which adds an axiom $i : D$ to the Instance Store; `remove`, which removes any axiom of the form $i : C$ (for some concept C) from the Instance Store; and `retrieve`, which returns the set of individuals that instantiate a query concept Q. As the Instance Store ABox can only contain one axiom for each individual, asserting $i : D$ when $i : C$ is already in the ABox is equivalent to first removing i and then asserting $i : (C \sqcap D)$.

In the current implementation, we make the simplifying assumption that the TBox itself does not change. Extending the implementation to deal with monotonic extensions of the TBox would be relatively straightforward, but deleting information from the TBox might require (in the worst case) all realisations to be recomputed.

[2]http://hsqldb.sourceforge.net/
[3]http://www.mysql.com/
[4]http://www.oracle.com/

Our experiments can be found in deliverable D2.5.2 "Report on Query Language Design and Standardisation". These show that the Instance Store provides stable and effective reasoning for role-free ABoxes, even those containing very large numbers of individuals. In contrast, full ABox reasoning using the RACER system exhibited accelerating performance degradation with increasing ABox size, and was not able to deal with the larger ABoxes used in this test.[5]

3.4 Discussion

Our experiments show that the Instance Store provides stable and effective reasoning for role-free ABoxes, even those containing very large numbers of individuals. In contrast, full ABox reasoning using the RACER system exhibited accelerating performance degradation with increasing ABox size, and was not able to deal with the larger ABoxes used in this test. (It may be possible to fix this problem by changing system parameters, but we had no way to investigate this.) The pseudo-individual approach to role-free ABox reasoning was more promising, and may be worth further investigation. It does not, however, have the Instance Store's advantage of ABox persistence, and it appears to be less likely to scale to even larger ABoxes: it does not cope well with large answer sets, and is inherently limited by the fact that DL reasoners (at least in current implementations) keep the entire TBox in memory. Moreover, it is not clear how the pseudo-individual approach could be extended to deal with ABoxes that are not role-free.

The acceptability of the Instance Store's performance would obviously depend on the nature of the application and the characteristics of the KB and of typical queries. It is likely that the performance of the Instance Store can be substantially improved simply by dealing with constant factors such as communication overheads—in the current implementation, communication overheads between the Instance Store and the DL reasoner account for nearly half the time taken to answer queries that require significant amounts of DL reasoning to compute the answer (i.e., when I_2 is large). It may also be possible to improve the performance of the database, e.g., using techniques such as indexing and clustering, or by reformulating queries.

As well as dealing with the above mentioned performance bottlenecks, future work will include the investigation of additional optimisations and enhancements. Possible optimisations include *semantic indexing feedback*—adding new indexing concepts to the ontology for the purpose of query optimisation; *description canonicalisation*—canonicalising the descriptions passed to the Instance Store, so that equivalent descriptions can be more effectively identified; *cardinality estimation*—estimating the cardinality of the result (and in particular of I_2) before executing a query, and giving users the chance to refine queries if the cost of answering them is likely to be very high;

[5] It may be possible to fix this problem by changing system parameters, but we had no way to investigate this.

and *result caching*—caching the results of queries and of DL subsumption tests in order to avoid DL reasoning when answering subsequent queries. Possible enhancements include providing a more sophisticated query interface with support for, e.g., conjunctive queries [Tessaris, 2001].

As discussed in [Pan *et al.*, 2004], we are currently engaged in extending the Instance Store to deal with ABoxes that are not role-free. The impact that this will have on performance is likely to be heavily dependent on the structure of the given ABox. In particular, the Instance Store is not likely to perform well with ABoxes that result in highly non-deterministic precompletions. ABoxes that are highly interconnected and/or contain many cyclical connections are also likely to have an adverse affect on performance. An evaluation of the effectiveness of the extended Instance Store will therefore have to wait for the completion of the prototype, and on the development of application ontologies containing large numbers of individuals—currently these are in rather short supply, but we hope that development of such ontologies will be encouraged by the existence of the extended Instance Store.

Chapter 4

Towards Resolution-Based Approximate Reasoning for OWL-DL

by PASCAL HITZLER & BORIS MOTIK

We propose a new technique for approximate ABox reasoning with OWL-DL ontologies. It comes as a side-product of recent research results on the relationship between OWL-DL and disjunctive datalog [Hustadt *et al.*, 2004a, Hustadt *et al.*, 2004c, Hustadt *et al.*, 2004b, Motik *et al.*, 2004]. Essentially, it relies on a new transformation of OWL-DL ontologies into negation-free disjunctive datalog, and on the idea of performing standard resolution over disjunctive rules by treating them as if they were non-disjunctive ones.

4.1 Introduction and Motivation

Knowledge representation and reasoning on the Semantic Web is done by means of ontologies. While the quest for suitable ontology languages is still ongoing, OWL [Patel-Schneider *et al.*, 2002] 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 [Baader *et al.*, 2003a] and can thus be considered to be fragments of first-order predicate logic.

OWL ontologies can be understood to consist of two parts, one intensional, the other extensional. In description logics terminology, the intensional part consists of a TBox and an RBox, and contains knowledge about concepts (called *classes*), relations between concepts (called *roles*), and information about hierarchical dependencies among classes and among roles. The extensional part consists of an ABox, and contains knowledge about entities and how they relate to the classes and roles from the intensional part. For the Semantic Web, the ABox corresponds to *annotations*, i.e. pointers delivered e.g. with web pages, indicating how items occuring on the web pages relate to the intensional

knowledge.

With an estimated 25 million active websites today and correspondingly more webpages, it is apparent that reasoning on the Semantic Web will have to deal with very large ABoxes. Complexity of ABox reasoning — also called *data complexity* — thus measures complexity in terms of ABox size only, while considering the intensional part of the ontology to be of constant size. For the different OWL variants, data complexity is at least NP-hard, which indicates that it will not scale well in general. Methods are therefore being sought to cope with large ABoxes in an approximate manner.

The approach which we propose is based on the fact that data complexity is polynomial for non-disjunctive datalog. We utilise recent research results [Hustadt *et al.*, 2004a, Hustadt *et al.*, 2004c, Hustadt *et al.*, 2004b, Motik *et al.*, 2004] which allow the transformation of OWL-DL ontologies into disjunctive datalog. 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. We are also able to give a characterization of the resulting approximate inference by means of standard methods from logic programming semantics.

This chapter is structured as follows. We first introduce formal terminology and notation for OWL-DL, a part of which can safely be skipped by any reader who is familiar with OWL-DL and description logic syntax. We also briefly review Datalog and SLD-Resolution. Then, in Section 4.3, we explain how OWL ontologies can be transformed into disjunctive datalog. In Section 4.4 we introduce the new approximate SLD-resolution procedure which we propose. A short analysis of the new procedure will be followed by conclusions in Section 4.5.

4.2 Preliminaries

4.2.1 OWL-DL Syntax and Semantics

OWL-DL is a syntactic variant of the $\mathcal{SHOIN}(\mathbf{D})$ description logic [Horrocks and Patel-Schneider, 2004]. Hence, although several XML and RDF syntaxes for OWL-DL exist, it will be convenient to use the traditional description logic notation since it is more compact. For the correspondence between this notation and various OWL-DL syntaxes, see [Horrocks and Patel-Schneider, 2004].

$\mathcal{SHOIN}(\mathbf{D})$ supports reasoning with concrete datatypes, such as strings or integers. For example, it is possible to define a minor as a person whose age is less than or equal to 18 in the following way: $Minor \equiv Person \sqcap \exists age.\ \leq_{18}$. Instead of axiomatizing concrete datatypes in logic, $\mathcal{SHOIN}(\mathbf{D})$ employs an approach similar to [Baader and Hanschke, 1991], where the properties of concrete datatypes are encapsulated in so-called *concrete domains*. A *concrete domain* is a pair $(\triangle_{\mathbf{D}}, \Phi_{\mathbf{D}})$, where $\triangle_{\mathbf{D}}$ is

an interpretation domain and $\Phi_{\mathbf{D}}$ is a set of concrete domain predicates with a predefined arity n and an interpretation $d^{\mathbf{D}} \subseteq \triangle_{\mathbf{D}}^n$. An *admissible* concrete domain $\mathbf{D}$ is equipped with a decision procedure for checking satisfiability of finite conjunctions over concrete predicates. Satisfiability checking of admissible concrete domains can successfully be combined with logical reasoning for many description logics [Lutz, 2003].

We use a set of concept names N_C, sets of abstract and concrete individuals N_{I_a} and N_{I_c}, respectively, and sets of abstract and concrete role names N_{R_a} and N_{R_c}, respectively. An *abstract role* is an abstract role name or the inverse S^- of an abstract role name S (concrete roles do not have inverses). In the following, we assume that $\mathbf{D}$ is an admissible concrete domain.

An *RBox* $\mathcal{R}$ consists of a finite set of transitivity axioms $\mathsf{Trans}(R)$, and role inclusion axioms of the form $R \sqsubseteq S$ and $T \sqsubseteq U$, where R and S are abstract roles, and T and U are concrete roles. The reflexive-transitive closure of the role inclusion relationship is denoted with $\sqsubseteq^*$. A role not having transitive subroles (w.r.t. $\sqsubseteq^*$, for a full definition see [Horrocks *et al.*, 2000]) is called a *simple* role.

The set of $\mathcal{SHOIN}(\mathbf{D})$ *concepts* is defined by the following syntactic rules, where A is an atomic concept, R is an abstract role, S is an abstract simple role, $T_{(i)}$ are concrete roles, d is a concrete domain predicate, a_i and c_i are abstract and concrete individuals, respectively, and n is a non-negative integer:

$$\begin{array}{rcl} C & \rightarrow & A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \geq n\, S \mid \leq n\, S \mid \{a_1, \ldots, a_n\} \mid \\ & & \mid \geq n\, T \mid \leq n\, T \mid \exists T_1, \ldots, T_n.D \mid \forall T_1, \ldots, T_n.D \\ D & \rightarrow & d \mid \{c_1, \ldots, c_n\} \end{array}$$

A *TBox* $\mathcal{T}$ consists of a finite set of concept inclusion axioms $C \sqsubseteq D$, where C and D are concepts; an *ABox* $\mathcal{A}$ consists of a finite set of concept and role assertions and individual (in)equalities $C(a)$, $R(a, b)$, $a \approx b$, and $a \not\approx b$, respectively. A $\mathcal{SHOIN}(\mathbf{D})$ *knowledge base* $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ consists of a TBox $\mathcal{T}$, an RBox $\mathcal{R}$, and an ABox $\mathcal{A}$.

The $\mathcal{SHIQ}(\mathbf{D})$ description logic is obtained from $\mathcal{SHOIN}(\mathbf{D})$ by disallowing nominal concepts of the form $\{a_1, \ldots, a_n\}$ and $\{c_1, \ldots, c_n\}$, and by allowing qualified number restrictions of the form $\geq n\, S.C$ and $\leq n\, S.C$, for C a $\mathcal{SHIQ}(\mathbf{D})$ concept and S a simple role.

Instead of using a direct model-theoretic semantics for $\mathcal{SHOIN}(\mathbf{D})$ [Horrocks *et al.*, 2000], we present an equivalent semantics by translation into multi-sorted first-order logic. We do this because our approach hinges heavily on the results from [Hustadt *et al.*, 2004a, Hustadt *et al.*, 2004b], where this approach had been taken. To separate the interpretations of the abstract and the concrete domain, we introduce the sorts a and c, and use the notation x^{c} and f^{c} to denote that x and f are of sort c. We translate each atomic concept into a unary predicate of sort a, each n-ary concrete domain predicate into a predicate with arguments of sort c, and each abstract (concrete)

Mapping Concepts to FOL	
$\pi_y(\top, X)=\top$	$\pi_y(\bot, X)=\bot$
$\pi_y(A, X)=A(X)$	$\pi_y(\neg C, X)=\neg\pi_y(C, X)$
$\pi_y(C \sqcap D, X)=\pi_y(C, X) \wedge \pi_y(D, X)$	$\pi_y(C \sqcup D, X)=\pi_y(C, X) \vee \pi_y(D, X)$
$\pi_y(\forall R.C, X)=\forall y : R(X, y) \rightarrow \pi_x(C, y)$	$\pi_y(\exists R.C, X)=\exists y : R(X, y) \wedge \pi_x(C, y)$
$\pi_y(\{a_1 \ldots, a_n\}, X)=$	$X \approx a_1 \vee \ldots \vee X \approx a_n$
$\pi_y(\leq n\, R.C, X)=$	$\forall y_1, \ldots, y_{n+1} : \bigwedge R(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \rightarrow \bigvee y_i \approx y_j$
$\pi_y(\geq n\, R.C, X)=$	$\exists y_1, \ldots, y_n : \bigwedge R(X, y_i) \wedge \bigwedge \pi_x(C, y_i) \wedge \bigwedge y_i \not\approx y_j$
$\pi_y(\forall T_1, \ldots, T_m.d, X)=$	$\forall y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}} : \bigwedge T_i(X, y_i^{\mathsf{c}}) \rightarrow d(y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
$\pi_y(\exists T_1, \ldots, T_m.d, X)=$	$\exists y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}} : \bigwedge T_i(X, y_i^{\mathsf{c}}) \wedge d(y_1^{\mathsf{c}}, \ldots, y_m^{\mathsf{c}})$
$\pi_y(\leq n\, T, X)=$	$\forall y_1^{\mathsf{c}}, \ldots, y_{n+1}^{\mathsf{c}} : \bigwedge T(X, y_i^{\mathsf{c}}) \rightarrow \bigvee y_i^{\mathsf{c}} \approx y_j^{\mathsf{c}}$
$\pi_y(\geq n\, T, X)=$	$\exists y_1^{\mathsf{c}}, \ldots, y_n^{\mathsf{c}} : \bigwedge T(X, y_i^{\mathsf{c}}) \wedge \bigwedge y_i^{\mathsf{c}} \not\approx y_j^{\mathsf{c}}$
Mapping Axioms to FOL	
$\pi(C(a))=\pi_y(C, a)$	$\pi(R(a, b))=R(a, b)$
$\pi(a \approx b))=a \approx b$	$\pi(a \not\approx b))=a \not\approx b$
$\pi(C \sqsubseteq D)=$	$\forall x : \pi_y(C, x) \rightarrow \pi_y(D, x)$
$\pi(R \sqsubseteq S)=$	$\forall x, y : R(x, y) \rightarrow S(x, y)$
$\pi(\mathsf{Trans}(R))=$	$\forall x, y, z : R(x, y) \wedge R(y, z) \rightarrow R(x, z)$
Mapping KB to FOL	
$\pi(KB)=$	$\bigwedge_{R \in N_R} \forall x, y : R(x, y) \leftrightarrow R^-(y, x) \wedge \bigwedge_{\alpha \in KB_{\mathcal{R}} \cup KB_{\mathcal{T}} \cup KB_{\mathcal{A}}} \pi(\alpha)$
where X is a meta variable and is substituted by the actual variable and π_x is defined as π_y by substituting x and x_i for all y and y_i, respectively.	

Table 4.1: Translation of $\mathcal{SHOIN}(\mathbf{D})$ into FOL

role into a binary predicate of sort $\mathsf{a} \times \mathsf{a}$ ($\mathsf{a} \times \mathsf{c}$). The translation operator π is presented in Table 4.1.

4.2.2 Datalog and SLD-Resolution

A (*definite* or *negation-free*) *disjunctive logic program* P consists of a finite set of *clauses* or *rules* of the form

$$\forall x_1 \ldots \forall x_n.(H_1 \vee \cdots \vee H_m \leftarrow A_1 \wedge \cdots \wedge A_k),$$

commonly written as

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

where $x_1, \ldots, x_n$ are exactly all variables occuring in $H_1 \vee \cdots \vee H_m \leftarrow A_1 \wedge \cdots \wedge A_k$, and all H_i and A_j are atoms over some given first-order signature Σ. The disjunction $H_1 \vee \cdots \vee H_m$ is called the *rule head*, and the conjunction $A_1 \wedge \cdots \wedge A_k$ is called the *rule body*. The set of all ground instances of atoms defined over Σ is called the *Herbrand base* of P and is denoted by B_P. The set of all ground instances of rules in P is denoted by $\mathsf{ground}(P)$. A rule is said to be *non-disjunctive* if $m = 1$. It is called a *fact* if $k = 0$. We abstract from the order of the atoms in the heads respectively bodies; it is not important for our results. A disjunctive logic program is called a *Datalog* program if it does not contain function symbols.

Note that we do not consider a logic program to come with one specific semantics. Some people for example associate Datalog with the minimal model semantics only. For our treatment, Datalog and logic programs are defined via syntax only. We do not specify a specific semantics because in the following we will discuss *different* semantics for logic programs in their relation to proof procedures. One of the semantics we will consider is the semantics coming from interpreting logic programs as a set of first order formulae, and in this case we use $\models$ to denote entailment in first-order predicate logic.

SLD-resolution (see e.g. [Lloyd, 1988]) is an efficient top-down query-answering technique for programs consisting of non-disjunctive rules, and has been implemented and successfully applied in standard Prolog systems (e.g. [SWI-Prolog, 2004]). In this framework, a ground atom can be derived from a program if and only if it is true in the least (and thus in all) Herbrand models of the program.

In the following, we mean by a *conjunctive query* simply a conjunction $B_1 \wedge \cdots \wedge B_n$ of atoms. The query is called *ground* if it does not contain any variables.

Given a conjunctive query $B_1 \wedge \cdots \wedge B_n$, an *SLD-resolution step* on the atom B_i with a non-disjunctive rule $H \leftarrow A_1, \ldots, A_k$ produces a conjunctive query

$$B_1\theta \wedge \cdots \wedge B_{i-1}\theta \wedge A_1\theta \wedge \cdots \wedge A_k\theta \wedge B_{i+1}\theta \wedge \cdots \wedge B_n\theta$$

where θ is the most general unifier of B_i and H. An *SLD-refutation* of a conjunctive query $B_1 \wedge \cdots \wedge B_n$ in a non-disjunctive program P is a finite sequence of conjunctive queries $Q_0, \ldots, Q_n$, where (*i*) $Q_0 = B_1 \wedge \cdots \wedge B_n$, (*ii*) each Q_i with $i > 0$ is obtained from Q_{i-1} by an SLD-resolution step with some rule from P on some literal B_i, and (*iii*) $Q_n = \Box$, i.e. the conjunctive query Q_n does not contain any literals. If an SLD-refutation of $B_1 \wedge \cdots \wedge B_n$ in P exists, we write $P \vdash B_1 \wedge \cdots \wedge B_n$.

One of the fundamental results in logic programming states that $A \in B_P$ can be proven by SLD-resolution if and only if A is a logical consequence of P, i.e. if and only if A is true in the least Herbrand model of P:

Theorem 3 ([Apt and van Emden, 1982, Lloyd, 1988]) For a ground conjunctive query $B_1 \wedge \cdots \wedge B_n$ and a non-disjunctive program P, $P \vdash B_1 \wedge \cdots \wedge B_n$ if and only if $P \models B_1 \wedge \cdots \wedge B_n$. In other words, entailment of ground conjunctive queries under SLD-resolution is entailment in predicate logic.

SLD-resolution also allows to derive answers to non-ground queries: For a conjunctive (and not necessarily ground) query Q there exist an SLD-refutation if and only if $P \models \exists x_1 \ldots \exists x_n.Q$, where $x_1, \ldots, x_n$ are the variables occuring in Q. By keeping track of the most general unifiers used in the process, it is also possible to obtain bindings for (some of) the x_i in the form of (answer) substitutions θ, such that $P \models \exists y_1 \ldots \exists y_k(Q\theta)$, where the y_i are exactly those variables occurring in $Q\theta$. In order to keep our exhibition focused, we will only deal with ground queries.

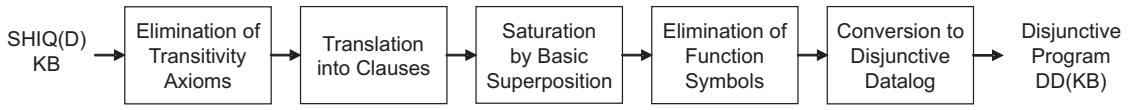

Figure 4.1: Algorithm for Reducing $\mathcal{SHIQ}(\mathbf{D})$ to Datalog Programs

4.3 Reducing OWL-DL Knowledge Bases to Disjunctive Datalog Programs

Our approach is based on reducing the description logic knowledge base to a disjunctive logic program which entails the same set of ground facts as the original knowledge base. The full presentation of the translation with corresponding proofs of its correctness are technically involved and lengthy. Here, we just provide an overview of the procedure, without going into details. For a complete presentation we direct the interested reader to [Hustadt *et al.*, 2004a, Hustadt *et al.*, 2004b].

Our approach does not support all of $\mathcal{SHOIN}(\mathbf{D})$ since it does not support nominals: to the best of our knowledge, no decision procedure has yet been implemented for $\mathcal{SHOIN}(\mathbf{D})$: The combination of nominals, inverse roles, and number restriction is known to be difficult to handle, which is confirmed by the increase in complexity from EXPTIME to NEXPTIME [Tobies, 2001]. Complexity, in this case, is *combined complexity*, measured in the overall size of the knowledge base.

Let KB be a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base. The reduction of KB to a disjunctive datalog program $\mathsf{DD}(KB)$ can be computed by an algorithm schematically presented in Figure 4.1. We next explain each step of the algorithm.

Elimination of Transitivity Axioms. Our core algorithms cannot handle transitivity axioms, basically because in their first-order logic formulation they involve three variables. However, we can eliminate transitivity axioms by encoding KB into an equisatisfiable knowledge base $\Omega(KB)$. Roughly speaking, for each transitive role S, each role $S \sqsubseteq^* R$, and each concept $\forall R.C$ occurring in KB, it is sufficient to add an axiom $\forall R.C \sqsubseteq \forall S.(\forall S.C)$. Intuitively, this axiom propagates all relevant concept constraints through transitive roles.

Translation into Clauses. The next step is to translate $\Omega(KB)$ into clausal first-order logic. We first use π as defined in Table 4.1 and then transform the result $\pi(\Omega(KB))$ into clausal form using *structural transformation* to avoid exponential blow-up [Nonnengart and Weidenbach, 2001]. We call the result $\Xi(KB)$.

Saturation by Basic Superposition. We next saturate the RBox and TBox clauses of $\Xi(KB)$ by basic superposition [Bachmair *et al.*, 1995] — a clausal calculus optimized for theorem proving with equality. In this key step of the reduction, we compute all non-ground consequences of KB. We can prove that saturation terminates because application of each rule of basic superposition produces a clause with at most one variable and with functional terms of depth at most two. This yields an exponential bound on the number of clauses we can compute, and thus an exponential time complexity bound for our translation algorithm so far.

Elimination of Function Symbols. Saturation of RBox and TBox of $\Xi(KB)$ computes all non-ground consequences of KB. If we add ABox assertions to this saturated clause set, all "further" inferences by basic superposition will produce only ground clauses. Moreover, the resulting ground clauses contain only ground functional terms of depth one. Hence, it is possible to simulate each functional term $f(a)$ with a new constant a_f. For each function symbol f, we introduce a binary predicate S_f, and for each individual a, we add an assertion $S_f(a, a_f)$. Finally, if a clause contains the term $f(x)$, we replace it with a new variable x_f and add the literal $\neg S_f(x, x_f)$, as in the following example:

$$\neg C(x) \vee D(f(x)) \Rightarrow \neg S_f(x, x_f) \vee \neg C(x) \vee D(x_f)$$

We denote the resulting function-free set of clauses with $\mathsf{FF}(KB)$. In [Hustadt *et al.*, 2004a], we show that each inference step of basic superposition in $\Xi(KB)$ can be simulated by an inference step in $\mathsf{FF}(KB)$, and vice versa. Hence, KB and $\mathsf{FF}(KB)$ are equisatisfiable.

Conversion to Disjunctive Datalog. Since $\mathsf{FF}(KB)$ does not contain functional terms and all its clauses are safe, we can rewrite each clause into a negation- and function-free disjunctive rule. We use $\mathsf{DD}(KB)$ for the result of this rewriting.

The following theorem summarizes the properties of our algorithm (we use $\models_c$ for cautious entailment in disjunctive datalog, which coincides on ground facts with first-order entailment for negation-free datalog programs [Eiter *et al.*, 1997a]):

Theorem 4 ([Hustadt *et al.*, 2004a]) Let KB be an $\mathcal{SHIQ}(\mathbf{D})$ knowledge base, defined over an admissible concrete domain $\mathbf{D}$, such that satisfiability of finite conjunctions over $\Phi_{\mathbf{D}}$ can be decided in deterministic exponential time. Then the following claims hold:

1. KB is unsatisfiable if and only if $\mathsf{DD}(KB)$ is unsatisfiable.
2. $KB \models \alpha$ if and only if $\mathsf{DD}(KB) \models_c \alpha$, for α of the form $A(a)$ or $S(a, b)$, A an atomic concept, and S a simple role.

3. $KB \models C(a)$ if and only if $\mathsf{DD}(KB \cup \{C \sqsubseteq Q\}) \models_c Q(a)$, for C a non-atomic concept, and Q a new atomic concept.

4. Let $|KB|$ be the length of KB with numbers in number restrictions coded in unary. The number of rules in $\mathsf{DD}(KB)$ is at most exponential in $|KB|$, the number of literals in each rule is at most polynomial in $|KB|$, and $\mathsf{DD}(KB)$ can be computed in time exponential in $|KB|$.

For our approximate reasoning approach, the first two steps of the translation may suffice in many cases. This would avoid the worst-case exponential complexity of subsequent steps, but comes at a price: The resulting program would contain function symbols, which may cause proof-search to fail to terminate in some cases. Hence the procedure would be incomplete in general.

4.4 Approximate Resolution

In the previous section we have shown how $\mathcal{SHIQ}(\mathbf{D})$ ontologies can be translated to (negation- and function-free) disjunctive datalog. In Section 4.2.2 we have reviewed SLD-resolution as deduction procedure for non-disjunctive datalog. In this section, we will show how to deal with disjunctive rules in a non-disjunctive fashion.

4.4.1 Approximate SLD-Resolution

Having to deal with disjunctive heads results in a considerable increase in reasoning complexity. We propose the following approximate reasoning technique to avoid this increase. Given a conjunctive query $B_1 \wedge \cdots \wedge B_n$, an *approximate SLD-resolution step* on the atom B_i with a disjunctive rule $H_1 \vee \cdots \vee H_m \leftarrow A_1, \ldots, A_k$ is a conjunctive query

$$B_1\theta \wedge \cdots \wedge B_{i-1}\theta \wedge A_1\theta \wedge \cdots \wedge A_k\theta \wedge B_{i+1}\theta \wedge \cdots \wedge B_n\theta$$

such that θ is the most general unifier of B_i and some H_j. *Approximate SLD-refutation* is defined analogously to SLD-refutation, where approximate SLD-resolution steps are used instead of (usual) SLD-resolution steps.

It is necessary to pursue the question of what notion of entailment underlies the approximate reasoning technique we propose. For this purpose, we need the following notion, which is derived from standard notions in logic programming.

Definition 5 (cf. [Apt *et al.*, 1988, Fages, 1994, Hitzler and Wendt, 2005]) A model M of a disjunctive program P is called *well-supported* if there exists a function $l : B_P \to \mathbb{N}$ such that for each $A \in M$ there exists a rule $A \vee H_1 \vee \cdots \vee H_m \leftarrow A_1, \ldots, A_k$ in $\mathsf{ground}(P)$ with $M \models A_i$ and $l(A) > l(A_i)$ for all i and k.

Definition 5 is a straightforward adaptation of the notion of well-supported model for non-disjunctive programs, as given in [Fages, 1994]. For non-disjunctive (negation-free) programs, the well-supported models are exactly the minimal ones, but this is not in general the case for disjunctive programs: just consider the program consisting of the single rule $p \vee q \leftarrow$. Then $\{p, q\}$ is a well-supported model, but is not minimal. To our knowledge, well-supported models have not been studied for disjunctive programs before.

Lifted appropriately to (non-disjunctive) programs with negation, well-supported models also provide an alternative means for characterizing the well-known stable models. This was done in [Fages, 1994], but we will not need this for our purposes. Stable models [Gelfond and Lifschitz, 1991] provide the base for the most popular non-monotonic reasoning paradigm called *Answer Set Programming*, of which the two most prominent implementations are DLV and SMODELS [Eiter *et al.*, 1997b, Simons *et al.*, 2002]. Our results thus stand well within this well-established tradition.

It is apparent that $A \in B_P$ follows from a program P by approximate SLD-resolution if and only if it is true in at least one well-supported model of P. This is called *brave reasoning with well-supported models*.

Proposition 6 Entailment of ground conjunctive queries under approximate SLD-resolution is brave reasoning with well-supported models.

As an example, consider the (propositional) program consisting of the two rules $p \vee q \leftarrow$ and $r \leftarrow p \wedge q$. Its minimal models are $\{q\}$ and $\{p\}$ so r is not bravely entailed by reasoning with minimal models. However all of $\{q\}$, $\{p\}$, $\{p, q\}$ and $\{p, q, r\}$ are well-supported models, so r is bravely entailed by reasoning with well-supported models.

There is an alternative way of formalizing approximate SLD-resolution using a modified notion of *split program* [Sakama and Inoue, 1994], as follows. Given a rule

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

define the *derived split rules* as the following:

$$\begin{aligned} H_1 &\leftarrow A_1, \ldots, A_k \\ &\vdots \\ H_m &\leftarrow A_1, \ldots, A_k. \end{aligned}$$

For a given disjunctive program P define its *split program* P' to be the collection of all split rules derived from rules in P. Approximate SLD-resolution on P is obviously identical to SLD-resolution over P'.

Minimal models are well-supported. This can be seen for example from the following result which was obtained along the lines of research laid out in [Hitzler, 2003, Hitzler and Wendt, 2005].

Theorem 7 ([Knorr, 2003]) Let P be a disjunctive program. Then a model M of P is a minimal model of P if and only if there exists a function $l : B_P \rightarrow \mathbb{N}$ such that for each A which is true in M there exists a rule $A \vee H_1 \vee \cdots \vee H_m \leftarrow A_1, \ldots, A_k$ in $\mathsf{ground}(P)$ with $M \models A_i$, $M \not\models H_k$ and $l(A) > l(A_i)$ for all i and k.

We hence have the following result, noting that $P \models Q$ for any ground conjunctive query Q and program P if and only if Q is true in all minimal models of P.

Proposition 8 Let P be a (possibly disjunctive) program and Q be a ground conjunctive query with $P \models Q$. Then there exists an approximate SLD-refutation for Q.

We remark that for negation-free programs minimal models coincide with stable models or *answer sets* [Gelfond and Lifschitz, 1991], as in the currently evolving *Answer Set Programming Systems*, of which the two most prominent implementations are DLV and SMODELS [Eiter *et al.*, 1997b, Simons *et al.*, 2002], as already mentioned.

4.4.2 Approximate Resolution for OWL-DL

Our proposal is based on the idea of converting a given OWL-DL knowledge base into a (possibly function-free) definite disjunctive logic program, and then to apply approximate resolution for ABox reasoning.

In order to be able to deal with all of OWL-DL, we need to add a preprocessing step to get rid of nominals. 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 4.3 in order to obtain a negation-free disjunctive datalog program.
3. Apply approximate SLD-resolution for query-answering.

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. From our discussions, we can conclude the following properties of approximate ABox reasoning for $\mathcal{SHIQ}(\mathbf{D})$.

- It is complete with respect to first-order predicate logic semantics.
- It is sound and complete with respect to brave reasoning with well-supported models.
- Data complexity of our approach is polynomial.

4.5 Conclusions

In a nutshell, our proposed procedure approximates reasoning by disregarding non-Horn features of OWL-DL ontologies. We argue that this is a reasonable approach to approximate reasoning with OWL-DL in particular because many — if not most — of the currently existing ontologies fall in the Horn fragment of OWL-DL anyway. A short survey in [Volz, 2004] substantiates this claim.

Our approach provides ABox reasoning with polynomial time complexity. While it is complete, it is also unsound with respect to first-order logic. We have shown, however, that the inference underlying our approach can be characterized using standard methods from the area of non-monotonic reasoning.

The checking whether a conjunctive query is a predicate logic consequence of a (negation-free) disjunctive logic program P amounts to checking whether the query is valid in *all* minimal models of P, i.e. corresponds to *cautious* reasoning with minimal models. Theorem 7 suggests how an anytime algorithm for this might be obtained: after performing approximate SLD-resolution, it remains to be checked whether there is any (ground instance of a) rule used in the refutation of the query, which has an atom A in its head besides the one used in the refutation and such that A is (cautiously) entailed by the program. Such an algorithm might then first find a brave proof of a query, and then substantiate this proof by subsequent calculations. We note again that this approach does not cover nominals in full, so the language weakening step described earlier is still necessary.

Further work includes the development of anytime algorithms as just laid out, the implementation of our procedure, and more detailed semantic analysis, in particular with respect to the language weakening preprocessing step proposed.

Chapter 5

Conclusion

by HOLGER WACHE

In this deliverable three approaches for approximation are discussed. In general approximation methods can be classified into three groups: language weakening, knowledge compilation, and approximate deduction. The three approaches are examples for these classes.

In Chapter 2 an approximated deduction approach originally developed by Cadoli and Schaerf [Schaerf and Cadoli, 1995] is investigated. Their approximation method should demonstrate its computational improvements in a more real application scenario, i.e., classification in large and real ontologies. Classification can be regarded as a sequence of subsumption tests. The classification is now approximated with the help of an approximation function which simplifies the subsumption queries by replacing sub concepts. Cadoli and Schaerf discuss two possibilities which sub concepts are replaced but it is fundamental for their approach that the sub concepts are replaced by $\top$ or $\bot$.

Unfortunately the proposed method shows only mixed results in any evaluated ontology. A large set of obviously unnecessary subsumption tests is caused by the effect of term collapsing where the approximated query often is reduced to meaningless queries, i.e., "over-simplified". Term collapsing shows that it is not so important which sub concepts are replaced. More important seems to be trough what the sub concepts are replaced. The experiments suggest that the replacement with $\top$ or $\bot$ is inadequate in more practical setting. Advanced approximation functions must consider more meaningful replacement with respect to the ontology.

The second approach is an example for language weakening. The representation language for the individuals is reduced in order to allow well-established database methods. In InstanceStore relationships between individuals can not be represented explicitly in the ABox. However, an individual can instantiate a concept which has some relationships to other concepts. Relationships can be expressed but only in the terminological knowledge. As a consequence of the restriction many individuals can be selected with normal database queries. The expensive description logic reasoning is mainly reduced to classify

the query and to select a few additional individuals in some situations. The experiments impressively demonstrate the performance improvements and scalability of this approach.

The third approach in Chapter 4 combines knowledge compilation and approximate deduction. In a first step, the knowledge compilation, the ontology in OWL-DL is transformed into disjunctive datalog clauses (DDL) which allows disjunction (only) in the head of a clause. During the transformation the implicit knowledge is determined and explicitly encoded in (additional) clauses. The resulting DDL-program can be executed by an appropriate resolution.

For practical ontologies it can be observed that the disjunctions appear only rarely. For an approximation it is reasonable to omit the disjunction. As a side effect the inference engine can be reduced to the well-known SLD-resolution. The resulting approximation is unsound but complete. However, the effect and the promising benefit must be proven in practical environments.

Bibliography

[Apt and van Emden, 1982] Krzysztof R. Apt and Maarten H. van Emden. Contributions to the theory of logic programming. *J. ACM*, 29(3):841–862, 1982.

[Apt *et al.*, 1988] Krzysztof R. Apt, Howard A. Blair, and Adrian Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 89–148. Morgan Kaufmann, Los Altos, CA, 1988.

[Baader and Hanschke, 1991] F. Baader and P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In *Proc. of the 12th Int'l Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, Australia, 1991.

[Baader *et al.*, 2000] F. Baader, R. Küsters, and R. Molitor. Rewriting concepts using terminologies. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, pages 297–308, San Francisco, 2000. Morgan Kaufman.

[Baader *et al.*, 2003a] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, January 2003.

[Baader *et al.*, 2003b] 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.

[Bachmair *et al.*, 1995] L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic Paramodulation. *Information and Computation*, 121(2):172–192, 1995.

[Baker *et al.*, 1998] P.G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. An Overview. In *Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology (ISMB'98)*, pages 25–34, Menlow Park, California, June 28-July 1 1998. AAAI Press.

[Bechhofer *et al.*, 2003] Sean Bechhofer, Ralf Möller, and Peter Crowther. The dig description logic interface. In *Proceedings of DL2003 International Workshop on Description Logics*, Rome, September 2003.

[Bechhofer, 2003] Sean Bechhofer. The DIG description logic interface: DIG/1.1. In *Proceedings of the 2003 Description Logic Workshop (DL 2003)*, 2003.

[Borgida and Etherington, 1989] Alex Borgida and David W. Etherington. Hierarchical knowledge bases and efficient disjunctive reasoning. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *KR'89: Principles of Knowledge Representation and Reasoning*, pages 33–43. Morgan Kaufmann, San Mateo, California, 1989.

[Brandt *et al.*, 2002] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 203–214, San Francisco, CA, 2002. Morgan Kaufman.

[Caprotti *et al.*, 2004] O. Caprotti, M. Dewar, and D. Turi. Mathematical service matching using description logic and owl. In *To appear in Proceedings 3rd Int'l Conference on Mathematical Knowledge Management (MKM'04)*, volume 3119 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[Dill *et al.*, 2003] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the twelfth World Wide Web conference (WWW12)*, 2003.

[Donini *et al.*, 1992] F.M. Donini, B. Hollunder, M. Lenzerini, A. Marchetti Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53:309–327, 1992.

[Eiter *et al.*, 1997a] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.

[Eiter *et al.*, 1997b] Thomas Eiter, Nicola Leone, Christinel Mateis, Gerald Pfeifer, and Francesco Scarcello. A deductive system for nonmonotonic reasoning. In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'97*, volume 1265 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, 1997.

[Fages, 1994] François Fages. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science*, 1:51–60, 1994.

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[GO,] GO project. European Bioinformatics Institute. `http://www.ebi.ac.uk/go`.

[Groot *et al.*, 2004] P. Groot, A. ten Teije, and F. van Harmelen. Towards a Structured Analysis of Approximate Problem Solving: a Case Study in Classification. In D. Dubois, C. Welty, and M. Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR 2004)*, pages 399–406, Whistler, BC, Canada, June 2004. AAAI Press.

[Grosof *et al.*, 2003] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proc. of the Twelfth Int'l World Wide Web Conf. (WWW 2003)*, pages 48–57. ACM, 2003.

[Haarslev and Möller, 1999] V. Haarslev and R. Möller. Race system description. In *Proceedings of the 1999 Description Logic Workshop (DL'99)*, CEUR Electronic Workshop Proceedings, pages 130–132, 1999.

[Haarslev and Möller, 2001a] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, 2001.

[Haarslev and Möller, 2001b] V. Haarslev and R. Möller. Racer system description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, 2001.

[Haarslev and Möller, 2001c] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, 2001.

[Hitzler and Wendt, 2005] Pascal Hitzler and Matthias Wendt. A uniform approach to logic rogramming semantics. *Theory and Practice of Logic Programming*, 5(1–2):123–159, 2005.

[Hitzler, 2003] Pascal Hitzler. Towards a systematic account of different logic programming semantics. In Andreas Günter, Rudolf Kruse, and Bernd Neumann, editors, *Proceedings of the 26th German Conference on Artificial Intelligence, KI2003, Hamburg, September 2003*, volume 2821 of *Lecture Notes in Artificial Intelligence*, pages 355–369. Springer, Berlin, 2003.

[Hollunder, 1996] Bernhard Hollunder. Consistency checking reduced to satisfiability of concepts in terminological systems. *Ann. of Mathematics and Artificial Intelligence*, 18(2–4):133–157, 1996.

[Horrocks and Patel-Schneider, 2003] Ian Horrocks and Peter F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In *Proc. of the 2nd International Semantic Web Conference (ISWC)*, 2003.

[Horrocks and Patel-Schneider, 2004] I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proc. of the Thirteenth Int'l World Wide Web Conf.(WWW 2004)*. ACM, 2004.

[Horrocks *et al.*, 2000] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[Horrocks, 1998a] I. Horrocks. The FaCT system. In *Proceedings of the second International Conference on Analytic Tableaux and Related Methods*, volume 1397 of *Lecture Notes in Artificial Intelligence*, pages 307–312. Springer, 1998.

[Horrocks, 1998b] Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[Horrocks, 2003] I. Horrocks. Implementation and optimisation techniques. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 306–346. Cambridge University Press, 2003.

[Hustadt *et al.*, 2004a] U. Hustadt, B. Motik, and U. Sattler. Reasoning for Description Logics around $\mathcal{SHIQ}$ in a Resolution Framework. Technical Report 3-8-04/04, FZI, Karlsruhe, Germany, April 2004. http://www.fzi.de/wim/publikationen.php?id=1172.

[Hustadt *et al.*, 2004b] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In *Proc. of the 9th Conference on Knowledge Representation and Reasoning (KR2004)*. AAAI Press, June 2004.

[Hustadt *et al.*, 2004c] Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics with a concrete domain in the framework of resolution. In Ramon López de Mántaras and Lorenza Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, pages 353–357. IOS Press, 2004.

[Knorr, 2003] Matthias Knorr. Level mapping characterizations for quantitative and disjunctive logic programs. Bachelor's Thesis, Department of Computer Science, Technische Universität Dresden, Germany, 2003.

[Li and Horrocks, 2003] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the twelfth World Wide Web conference (WWW12)*, pages 331–339. ACM, 2003.

[Lloyd, 1988] John W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 1988.

[Lutz, 2003] C. Lutz. Description Logics with Concrete Domains—A Survey. In *Advances in Modal Logics*, volume 4. King's College Publications, 2003.

[Masolo *et al.*, 20003] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. *Ontology Library. WonderWeb Deliverable D18*. Laboratory For Applied Ontology - ISTC-CNR, 20003.

[McAllester, 1990] D. McAllester. Truth maintenance. In *Proceedings of AAAI'90*, pages 1109–1116. Morgan Kaufmann, 1990.

[Motik *et al.*, 2004] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. In *Proceedings of the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, November 2004*, 2004. To appear.

[Nonnengart and Weidenbach, 2001] A. Nonnengart and C. Weidenbach. Computing Small Clause Normal Forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science, 2001.

[OWL, a] OWL test suite. `http://www.w3.org/TR/owl-test/`.

[OWL, b] Bechhofer, S.: OWL Reasoning Examples. University of Manchester (2003)
`http://owl.man.ac.uk/2003/why/latest/`.

[Pan *et al.*, 2004] Jeff Z. Pan, Enrico Franconi, Sergio Tessaris, Birte Glimm, Giorgos Stamou, Vassilis Tzouvaras, Ian Horrocks, Lei Li, and Holger Wache. Report on Query Language Design and Standardisation. Technical report, The Knowledge Web project, Dec 2004.

[Patel-Schneider *et al.*, 2002] P. F. Patel-Schneider, P. Hayes, I. Horrocks, and F. van Harmelen. OWL Web Ontology Language; Semantics and Abstract Syntax, W3C Candidate Recommendation. http://www.w3.org/TR/owl-semantics/, November 2002.

[Rector *et al.*, 1993] A. L. Rector, W. A. Nowlan, and A. Glowinski. Goals for concept representation in the galen project. In *Proceedings of the Seventeenth Annual Symposium on Computer Applications in Medical Care (SCAMC-93)*, pages 414–418, Washington DC, USA, 1993.

[Sakama and Inoue, 1994] 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.

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

[Simons *et al.*, 2002] Patrik Simons, Ilkka Niemelä, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1–2):181–234, 2002.

[SWI-Prolog, 2004] http://www.swi-prolog.org/, 2004.

[ten Teije and van Harmelen, 1996] A. ten Teije and F. van Harmelen. Computing approximate diagnoses by using approximate entailment. In G. Aiello and J. Doyle, editors, *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-96)*, Boston, Massachusetts, November 1996. Morgan Kaufman.

[ten Teije and van Harmelen, 1997] A. ten Teije and F. van Harmelen. Exploiting domain knowledge for approximate diagnosis. In M.E. Pollack, editor, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, volume 1, pages 454–459, Nagoya, Japan, August 1997. Morgan Kaufmann.

[Tessaris, 2001] Sergio Tessaris. *Questions and Answers: Reasoning and Querying in Description Logic*. PhD thesis, University of Manchester, Department of Computer Science, April 2001.

[Tobies, 2001] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.

[Uschold *et al.*, 2003] Michael Uschold, Peter Clark, Fred Dickey, Casey Fung, Sonia Smith, Stephen Uczekaj Michael Wilke, Sean Bechhofer, and Ian Horrocks. A semantic infosphere. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proceedings of the Second International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 882–896. Springer, 2003.

[Volz, 2004] Raphael Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, AIFB, University of Karlsruhe, 2004.

Related deliverables

A number of Knowledge web deliverable are clearly related to this one:

Project	Number	**Title** and relationship
KW	D2.1.1	**D2.1.1 Survey of Scalability Techniques for Reasoning with Ontologies** gives an overview of methods for approximating the reasoning.
KW	D2.5.2	**D2.5.2 "Report on Query Language Design and Standardisation"** contains the experiments of InstanceStore