D2.1.1 Survey of Scalability Techniques for Reasoning with Ontologies

Holger Wache (Vrije Universiteit Amsterdam)
Luciano Serafini (Centro per la ricerca scientifica e tecnologica)
Raúl García-Castro (Universidad Politécnica de Madrid)

with contributions from:

Perry Groot (Vrije Universiteit Amsterdam)
Asunción Gómez-Pérez (Universidad Politécnica de Madrid)
Mustafa Jarrar (Vrije Universiteit Brussel)
Yiannis Kompatsiaris (Centre for Research and Technology Hellas)
Diana Maynard (University of Sheffield)
Jeff Pan (University of Manchester)
Rosario Plaza-Arteche (Universidad Politécnica de Madrid)
Floris Roelofsen (Centro per la ricerca scientifica e tecnologica)
Stefano Spaccapietra (École Polytechnique Fédérale de Lausanne)
Giorgos Stamou (National Technical University of Athens)
Andrei Tamilin (Università degli Studi di Trento)
Ilya Zaihrayeu (Università degli Studi di Trento)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.1.1 (WP2.1)

This deliverable gives an overview of methods necessary for achieving scalability. This includes general methods for approximating symbolic inference engines and for compiling knowledge bases, different methods for modularisation and distributed reasoning, and a survey of benchmarking to evaluate the proposed techniques for the Semantic Web.
Keyword list: state-of-the-art, scalability, approximation, modularisation, distribution, symbolic reasoning

Document Identifier	KWEB/2004/D2.1.1/v1.2
Project	KWEB EU-IST-2004-507482
Version	v1.2
Date	02. August, 2004
State	final
Distribution	public

Knowledge Web Consortium

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Work package participants

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

Centre for Research and Technology Hellas
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
0.1	10.05.04	Holger Wache	creation
0.2	15.06.04	Holger Wache	updating the three parts
0.3	28.06.04	Holger Wache	including the comments
1.0	29.06.04	Holger Wache	finalized
1.1	18.07.04	Holger Wache	including the comments from WP leader, p.p. Frank van Harmelen
1.2	02.08.04	Holger Wache	including the comments from quality controller, Jérôme Euzenat

Executive Summary

Scalability is a very important requirement for Semantic Web techniques to be usable in real world applications. This report gives an overview of techniques which may help to achieve scalability.

Current proposals for Semantic Web languages such as the Web Ontology Language OWL are based on formal logic. Consequently they share the advantages and disadvantages of formal logic: a well-founded semantic can be used to derive implicit information, however, at the price of a high computational complexity.

Techniques of approximate logical reasoning techniques are one option for dealing with the complexity of reasoning. We look at general approximation techniques for logical theories. This includes anytime algorithms, approximate entailment and abstraction techniques. In order to be useful on the Semantic Web these techniques need to be examined and adapted to the particular needs of ontological knowledge encoded in semantic web languages.

Distribution and parallelization of inference is another option for achieving scalability. Partitioning and modularisation of ontologies is a first step in this direction. With modularisation the amount of information that must be taken into account at the same time can be reduced. Techniques from a wide range of different fields like database integration and modularisation, partition-based reasoning, or ontology coordination are investigated. Decentralised representations, however, raise the problem of heterogeneity which establishes a thematic relation to working package 2.2 Heterogeneity.

Benchmarking is needed to evaluate the effectiveness of the proposed approaches to increase scalability by measuring system performance in a deterministic and reproducible manner. We give an overview of benchmarking and measurement from relevant research areas and survey existing work on the evaluation of ontology-based tools.

Contents

Chapter 1

Introduction

by PERRY GROOT & HOLGER WACHE

To increase the scalability of symbolic problem solving systems, one can use a number of techniques like approximation, modularisation, or distribution. There is already a vast amount of research done in these areas. However, there are a number of characteristics of symbolic problem solving methods applied to the Semantic Web one should keep in mind before employing those kind of techniques. These characteristics are the following:

Use of logic: Many symbolic problem solving systems — especially those for the Semantic Web — use some form of logic as representation and logical inference to derive a solution. Hence, we are not dealing with numerical problems. This means there is no obvious metric that tells us "how far we are" from the right answer to an inference problem. Furthermore, a logical theory may be simpler to modularise and distributed.

Multiple components: Scalability in symbolic problem solving can be achieved in more

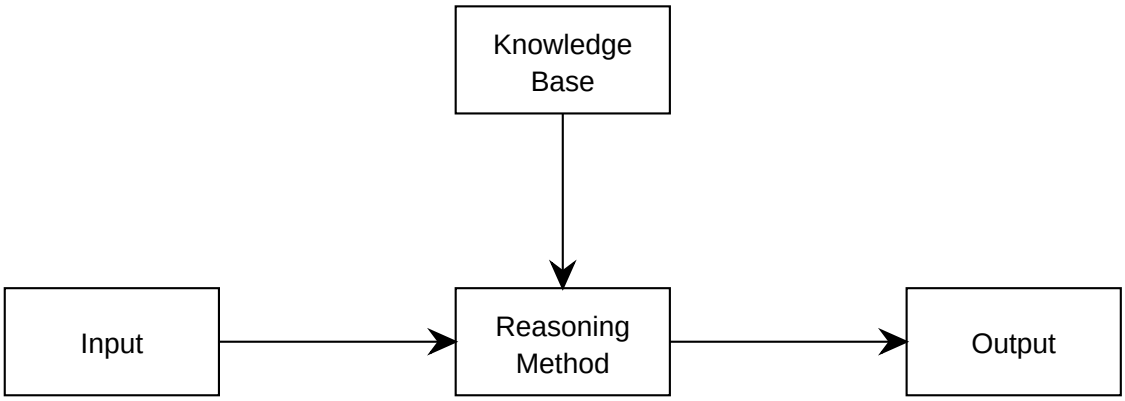

Figure 1.1: A typical architecture for solving problems in symbolic problem solving.

than one way. In general a typical architecture for symbolic problem solving is shown in Figure 1.1. It consists of a reasoning method (i.e., algorithm), a knowledge base, and an input, which together produce an output. For example, in diagnosis the knowledge base consists of rules and facts about the way some system is expected to behave, while the input consists of the observed behaviour of the system. The reasoning method can be some form of logical inference that gives a diagnosis as output, which gives an explanation whenever a discrepancy is found between the observed behaviour and the expected behaviour.

Hence, to enhance the scalability of a system that fits the architecture in Figure 1.1, one can apply a technique on *three* different components: the reasoning method, the knowledge base, or the input. Either one of the components can be approximated, modularised, and/or distributed to enhance the scalability of the system. For example, the inferences can be approximated with the help of anytime algorithms in order to produce some output when needed. Also the inferences can be partitioned and distributed over a network of computational resources.

But only to claim the scalability of some new techniques and tools is not enough. In order to promote the Semantic Web it is also necessary to prove the effectiveness in general and in particular the scalability. Therefore benchmarking is a fundamental part of this working package.

The rest of this report gives a background of concepts and methods that can enhance the scalability of symbolic problem solving methods and how benchmarking can be established. First several approximation techniques for inferences and knowledge bases are investigated. Because appropriate approximation techniques are rare more general techniques are described. In the modularisation and distribution chapter a wide range of different techniques and research areas are investigated and sorted into an appropriate classification schema. Before this report conclude an overview about the benchmarking is given. Because only a few studies for evaluating ontology-based tools exists and as benchmarking activities appear all over Knowledge Web, we have chosen to present in this deliverable a broader viewpoint of benchmarking and its related areas trying to establish a discussion base.

Chapter 2

Approximation

by PERRY GROOT

There is a large amount of literature dealing in some way with approximation. In our analysis we dont restrict ourselves to considering approaches which are specifically and explicitly designed to deal with ontologies. On the contrary, both from a formal and a practical point of view, we think its relevant to also take into account certain rather more general approaches, and to study how they can be deployed for reasoning with ontologies.

However, it falls outside the scope of this deliverable to describe this vast amount in detail. This section is limited to those concepts that have a major influence in the field of symbolic problem solving which is the fundament of the reasoning techniques for the Semantic Web. This related work section is divided according to Figure 1.1. Section 2.1 discusses approximations related to the reasoning method where section 2.2 discusses approximations related to the knowledge base. Approximation related to the input is omitted because it falls not directly in the scope of this deliverable.

2.1 Approximations related to the reasoning method

Approximating the reasoning method to solve a problem approximately is probably the most well known form of approximation among the forms of approximation identified in Figure 1.1. A simple example of an approximation algorithm is demonstrated in the following two player game. Given an interval $[a, b]$, either continuous or discrete, and two players A and B, player A picks a number n from the interval $[a, b]$ and player B has to guess it. Player B may repeatedly pick any number m from the interval $[a, b]$ and player A will tell him if $n < m$, $n > m$ or $n = m$ holds. An approximation algorithm for player B would be to repeatedly pick a number from the remaining interval that contains n.

Although simple, this algorithm belongs to an important group of approximation algorithms called ‘anytime algorithms’ which will be introduced in the next section. A more detailed view on anytime algorithms for approximating the logical entailment op-

erator will follow. Because the key inferences for the Semantic Web are logic-based the universal logical entailment operator is focused. A general overview over another group of approximation — abstraction — will be given in the last section.

2.1.1 Anytime algorithms

Anytime algorithms are algorithms that exchange execution time for quality of results. The term anytime algorithm was coined by Dean and Boddy in the late 1980s in their work on time-dependent planning [Boddy and Dean, 1989, Dean and Boddy, 1988]. A similar idea, called flexible computation was introduced in [Horvitz, 1987] in 1987 in solving time-critical decision problems.

Anytime algorithms are important for symbolic problem solving for two reasons. First, although many problems require a lot of resources (e.g., time) to solve them, many systems can already produce good partial solutions in a short amount of time. A system that can reason about how much time is needed to obtain an adequate result may be more adaptive in complex and changing environments. Second, a technique for reasoning about allocating time need not be restricted to the internals of a system. Intelligent agents must be able to reason about how fast they and other agents can manipulate and respond to their environment.

Not every algorithm that trades execution time for quality of results is necessarily an anytime algorithm. The properties desirable for anytime algorithms are the following [Zilberstein, 1996]:

Measurable quality: The quality of an approximate result can be determined precisely.

Recognisable quality: The quality of an approximate result can easily be determined at run time.

Monotonicity: The quality of the result is a nondecreasing function of time and input quality.

Consistency: The quality of the result is correlated with computation time and input quality.

Diminishing returns: The improvement in solution quality is larger at the early stages of the computation, and it diminishes over time.

Interruptibility: The algorithm can be stopped at any time and provide some answer.

Preemptability: The algorithm can be suspended and resumed with minimal overhead.

The algorithm described in the beginning of this section satisfies these properties. The result of the algorithm after each step is the smallest remaining interval that contains the

number n we seek. By repeatedly choosing a number from the interval that contains n, it will become smaller and therefore be a better approximation of n. Let us denote the interval given as output by the algorithm by I, its length by $l(I)$, and define the quality of the algorithm by

$$Q = \frac{(b-a) - l(I)}{(b-a)}.$$

Then Q is a non-decreasing function which can easily be computed. In fact, when we divide the interval each time in half and the number n has not be found yet, the quality of the algorithm can be computed exactly beforehand without running the algorithm and can graphically be represented as is done in Figure 2.1. This graph clearly demonstrates some of the desired properties (e.g., monotonicity, diminishing returns, etc.). A graph like Figure 2.1 in which the quality of an algorithm is plotted against execution time is called a *performance profile*.

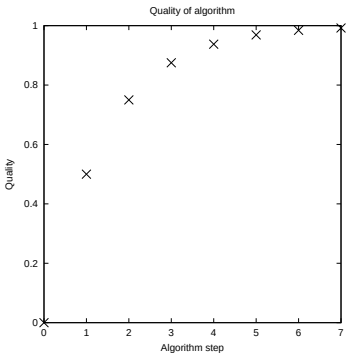

Figure 2.1: Quality of example algorithm for first seven steps.

Since the work of Dean and Boddy [Boddy and Dean, 1989, Dean and Boddy, 1988], the context in which anytime algorithms have been applied has broadened from planning and decision making to include problems from sensor interpretation to database manipulation, and the methods for utilising anytime techniques have become more powerful. In 1991, S.J. Russel and S. Zilberstein completed work on composing anytime algorithms into more complicated systems [Russell and Zilberstein, 1991, Zilberstein, 1993, Zilberstein and Russell, 1996]. By proving composition could be optimally performed (i.e., using information about the algorithm's performance to determine the best way to divide the time between the components), it became possible to build complex anytime systems by combining simple anytime components.

2.1.1.1 An Example: Query Approximation

by HOLGER WACHE

An example for anytime algorithms in the context of ontology reasoning was proposed by Stuckenschmidt and van Harmelen [2002]. They propose an approach for approximating the answers to terminological queries based on non-equivalent transformations of increasing exactness.

First ontology based queries as well as the notion of answers to and relations between queries must be explained. Queries are formalised as conjuncts of predicates that correspond to classes and relations of the ontology. Further, variables in a query may only be instantiated by constants that correspond to objects/instaces in that ontology.

Definition 2.1.1 (Terminological Queries) *Let V be a set of variables then a terminological query Q over a knowledge base T is an expressions of the form*

$$Q \longleftarrow q_{1_i} \wedge ... \wedge q_{m_i}$$

where q_i are query terms of the form $x : C$ or $(x; y) : R$ such that $x; y$ are variables or instances, C is a concept name, and R is a relation name.

The fact that all conjuncts relate to elements of the ontology allows us to determine the answer to terminological queries in terms of instantiations of the query that are logical consequences of the knowledge base it refers to. Therefore the set of answers $res(Q)$ for a query Q consists of tuples $(i_1; ...; i_k)$, where i_j are instances of the knowledge base.

Based on the possible answers to a query, semantic relations between different queries can be defined that will later be used to characterise approximations.

Definition 2.1.2 (Query Containment and Equivalence) *Let T be a knowledge base and $Q_1; Q_2$ conjunctive queries over T. Q_1 is said to be contained in another query Q_2 denoted by $Q_1 \sqsubseteq Q_2$ if for all possible sets of object definitions of a terminological knowledge base the answers for Q_1 is a subset of the answers for Q_2 : $(res(Q_1) \subseteq res(Q_2))$. The two queries are said to be equivalent, denoted as $Q_1 \equiv Q_2$ iff $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$*

The underlying assumption of their approach is that less complex queries can be answered in shorter time. Following this idea, Stuckenschmidt and van Harmelen propose to compute a sequence $Q_1; ...; Q_n$ of queries starting with very simple ones while gradually increasing their complexity. In order to use these simpler queries as approximations for the original query they ensure the following properties of the sequence of queries:

1. $i < j \Longrightarrow Q_i \sqsubseteq Q_j$
2. $Q_n \equiv Q$

The first property ensures that the quality of the results of the queries is not decreasing (refer to "Monotonicity" in the previous section). The second claims that the last query computed returns the desired exact result. Together, these properties ensure that the sequence of queries approximate the exact result in an anytime behaviour.

Stuckenschmidt and van Harmelen discussed how to determine queries to be used for the approximation. The process starts with the universal query Q_0 that returns all objects in the knowledge base and successively adds conjuncts from the original query which leads to a sequence of subsumed query. Further, as the original query has a finite number of conjuncts, adding conjuncts necessarily leads to the original query after a finite number of steps.

The critical part is the step of deciding which conjunct(s) to add next in order to generate the next query in the sequence. This choice has a strong influence on the quality of the approximation. Stuckenschmidt and van Harmelen investigated two different strategies for determining sequences of subsuming queries that try to avoid the problem mentioned. They argue, that the next conjuncts added to the query expression have to directly apply to the objects in the answer set. This in turn depends on the dependencies between the variables in the query. The proposed approaches base on a query graph [Horrocks and Tessaris, 2000], which they restrict to a query tree. The approaches differ in how the tree is traversed. [Stuckenschmidt and van Harmelen, 2002] gives more details.

2.1.2 Approximate entailment

In the beginning of this introduction we stated that a characteristic of systems in symbolic problem solving is the use of logic. This section looks at some approximate reasoning methods that can be used to approximate any logical inference problem that uses the logical entailment operator.

2.1.2.1 Boolean Constraints Propagation

One of those methods is Boolean Constraints Propagation (BCP) which is a variant of unit resolution [McAllester, 1990]. BCP is a sound, but incomplete linear-time reasoner that can be used to approximate the logical entailment relation. Sometimes BCP is also called *clausal BCP*, because BCP is usually restricted to clauses. This restriction makes BCP tractable. If BCP is not restricted to clauses, but general formulae are allowed, the reasoner is called *formula BCP* and the method becomes intractable.

In [de Kleer, 1990] two techniques are suggested for using clausal BCP for theories containing arbitrary formulae. In one, *CNF-BCP*, the formulae are first converted into CNF, and in the other, *Prime-BCP*, clausal BCP is applied to the prime implicants of each formula. Since computing prime implicants is itself an intractable problem, Prime-BCP is also inherently intractable.

For CNF-BCP there are two methods to transform the formulae into CNF. If no new symbols are added, then the conversion to CNF may lead to an exponential increase of the size of the given theory. The transformation of a theory to CNF can be done in linear time and space if new symbols are allowed to be added to the theory [Cook, 1971]. Each new symbol will be used to represent some sub-formula of the theory. However, with this method, reasoning with CNF-BCP is strongly inhibited.

Another method that extends BCP to non-clausal theories is Fact Propagation (FP) [Dalal, 1992]. FP can be specified using a confluent rewrite system, for which an algorithm can be written that has quadratic-time complexity in general, but is still linear-time for clausal theories. Another advantage of FP is that it sometimes performs more infer-

ences than CNF-BCP. A restricted form of FP (RFP) also exists which infers exactly the same literals as CNF-BCP.

All the discussed methods (i.e., BCP, FP, and RFP) are incomplete reasoners. In [Dalal, 1996a, Dalal, 1996b] a general technique is presented that can extend any incomplete propositional reasoner satisfying certain properties to a family of increasingly-complete, sound and tractable reasoners. Such a family of increasingly-complete, sound and tractable reasoners is called an 'anytime family' of propositional reasoners, which, as the name implies, can be used as an anytime algorithm.

These BCP-methods have not yet been studied in the specific context of reasoning with ontologies. In particular, questions must be answered on how the inherent incompleteness of these methods affects the typical logical structures that are found in ontologies (such as class hierarchies), and in general a logical fragment restricted to unary and binary predicates with only limited forms of quantification and negation.

2.1.2.2 S-1- and S-3-entailment

Another method for approximating logical inferencing was devised by Cadoli and Schaerf and is called S-1- and S-3-entailment [Schaerf and Cadoli, 1995]. This method uses a semantic approach and is based on a 'classic' method for incomplete reasoning, which has been introduced by Levesque [Levesque, 1984, Levesque, 1988, Levesque, 1989] and has since been studied by many other authors.

The method of Cadoli and Schaerf allows both sound approximations and complete approximations and the approximate answers can be improved when more resources (e.g., computation time) are given to the reasoner. The approximate answer will converge to the right answer provided there is enough time and motivation. Cadoli and Schaerf proposed the following guidelines which are fulfilled by S-1- and S-3-entailment and may be desirable for any approximation method:

Semantically well-founded: Approximate answers should give semantically clear information about the problem at hand.

Computationally attractive: Approximate answers should be easier to compute than answers to the original problem.

Improvable: Approximate answers can be improved, and eventually they converge to the right answer (provided there is enough time and motivation).

Dual: Both sound approximations and complete ones should be described.

Flexible: The approximation method should be general enough to be applicable to a wide range of reasoning problems.

For a precise definition of the approximate entailment operators by Cadoli and Schaerf [1995] we assume that there is an underlying finite language L used for building all the sentences. Symbols t and f are used for denoting special propositional letters, which are always mapped into 1 and 0, respectively. In the following we denote with S a subset of L.

Definition 2.1.3 (S-3-interpretation) *An S-3-interpretation of L is a truth assignment which maps every letter l of S and its negation $\neg l$ into opposite values. Moreover, it does not map both a letter l of $L \setminus S$ and its negation $\neg l$ into 0.*

Definition 2.1.4 (S-1-interpretation) *An S-1-interpretation of L is a truth assignment which maps every letter l of S and its negation $\neg l$ into opposite values. Moreover, it maps every letter l of $L \setminus S$ and its negation $\neg l$ into 0.*

The names given to the interpretations defined above can be explained as follows. For an S-1-interpretation there is *one* possible assignment for letters outside S, namely false for both x and $\neg x$. For an S-3-interpretation there are *three* possible assignments for letters outside S, namely the two classical assignments, plus true for both x and $\neg x$. (As a classical interpretation allows *two* possible assignments for letters, such an interpretation is sometimes referred to as a 2-interpretation.)

Satisfaction of a formula by an S-1- or S-3-interpretation is defined as follows. The formula is satisfied by an interpretation σ if σ evaluates the formula written in Negated Normal Form (NNF) into true using the standard rules for the connectives.

The notions of S-1- and S-3-entailment are now defined in the same way as classical entailment: A theory T S-1-entails a formula ϕ, denoted by $T \models_1^S \phi$, iff every S-1-interpretation that satisfies T also satisfies ϕ. S-3-entailment is defined analogously and denoted by $T \models_3^S \phi$.

Let $S, S' \subseteq L$, T a generic propositional CNF formula, and ϕ a generic propositional clause not containing both a letter l and its negation $\neg l$. We use the shorthand $\models_i^S \Rightarrow \models_i^{S'}$ denote $T \models_i^S \phi \Rightarrow T \models_i^{S'} \phi$. These definitions then lead to the following result [Schaerf and Cadoli, 1995]:

Theorem 2.1.5 (Approximate Entailment) *Let $S, S' \subseteq L$, such that $S \subseteq S'$, then*

$$\models_3^{\varnothing} \Rightarrow \models_3^S \Rightarrow \models_3^{S'} \Rightarrow \models_2 \Rightarrow \models_1^{S'} \Rightarrow \models_1^S \Rightarrow \models_1^{\varnothing} .$$

This theorem tells us that $\models_3^S$ is a sound but incomplete approximation of the classical entailment $\models_2$, whereas $\not\models_1^S$ is a sound but incomplete approximation of $\not\models_2$ (i.e., $\not\models_1^S \Rightarrow \not\models_1^{S'} \Rightarrow \not\models_2$). Furthermore, the theorem states that the accuracy of the approximations can be improved by increasing the parameter S until the approximations coincide with the classical entailment.

Theorem 2.1.5 holds even if T is a NNF formula and ϕ is a generic formula in CNF. This aspect has been analysed in [Cadoli and Schaerf, 1995], which analyses other normal forms for which the result holds.

We will continue with some results, which show that S-i-entailment can be reduced to S-i-satisfiability. Before doing so we introduce the following definition:

Definition 2.1.6 *We denote with* $letters(\gamma)$ *the set* $\{l \in L \mid l\ occurs\ in\ \gamma\} \cup \{l \in L \mid \neg l\ occurs\ in\ \gamma\}$.

The next two theorems show that S-1- and S-3-entailment can be reduced to S-1- and S-3-satisfiability, respectively.

Theorem 2.1.7 (Reducing S-1-entailment to S-1-satisfiability) *Let* γ *be* $\gamma_S \cup \gamma_{\overline{S}}$*, where both letters(*$\gamma_S$*)* $\subseteq S$ *and letters(*$\gamma_{\overline{S}}$*)* $\cap S = \varnothing$ *hold.* $T \models_1^S \gamma$ *holds iff* $T \cup \{\neg\gamma_S\}$ *is not* S*-1-satisfiable.*

Theorem 2.1.8 (Reducing S-3-entailment to S-3-satisfiability) *Let letters(*γ*)* $\subseteq S$ *hold.* $T \models_3^S \gamma$ *holds iff* $T \cup \{\neg\gamma\}$ *is not* S*-3-satisfiable.*

Note that the condition $letters(\gamma) \subseteq S$ in Theorem 2.1.8 is not a restriction since [Schaerf and Cadoli, 1995] also prove that $T \models_3^S \gamma$ iff $T \models_3^{S \cup letters(\gamma)} \gamma$.

These results extend the well-known relation existing between classical entailment and satisfiability, namely $T \models \gamma$ iff $T \wedge \neg\gamma$ is unsatisfiable. The importance of such a result is that S-3-satisfiability can be tested in the following way:

1. replace by t all occurrences (both positive and negative) in T of letters which belong to $L \setminus S$, thus obtaining the formula $[T]_3^S$.
2. test standard (2-valued) satisfiability of $[T]_3^S$.

In a similar way S-1-satisfiability can be tested in the following way:

1. replace by f all occurrences (both positive and negative) in T of letters which belong to $L \setminus S$, thus obtaining the formula $[T]_1^S$.
2. test standard (2-valued) satisfiability of $[T]_1^S$.

Hence, considering the above tests we can clarify S-1- and S-3-satisfiability by the following syntactic operations. For a theory T in clausal form, T is S-1-satisfiable iff T is classically satisfiable after removing from every clause any literals with a letter outside S. When this results in an empty clause, the theory becomes the inconsistent theory $\bot$. Similarly, T is S-3-satisfiable iff T is classically satisfiable after removing every clause

from the theory that contains a literal with a letter outside S. This may result in the empty theory $\top$. Because of the close correspondence between S-1-, S-3-satisfiability and these syntactic operations, we prefer to write $\vdash_i^S$ instead of $\models_i^S$.

Cadoli and Schaerf present an algorithm that can be used to compute the S-1-satisfiability and S-3-unsatisfiability of a generic formula. This algorithm runs in time exponential in $|S|$ and uses polynomial space. Furthermore, the algorithm can benefit from previous computations. More precisely, when $S' \supset S$, computing satisfiability with respect to S' can be done by using information gained in a previous step when the satisfiability was computed with respect to S. Hence, the method can be used to approximate the classical entailment operator from two directions (by using $\not\vdash_1^S$ and $\vdash_3^S$ instead of the $\vdash$ operator) in a stepwise fashion and is not harder (and usually easier) to compute than the original problem. Furthermore some applications of this method to approximate diagnosis are reported in [ten Teije and van Harmelen, 1997, ten Teije and van Harmelen, 1996].

In their original 1995 paper, Cadoli and Schaerf themselves already studied the use of S1,3-entailment for Description Logics, and the provide the straightforward extension of their method to first-order theories that is required for this. They are able to give a precise approximate semantics for approximating a concept by a sequence of ever weaker and stronger sub- or super-concepts. They apply their method to languages like ALC, which are much weaker in expressivity then the languages currently being proposed for ontology modelling on the Semantic Web such as OWL. Questions on the applicability of their method to OWL are still open.

2.1.3 Abstraction

by HOLGER WACHE

Another possible way to approximate a reasoning method is abstraction. The earliest and "undoubtedly one of the most influential work"[Giunchiglia *et al.*, 1997] was proposed by Sacerdoti [1973] with his ABSTRIPS. The complexity is reduced while abstracting the operator descriptions in order to simplify the planning process finding a valid sequence of operators which achieves some goals. Then the generated abstract plan can be refined to a concrete plan solving the concrete problem. Most abstraction techniques follow this principle: solving a problem on an abstract, simple level and refining the abstract solution to the concrete problem to receive a concrete solution.

Abstraction is a very general techniques applied in a wide range of application domains. Therefore the meanings and the intuitive understanding of the term "abstractions" differ from application to application and from domain to domain. In this deliverable we shall concentrate on abstraction for logical reasoning systems [Giunchiglia *et al.*, 1997, Nayak and Levy, 1995, Giunchiglia and Walsh, 1992] because as already mentioned they are the key inferences in the Semantic Web.

Giunchiglia and Walsh [1992] were the first who developed a general framework which unifies past work and provides a vocabulary to discuss different types of abstrac-

tion [Nayak and Levy, 1995]. They define abstraction as a syntactic mapping of a problem representation (the "base" representation Σ_B) into a simpler one (the "abstract" representation Σ_A) that satisfies some desirable properties and simplifies the complexity of the inference task [Frei and Faltings, 2000]. Formally they use the notion of a formal system $\Sigma = \langle \Lambda, \Theta \rangle$ for representing problems, which consists of a set of formulae Θ written in the language Λ.

Definition 2.1.9 (Syntactic Abstraction [Giunchiglia and Walsh, 1992]) *An abstraction $f : \Sigma_B \Rightarrow \Sigma_A$ is a pair of formal systems $\langle \Sigma_B, \Sigma_A \rangle$ with the languages Λ_B and Λ_A resp. and a total function $f_\Lambda : \Lambda_B \rightarrow \Lambda_A$.*

This definition allows to classify abstractions according to the relationship between the set of theorems of the abstract theory $TH(\Sigma_A) = \{\alpha | \vdash_{\Sigma_A} \alpha\}$ and the set of theorems of the base theory $TH(\Sigma_B)$. If $TH(\Sigma_A)$ is a subset of (resp. superset of or equal to) $TH(\Sigma_B)$, then the abstraction is called a TD (resp. TI or TC abstractions). Consequently the set of theorems which would make a theory Σ inconsistent (i.e. $NTH(\Sigma) = \{\alpha | \Sigma \cup \{\alpha\} inconsistent\}$ can be used to define NTD, NTI, and NTC abstractions respectively.

The theory of Giunchiglia and Walsh [1992] captures some important aspects of many abstractions. For example any T* abstraction with additional property $f(\alpha \rightarrow \beta) = f(\alpha) \rightarrow f(\beta)$ preserves the deducibility. Also for refutation systems it can be shown that NT* abstractions which preserve negation (i.e. $f(\neg\alpha) = \neg f(\alpha)$) also preserve inconsistency.

[Giunchiglia and Walsh, 1992] pointed out that in literature most practical abstractions are related to the (N)TI abstraction. For example, most complete and well developed abstractions for resolution systems developed by Plaisted [Plaisted, 1981, Plaisted, 1980] are excellent examples for NTI-abstraction. The complete discussion of the theory of abstraction and more examples can be found in [Giunchiglia and Walsh, 1992].

However, Nayak and Levy argue in [Nayak and Levy, 1995] that viewing abstractions as syntactic mappings as introduced by Giunchiglia and Walsh [1992] captures only one aspect of abstractions — it omits to "capture the underlying justification that leads to that abstraction". As consequence the syntactic theory does not allow to compare or to rank different (TD) abstractions e.g. in order to determine which is more natural. Therefore Nayak and Levy introduce MI-abstractions, a strict subset of the TD abstractions. The MI abstraction is performed in two steps: first, the domain model is abstracted and then the intension of the abstraction is captured with some additional formulae which justify the domain model abstraction. These additional formulae can be seen as a formalisation of the semantic which bias the interpretations of the ground and abstract language Λ_B and Λ_A.

Definition 2.1.10 (Semantic Abstraction [Nayak and Levy, 1995]) *A semantic abstraction is defined as $f : I(\Lambda_B) \rightarrow I(\Lambda_A)$, where $I(\Lambda)$ is the set of interpretations of the language Λ.*

The MI abstraction provides some nice features. The resulting abstraction can be proven to be the strongest one. Furthermore the MI abstraction can be determined automatically [Nayak and Levy, 1995]. However, Giunchiglia et.al. [1997] like to see this as a drawback because it complicates the process of defining an abstraction and generating the abstract theory.

One of the motivation for using abstraction for approximating logical inferences is the reduction of complexity. Korf [1990] was the first who shows that a search could be reduced from polynomial to linear in a planning domain. Knoblock [1989] and Giunchiglia and Walsh [1990] discovered similar results. However, Bäckström and Jonsson [1995] argue that some abstractions can also slow down the planning process exponentially. In practice this result is only of a technical matter which says in the set of all possible abstractions there are some (namely H_1 and H_2 cf. [Bäckström and Jonsson, 1995]) with exponential growth of search space (H_2 w.r.t. H_1). But there will also be some abstractions with better complexity reduction [Giunchiglia *et al.*, 1997].

2.2 Approximations related to the knowledge base

One can enhance the scalability of a system by reducing the complexity of inferencing from a knowledge base, as the computational complexity of reasoning is a well known problem in symbolic problem solving. An area that deals with this problem for knowledge bases written in some logical language is *knowledge compilation*. The underlying idea of knowledge compilation is that a knowledge base does not change much over time. The goal of knowledge compilation is to translate the knowledge into (or approximate by) another knowledge base with better computational properties. This 'compiled' knowledge base can be re-used to solve many problem instances (so-called 'on-line'), thereby saving time and computational resources when compared to solving the problem instances with the original knowledge base.

However, it is not always possible to make on-line reasoning more efficient in all cases. For some important problems the requirements are unlikely to be achieved. Two classical approaches have been developed for addressing the computational hardness of reasoning problems, which are *language restriction* and *theory approximation*. Traditionally these methods have been used for achieving tractability in reasoning problems. They can be described as follows:

Language restriction. The language used to represent knowledge may be restricted. Then knowledge is compiled into the restricted language. One can still represent interesting cases and one can compile the resulting problems.

Theory approximation. One can compile the theory into another "easier" theory. During the compilation one give up the soundness or completeness in the answer to the original reasoning problem. This means that either certain information is lost in the

compilation phase, or that the theory is compiled into an equivalent theory and the soundness or completeness in the answer is lost in the on-line reasoning phase.

The language restriction has not been considered often in the knowledge compilation setting. On the other hand, the theory approximation approach has had some success.

In section 2.2.1 we look at knowledge compilation techniques in general; the next sections gives an example how to achieve knowledge compilation. Approaches for exact knowledge compilation are presented in the section 2.2.2. Approximation techniques come in the following section 2.2.3. In section 2.2.4 we refer to the most promising approach restricting the representation language.

Please note that the techniques presented here are most often concerned with propositional languages.

2.2.1 Knowledge compilation

Many systems in symbolic problem solving for the Semantic Web use logic as representation language. However, the logical approach also has some drawbacks. The complexity of logical entailment is such a drawback. It is well known that deduction in a logical formalism is very much demanding from a computational point of view. Many problems and tasks (e.g., planning, diagnosis, and configuration) that we are typically dealing with in symbolic problem solving are already intractable for the simple varieties. Since such tasks still have to be performed, several methods were developed to deal with this kind of problem.

A technique called *knowledge compilation* is such a method that can be used to deal with computational difficulties. The underlying idea of knowledge compilation is that many reasoning problems can be split into two parts: a knowledge base and a query. For example, in diagnosis the knowledge base consists of rules and facts about the way some system is expected to behave. When there exists a discrepancy between the observed behaviour and the way the system is expected to behave, the knowledge base is queried to give a cause for this discrepancy. In this case the query can be a conjunction of specific facts reflecting the current state (i.e., observations), which implies the cause for discrepancy (in the context of the knowledge base). More specifically, in diagnosis the knowledge of the expected behaviour of a system is represented by a theory T, the current state is represented by some formula F (e.g., a conjunction of facts), and some cause is represented by a literal l. The problem of determining $T \cup F \vdash l$ is logically equivalent to $T \vdash F \Rightarrow l$. This problem can be considered to have two parts: the theory T is the 'knowledge base', and $F \Rightarrow l$ is the 'query' of the problem.

In a typical scenario, the knowledge base remains unchanged over a long period of time and is used to answer many queries. In knowledge compilation the idea is to split this kind of reasoning into two phases:

1. In the first phase the knowledge base is pre-processed by translating it into an appropriate data structure, which allows for more efficient query answering. (This phase is also called *off-line reasoning*.)
2. In the second phase, the data structure, which resulted from the previous phase, is now used to answer the query. (This phase is also called *on-line reasoning*.)

The goal of the pre-processing is to make on-line reasoning computationally easier with respect to reasoning in the case when no pre-processing is done at all.

Pre-processing is quite common in Computer Science. For example, compilers usually optimise object code or a graph can be pre-processed to obtain a data structure that allows for a fast node reachability test. However, in Computer Science pre-processing is usually done for problems, which are already solvable in polynomial time. What characterises the same study of such techniques in the context of symbolic problem solving is that reasoning problems are often NP-hard.

The rest of this section is divided as follows. Next we give the terminology we will use as well as some formal definitions. Thereafter, we discuss several methods used in knowledge compilation. These methods are divided into exact methods and approximate methods.

2.2.1.1 Knowledge compilation: terminology

First we introduce a simple reasoning problem which will be used as running example. A reasoning problem is always specified by means of (1) its instances, and (2) the question we are expected to solve. The *Literal Entailment problem*, which is our running example, is specified as follows:

Instance: Finite set L of Boolean variables, a propositional formula in Conjunctive Normal Form T, and a literal l (both T and l are built using variables in L).

Question: Is it true that all models of T are models of l (i.e., that $T \models l$)?

Usually a problem is represented as a pair `Instance/Question`. However, this representation does not tell us which part of `Instance` is fixed. Another representation is therefore needed that clearly splits an instance into a fixed and a variable part. For example, we stated before that many reasoning problems can be considered to consist of two parts: a knowledge base and a query. The knowledge base is not changed often and can therefore be pre-processed and used to answer many problem instances. The query on the other hand is posed to the knowledge base and will be different for each instance. To address the pre-processing aspects in knowledge compilation, we will therefore use the following terminology [Cadoli, 1993, Cadoli, 1996]:

Fixed part of a problem: The part of a problem that goes to pre-processing when a problem is compiled (e.g., a propositional formula T in CNF). [1]

Variable part of a problem: The part that does not go through pre-processing (e.g., a literal l).

(Structured) problem: A triple consisting of the type of question that we ultimately want to solve, its fixed part, and its variable part (e.g., $[T \models l, T, l]$).

Which part of the problem is considered fixed or variable may depend on the knowledge about the domain.

We stated before that the goal of knowledge compilation is to make on-line reasoning easier with respect to reasoning in the case when no pre-processing is done at all. An example problem for which this goal can be attained is our running example the Literal Entailment problem. Without pre-processing, the Literal Entailment problem is coNP-complete, but after compiling it, the problem $[T \models l, T, l]$ can be solved on-line in time polynomial in $|T| + |l|$. This can be done as we can record on a table, for each literal l occurring in T, whether $T \models l$ or not. The size of the table is in $O(n)$, where n is the cardinality of the alphabet L of T. The table can be consulted in $O(n)$ time. Note that creating the entire table means solving $O(n)$ instances of a coNP-problem, but this is done off-line and in knowledge compilation one is not concerned with these off-line computational costs.

The compilation of our running example $[T \models l, T, l]$ contains two aspects which are important:

1. The output of the pre-processing part is a data structure (e.g., a set of propositional formulae), which has size polynomial with respect to the fixed part.
2. On-line reasoning takes time polynomial in the size of the data structure and the size of the variable part.

Furthermore, it is believed that the same pre-processing should facilitate the answer to a whole class of queries – not just one. Intuitively, the effort spent in the pre-processing phase pays off when its computation time is amortised over the answers to many queries. Finally, even if the compilation process can take a substantial amount of time, it must always terminate. The aspects mentioned above can be used as guidelines to formalise the notion of a compilable problem. The following definition is from [Cadoli *et al.*, 1994]:

Definition 2.2.1 (Compilable problem) *A problem [P, F, V] is compilable if there exist two polynomials p_1, p_2 and an algorithm ASK such that for each instance f of F there is a data structure D_f such that:*

[1] Note that the vague term 'the part' is intentionally used as knowledge compilation can be used on many kinds of data structures (e.g., formulae, models). However, within this report we only consider knowledge compilation of propositional formulae and 'the part' can be considered to be a set of propositional formulae representing for example a knowledge base.

1. $|D_f| \leq p_1(|f|)$.
2. *for each instance v of V the call ASK(D_f, v) returns yes if and only if (f, v) is a "yes" instance of P.*
3. *ASK(D_f, v) requires time $\leq p_2(|v| + |D_f|)$.*

Remember, that in Definition 2.2.1 P stands for the type of question we ultimately want to solve. For example, does the knowledge base entail a certain literal, or can a certain cause explain the discrepancy between observations and system description. Furthermore, in Definition 2.2.1 F stands for the fixed part of the problem, and V stands for the variable part of the problem. In Definition 2.2.1, Constraint 1 states that the size of some data structure D_f is polynomial in the size of some instance f of F. Constraint 2 states that D_f can be used to answer an instance (f, v) of P for any instance v of V. Hence, from Constraints 1 and 2 follows that D_f stands for the compilation of f. Finally, Constraint 3 states that any instance (f, v) can be answered in time polynomial in the size $|v| + |D_f|$.

In case Definition 2.2.1 does not hold for a problem $[P, F, V]$ such a problem is said to be *incompilable*.

In the rest of this section, we will describe various knowledge compilation methods (for propositional theories). The methods are divided into methods which exactly translate the original theory into another form (Section 2.2.2) and methods which translate the original theory into a form that approximates the original theory (Section 2.2.3).

2.2.2 Exact knowledge compilation: Prime implicants and prime implicates

A knowledge compilation method is called exact when the original theory is compiled into a *logically equivalent* theory. Proposals for exact knowledge compilation can be classified in three main methods [Cadoli, 1993]:

1. use prime implicants or prime implicates.
2. add to the knowledge base only those prime implicates that make any deduction possible by unit resolution.
3. use prime implicates with respect to a tractable theory.

Note that an *implicant* of a theory Σ is a conjunction of literals D such that $D \models \Sigma$ and D does not contain two complementary literals; a *prime implicant* is a minimal implicant with respect to set containment. An *implicate* of a theory Σ is a disjunction of literals C (a *clause*) such that $\Sigma \models C$ and C does not contain two complementary literals; a *prime implicate* is a minimal implicate with respect to set containment.

The simplest proposals on exact knowledge compilation use the fact that knowledge bases have normal forms (e.g., CNF and DNF) from which consequences can be easily computed.

For example by taking the disjunction of all prime implicants $D_1, \ldots, D_k$ of a knowledge base KB one obtains a DNF formula $D_1 \vee \cdots \vee D_k$ which is equivalent to KB [Quine, 1959], such that for every query Q, $KB \models Q$ iff for every prime implicant D_i, $D_i \models Q$. If Q is in CNF this amounts to verify that every clause of Q has a non-empty intersection with each D_i. Hence, entailment of CNF queries can be computed in time polynomial in the size of the set of prime implicants plus the size of the query.

Dually one can take the conjunction of all prime implicates $C_1, \ldots, C_k$ of a knowledge base KB to obtain a CNF formula $C_1 \wedge \cdots \wedge C_k$ which is equivalent to KB [Reiter and de Kleer, 1987]. For every CNF query Q, $KB \models Q$ iff for each nontautologous clause C' of Q there is a prime implicate C of KB such that $C \models C'$, i.e., $C \subseteq C'$. Hence, the entailment of CNF queries can be computed in time polynomial in the size of the set of prime implicates plus the size of the query.

Intuitively all prime implicates of a CNF knowledge base can be found by resolving clauses (each resolvent is an implicate) and discarding implicates which are not prime. However, this method may require too many resolution steps. Research on algorithms for computing prime implicates already started a long time ago and can be found for example in [Tison, 1967, Jackson and Pais, 1990, de Kleer, 1992, Simon and del Val, 2001].

However, the number of prime implicants and prime implicates of a knowledge base with n variables was shown in [Chandra and Markowsky, 1978] to be exponential in n in the worst-case. In [Schrag and Crawford, 1996b] an experimental study of the number of prime implicants and prime implicates for CNF knowledge bases of increasing size was performed.

Unit-resolution-complete compilation Since the number of prime implicates can be exponential an enhanced method was proposed in [del Val, 1994]. The method is based on the observation that entailment of a CNF query and a prime-implicates-compiled knowledge base can be done by checking whether each query clause is contained in a prime implicate. This check for containment is a form of *unit-resolution* refutation, which is defined as follows:

Definition 2.2.2 *Unit resolution is a form of resolution where at least one of the two clauses to be resolved is a single literal.*

Unit resolution is sound, but incomplete, i.e., not all refutations can be found by unit resolution. Negating a clause in the query yields a set of negated literals, and by unit resolution one obtains the empty clause if and only if there is a prime implicate made by a subset of the literals in the clause.

By substituting the set-containment check with unit resolution refutation, one does not need to keep all prime implicates, but only the subset of prime implicates from which each prime implicate can be derived by unit resolution. Since every unit resolution refutation needs polynomial time in the size of the initial clauses to be refuted, this method also

turns a coNP-complete method into a problem solvable in polynomial time with respect to the size of the formula produced by pre-processing.

In [del Val, 1994] cases are given in which unit refutation can discard an exponential number of prime implicates. However, the method is limited to CNF knowledge bases (i.e., although any formula can be translated into an equivalent CNF formula, this may increase its size exponentially).

Theory prime implicates Another method was developed by Marquis [1995]. He starts observing that prime implicants and prime implicates methods are based on transforming the problem $KB \models Q$ (Q being a clause), involving the entire knowledge base KB, into *local* tests involving one prime implicant/implicate at a time. He proposes to enhance such local tests with a theory, while keeping its complexity to be polynomial-time.

Definition 2.2.3 *A* theory prime implicate *of a knowledge base KB with respect to a theory Φ is a clause C such that $KB \cup \Phi \models C$ and for each other clause C' if $KB \cup \Phi \models C'$ and $\Phi \cup C' \models C$ then also $\Phi \cup C \models C'$.*

The theory prime implicates of a knowledge base KB with respect to a theory Φ will be denoted by TPI(KB,Φ). Note that when Φ is an empty knowledge base one obtains the definition of prime implicate, i.e., TPI(KB,$\varnothing$) = PI(KB). Hence, theory prime implicates extend prime implicates.

Observe that checking $\Phi \cup \{C\} \models C'$ is equivalent to check, for each literal $l_i \in C$, whether $\Phi \models \neg l_i \cup C'$. The key point is that *if deduction in the theory Φ can be done in polynomial time* entailment of CNF queries can be computed in time polynomial in the size of the set of theory prime implicates.

Marquis suggests as good candidates for Φ the set of all Horn clauses of KB, the set of all binary clauses of KB, and many others. In general, any subset of KB such that entailment is tractable can be used. However, for a knowledge base KB and two theories Φ and Φ' such that $KB \models \Phi'$ and $\Phi' \models \Phi$ holds we have that the number of clauses of TPI(KB,Φ) can never be larger than TPI(KB,Φ') (Corollary 2 in [Marquis, 1995]). Hence, to get the best theory prime implicate compilation we only have to consider the largest subsets (with respect to set inclusion) of a knowledge base KB.

Marquis gives examples in which the number of theory prime implicates is exponentially smaller than the number of del Val's filtered prime implicates. Furthermore, in [Marquis and Sadaoui, 1996] Marquis and Sadaoui give an algorithm for computing theory prime implicates, which is based on Binary Decision Diagrams. With this algorithm, the initial knowledge base does not need to be in CNF, and the prime implicates of KB need not be generated. This reduces considerably compilation operations, as shown in some experiments.

In summary, the three classes of exact knowledge compilation methods given in the beginning of Section 2.2.2 are ordered according to the effectiveness of the method.

Each method is at least as good as its predecessors, and for each method there exists a theory that can be compiled exponentially smaller than when using one of the earlier described knowledge compilation methods. However, according to a theorem in [Selman and Kautz, 1996] it is highly unlikely that one of the described methods (or any exact knowledge compilation method) can compile every theory into a polynomial data structure.

To overcome some of the drawbacks of exact knowledge compilation methods some of the requirements might be weakened. We will look at some of these methods in the following section.

2.2.3 Approximate knowledge compilation: Theory approximation

The theory approximation approach is analogous to optimisation problems. In both cases, we are interested in approximate solutions that are meaningful. However, in Knowledge Representation we are not dealing with numerical problems. This means there is no obvious metric that tells us "how far we are" from the right answer to an entailment problem. The approximation of the answer to an entailment problem should therefore be grounded on a semantical basis.

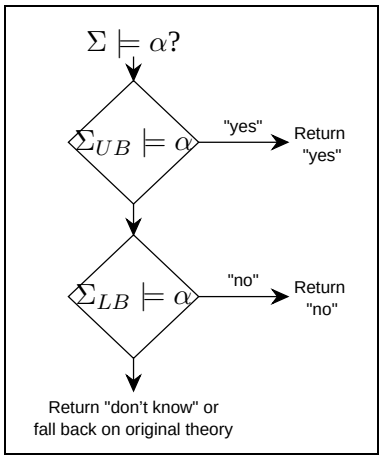

Figure 2.2: Fast querying using theory approximations.

The underlying idea in approximate knowledge compilation is that answers to a query can be approximated from two sides. Either with an answer that is sound but incomplete or with an answer that is complete but unsound. A sound-but-incomplete answer approximates the correct answer as a 'yes' answer is correct while a 'no' answer is in fact a 'don't know'. A complete-but-unsound answer approximates the correct answer as a 'no' answer is correct while a 'yes' answer is in fact a 'don't know'. Obviously, in both cases one wants to have an approximation that can be computed using fewer resources.

The ideas mentioned above can be formalised as follows. An approximation A of a knowledge base Σ is *sound* when for every query Q, if $A \models Q$ then $\Sigma \models Q$. In this case A is called an *upper bound* for Σ. Observe that A is an upper bound (UB) for Σ if and only if $\Sigma \models A$ holds. Dually, an approximation B of a knowledge base Σ is *complete* when for every query Q, if $B \not\models Q$ then $\Sigma \not\models Q$. In this case, B is called a *lower bound* (LB) for Σ, and $B \models \Sigma$ holds.

The approximations can be used to improve the efficiency of query answering. Suppose we have a knowledge base Σ and we want to determine if a formula α is implied by the knowledge base Σ. This can be done as depicted in Figure 2.2 where Σ_{UB} is an upper

bound of Σ and Σ_{LB} is a lower bound of Σ. First, the system tries to answer the query quickly by using the approximations. If $\Sigma_{UB} \models \alpha$ then it returns 'yes', or if $\Sigma_{LB} \not\models \alpha$ then it returns 'no'. In case no answer is obtained, the system could simply return 'don't know', or it could decide to spend more time and use a general inference procedure to determine the answer directly from the original theory. In the latter case, the approximations could still be used to prune the search space of the inference procedure. For example, in [Kautz and Selman, 1994] queries are answered using a knowledge base Σ and also answered using the knowledge base Σ conjoined with its unit LUB. The latter is shown to speed up the query answering in their experimental setup.

2.2.3.1 Anytime versions of exact compilation methods

Any of the exact knowledge compilation methods discussed previously in Section 2.2.2 can be turned into an approximate method by stopping it before it is completed, because these methods are anytime algorithms. In fact, we can be a bit more precise about the approximations of some algorithms.

The methods based on implicates as del Val's [1994] and Marquis' [1995], yield upper bounds when stopped before the entire compilation is finished. As for each implicate C by definition $\Sigma \models C$ holds, it also holds that $\Sigma \models PI_n(\Sigma)$, with $PI_n(\Sigma)$ denoting all implicates computed after n steps of one of the algorithms described in Section 2.2.2. Hence $PI_n(\Sigma)$ is an upperbound of Σ.

The methods computing implicants like Schrag's [1996a], yield lower bounds when stopped before the entire compilation is finished. As for each implicant D of Σ it holds by definition that $D \models \Sigma$, it follows that whenever $D \not\models Q$ for each already computed implicant and for some query Q that $\Sigma \not\models Q$. Hence, the computed implicants form a lowerbound of the theory Σ.

2.2.3.2 Horn approximations

In [Selman and Kautz, 1991] an original method to approximate knowledge bases was developed which was extended in [Selman and Kautz, 1996]. The idea is to compile a knowledge base into a formula which belongs to a syntactic class which guarantees polynomial-time inference.

In the method developed in [Selman and Kautz, 1991] a knowledge base is approximated by a Horn formula. The basic idea is to bound a set of models of the original theory from below (i.e., complete) and from above (i.e., sound) which is formalised in the following definition.

Definition 2.2.4 *Let Σ be a set of clauses. The set Σ_{LB} and Σ_{UB} of Horn clauses are respectively a Horn lower bound and a Horn upper bound of Σ if and only if*

$$\mathcal{M}(\Sigma_{LB}) \subseteq \mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{UB})$$

or, equivalently,

$$\Sigma_{LB} \models \Sigma \models \Sigma_{UB}.$$

Instead of using any pair of bounds to characterise the original theory, we would like to use the best possible bounds. This leads to a *greatest* Horn lower bound and a *least* Horn upper bound.

Definition 2.2.5 *Let Σ be a set of clauses. The set Σ_{GLB} of Horn clauses is a greatest Horn lower bound of Σ if and only if $\mathcal{M}(\Sigma_{GLB}) \subseteq \mathcal{M}(\Sigma)$ and there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma_{GLB}) \subset \mathcal{M}(\Sigma') \subseteq \mathcal{M}(\Sigma)$.*

Definition 2.2.6 *Let Σ be a set of clauses. The set Σ_{LUB} of Horn clauses is a least Horn upper bound of Σ if and only if $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma_{LUB})$ and there is no set Σ' of Horn clauses such that $\mathcal{M}(\Sigma) \subseteq \mathcal{M}(\Sigma') \subset \mathcal{M}(\Sigma_{LUB})$.*

Each theory has a unique LUB (up to logical equivalence), but can have many different GLBs.

As shown in Figure 2.2, inference can be approximated by using the Horn GLBs and Horn LUBs. In this way, inference could be unsound or incomplete, but it is always possible to spend more time and use a general inference procedure on the original theory.

Similar to Horn bounds other bounds can also be used. In [Selman and Kautz, 1996] the GLB and LUB are computed, which only contain unit clauses (i.e., substitute in the definitions above unit clause for Horn clause). Their experimental results show that such a restricted language for the bounds can already lead to a substantial savings in computation time.

2.2.4 Approximation in ABox Reasoning

by JEFF 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 [Haarslev and Möller, 2001a, Horrocks, 1998], 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 — even in scenarios where scalability matters.

In this section, we describe an approach to ABox reasoning that restricts the language and 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.

2.2.4.1 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; such concepts will be called *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 $A_Q \cup A'_Q$.

It can be shown that the above procedure is sound and complete. 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.

2.2.4.2 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, one 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 should be avoided), the optimised Instance Store only checks for syntactic equality using a database lookup.[2] Individuals are grouped into equivalence sets, where each individual in the set is asserted to be an instance of a

[2]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.

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, repeated computations of sub and super-concepts for the same concept (e.g., when repeating a query) can be avoided 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 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.

2.2.4.3 Implementation

The Instance Store has been implemented 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, 1998] and RACER [Haarslev and Möller, 2001b]. It also connects to a DB via standard interfaces, and has been tested with HSQL[3], MySQL[4] and Oracle[5].

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

Figure 2.3: Instance Store basic functionality

The basic functionality of the Instance Store is illustrated by Figure 2.3. 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 $i : C$

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

(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.

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.[6]

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

Chapter 3

Distributed and Modular Knowledge Representation & Reasoning

by LUCIANO SERAFINI

3.1 Introduction

A promising way to deal with large ontologies is to *decompose* them into a collection of smaller, more specific ontologies, which, together with the relations between them, constitute a representation that is semantically equivalent to the original ontology. Conversely, it may be desirable to *compose* a set of ontologies into a coherent network of ontologies that can be referred to as a single entity.

In both cases, the ultimate ontology is *modular* - it comprises a set of autonomous modules, which are interrelated by semantically meaningful links. Figure 3.1 depicts a very high level reference model of such a modular ontology.

The objective of this section of the report is to give an overview of the frameworks that have been developed in various ontology-related fields of computer science and artificial intelligence in order to represent and reason with modular and distributed knowledge.

Like in the previous chapter we don't restrict in our analysis ourselves to considering approaches which are specifically and explicitly designed to deal with ontologies. Moreover it's relevant to also take into account certain rather more general approaches to distributed knowledge bases, distributed knowledge representation, and distributed databases.

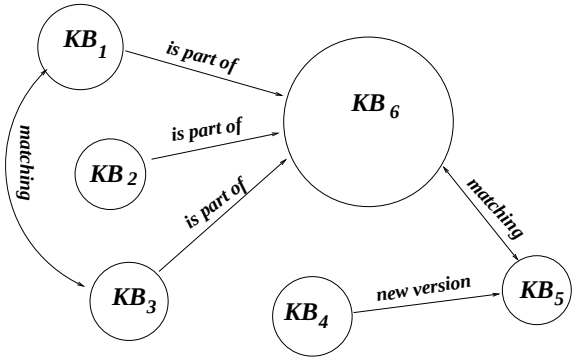

Figure 3.1: Modular ontology reference model

3.2 Frameworks

We distinguish between *modularisation* (decomposition), *integration* (compile time composition), and *coordination* (run time composition) based approaches. With respect to this classification we consider the following frameworks:

- Modularisation
  - Database Modularisation
  - Partition-Based Reasoning
- Integration
  - Ontology Integration
  - Database Integration
- Coordination
  - Ontology Coordination
    - Distributed Description Logics
    - C-Owl
  - Database Coordination
    - Peer to Peer Databases
    - Local Relational Model
    - Cooperative Information Agents
  - Contextual Reasoning
    - Propositional Logic of Context
    - Multi-Context Systems
  - XML namespaces
  - Emergent Semantics

3.3 Evaluation criteria

We summarise each of these frameworks with respect to the following criteria (the abbreviations will be used in a unified table at the end of each section):

COMPOS Does the framework provide for composition or decomposition of ontologies, or knowledge in general? Or does it enable both?

SYNTAX Does the framework provide a formal language for the specification of modular ontologies and relations between them?

SEMANT Does this language come with a formal semantics?

HETERO Does the framework allow for heterogeneous representation of individual modules?

REASON Is the framework equipped with concrete decision procedures?

SYSTEM Are there any systems that implement these decision procedure?

The first criterion asks for the basic principle behind a framework. This fundamental criterion is a first characterisation of the different frameworks and helps to separate the frameworks.

One of the central questions in an modularisation/distribution framework is how the distributed resources/modules are related. The next two criteria gather the formal characteristics of the formalism in which the relationship can be expressed. Only approaches with a clear and formal syntax and semantics are useful for the semantic web because the definition should not depend on the behaviour of an implementation.

Also in the semantic web it can not be ensured that all information is represented in one formalism. Therefore the framework should be able to handle resources with heterogeneous representations. The forth criterion asks for this ability.

The last two criteria are concerned with the implementation. They allow to identify those approaches which solve their feasibility and try to prove their usefulness in practical situations.

The table 3.1 gives an overview of this summarisation.

APPROACH	(DE)COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
Coop. Information Systems	compose	yes	yes	yes	no	no
C-OWL	compose	yes	yes	yes	no	no
DFOL	both	yes	yes	yes	yes	yes
DB Modularisation	decompose	yes	yes	yes	no	no
DB Integration	compose	yes	yes	yes	no	no
Multi Context Systems	both	yes	yes	yes	yes	no
Partitioned repres.	decompose	yes	yes	yes	yes	yes
Prop. Logic of Context	none	yes	yes	no	no	no
Partitioned Theorem Proving	decompose	no	no	yes	yes	yes
P2P Datbases	compose	no	no	yes	no	yes
XML namespaces	compose	yes	no	no	no	yes

Table 3.1: Overview of the characterisation of all frameworks

3.4 Modularisation

3.4.1 Database Modularisation

by STEFANO SPACCAPIETRA

Motivation Traditional database technology has addressed modularisation in different ways and under different headings. At the data instance level, the work on fragmentation in distributed databases corresponds to a modularisation aiming at splitting the database into possibly overlapping subsets to be stored as local databases at different sites. This type of modularisation is basically human-driven (i.e., defined by the database designer), although a number of algorithms have been proposed to optimise data distribution versus access requirements. This technology is currently available in major commercial DBMS. At the metadata (schema) level, modularisation is a concern in the management of large schemas, with hundreds or even thousands of object and relationship types. Clearly, no human DB administrator can handle such complexity at a glance, and no normal browsing techniques can provide an easy-to-understand visualisation of such huge schemas. Finally, another form of modularisation is personalisation, a technique leading to modules defined by their owner.

Description Modularisation is often seen as cutting something into pieces, like when using scissors to cut a newspaper page into its components (articles and advertisements). Yet modularisation does not exclude cutting a whole into overlapping parts, as in the process of defining data fragments for distributed storage. This is specifically the case when defining a distributed database, i.e. a database that is centrally managed but whose data is split into sub-databases stored at multiple sites. In this setting, each element of the database schema (e.g., a relational table, an object type) is candidate for fragmentation. Fragments are defined by a combination of selection criteria (known as horizontal

splitting) and projection criteria (known as vertical splitting). Fragments overlap: all fragments need to include a key that allows correct re-composition of the whole from the fragments. Each fragment may also be stored in multiple sites if duplication is desirable for performance reasons. Modularising a huge database schema to make it more manageable can be done automatically or based on manual specifications. In automatic approaches, concepts of semantic closeness are defined to determine which elements of the schema can be clustered together to build a higher-level element. For example, an object type and its subtypes can be clustered together [Teorey *et al.*, 1989]. Clustering can be done iteratively by applying a sequence of criteria. Clusters at each level can be seen as modules in the original schema. In object-oriented databases, identification of interesting clusters may additionally take into account use relationships showing which methods use which object types. Thus, an object type can be clustered with the other object types that are accessed while executing its associated methods. In terms of modularisation through personalisation, the traditional database approach relies on the view definition mechanism. In relational databases, views can be freely defined as SQL queries. The subschema that is relevant for a given user can then be composed by gathering all tables and views that are accessible to the user (as specified by the definition of access rights attached to each table and view). To a certain extent, modularisation, contextualisation (i.e., modules defined as all elements relevant to a given context), and personalisation (where the context is the data owner) can all be subsumed by a generic technique that: 1) identifies the different pieces (and combinations of pieces) we are interested in (module, context, person or agent), and 2) defines for each data item (e.g., an axiom, a value, an object type, an ontology) for which piece (or combination of pieces) it is relevant for or belongs to (e.g., this axiom belongs to module m1, this contribution is relevant for contexts Kweb2.1 and DIP1.5). Such a generic technique has been developed at EPFL-LBD to enhance conceptual data models with support of multiple coexisting perceptions, leading to multiple representations of the same real world phenomena (e.g., objects, links, properties) in the same database. The basic mechanism that is used is to associate to each element the identification of the perceptions it is relevant for. This is done from the meta-level (data descriptions in the database schema) down to the data level (instances and values). The technique allows defining any component of the database as perception-varying. For example, a schema item that is shared by many perceptions may have two or more definitions, up to one per perception, and still be understood as a single data item. Similarly, the set of instances of a class as well as the value of an instance may be different from one perception to another perception. Whenever multiple perceptions lead to multiple representations, these representations may be organised separately or integrated into a single container. For example, if two perceptions of hotels coexist and require different representations, it is possible to define a single object type Hotel and have the two representations merged within the object type. This is relatively easy to achieve if one hotel in one perception maps to at most one hotel in the other perception (partial or total bijective mapping). Alternatively, it is possible to define two separate object types, and link them with a relationship type whose semantics is to express that the two linked instances represent the same real world hotel. Both solutions allow the system to be aware of the existence of multiple

representations for multiple perceptions (just as an is-a link between two object types instructs the system that the related instances result from alternative classifications of the same real world object). Consistency rules are defined to guarantee that the database is designed so that, when restricted to a single perception, the database schema is a traditional database schema and the data instances form a traditional database. Similar rules govern how transactions see the database. When a transaction adheres to a single perception, it sees a traditional database. When a transaction uses multiple perceptions, it has an integrated vision of the corresponding collection of single-perception databases and the inter-perception data that spans over the visible perceptions. In all cases, the data that is returned in response to queries is consistent and visible to the transaction (i.e., it belongs to the perception that the transaction holds). The proposed technique offers two advantages over traditional personalisation techniques such as views and roles (in the object-oriented sense). First, it allows implementing the concept of perception as a consistent subset of the whole database. In this sense a perception may be seen as a module. Second, it allows establishing bridges between perceptions and using them for multi-perception applications, including applications checking the semantic consistency of the different representations. Third, it allows keeping into a single schema and database all the perceptions, so that an integrated view is possible, in particular for the database administrator. This has immediate benefits in reducing error risk in defining the database and in making its definition evolutionary. Fourth, as the technique applies to all database items, it can straightforwardly be ported to data models that have non-standard features, such as constructs for the description of spatial and temporal information. Finally, the technique does not depend on the underlying data model. It may be implemented as an extension to any data model. For instance, its application to an ontology language has been proposed as a solution to support context-dependent ontologies.

Results and Applications Modularisation as fragmentation in distributed database systems is part of the functionality routinely supported by major DBMS on the marketplace. Similarly for modularisation as personalisation through the view mechanism. As far as we know, modularisation as schema decomposition has interesting results at the academic level only [Massart, 1997]. Modularisation through definition of perception-varying schemas has been implemented in a prototype and tested in two case studies as part of an EEC sponsored project [Consortium, 2000].

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
decompose	yes	yes	yes	no	no

3.5 Partition-Based Reasoning

by FLORIS ROELOFSEN & HOLGER WACHE

Motivation There is a growing interest in building large (common sense) knowledge bases [Lenat and Guha, 1990, Cohen *et al.*, 1998, Fikes and Farquhar, 1999]. General-purpose reasoners for such knowledge bases tend to suffer from combinatorial explosion. One promising approach to resolving this problem, pursued at the Knowledge Systems Laboratory at Stanford University, is to structure knowledge bases into multiple domain- or task-specific partitions, and then to *exploit* that structure so as to make reasoning more focused and efficient.

Description The basic idea is to partition a first-order or propositional theory T into tightly coupled subtheories $T_1, \ldots, T_n$, which are related by (and should agree on) the overlap between the signatures of their respective languages. Amir and McIlraith [2004] describe a greedy algorithm that automatically establishes such a partitioning. The algorithm optimises the size of each partition and the overlap between partitions. The algorithm manipulates a graphical representation of the axioms that constitute the initial theory, which resembles a dual constraint graph in constraint satisfaction problems. It yields a graph, whose nodes represent partitions and whose arcs represents the languages shared by overlapping partitions. Efficient graph-based algorithms can be applied to convert this representation into a tree structure.

A family of message-passing (MP) algorithms has been developed for reasoning with such tree-structured partitions of propositional or first-order axioms. Reasoning is performed locally in each partition, and relevant results are propagated toward the goal partition so as to provide a global solution. Different (possibly special-purpose) reasoning engines may be used in different partitions. Global soundness and completeness follow from the soundness and completeness of each local reasoner. Performance is shown to be linear in the tree structure, and worst-case exponential within the individual partitions. To maximise the effectiveness of the algorithms, (1) the coupling between partitions should be minimised, in order to reduce information being passed between them, and (2) local inference within individual partitions should be maximally "focused", that is, it should involve a minimised set of tightly related variables.

It is useful to observe that the semantics of a partitioned theory can be seen as the projection of a global semantics for T onto each local language T_i. Or, the other way around, a model for T is the combination of one model for each T_i. In other words, a model of T_i is *partial* in the sense that it concerns only part of the total knowledge base, but it is *complete* in the sense that it expresses a unique interpretation (complete knowledge) of that part of the knowledge base. As regards the inter-partition relationships, it is worth remarking that the approach is limited to considering overlap between *pairs* of partitions, which is exclusively described by *symmetric* (as opposed to *directional*) relations.

Results This approach has only fairly recently been initiated. Amir and McIlraith [2004] provide a comprehensive presentation of the theoretical results obtained so far. Another reliable source of information is the project's website:
`http://www.ksl.stanford.edu/projects/RKF/Partitioning/`.

A first empirical analysis [MacCartney *et al.*, 2003] shows that using message-passing algorithms on partitions which are generated automatically does in general not yield any efficiency improvements. Only when specialised and adapted to the particular process of inference the algorithms are shown to be more efficient than global reasoning systems.

Applications Prototypes of the algorithms was implemented into Stanford and SRI's high-performance knowledge bases [Fikes and Farquhar, 1999, Cohen *et al.*, 1998]. Detailed results, however, have not been reported yet. Initial efforts have been directed toward the development of a partition-based theorem prover [MacCartney *et al.*, 2003] as well as a partition-based planner [Amir and Engelhardt, 2003].

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
decomp.	yes	yes	yes	yes	yes

3.6 Integration

3.6.1 Ontology Integration

by LUCIANO SERAFINI

Motivation With the growing availability of large and specialised online ontologies, the questions about the combined use of independently developed ontologies have become even more important. Michel Klein in [Klein, 2001] proposes an exhaustive list of tasks that have to be taken into account in managing heterogeneous and partially autonomously developed ontologies. In this list *ontology integration* is defined as

> "... Creating a new ontology from two or more existing ontologies with overlapping parts, which can be either virtual or physical".

The most critical point in ontology integration is the problem of discovering the *overlapping parts* between two or more ontology. This problem, however, is out of the scope of this work-package. Actually, it is the main task of the Work-Package WP2.2 Heterogeneity. As it is explained in the deliverable D2.2.1 of this WP the overlapping between two ontologies can be expressed via mappings. Roughly speaking a mapping is an expression

stating that the semantics (meanings) of $term_i$ in an ontology O_1 and $term_2$ in the ontology O_2 are in a certain *semantic relation*. (e.g., more-general-than, less-general-then, equivalent, disjoint, etc.).

The problem In this document, therefore, we are not interested in the approaches that allow to *discover* the mappings. We suppose a set of ontologies with a set of mappings between them, and we look at the approaches that allow us to integrate these ontologies using mappings.

Proposed solutions An overview of the state-of-the-art methodologies and tools for ontology integration can be found in [Wache *et al.*, 2001]. In the following we report only a subset of the most significant approaches to ontology integration.

From our perspective, the approaches of finding commonalities between two different ontologies A and B and generating a new integrated ontology C in which these commonalities are represented, can be clustered in two groups depending on whether the new ontology C replaces A and B, or it is used only as a mediator between A and B which are parts of the integrated ontology.

Ontology Merging The result of the integration is a unique ontology, in which every source ontology is explicitly represented together with the mappings between them. The tools for ontology merging usually integrate also an alignment algorithm which is capable of finding matches between the two ontologies. Examples of these tools are:

PROMPT [Noy and Musen, 2001] by Natasha Noy. PROMPT allows the management of multiple ontologies in Protégé, mainly to compare versions of the same ontology, move frames between included and including project, *merge two ontologies into one* extract a part of an ontology.

MoA by Jaehong Kim. MoA [Kim,] is an OWL ontology merging and alignment library and tool composed by a set of basic libraries for add, remove, merge and align operations on an ontology model, and similarity comparison between two concepts.

HCONE by Konstantinos Kotis and George A. Vouros. The HCONE approach [Kotis and Vouros, 2004] to ontology merging is based on (a) capturing the intended informal interpretations of concepts by mapping them to WordNet senses using lexical semantic indexing, and (b) exploiting the formal semantics of concepts definitions by means of description logics reasoning services.

Ontomerge by D. Dou, D. McDermott and P. Qi. Ontomerge [Dou *et al.*, 2002] is a tool for ontology translation which is based on a method for ontology merging. The merge of two related ontologies is obtained by taking the union of the terms and the axioms defining them, and by adding bridging axioms. Bridging

axioms not only expresses bridges between the semantics of the terms in two related ontologies but also make this merge into a complete new ontology for further merging with other ontologies.

FCA-merge by G. Stumme and A. Maedche. Ontologies can be merged by FCA-merge [Stumme and Maedche, 2001] following a bottom-up approach which offers a global structural description of the merging process. For the source ontologies, it extracts instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances it applies Formal Concept Analysis techniques to derive a lattice of concepts The produced result is manually transformed to the integrated ontology, and acts as a global ontology that allows access to the local "federated ontologies".

Ontology Mediation[1] The result of the integration is a *global ontology* that allow access to the source ontologies via mappings between global ontologies and source ontologies. An example that can be classified under this cluster is

Formal Framework for Ontology Integration Calvanese et. al in [2001] provides a formal framework for specifying the mapping between the global ontology and the source ontologies. In this paper it is argued that mappings should be viewed as queries from the global ontology to the local ones. This approach is supported by the intuition that a concept in the global ontology corresponds to a view (i.e., a query) over the other ontologies.

3.6.2 Database Integration

by STEFANO SPACCAPIETRA

Motivation The need for integrating digital data residing in different data stores exists since networks have offered functionality to transfer data from one site to another. Consequently, research on database integration has a long history, starting back in the 70s, and is still ongoing. Among the many commonalities that exist between ontologies and databases, the areas of ontology integration and database integration clearly share quite many issues and solutions. Ontology composition, as defined in section 1, is thus very similar to the approach that deals with building a federated database from a set of existing databases. Federated databases correspond to modern requirements from organisations and enterprises whose work calls for data coming from a variety of sources (the most frequent pattern today). Data warehousing is another approach that heavily depends on successful data integration. It is foreseeable that ontology integration becomes an equally essential component of future information management systems. It is therefore worth investigating which achievements from the database community may be reusable in addressing issues related to distributed and modular knowledge representation.

Description Despite the variety of approaches and proposals to be found in the overwhelming literature on database integration, a methodological consensus seems to exist for decomposing the information integration process into three steps:

- Source homogenisation: in this step all data sources are described using the same data model (i.e., existing descriptions, based on heterogeneous data models, are translated into equivalent descriptions using a single common data model). Wrappers, for instance, are tools designed to implement this homogenisation process [Wiederhold, 1992]. Homogenisation could also include other considerations, such as, for instance, conforming the schemas of the input sources to agreed design rules (e.g., normalisation). Some integration methodologies use meta-modelling techniques to enable skipping the homogenisation step [Nicolle and Yetongnon, 2001], or just achieve the same goal by directly building semantic bridges among source schemas [Parent and Spaccapietra, 2000].

- Schema and data matching: this step aims at finding and describing all semantic links between elements of the input schemas and the corresponding data. Fully automated matching is considered impossible, as a computer process can hardly make ultimate decisions about the semantics of data. But even partial assistance in discovering of correspondences (to be confirmed or guided by humans) is beneficial, due to the complexity of the task. All proposed methods rely on some similarity measures that try to evaluate the semantic distance between two descriptions [Rahm and Bernstein, 2001]. Notice that schema matching is frequently termed alignment in works on ontology integration.

- Integration: this step takes as input the data sources and the mappings in-between and produces an integrated schema and the mappings between the (global) integrated schema and the (local) existing sources. The global-to-local mappings are needed to support queries on the integrated schema. This integration step relies on the definition of a set of generic integration rules, which state how to solve all types of semantic conflicts that may exist between two data sources. Integration rules need not be deterministic, i.e. there may be alternative solutions to the same semantic conflict. The designer responsible for the integrated schema should choose an integration policy stating which solutions have to be preferred [Dupont, 1994]. Research has identified two methods to set up mappings between the integrated schema and the input schemas [Cali *et al.*, 2003]. One method is called GAV for Global As View, and proposes to define the integrated schema as a view over input schemas. The other method is called LAV, and proposes to define the local schemas as views over the integrated schema. GAV is usually considered simpler and more efficient for processing queries on the integrated database, but is weaker in supporting evolution of the global system through addition of new sources. LAV generates issues of incomplete information, which adds complexity in handling global queries, but it better supports dynamic addition and removal of source. Proposals also exist that suggest merging the two techniques.

A current trend is to develop ontology-based integration techniques. Here, one or more ontologies play the role of the human expert in assessing the likeliness of a semantic similarity, or in suggesting alternatives (e.g., synonyms) in measuring similarity. This technique is likely to lead to solutions that could in particular be used to implement information services for mobile users. In a mobility context, it is not possible to rely on advice from human experts to solve semantic heterogeneity. Everything has to be done by agents that rely on locally available information to provide the requested service. There is a need for integration, but the process has to be done on the fly, which means that completeness and to some extent correctness has to be traded off for quick response. Finally, database integration issues and their solutions are currently being transferred to the very similar problem of ontology integration (also called ontology fusion or merging). Although in ontology integration the focus has been on terminological comparisons, it is clearly evolving towards the full complexity of semantic conflicts as identified by research in database integration. The only tangible difference between database integration and ontology integration seems to be in the fact that more often, in the ontology world, some ontologies may take a form of leadership (e.g., WordNet for ontologies about terminology) over the other ontologies. In this case, the integration process (which usually gives no preference to a source over the others) may turn into a conformance process, where integration means that a new source is used to enrich an existing reference ontology.

Results Results in information integration include the development of a large know-how for the domain as well as tools and prototypes for the various tasks composing the integration process. There is, however, no tool that provides a single compete solution for the whole process. Regarding homogenisation, there is abundance of prototypes for data model translation (e.g., between the relational data model and the object-oriented data model) and wrappers. In terms of tools, there is a family of CASE database design tools that accept ER-like specifications and turn them into relational schemas. UML is a broader approach that allows expressing specifications for both static and dynamic aspects in data and process modelling and generates a relational design. DBMain [Hainaut, 1999] [Thiran and Hainaut, 2001] is a more flexible approach, which can convert a variety of input formats into one of many possible output formats. Similar goals are achievable with tools and languages oriented towards data exchange (e.g., XML and, in the geographic domain, FME). Regarding tools helping in finding semantic links among input sources, a number of prototypes have been developed. A significant recent effort is the Rondo project [Melnik *et al.*, 2003]. A detailed discussion on the nature of semantic links that can be established, and how to characterise them, can be found in [Parent and Spaccapietra, 2000]. Matching algorithms that operate at the instance level have been developed for geographic databases [Badard, 1999]. Regarding the final integration step, the variety of semantic conflict that can arise has been largely investigated [Larson *et al.*, 1989], [Sheth and Kashyap, 1992] and the issue can be considered as comprehensively analysed. Integration rules have also been extensively discussed (e.g., in [Parent and Spaccapietra, 2000]).

Applications As stated above, a number of tools exist that are used in one of the phases of the information integration process. Full integration in real applications remains a basically manual activity.

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
compose	yes	yes	yes	no	no

3.7 Coordination

3.7.1 Ontology Coordination

3.7.1.1 Distributed Description Logics

by ANDREI TAMILIN

Motivation The current situation on the web is characterised by a steady proliferation of local ontologies. We use the term ”local” to emphasise the fact that each ontology describes the domains of interest from its local and subjective point of view, using the language of desired complexity. In this circumstances, the same domain can be covered by different local ontologies in heterogeneous ways. The common solution for resolving semantic interoperability problem between such ontologies is based on the discovery of semantic mappings relating concepts of these ontologies. Having a set of ontologies related via semantic mappings is not enough to guarantee the interoperability. One has to provide the capability of reasoning within such system. Reasoning is necessary for checking consistency of mappings, discovering new ones, and computing new ontological properties that derive from the combination of ontologies and mappings. To reflect the situation given above, the reason-able formalisation for dealing with multiple ontologies interrelated via semantic mappings is required.

Description The main purpose of *Distributed Description Logics (DDL)* is to provide a syntax and semantics, formalising the case of multiple local ontologies pairwise linked by semantic mappings. In DDL, local ontologies are represented by description logic theories (T-boxes), while semantic mappings are represented by bridge rules. We briefly recall the definition of DDL as given in [Borgida and Serafini, 2003].

Given a nonempty set of local ontologies $\{O_i\}_{i\in I}$, let $\{\mathcal{D}L_i\}_{i\in I}$ be a collection of description logics, and $\mathcal{T}_i$ be T-boxes in $\mathcal{D}L_i$. T-boxes $\mathcal{T}_i$ are formalisations of ontologies O_i in description logics $\mathcal{D}L_i$. To avoid confusions, every description C of $\mathcal{T}_i$ is labelled with its index, written $i : C$.

Semantic mappings between pairs of ontologies represented in DDL by bridge rules between pairs of corresponding T-boxes. A *bridge rule* from i to j is an expression of the following two forms:

1. $i : C \xrightarrow{\sqsubseteq} j : D$, an *into-bridge rule*
2. $i : C \xrightarrow{\sqsupseteq} j : D$, an *onto-bridge rule*

where C and D are two concepts of $\mathcal{T}_i$ and $\mathcal{T}_j$ respectively. Bridge rules from i to j express relations between i and j as viewed from the *subjective* point of view of the j-th ontology. Intuitively, the into-bridge rule $i : C \xrightarrow{\sqsubseteq} j : D$ states that, from the j-th point of view the concept C in i is less general than its local concept D. Similarly, the onto-bridge rule $i : C \xrightarrow{\sqsupseteq} j : D$ expresses the fact that, according to j, C in i is more general than D in j.

A collection of T-boxes $\{\mathcal{T}_i\}_{i\in I}$, and a collection of bridge rules between them $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{i\neq j\in I}$, form a *distributed T-box* in DDL $\mathfrak{T} = \langle\{\mathcal{T}_i\}_{i\in I}, \mathfrak{B}\rangle$.

The semantics of DDL represents a customisation of the Local Models Semantics for Multi Context Systems [Ghidini and Giunchiglia, 2001a, Ghidini and Serafini, 2000]. The underlying idea, is that each ontology $\mathcal{T}_i$ is *locally interpreted* on its *local domain*, and characterised by local interpretation $\mathcal{I}_i$. Local domains are pairwise connected by a set of *domain relations* r_{ij}, representing a possible way of mapping the elements of i-th local domain into j-th local domain, seen from j's perspective. Domain relations, therefore, give a semantics for bridge rules.

A collection of local interpretations $\mathcal{I}_i$ for each $\mathcal{T}_i$ on local domains $\Delta^{\mathcal{I}_i}$, and a family of domain relations r_{ij} between these local domains, define a *distributed interpretation* $\mathfrak{I} = \langle\{\mathcal{I}_i\}_{i\in I}, \{r_{ij}\}_{i\neq j\in I}\rangle$ of distributed T-box $\mathfrak{T}$.

Results [Borgida and Serafini, 2003] introduces basic definitions of DDL framework. [Serafini and Tamilin, 2004] characterises reasoning in DDL as a problem of calculating a *global subsumption*. As well [Serafini and Tamilin, 2004] describes a sound and complete reasoning procedure, which is based on a *distributed tableau* and constitutes a method for combining existing tableaux reasoning procedures for description logics.

Applications Distributed tableau algorithm for computing global subsumption in DDL is implemented in the distributed reasoning system *D-Pellet* [Serafini and Tamilin, 2004]. D-Pellet is a mapping-aware extension of open source Pellet OWL DL Reasoner[2], implementing description logics tableau reasoning algorithm. Roughly, every D-Pellet maintains a set of local OWL ontologies, and C-OWL mappings established between local ontologies and ontologies of foreign D-Pellets. D-Pellets are organised into a peer-to-peer network (currently acyclic) and are capable of providing global reasoning within local ontologies they maintain.

[2] `http://www.mindswap.org/2003/pellet`

To facilitate the process of development of an ontology, mapped via C-OWL mappings to other ontologies, and to provide mapping-aware ontology classifier, D-Pellet can be integrated into Protégé development platform as a plug-in [Serafini and Tamilin, 2004].

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
both	yes	yes	yes	yes	yes

3.7.1.2 C-OWL

by LUCIANO SERAFINI

Motivation The need for explicit models of semantic information (terminologies and background knowledge) in order to support information exchange in the semantic web has been widely acknowledged by the research community. Several different ways of describing information semantics have been proposed and used in applications. However we can distinguish two broad approaches which follow somehow opposite directions:

Ontologies are *shared* models of some domain that encode a view which is common to a set of different parties [Patel-Schneider *et al.*, 2003];

Contexts are *local* (where *local* is intended here to imply *not shared*) models that encode a party's view of a domain [Giunchiglia, 1993a, Ghidini and Serafini, 1998, Ghidini and Giunchiglia, 2001b].

Ontologies are best used in applications where the core problem is the use and management of common representations. Contexts, instead, are best used in those applications where the core problem is the use and management of local and autonomous representations with a need for a limited and controlled form of globalisation (or, using the terminology used in the context literature, maintaining *locality* still guaranteeing semantic *compatibility* among representations [Ghidini and Giunchiglia, 2001b]). Contexts and ontologies have both strengths and weaknesses, see [Bouquet *et al.*, 2003a] for a deeper discussion, and we believe that they should be integrated in the representational infrastructure of the Semantic Web.

The C-OWL language is an extension of the OWL language for expressing ontologies, with *bridge rules*, which allow to relate concepts, roles and individuals in different ontologies. We call a set of bridge rules between two ontologies a *context mapping*. Thus a *contextual ontology* is an OWL ontology embedded in a space of other OWL ontologies and related to them via context mappings. C-OWL can be used to express alignment (i.e., weak integration) of a set of independently developed ontologies.

Description OWL syntax is obtained by extending OWL syntax with expressive mappings. The semantics of C-OWL is obtained by modifying the original OWL semantics [Patel-Schneider *et al.*, 2003]. The semantic of C-OWL uses the ideas and notions originally developed in [Borgida and Serafini, 2003], which is based on the semantics of context (the, so called, Local Models Semantics [Ghidini and Serafini, 1998]).

The complete description of C-OWL is given in [Bouquet *et al.*, 2003a]. The main items defined in C-OWL are:

Ontology An ontology written in OWL

Local interpretation of an ontology Either a model of the T-box and the A-box defined by the ontology, or a *hole* i.e., an interpretation of an inconsistent ontology. In a hole every subsumption is satisfied.

Ontology space: Is an indexed set of local ontologies.

Bridge rule: Is a special kind of inference rules from an ontology O_i to an ontology O_j, which allow to relate concepts, roles and individuals of the ontology O_i with the concepts, roles and individuals of the ontology O_j. We call a set of bridge rules between two ontologies a *context mapping*.

Contextual ontology: It is a local ontology plus a set of bridge rules (context mappings). We sometimes write context meaning contextual ontology.

Context space: A context space is the pair OWL space $\langle i, O_i \rangle$ (of local ontologies) family M_{ij} of (context) mappings from i to j, for any pair i, j

Interpretation for context spaces . It is a function that associates to each contextual ontology a local interpretation, and to each pair of local ontologies O_i and O_j a domain relation r_{ij} between the domain of interpretation of O_i and O_j.

Results So far C-OWL is a specification language, an abstract syntax, a concrete syntax and a semantics. The abstract syntax is based on the description logics for OWL, and bridge rules. The concrete syntax is OWL extended with mappings. The following is an example of C-OWL mappings.

```
<cowl:mapping>
  <rdfs:comment>Example of a mapping of wine into vino</rdfs:comment>
  <cowl:sourceOntology  rdf:resource="http://www.example.org/wine.owl"/>
  <cowl:targetOntology  rdf:resource="http://www.example.org/vino.owl"/>

  <cowl:bridgRule cowl:br-type="equiv">
     <cowl:sourceConcept rdf:resource="http://www.example.org/wine.owl#wine"/>
     <cowl:targedConcept rdf:resource="http://www.example.org/vino.owl#vino"/>
  </cowl:bridgRule>

  <cowl:bridgRule cowl:br-type="onto">
     <cowl:sourceConcept rdf:resource="http://www.example.org/wine.owl#RedWine"/>
     <cowl:targedConcept rdf:resource="http://www.example.org/vino.owl#VinoRosso"/>
```

```
</cowl:bridgRule>

<cowl:bridgRule cowl:br-type="into">
   <cowl:sourceConcept rdf:resource="http://www.example.org/wine.owl#Teroldego"/>
   <cowl:targedConcept rdf:resource="http://www.example.org/vino.owl#VinoRosso"/>
</cowl:bridgRule>

<cowl:bridgRule cowl:br-type="compat">
   <cowl:sourceConcept rdf:resource="http://www.example.org/wine.owl#WhiteWine"/>
   <cowl:targedConcept rdf:resource="http://www.example.org/vino.owl#Passito"/>
</cowl:bridgRule>

<cowl:bridgRule cowl:br-type="incompat">
   <cowl:sourceConcept rdf:resource="http://www.example.org/wine.owl#WhiteWine"/>
   <cowl:targedConcept rdf:resource="http://www.example.org/vino.owl#VinoNero"/>
</cowl:bridgRule>
```

Applications The need for terminology integration has been widely recognised in the medical area leading to a number of efforts for defining standardised terminologies. The notion of contextualised ontologies can provide such an alignment by allowing the co-existence of different, even in mutually inconsistent models that are connected by semantic mappings. As discussed above, the nature of the proposed semantic mappings satisfies the requirements of the medical domain, because they do not require any changes to the connected ontologies and do not create logical inconsistency even if the models are incompatible.

In [Stuckenschmidt *et al.*, 2004] an experience from using C-OWL for the alignment of medical ontologies Galen, Tambis, and UMLS is reported.

Galen One of the results of the GALEN project [Rector and Nowlan, 1993] is their GALEN Coding Reference model. This reference model is an ontology that covers general medical terms, relations between those terms as well as complex concepts that are defined using basic terms and relations. For the study we used an OWL translation of the GALEN model that contains about 3100 classes and about 400 relations.

Tambis The Tambis Ontology [Baker *et al.*, 1999] is an explicit representation of biomedical terminology. The complete version of Tambis contains about 1800 terms. The DAML+OIL version we used in the case study actually contains a subset of the complete ontology. It contains about 450 concepts and 120 Relations.

UMLS The Unified Medical Language System UMLS [Nelson and Powell, 2002] is an attempt to integrate different medical terminologies and to provide a unified terminology that can be used across multiple medical information sources. Examples of medical terminologies that have been integrated in UMLS are MeSH [Nelson *et al.*, 2001] and SNOMED [Côté, 1993]. In our case study, we used the UMLS semantic network. The corresponding model that is available as OWL file contains 134 semantic types organised

in a hierarchy as well as 54 relations between them with associated domain and range restrictions.

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
compose	yes	yes	yes	no	no

3.7.2 Database Coordination

3.7.2.1 Cooperative Information Agents

by HOLGER WACHE

Motivation The information agent technology in general and cooperative information agents in particular emerged as response to the continuing information overload where the data, system and semantic heterogeneity causes one of the problems. The idea behind information agents is the development and utilisation of autonomous computational software entities which access multiple, heterogeneous and distributed information sources while retrieving, analysing, integrating and visualising of the information [Klusch, 2001]. Especially their abilities for cooperation and to be mobile are very interesting in the context of distributed reasoning.

Description Intelligent agents differ from traditional software systems in their pro-activeness (taking the initiative to reach given objectives), their reactivity or deliberations (perceiving the environment), and acting social in groups when it is needed (cf. [Wooldridge and Jennings, 1995]). Information agents are special kinds of intelligent agents and are defined as agents which have access to one or more heterogeneous and distributed information sources and acquire, mediate and maintain relevant information [Klusch, 2001]. In contrast to existing federated database systems information agents have the ability of pro-active information discovery leading to the area of cooperative information systems originated from [Papazoglou *et al.*, 1992].

According to one or more of the following features information agents can be classified into [Klusch, 1999]:

Non-cooperative or collaborating agents have the ability to cooperate and to collaborate for a given task or not.

Adaptive agents are be able to adapt themselves to any changes in the agent network or in the environment.

Rational agents try to increase their own benefits in an economical sense.

Mobile agents can travel autonomously through the internet.

Cooperative Information Agents (CIA) are based on a meaningful communication and coordination. First of all the agents need to understand the meanings of concepts and notions across multiple domains. Related efforts include (semi-)automated, ontology-based interoperation with knowledge representation and ontologies also in the area of intelligent agent technology.

In [Omicini *et al.*, 2001] a comprehensive overview of coordination mechanism is given. High-level collaboration of an information agent with other agents includes, for example, brokering, matchmaking, negotiation, coalition formation, and collaborative (social) filtering. A matchmaker tells a service requester which is the appropriate service provider for the request. A broker also forwards the request to the service provider and returns the answer to the requester. Negotiating allows a couple of agents to agree on a common and shared statement by applying e.g. trading mechanism like auctions. In order to gain and share benefits CIAs can also negotiate for stable (temporary) coalitions. With collaborative filtering agents communicate about the user's preferences in order to provide information items to the user which are similar to his previous requests and will match his interests.

Mobile agents are developed to be able to travel autonomously in the internet. Performing their tasks on different servers allows to balance the server performance and can exhibit intelligent strategies for actively searching information. They are suitable especially in dynamically changing environments, where in the case of a wireless network the connection may be lost sometimes [Klusch, 2001]. A comprehensive overview of mobile agent systems and their application to distributed information retrieval is given in [Brewington *et al.*, 1999].

Applications and Results Collaborative information agents are successfully applied to a wide field. Examples are Infosleuth [Woelk and Tomlinson, 1995] and IMPACT [Subrahmanian *et al.*, 2000] for information retrieval in a distributed and heterogeneous environment. InfoSpider [Menczer and Monge, 1999] is an example of an adaptive CIA. Mobile Agents are mainly applied to the area of telecommunications, where agents may be a part of the decentralised architecture for the next network generation, which integrates mobile devices like smart phones or PDAs with the (wireless) internet in a more sophisticated way [Pentland, 1998].

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
no	no	no	yes	no	no

3.7.2.2 Peer to Peer Databases

by ILYA ZAIHRAYEU

Motivation Existing interest from the research community to Peer-to-Peer (P2P) databases is motivated by the promises laying beyond this paradigm. First, it allows for integration of databases in a totally decentralised distributed manner by specifying *mappings* between pairs of databases (e.g., [Giunchiglia and Zaihrayeu, 2004]), also called *peers* or *nodes*, or (small) sets of peers (e.g., [Halevy *et al.*, 2003]). Peers use these mappings in order to (transitively) propagate queries, query results, and updates to each other. Second, totally decentralised solution allows for large scalability and fault tolerance of the network. Third, peers are largely autonomous in what data they share and what schema they use to describe the data, in what other peers they establish mappings with, in when they are online or offline, etc. All this gives low startup costs (no need to adjust local database), ease of participation (once online, the service is available), low contribution requirements for a single party (due to distributed processing), and exhaustive query results (due to collective query answering).

Due to the high autonomy of peers, adoption of data integration technologies, relying on the notion of a *global schema* [Lenzerini, 2002], is not possible. Peers come and go, import arbitrary schemas to the network, and may change their schemas at runtime. In such settings maintenance of a global schema becomes too expensive, and even infeasible. Therefore new methodologies, theories, mechanisms, and technologies must be provided in order to address the new challenges.

Description The description in this section base on [Giunchiglia and Zaihrayeu, 2004, Franconi *et al.*, 2004, Serafini *et al.*, 2003, Franconi *et al.*, 2003, Bernstein *et al.*, 2002]. Each peer on a P2P database network provides a *source* database described by a (*source*) *schema*, or supplies only the schema. In this latter case a node acts as a kind of mediator in transitive propagation of data. Peers establish pairwise mappings, called *coordination rules*, which have the following form:

$$i : CQ(x) \Rightarrow j : CQ(x, y)$$

where i, j are distinct indices denoting distinct peers, $CQ(x)$ is a conjunctive query over the schema of peer i, $CQ(x, y)$ is a conjunctive query over the schema of peer j, x is a set of variables, and y is a set of free variables. As from data integration literature [Lenzerini, 2002], this kind of mapping is called Global-Local-as-View, or GLAV.

Coordination rules solve the heterogeneity problem at the structural level (when for representing the same concepts, different databases use different names for relations and attributes, and/or use different number of relations and/or attributes), whereas the instance level heterogeneity (the same object is assigned different constants in different databases) is solved with the help of *domain relations* [Serafini *et al.*, 2003]. A domain relation, written $r_{i,j}(d_i) = d_j$, is a function that specifies that a constant d_i from the domain of

database at peer i is equal to constant d_j from the domain of database at peer j. A domain relation is not necessarily symmetric, i.e. the following may take place $r_{i,j}(d_i) = d_j$, $r_{j,i}(d_j) \neq d_i$. A typical example of this situation is currency conversion. Consider the following example:

Example 3.7.1 *Suppose we have two peers with the following schemas:*

Peer A	*Peer B*
movies(title, year, genre, budget)	*movie(title, budget, year)*
credits(name, title)	*genres(title, genre)*
	actors(name, title, role)

Peer A is acquainted with peer B w.r.t. the following coordination rule:

$$A : movies(t, y, g, b), credits(n, t) \Rightarrow B : movie(t, b, y), genres(t, g), actors(n, t, r)$$

Now suppose that peer A is in Italy and peer B is in USA. Then we need to specify domain relations function in order to translate values for $budget$ from US dollars to Euro and vice versa. Below are two examples of such a function:

$$r_{A,B}(b) = b * 1.18, r_{B,A}(b) = b * 0.84$$

At each peer, for different queries different coordination rules may be used for propagation. Thus, for any given query, and a node on the network, where the query is submitted, different propagation graphs, or *views* on the network, may take place. And, since "coordination rules graph" may be absolutely arbitrary, a view on the network may contain circles. Moreover, during the propagation of a query, the view may change, i.e. some nodes may leave the network, some nodes may join, or some coordination rules may be changed.

Results [Halevy *et al.*, 2003] proposes a language for mediating between peer schemas, which extends known data integration formalisms to a more complex architecture, and specifies the complexity of query answering in that language. [Ng *et al.*, 2003] present design and evaluation of a (relational) data sharing system, where query answering is mainly supported by mobile agents. [Serafini *et al.*, 2003] proposes a local relational model, that provides a language and a proper semantics for expressing mappings between databases, as well as introduces the notion of domain relations. [Giunchiglia and Zaihrayeu, 2004] proposes a data coordination model, where the main notions are Interest Groups and Acquaintances. The first notion allows for a global aggregation of nodes carrying similar information, while the second allows for a local logical point-to-point data exchange between databases. [Franconi *et al.*, 2004] proposes a distributed update algorithm, which is correct and complete with presence of circles in the coordination rules graph and network dynamics under certain semantics. The paper reports first experimental results which involved up to 64 database nodes.

Applications To our knowledge, Peer-to-Peer databases have not been yet applied as industrial applications, but some prototypes exist as testbeds in academia. Examples of such prototypes are: [Giunchiglia and Zaihrayeu, 2004, Franconi *et al.*, 2004, Ng *et al.*, 2003, Halevy *et al.*, 2003].

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
...	yes	yes	yes	yes/no	yes/no

3.7.3 Contextual Reasoning

3.7.3.1 Propositional Logic of Context

by FLORIS ROELOFSEN

Motivation Formalising context was first proposed by McCarthy [1987] as a possible solution to the problem of *generality* in Artificial Intelligence. He observed that an axiom is true only with respect to a particular context. In any case, a more general context can be thought of in which the precise form of the axiom doesn't hold any more.

Description In the propositional logic of context (PLC), as described by McCarthy, Buvač, and Mason [Buvač and Mason, 1993, McCarthy and Buvač, 1998], contexts are represented by *sequences* of labels. Intuitively, a label sequence $\kappa_1\kappa_2$ denotes a context κ_2 *as seen from the viewpoint of context* κ_1. If $\mathbb{K}$ is a set of labels and $\mathbb{K}^*$ the set of finite sequences over $\mathbb{K}$, then the language of PLC is defined as a multi-modal language over a set of atomic propositions $\mathbb{P}$, with modal operators $\mathit{ist}(\overline{\kappa}, \varphi)$ for each label sequence $\overline{\kappa} = \kappa_1 \dots \kappa_n \in \mathbb{K}^*$. The intuitive meaning of a formula $\mathit{ist}(\kappa_2, \varphi)$, when stated in context κ_1, is that φ holds in context κ_2, from the standpoint of context κ_1.

A model $\mathfrak{M}$ for PLC associates to each context $\overline{\kappa}$ a set of partial truth value assignments $\mathfrak{M}(\overline{\kappa})$. Associating a *set* of assignments to every context is motivated by intuitions similar to those which underlie possible worlds semantics. A formula φ holds ("is known to be true") in context $\overline{\kappa}$ if it is satisfied by all the assignments associated to $\overline{\kappa}$.

Allowing *partial* assignments provides for the simulation of local languages – in each context, only a fragment of the global language is actually meaningful. Formally, a formula φ is meaningful in context $\overline{\kappa}$ if every assignment in $\mathfrak{M}(\overline{\kappa})$ fully determines the truth of φ. So $\mathfrak{M}$ defines a function $\mathsf{Vocab}(\mathfrak{M})$, which associates to every context $\overline{\kappa}$ a set $\mathsf{Vocab}(\mathfrak{M})(\overline{\kappa})$ of meaningful formulae.

Now for a model $\mathfrak{M}$, a context $\overline{\kappa}$, an an assignment $\nu \in \mathfrak{M}(\overline{\kappa})$, and a formula $\varphi \in \mathsf{Vocab}(\mathfrak{M})(\overline{\kappa})$, satisfaction is defined as follows:

1. $\mathfrak{M}, \nu \models_{\overline{\kappa}} p$ iff $\nu(p) = \mathrm{true}$
2. $\mathfrak{M}, \nu \models_{\overline{\kappa}} \neg\varphi$ iff not $\mathfrak{M}, \nu \models_{\overline{\kappa}} \varphi$
3. $\mathfrak{M}, \nu \models_{\overline{\kappa}} \varphi \supset \psi$ iff not $\mathfrak{M}, \nu \models_{\overline{\kappa}} \varphi$ or $\mathfrak{M}, \nu \models_{\overline{\kappa}} \psi$
4. $\mathfrak{M}, \nu \models_{\overline{\kappa}} \mathit{ist}(\kappa, \varphi)$ iff for all $\nu' \in \mathfrak{M}(\overline{\kappa}\kappa)$, $\mathfrak{M}, \nu' \models_{\overline{\kappa}\kappa} \varphi$
5. $\mathfrak{M} \models_{\overline{\kappa}} \varphi$ iff for all $\nu \in \mathfrak{M}(\overline{\kappa})$, $\mathfrak{M}, \nu \models_{\overline{\kappa}} \varphi$

If the precondition $\varphi \in \mathsf{Vocab}(\mathfrak{M})(\overline{\kappa})$ does not hold, then neither $\mathfrak{M}, \nu \models_{\overline{\kappa}} \phi$ nor $\mathfrak{M}, \nu \models_{\overline{\kappa}} \neg\phi$. A formula φ is satisfiable in a context $\overline{\kappa}$ if there is a model $\mathfrak{M}$ such that $\mathfrak{M} \models_{\overline{\kappa}} \varphi$.

Results The mentioned basic framework is described in [McCarthy and Buvač, 1998, Buvač and Mason, 1993], building on earlier and still relevant work by Guha [1991]. An extension to incorporate quantifiers in the local languages has been provided by Buvač [1996a]. The complexity of reasoning with purely propositional contexts has been investigated by Massacci [1996], and more recently by Roelofsen and Serafini [2004, 2004]. PLC has been shown to be a special instance of the more general multi-context systems framework discussed in section 3.7.3.2 [Serafini and Bouquet, 2004].

Applications The propositional logic of context has most notably been implemented into the CYC common sense knowledge base by Lenat and Guha [1990]. Moreover, it has been applied to fields such as planning [Buvač and McCarthy, 1996], machine translation [Buvač and Fikes, 1995], and word sense disambiguation [Buvač, 1996b].

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
none	yes	yes	no	no	no

3.7.3.2 Multi-Context Systems

by FLORIS ROELOFSEN

Motivation The multi-context system (hereafter MCS) framework is motivated by the work of Giunchiglia [1993b], which emphasises the *principle of locality*: reasoning based on large (common sense) knowledge bases can only be effectively pursued if confined to a manageable subset (context) of that knowledge base. Complementary to this idea is the *principle of compatibility* [Ghidini and Giunchiglia, 2001c]: there must be certain constraints between reasoning processes in different contexts so as to guarantee their compatibility. Together, these two principles have fostered the investigation of contextual

reasoning, viewed as an assemblage of heterogeneous reasoning mechanisms that operate on local, interrelated knowledge fragments.

Description A simple but effective illustration of the intuitions underlying MCS is provided by the so-called "magic box" example, depicted below.

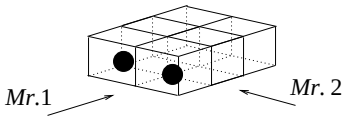

Figure 3.2: The magic box

Example 3.7.2 *Mr.1 and Mr.2 look at a box, which is called "magic" because neither of the observers can make out its depth. Both Mr.1 and Mr.2 maintain a local representation of what they see. These representations must be coherent – if Mr.1 believes the box to contain a ball, for instance, then Mr.2 may not believe the box to be empty.*

The formal description of such interrelated local representations departs from a set of indices I. Each index $i \in I$ denotes a *context*, which is described by a *local* formal language L_i. To state that a formula $\varphi \in L_i$ holds in context i one utilises *labelled formulae* of the form $i : \varphi$. formulae that apply to different contexts may be related by *bridge rules*, which are expressions of the form:

$$i_1 : \phi_1, \ldots, i_n : \phi_n \rightarrow i : \varphi \tag{3.1}$$

Example 3.7.3 *The situation described in example 3.7.2 may be formalised by an MCS with two contexts* 1 *and* 2*, described by propositional languages* $L_1 = L(\{l, r\})$ *and* $L_2 = L(\{l, c, r\})$*, respectively. The constraint that Mr.2 may not believe the box to be empty if Mr.1 believes it to contain a ball can be captured by the following bridge rule:*

$$1 : l \vee r \quad \rightarrow \quad 2 : l \vee c \vee r$$

[Ghidini and Giunchiglia, 2001c] call the proposed semantics for this formalism *local model semantics*. An MCS (collection of local languages plus set of bridge rules) is interpreted in terms of a *chain*: a collection of *sets of local models*, one set for each context. A *local model* is merely a standard interpretation of the language of the context that it serves to interpret. A chain can be thought of as a set of "epistemic states", each corresponding to a certain context. The fact that its i^{th} element contains more than one local model amounts to L_i being interpretable in more than one unique way (partial knowledge). Exactly one local model corresponds to complete knowledge, whereas an empty set of local models indicates an inconsistent context.

Example 3.7.4 *Consider the situation depicted in Figure 3.2. Both agents have complete knowledge, corresponding to the chain* $\{\{[l, r]\}, \{[l, \neg c, \neg r]\}\}$. *We can imagine a scenario however, in which Mr.1 and Mr.2's views are restricted to the right half and the left-most section of the box, as depicted in Figure 3.3.*

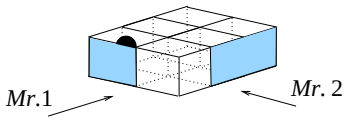

Figure 3.3: A partially hidden magic box.

Now, both Mr.1 and Mr.2 have only partial knowledge. This is reflected by a chain, whose elements contain more than one local model:

$$\left\{ \begin{array}{c} \{[l, \neg r], [\neg l, \neg r]\}, \\ \{[l, \neg c, \neg r], [l, \neg c, r], [l, c, \neg r], [l, c, r]\} \end{array} \right\}$$

Satisfaction is a local concept in this framework: a chain c satisfies a labelled formula $i : \varphi$, regarding context i, if all the local models comprised by the i^{th} element of c satisfy φ in a classical sense. Compatibility of different contexts is captured by the concept of *bridge rule compliance*: a chain complies with a bridge rule if it satisfies its consequence or does not satisfy one of its premises. In order for a chain to *consistently satisfy* a formula $i : \varphi$, it should satisfy $i : \varphi$ and comply with all the bridge rules of the system. Moreover, its i^{th} element should not be empty (the corresponding context should not be inconsistent).

Results The framework has been worked out and described by Giunchiglia and Serafini [1994]. The local model semantics is due to Ghidini and Giunchiglia [2001c]. Complexity issues and decision procedures have been investigated by Serafini and Roelofsen [2004, 2004]. Multi-context systems are the most general framework for contextual reasoning proposed to date [Serafini and Bouquet, 2004].

Applications Multi-context systems have been successfully applied to various fields of computer science and artificial intelligence, including

- meta-reasoning and propositional attitudes [Giunchiglia and Serafini, 1994],
- reasoning with different viewpoints [Attardi and Simi, 1995],
- common sense reasoning [Bouquet and Giunchiglia, 1995],
- reasoning about beliefs [Benerecetti *et al.*, 1998a, Fisher and Ghidini, 1999] [Ghidini, 1999, Giunchiglia and Giunchiglia, 1996, Giunchiglia *et al.*, 1993],
- multi-agent systems [Benerecetti *et al.*, 1998b, Cimatti and Serafini, 1995],

- automated reasoning with modal logics [Giunchiglia and Sebastiani, 2000], and
- contextual ontology design [Bouquet *et al.*, 2003b, also see section 3.7.1.2 of this report].

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
none	yes	yes	yes	yes	no

3.7.4 XML namespaces

by GIORGOS STAMOU & YIANNIS KOMPATSIARIS

Introduction The main aim of this report is to present the use and the operation of namespaces in Extensible Markup Language (XML). The role of namespaces is very important and in order to realise this we can think of applications of Extensible Markup Language (XML) in which a single document may contain elements and attributes that are defined for and used by multiple software modules. In such cases if a markup vocabulary exists which is well understood and for which there is useful software available, it is better to re-use this markup than re-invent it.

But in documents like these, containing multiple markup vocabularies, there is danger in recognition and collision. Software modules need to be able to recognise the tags and attributes which they are designed to process, even in the face of "collision" occurring when markup intended for some other software package uses the same element type or attribute name.

At that point namespaces are the solution for this problem. The definition given by W3C is that "an XML namespace is a collection of names, identified by a URI reference which are used in XML documents as element types and attributes names. XML namespaces differ from the "namespaces" conventionally used in computing disciplines that the XML version has internal structure and is not mathematically speaking, a set". Furthermore "URI references, which identify namespaces, are considered identical when they are exactly the same character-for-character, noting that URI references which are not identical in this sense may in fact be functionally equivalent".

Names from XML namespaces may appear as qualified names, which contain a single colon, separating the name into a namespace prefix and a local part. The prefix, which is mapped to a URI reference, selects a namespace. The combination of the universally managed URI namespace and the document's own namespace produces identifiers that are universally unique and mechanisms are provided for prefix scooping and defaulting. On the other hand URI references can contain characters not allowed in names so cannot be used directly as namespace prefixes. Therefore, the namespace prefix serves as a proxy for a URI reference.

Declaring Namespaces A namespace is declared using a family of reserved attributes. Such an attribute's name must either be `xmlns` or have `xmlns:` as a prefix. These attributes, like any other XML attributes, may be provided directly or by default.

The attribute's value, a URI reference, is the namespace name identifying the namespace. The namespace name, to serve its intended purpose, should have the characteristics of uniqueness and persistence.

Furthermore if the attribute name matches `PrefixedAttName`, then the `NCName` gives the namespace prefix, used to associate element and attribute names with the namespace name in the attribute value in the scope of the element to which the declaration is attached. Also in some of those declarations, the namespace name may not be empty.

Finally if the attribute name matches `DefaultAttName`, then the namespace name in the attribute value is that of the default namespace in the scope of the element to which the declaration is attached.

Qualified Names In conforming XML documents to the W3C specification some names may be given as qualified names. Qualified names are declared using a prefix, which provides the namespace prefix part of the qualified name, and must be associated with a namespace URI reference in a namespace declaration and also the Local part that provides the local part of the qualified name.

Applying namespaces to Elements and attributes

The first Namespace Scoping The namespace declaration is considered to apply to the element where it is specified and to all elements within the content of that element, unless overridden by another namespace declaration with the same `NSAttName` part. Additionally multiple namespace prefixes can be declared as attributes of a single element.

Namespace Defaulting A default namespace is considered to apply to the element where it is declared and to all elements with no prefix within the content of that element. If the URI reference in a default namespace declaration is empty, then unprefixed elements in the scope of the declaration are not considered to be in any namespace. The default namespace can be set to the empty string that has the same effect, within the scope of the declaration, of there being no default namespace.

Uniqueness of Attributes In XML documents conforming no tag may contain two attributes which

- have identical names and

- have qualified names with the same local part and with prefixes which have been bound to namespace names that are identical.

Conformance of Documents In XML documents that conform to the W3C specification, element types and attribute names must match the production for `QName` and must satisfy the "Namespace Constraints".

An XML document conforms this specification if all other tokens in the document which are required, for XML conformance, to match the XML production for Name match this specification's production for `NCName`.

The effect of conformance is that in such a document

- all element types and attribute names contain either zero or one colon and
- no entity names, PI targets, or notation names contain any colons.

Strictly speaking attribute values which are declared to be of types `ID`, `IDREF(S)`, `ENTITY(IES)`, and `NOTATION`, are also names, and thus should be colon-free. However the declared type of attribute values is only available to processors which read markup declaration, for example validating processors. Thus unless the use of a validating processor has been specified, there can be no assurance that the contents of attribute values have been checked for conformance to this specification.

Summary With respect to the evaluation criteria proposed in section 3.3 the framework may be summarised as follows:

COMPOS	SYNTAX	SEMANT	HETERO	REASON	SYSTEM
compose	yes	no	no	no	yes

3.8 Emergent Semantics

by MUSTAFA JARRAR

In what follows, we summarise the status of a collaborative effort on the development of the notion of "emergent semantics", which has been initiated by the IFIP 2.6 Working Group on Data Semantics. This summary is based on [Aberer *et al.*, 2004b, Aberer *et al.*, 2004a].

This approach is motivated by the belief that global semantic interoperability emerges from large numbers of purely local, pair-wise interactions (see also 3.7.1.2). "Semantic interoperability is viewed as an emergent phenomenon constructed incrementally, and its state at any given point in time depends on the frequency, the quality and the efficiency

with which negotiations can be conducted to reach agreements on common interpretations within the context of a given task". This type of semantic interoperability is called "emergent semantics".

The key principles of this approach are:

- Agreements as a Semantic Handshake Protocol. Emergent semantics ("Dynamic ontologies") can be established on the bases of mutually accepted propositions between the interacting agents. The quality of "emerged semantics" depends on the strength of the agreed propositions, and their trustworthiness.
- Agreements emerge from negotiations. Information environments are assumed to be dynamic. Thus, interactions between agents are necessary to identify and resolve semantic conflicts, and to verify whether a consensus leads to the expected actions. Interactions are message exchanges or references to distributed information resources.
- Agreements emerge from local interactions. Emergent semantics are assumed to be established incrementally, based on local agreements. Global agreements are obtained through aggregations of local agreements.

This approach is currently active in the area of peer-to-peer data management and integration, where local schema mappings are introduced in order to enable semantic interoperability. Local schema mappings can be seen as the local communication mechanisms for establishing consensus on the interpretation of data.

While the Semantic Web approach uses ontologies for obtaining semantic interoperability, the ambition of the emergent semantics approach is to obtain such interoperability in a more scalable and decentralised fashion, without necessarily using ontologies.

Chapter 4

Benchmarking Ontology Technology

by RAÚL GARCÍA-CASTRO

4.1 Introduction

It can be expected, that the Semantic Web will contain large volumes of information and, as time goes by, this information will increase exponentially. Therefore, a new necessity arises: to be able to deal with this information size; and scalability will become one of the main requirements for Semantic Web technology.

Research results (techniques and technology) obtained in the fields of approximation, distribution, and modularisation of ontologies must be assessed. This is primarily done by performing experiments that measure the scalability capabilities of these results.

Measurement and experimentation, although being a cornerstone of assessment, just deal with performing a comparative analysis of different techniques or technology. In order to learn from the best practices carried out in the area and continuously improve, benchmarking studies must also be performed over these techniques and technology.

The use of the term benchmarking regarding the search for continuous improvement and best practices emerged in the industry area. The Software Engineering community has adopted it and has performed many benchmarking studies in different fields like operating systems or database management systems.

Benchmarking activities are present all over the Knowledge Web Network of Excellence. Several working packages contain tasks concerning benchmarking. Hence, in this chapter we present a wide overview of the benchmarking process, experimentation, and measurement. This way, the contents of this chapter can be useful not only to the participants in the scalability work package, but to most of the Knowledge Web partners involved in benchmarking. This chapter is meant to serve as a possible starting point for benchmarking Semantic Web tools, techniques, and applications.

In this chapter we review definitions, classifications, and methods of benchmarking, as

well as of measurement and experimentation in Software Engineering. Then, we present the state of the art of benchmarking within the Semantic Web area. We have focussed on ontology-based technology evaluation, presenting a general evaluation framework and the different studies carried out in the field.

4.2 Benchmarking

Benchmarking has been broadly used in the industry area as a way of continuously improving and searching for best practices. In this section, we will summarise the most relevant definitions used in benchmarking, the main classifications proposed, and the different methodologies used to perform benchmarking.

4.2.1 Benchmark versus benchmarking

The IEEE Standard Glossary of Software Engineering Terminology [IEEE, 1991] defined **benchmark** as:

1. *A standard against which measurements or comparisons can be made.*
2. *A procedure, problem, or test that can be used to compare systems or components to each other or to the standards as in (1).*
3. *A recovery file.*

A few years later, Sill [1996] complemented the second IEEE benchmark definition at the comp.benchmarks FAQ by saying that:

> *A benchmark is a test that measures the performance of a system or subsystem on a well-defined task or set of tasks.*

Although in the above definitions a benchmark is supposed to be used only to assess systems, Sim and colleagues [2003] expanded the definition to benchmark techniques as well as systems.

In the last decades, the word benchmarking has become relevant within the business management community. The definitions widely known are those due to Camp [1989] and Spendolini [1992]. Camp defined benchmarking as *the search for industry best practices that lead to superior performance,* while Spendolini expanded it saying that *benchmarking is a continuous, systematic process for evaluating the products, services, and work processes of organisations that are recognised as representing best practices for the purpose of organisational improvement.*

A few years later, Ahmed and Rafiq [1998] stated that the central essence of benchmarking is to learn how to improve business activity, processes, and management. They identified the main benchmarking characteristics as:

- Measurement via comparison.
- Continuous improvement.
- Systematic procedure in carrying out benchmarking activity.

The Software Engineering community does not have a common **benchmarking** definition. Some of the most representative benchmarking definitions are:

- Kitchenham [1996] defined benchmarking as a software evaluation method. For her, benchmarking is *the process of running a number of standard tests using a number of alternative tools/methods and assessing the relative performance of the tools in those tests*.
- Weiss [2002] and Wohlin and colleagues [2002] adopted the business benchmarking definition. For Weiss benchmarking is *a method of measuring performance against a standard, or a given set of standards*; and for Wohlin benchmarking is *a continuous improvement process that strives to be the best of the best through the comparison of similar processes in different contexts*.

To sum up, the terms benchmark and benchmarking differ. While benchmarking refers to a process, the term benchmark refers to a test (maybe used in the benchmarking process). Table 4.1 summarises the main differences between benchmark and benchmarking.

Table 4.1: Main differences between benchmark and benchmarking

	Benchmark	**Benchmarking**
IS A	Test	Continuous process
PURPOSE	Measure Evaluate	Search for best practices - Measure - Evaluate Improve
TARGET	Method System	Product Service Process

4.2.2 Benchmarking classifications

One of the main purposes of this section is to provide a general understanding of the vocabulary used to classify benchmarking processes. We present two different classifications of benchmarking: one is more focussed on the participants involved in it, while the other is based on the nature of the objects under analysis.

The main benchmarking classification was presented by Camp [1989]. He categorised benchmarking depending on the kind of participants involved, and this classification has been adopted later by other authors like Sole and Bist [1995], Ahmed and Rafiq [1998] and Fernandez and colleagues [2001]. The four categories identified by Camp are:

- **Internal benchmarking**. It measures and compares the performance of activities as well as functions and processes within one organisation.
- **Competitive benchmarking**. In this case, the comparison is made with products, services, and/or business processes of a direct competitor.
- **Functional benchmarking** (also called industry benchmarking). It is similar to competitive benchmarking, except that the comparison involves a larger and more broadly defined group of competitors in the same industry.
- **Generic benchmarking**. Its aim is to search for general best practices, without regard to a specific industry.

Another classification categorises benchmarking according to the nature of the objects under analysis in the benchmarking. This classification appeared, although not explicitly separated from the previous one, in Ahmed and Rafiq's [1998] benchmarking classification. A few years later, Lankford [2000] established a separate classification and identified the following types of benchmarking:

- **Process benchmarking**. It involves comparisons between discrete work processes and systems.
- **Performance benchmarking**. It involves comparison and scrutiny of performance attributes of products and services.
- **Strategic benchmarking**: It involves comparison of the strategic issues or processes of an organisation.

4.2.3 Benchmarking methodologies

This section presents the traditional methodologies used to perform benchmarking. These methodologies belong to the business community but, as they are quite general, they can

be easily adapted to benchmark software. All of them have similar elements and coincide in the fact that benchmarking is a continuous process. Therefore, the steps proposed are just an iteration of the benchmarking cycle.

The methodology proposed by Camp [1989] includes the following four phases:

- **Planning phase**. Its objective is to schedule the benchmarking investigations. The essential steps of this phase are to:
  - Identify what is to be benchmarked.
  - Identify comparative companies.
  - Determine the data collection method and collect data.
- **Analysis phase**. This phase involves a careful understanding of current process practices as well as of those practices of benchmarking partners. The steps to follow in this phase are to:
  - Determine the current performance gap between practices.
  - Project the future performance levels.
- **Integration phase**. This phase involves planning to incorporate the new practices obtained from benchmark findings in the organisation. The main step of this phase is to:
  - Communicate benchmark findings and to gain acceptance.
- **Action phase**. In this phase, benchmarking findings and operational principles based on them are converted into actions. The steps recommended are to:
  - Establish functional goals.
  - Develop action plans.
  - Implement specific actions and monitor progress.
  - Recalibrate benchmarks.

Camp also identifies a **maturity state** that will be reached when best industry practices are incorporated into all business processes and benchmarking becomes institutionalised.

Another methodology is the one proposed by the American Productivity and Quality Centre. It has been broken down by Gee and colleagues [2001] in the following four phases:

- **Plan**. Its goal is to prepare the benchmarking study plan, to select the team and partners, and to analyse the organisational process. The steps to follow are to:
  - Form (and train, if needed) the benchmarking team.

  - Analyse and document the current process.
  - Identify the area of study on which the team will focus.
  - Identify the most important customer.
  - Identify the smaller subprocesses, especially problem areas.
  - Identify the critical success factors for the area and develop measures for them.
  - Establish the scope of the benchmarking study.
  - Develop a purpose statement.
  - Develop criteria for determining and evaluating prospective benchmarking partners.
  - Identify target benchmarking partners.
  - Define a data collection plan and determine how the data will be used, managed, and distributed.
  - Identify how implementation of improvements will be accomplished.

- **Collect**. The goals of the data collection phase are to: prepare and administer questions, capture the results, and follow-up with partners. The steps to follow are to:
  - Collect secondary benchmarking information in order to determine whom to target as benchmarking partners.
  - Collect primary benchmarking data from the benchmarking partners.

- **Analyse**. The goals of this phase are to: analyse performance gaps and identify best practices, methods, and enablers. The steps to follow are to:
  - Compare your current performance data with the partner's data.
  - Identify any operational best practices observed and the factors and practices that facilitate superior performance.
  - Formulate a strategy to close any identified gaps.
  - Develop an implementation plan.

- **Adapt**. The goals of this phase are: publish findings, create an improvement plan, and execute the plan. The steps to follow are to:
  - Implement the plan.
  - Monitor and report progress.
  - Document and communicate the study results.
  - Plan for continuous improvement.

Gee and colleagues [2001] also identify the **final steps** to carry out after the adaptation phase. These steps are:

- Document the benchmarking in a final report, capture any lessons learned that can be of future value, capturing also a variety of process information.
- Communicate the results of the benchmarking effort to management and staff.
- Send a copy of the final report to the benchmarking partners.
- Routinely review the performance of the benchmarked processes to ensure that goals are being met.
- Move on to what is next by identifying other candidate processes for benchmarking.

4.3 Experimental Software Engineering

The need for experimentation in Software Engineering was stated by Basili and colleagues [1986]. Experimentation helps to better evaluate, predict, understand, control, and improve the software development process and its associated products. In this section, we summarise the most relevant definitions used in experimental Software Engineering as well as the different classifications proposed and the different methodologies used to perform experimentation.

4.3.1 Definition

Basili [1993] defined experiment as:

> *An experiment is a test, trial or tentative procedure policy; an act or operation for the purpose of discovering something unknown or for testing a principle, supposition, etc.; an operation carried out under controlled conditions in order to discover an unknown effect or law, to test or establish a hypothesis, or to illustrate a known law.*

4.3.2 Classification of experiments

Basili and colleagues [1986] classified Software Engineering experiments in terms of the number of projects evaluated and the number of teams involved in each project. This classification is adopted by many Software Engineering experiments published. Basili proposes:

- **Blocked subject-project studies**. They examine one or more objects across a set of teams and a set of projects.
- **Replicated project studies**. They examine object(s) across a set of teams and a single project.
- **Multi-project variation studies**. They examine object(s) across a single team and a set of projects.
- **Single project studies**. They examine object(s) on a single team and a single project.

The DESMET project [Kitchenham *et al.*, 1994] classified experiments according to the control of the evaluation in the following three groups:

- **Formal experiment**. It uses the principles and procedures of experimental design to check whether a hypothesis can be confirmed. Formal experiments are specifically designed to minimise the effects of extraneous factors on the outcome of the experiment.
- **Case study**. It is a trial use of a method or tool on a full scale project. Control of extraneous factors is much more difficult than in a formal experiment.
- **Survey**. It is the collection and analysis of data from a wide variety of projects. The data obtained in a survey are not as controlled as those obtained from a formal experiment, but they can be analysed statistically to identify important trends.

4.3.3 Methodologies

Basili and colleagues [1986] proposed a framework for conducting experimentation that includes the following four phases:

- **Definition**. Its goal is to identify the motivation, object, purpose, perspective, domain, and scope of the experiments.
- **Planning**. Its goal is to design the experiments, choose the criteria to be used according to the experiments definition, and define the measurement process.
- **Operation**. This phase consists of the experiment preparation and execution, and of the analysis of the data obtained after their execution.
- **Interpretation**. In this phase, the results of the previous phase are interpreted in different contexts, they are extrapolated to other environments, they are presented, and the needed modifications are performed.

A few years later Kitchenham and colleagues [2002] proposed a set of guidelines for carrying out experiments. These guidelines consider what to do and what not to do in the following six basic experimentation areas:

- **Experimental context**
  - Specify as much as possible the industrial context. Define clearly the entities, attributes, and measures that are capturing the contextual information.
  - If a specific hypothesis is being tested, state it clearly prior to performing the tests and discuss the theory from which it is derived, so that its implications are apparent.
  - If the research is exploratory, state clearly and, prior to the data analysis, what questions the investigation is intended to address, and how it will address them.
  - Describe research that is similar to the actual research and how current work relates to it.
- **Experimental design**
  - Identify the population from which the experimental subjects and objects are drawn.
  - Define the process by which the subjects and objects are selected and assigned to treatments.
  - Restrict yourself to simple study designs or, at least, designs that are fully analysed in the literature.
  - Define the experimental unit.
  - For formal experiments, perform a pre-experiment or pre-calculation to identify or estimate the minimum required sample size.
  - Use appropriate levels of blinding.
  - Make explicit any vested interests, and report what has been done to minimise bias.
  - Avoid the use of controls unless you are sure that the control situation can be unambiguously defined.
  - Fully define all interventions.
  - Justify the choice of outcome measures in terms of their relevance to the objectives of the empirical study.
- **Conducting the experiment and data collection**
  - Define all software measures fully, including the entity, attribute, unit, and counting rules.

  - For subjective measures, present a measure of inter-rater agreement.
  - Describe any quality control method used to ensure completeness and accuracy of data collection.
  - For surveys, monitor and report the response rate, and discuss the representativeness of the responses and the impact of non-response.
  - For observational studies and experiments, record data about subjects who drop out from the studies. Also record other performance measures that may be adversely affected by the treatment, even if they are not the main focus of the study.

- **Analysis**
  - Specify all the procedures used to control multiple testing.
  - Consider using blind analysis.
  - Perform sensitivity analysis.
  - Ensure that the data do not violate the assumptions of the tests used on them.
  - Apply appropriate quality control procedures to verify your results.

- **Presentation of results**
  - Describe or cite a reference for all statistical procedures used.
  - Report the statistical package used.
  - Present quantitative results showing the magnitude of effects and the confidence limits.
  - Present the raw data whenever possible. Otherwise, confirm that they are available for confidential review.
  - Provide appropriate descriptive statistics and graphics.

- **Interpretation of results**
  - Define the population to which inferential statistics and predictive models apply.
  - Define the type of study taken into account.
  - Differentiate between statistical significance and practical importance.
  - Specify any limitations of the study.

4.4 Measurement

In order to evaluate and compare software qualitatively and quantitatively, it must be described through software measurements. In fact, Fenton [1991] stated that assessment is one of the two broad purposes of software measurement, the other one is predicting. In this section, we will summarise the most relevant definitions used in measurement, the different classifications proposed, and the different methodologies used for measurement.

4.4.1 Definitions

Fenton [1991] defined both measure and measurement as follows:

- *A measure is an empirical objective assignment of a number (or symbol) to an entity to characterise a specific attribute.*
- *Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules.*

The concept of metric is highly related to the terms defined above. Fenton set out that the term metric has been used in distinct ways in the Software Engineering literature, although every proposed definition could be accommodated within the framework of scientific measurement. Fenton and Neil [2000] proposed the following definition:

> *Software metrics is a collective term used to describe the very wide range of activities concerned with measurement in Software Engineering. These activities range from producing numbers that characterise properties of software code (these are the classic software 'metrics') through to models that help predict software resource requirements and software quality.*

4.4.2 Classification of software measures

Fenton [1991] proposed two classifications of software measures, one regarding the entities and attributes of interest involved in the measure, and another regarding the scope of the measure.

He distinguished three classes of entities whose attributes could be measured:

- **Processes**. They are any software related activities.
- **Products**. They are any artefacts, deliverables or documents which are output of the processes.

- **Resources**. They are items that are inputs to processes.

Processes, products, and resources have internal and external attributes:

- **Internal attributes** of a product, process, or resource are those that can be measured purely in terms of the product, process, or resource itself.
- **External attributes** of a product, process, or resource are those that can only be measured with respect to how the product, process, or resource relates to its environment.

Kitchenham and colleagues [1995] also differentiated between direct and indirect measures, defining them as follows:

- **Direct measure**. It is the measured value of an entity's attribute obtained through a measurement instrument.
- **Indirect measure**. It is the measure obtained from other measures when applying equations to them. These equations are also considered as a form of measurement instrument.

4.4.3 Scales of software measures

Measurement scales are derived from the rules we use for assigning values to attributes. Thus, different rules lead to different scales. Fenton [1991] classified measurement scales according to the transformations that can be made to a scale without changing its structure. Park and colleagues [1996] gave the following definitions to the scales categorised by Fenton:

- **Nominal scale**: A nominal scale provides a name or label as the value for an attribute. The order of values on the scale has no significance.
- **Ordinal scale**: An ordinal scale permits that measured results are placed in ascending (or descending) order. However, distances between locations on the scale have no meaning.
- **Interval scale**: An interval scale adds the concept of distance between values.
- **Ratio scale**: A ratio scale adds an origin (a meaningful, non arbitrary zero value). With a true origin, division and multiplication become meaningful, and all the mathematical operations we customarily use for real numbers are legitimate.
- **Absolute scale**: Absolute scales are special cases of ratio scales in which the only admissible multiplier is 1.

4.4.4 Measurement methods

Grady and Caswell [1987] described the implementation of a software metrics program in Hewlett-Packard. They proposed the following steps in order to define and implement metrics in an organisation:

- **To define company/project objectives for the program**. The objectives you define will frame the methods you use, the costs you are willing to incur, the urgency of the program, and the level of support you have from your managers.
- **To assign responsibilities**. The organisational location of responsibilities for software metrics and the specific people you recruit to implement your objectives is a signal to the rest of the organisation that indicates the importance of the software metrics program.
- **To do research**. Examining data external to the organisation in order to get ideas for conducting experiments and set expectations for the results.
- **To define initial metrics to collect**. You can start with a simple set.
- **To sell the initial collection of these metrics**. The success of a metrics program depends on the accuracy of the data collected, and this accuracy relies on the commitment of the personnel involved and the time required to collect them.
- **To get tools for automatic data collection and analysis**. Such type of tools help to simplify the task of collection, reduce the time expenditure, ensure accuracy and consistency, and reduce psychological barriers to collection.
- **To establish a training class in software metrics**. Training classes help ensure that the objectives for data collection are framed in the context of the company/project objectives. Training is also necessary to achieve the widespread usage of metrics.
- **To publicise success stories and to encourage exchange of ideas**. Publicity of success stories provides feedback to the people taking measurements that their work is valuable. It also helps to spread these successes to other parts of the organisation.
- **To create a metrics database**. A database for collected measurements is necessary to evaluate overall organisational trends and effectiveness. It also provides valuable feedback concerning whether the metric definitions you are using are adequate.
- **To establish a mechanism for changing the standard in an orderly way**. As the organisation understands its development process better, the process and the metrics you collect will evolve and mature. There must be a mechanism in place that basically repeats the previous steps.

Grady and Caswell also stated that a software metrics program must not have a strategy into itself. Collecting software metrics must not be an isolated goal, but a part of an overall strategy for improvement.

Goodman [1993] set up a framework for developing and implementing software metrics programmes within organisations. Goodman defined a generic model, in order to be tailored to each specific environment. The stages proposed for the model are the following:

- **Initialisation stage**. It is caused by some trigger, and it will be driven by an initiator. It is the time when the initial scope of the program is defined.
- **Requirements definition**. This stage is all about finding out what the various parts of the organisation want from a software metrics programme. It involves requirements gathering and specification.
- **Component design**. This stage encompasses both the choice of specific metrics together with the design of the infrastructure that will support the use of those metrics.
- **Component build**. This phase involves building the components of the software metrics program regarding the requirements and design obtained in the previous stages.
- **Implementation**. This phase involves implementing the components that form the measurement initiative into the organisation.

4.5 Ontology technology evaluation

Ontology technology has improved enormously since the creation of the first environments in the mid-1990s. In general, ontology technology has not been the object of software evaluation studies but, as the use of this technology spreads, in the last few years many studies involving ontology tools evaluation have been developed.

In this section, we will present a general framework for ontology technology evaluation as well as the different evaluation studies performed.

4.5.1 General framework for ontology tool evaluation

The OntoWeb deliverable 1.3 [OntoWeb, 2002] presented a general framework for comparing ontology related technology. This framework identified the following types of tools: ontology building tools, ontology merge and integration tools, ontology evaluation tools, ontology based annotation tools, and ontology storage and querying tools. For each

type of tool, the framework provided a set of criteria for comparing tools in each group as well as how different tools in each group satisfied these criteria. Table 4.2 shows the criteria used in the evaluation of each type of tool.

Table 4.2: Ontology tool evaluation criteria [OntoWeb, 2002]

Tools	**Criteria**
Ontology building tools	General description Software architecture and tool evolution Interoperability with other tools and languages Knowledge representation expressivity Inference services attached to the tool Usability
Ontology merge and integration tools	General description Software architecture and tool evolution Information used during the merge process Interoperability Work mode Management of different versions of ontologies Components that the tool merge Suggestions provided by the tool Conflicts detected by the tool Support of some methodology and techniques Help system Edition & visualisation Experience using the tool
Ontology evaluation tools	Interoperability Turn around ability Performance Memory allocation Scalability Integration into frameworks Connectors and interfaces
Ontology-based annotation tools	General description Documentation Tutorial material Available modes of working Automation Interoperability Ontology related points Kind of documents that can be annotated Usability
Ontology storage and querying tools	Query language Implementation language Storage database Inference support Update support API support Export data format Scalability Performance

4.5.2 Evaluation of ontology building tools

Most of the existing literature on evaluation of ontology tools deals with the evaluation of ontology building tool [Angele and Sure, 2002, Sure *et al.*, 2003, Sure *et al.*, 2004]. While some authors have proposed a general evaluation framework, other authors have

focused on specific criteria regarding these tools. In this section, we will compile the work related to the evaluation of ontology building tools.

Duineveld and colleagues [1999] proposed a framework for evaluating different ontology building tools (Ontolingua[1], WebOnto[2], ProtégéWin[3], Ontosaurus[4], and ODE[5]). The tools were evaluated on three dimensions: a general dimension, which refers to the aspects of the tools that can also be found in other types of programs; the ontology dimension, which refers to ontology-related issues found in the tools; and the cooperation dimension, which refers to the tool's support for constructing an ontology by several people at different locations.

Stojanovic and Motik [2002] dealt with the ontology evolution requirements that the tools provided. They evaluated three ontology editors (OilEd[6], OntoEdit[7], and Protégé-2000[8]) and the criteria used were the number of ontology evolution requirements that a platform fulfilled.

Sofia Pinto and colleagues [2002] evaluated the support provided by Protégé-2000 in ontology reuse processes. The criteria they considered were the usability of the tool and the time and effort for developing an ontology by reusing another instead of building it from scratch.

The participants in the First International Workshop on Evaluation of Ontology-based Tools (EON2002) [Angele and Sure, 2002] performed an experiment that consisted of modelling a tourism domain ontology in different tools (KAON[9], Loom[10], OilEd, OntoEdit, OpenKnoME[11], Protégé-2000, SemTalk[12], Terminae[13], and WebODE[14]) and exporting them to a common exchange language (RDF(S)). The criteria used in the evaluation were: expressiveness of the model attached to the tool, usability, reasoning mechanisms, and scalability.

Another general evaluation framework for ontology building tools was proposed by Lambrix and colleagues [2003]. They evaluated Chimaera[15], DAG-Edit[16], OilEd, and Protégé-2000 regarding several general criteria (availability, functionality, multiple in-

[1] http://www.ksl.stanford.edu/software/ontolingua/
[2] http://kmi.open.ac.uk/projects/webonto/
[3] http://protege.stanford.edu/
[4] http://www.isi.edu/isd/ontosaurus.html
[5] http://delicias.dia.fi.upm.es/webODE/
[6] http://oiled.man.ac.uk/
[7] http://www.ontoprise.de/products/ontoedit_en
[8] http://protege.stanford.edu/
[9] http://kaon.semanticweb.org/
[10] http://www.isi.edu/isd/LOOM/
[11] http://www.topthing.com/openknome.html
[12] http://www.semtalk.com/
[13] http://www-lipn.univ-paris13.fr/%7Eszulman/TERMINAE.html
[14] http://delicias.dia.fi.upm.es/webODE/
[15] http://www.ksl.stanford.edu/software/chimaera/
[16] http://godatabase.org/dev/

stance, data model, reasoning, sample ontologies, reuse, formats, visualisation, help, shortcuts, stability, customisation, extendibility, and multiple users) and user interface criteria (relevance, efficiency, attitude, and learnability).

In the Second International Workshop on Evaluation of Ontology-based Tools (EON-2003) (see [Corcho *et al.*, 2003, Isaac *et al.*, 2003, Calvo and Gennari, 2003, Fillies, 2003] and [Knublauch, 2003]), the experiment proposed was to evaluate the interoperability of different ontology building tools. This was performed by exporting and importing to an intermediate language and assessing the amount of knowledge lost during these transformations. The tools evaluated were: DOE[17], OilEd, SemTalk, Protégé-2000, and WebODE; and the intermediate languages used: DAML+OIL, RDF(S), OWL, and UML.

Gómez-Pérez and Suárez-Figueroa [2004] analyzed the behavior of several ontology building tools (OilEd, OntoEdit, Protégé-2000, and WebODE) according to their ability to detect taxonomic anomalies (inconsistencies and redundancies) when building, importing, and exporting ontologies.

Corcho and colleagues [2004] evaluated WebODE's performance, analysing the temporal efficiency and the stability of the methods provided by its ontology management API.

Table 4.3 summarises the criteria used by the mentioned authors when evaluating ontology building tools.

4.5.3 Evaluation of ontology-based annotation tools

by DIANA MAYNARD

The benchmarking of ontology-based annotation tools needs to comprise some metrics for evaluating the quality of the output. Such metrics must provide a simple mechanism for comparing different systems and different versions of the same system in a consistent and repeatable way. Evaluation of semi-automatic or automatic annotation tools can be performed by measuring the correctness of the semantic metadata they produce, with respect to a manually annotated set of data (documents) and an ontology, i.e. by evaluating the quality of the information extraction.

Currently there is no standard for ontology-based information extraction, because it is a relatively new area of research, although there are several well-established metrics for the evaluation of traditional information extraction systems. The most common metrics are those defined by MUC [ARPA, 1993] (Precision/Recall/F-measure) and ACE [ACE, 2004] (cost-based measure based on error rate). The needs of ontology-based information extraction metrics are rather different, however, because traditional methods are binary rather than scalar. This means that these methods assess an answer as correct or incorrect (occasionally allowing for partial correctness which is generally allocated a "half-score"). Ontology-based systems should, however, be evaluated in a scalar way, in

[17] http://opales.ina.fr/public/

Table 4.3: Ontology building tool evaluation criteria

Authors	Tools	Criteria
Duineveld et al., 1999	Ontolingua WebOnto ProtégéWin Ontosaurus ODE	General properties that can also be found in other types of programs Ontology properties found in tools Cooperation properties when constructing an ontology
Stojanovic and Motik, 2002	OilEd OntoEdit Protégé-2000	Ontology evolution requirements fulfilled by the tool
Sofia Pinto et al., 2002	Protégé-2000	Support provided in ontology reuse processes Time and effort for developing an ontology Usability
EON 2002	KAON Loom OilEd OntoEdit OpenKnoME Protégé-2000 SemTalk Terminae WebODE	Expressiveness of the knowledge model attached to the tool Usability Reasoning mechanisms Scalability
Lambrix et al., 2003	Chimaera DAG-Edit OilEd Protégé-2000	General criteria (availability, functionality, multiple instance, data model, reasoning, sample ontologies, reuse, formats, visualisation, help, shortcuts, stability, customisation, extendibility, multiple users) User interface criteria (relevance, efficiency, attitude, learnability)
EON 2003	DOE OilEd Protégé-2000 SemTalk WebODE	Interoperability Amount of knowledge lost during exports and imports
Gómez-Pérez and Suárez-Figueroa, 2003	OilEd OntoEdit Protégé-2000 WebODE	Ability to detect taxonomic anomalies
Corcho et al., 2004	WebODE	Temporal efficiency Stability

order to allow for different degrees of correctness. For example, a scalar method allows the score to be based on the position of the response in the ontology and its closeness to the correct position in the ontology.

When preparing corpus and metrics for ontology-based information extraction, the following activities are essential:

- To have well defined annotation guidelines, so that the annotation of the gold standard text is consistent.
- To carry out an analysis of the corpora with respect to the distributions of the different tags, an analysis of the complexity of the domain for the IE task, and a statistical profile of the tasks (i.e., how difficult the task is for the baseline system).
- To ensure that at least some portion of the corpus, if not all of it, is double-annotated or better still triple-annotated, and that there is a mechanism for conflict resolution where annotators do not agree.

- To measure inter-annotator agreement and to publish this, so systems can know when they have reached the ceiling (if people cannot achieve 100% correctness, then it is unlikely that systems ever can).
- To provide a pre-defined split of the corpus into training and testing data, allowing for measuring the statistical significance of the results.

When defining the metric itself, the following criteria were suggested by King [2003]. The metrics should:

- Reach their highest value for perfect quality.
- Reach their lowest value for worst possible quality.
- Be monotonic.
- Be clear and intuitive.
- Correlate well with human judgement.
- Be reliable and exhibit as little variance as possible.
- Be cheap to set up and to apply.
- Be automatic.

4.5.4 Evaluation of other ontology tools

In this section we summarise the evaluation studies performed for other type of ontology-based tools.

Giboin and colleagues' study [2002] proposed a scenario-based evaluation of ontology-based tools, and applied it to the CoMMA platform in order to evaluate its usability and utility.

Sure and Iosif [2002] evaluated two ontology-based search tools (QuizRDF[18] and Spectacle[19]) in order to compare them with a free text search tool (EnerSEARCHer). The criteria used were the information finding time and the number of user mistakes made during a search.

Noy and Musen [2002] and Lambrix and Edberg [2003] focused on ontology merging tools. Noy and Musen evaluated PROMPT[20] regarding the precision and recall of its suggestions when merging, and the differences between the result ontologies. Lambrix and

[18] http://i97.labs.bt.com/quizrdf-bin/rdfsearch/pmika2.89
[19] http://spectacle.aidministrator.nl/spectacle/index.html
[20] http://protege.stanford.edu/plugins/prompt/prompt.html

Edberg evaluated PROMPT and Chimaera[21] regarding some general criteria (availability and stability), merging criteria (functionality, assistance, precision and recall of suggestions and time to merge), and user interface criteria (relevance, efficiency, attitude and learnability).

Guo and colleagues' evaluation study [Guo *et al.*, 2003] concerned the evaluation of DLDB[22], a DAML+OIL ontology repository. The criteria used in the evaluation were the repository load time when storing the test data, the repository size, the query response time and the completeness of the repository regarding the queries.

Gómez-Pérez and Suárez-Figueroa [2003] analyzed the behavior of several RDF(S) and DAML+OIL parsers (Validating RDF Parser[23], RDF Validation Service[24], DAML Validator[25], and DAML+OIL Ontology Checker[26]) according to their ability to detect taxonomic anomalies (inconsistencies and redundancies).

Finally, Euzenat [2003] proposed the evaluation of ontology alignment methods regarding the distance between provided output and expected result and other measures of the amount of resource consumed (time, memory, user input, etc.).

Table 4.4 summarises the criteria followed by the different authors when evaluating ontology tools.

Table 4.4: Other ontology tool evaluation criteria

Author	**Type of tool**	**Criteria**
Giboin et al., 2002	Ontology-based tools (CoMMA)	Usability Utility
Sure and Iosif, 2002	Ontology-based search tools (QuizRDF and Spectacle)	Information finding time Mistakes during a search Mistakes during a search
Noy and Musen, 2002	Ontology merging tools (Prompt)	Precision and recall of the tools suggestions Difference between result ontologies
Lambrix and Edberg, 2003	Ontology merging tools (Prompt and Chimaera)	General criteria (availability, stability) Merging criteria (functionality, assistance, precision and recall of suggestions, time to merge) User interface criteria (relevance, efficiency, attitude and learnability)
Guo et al., 2003	Ontology repositories (DLDB)	Load time Repository size Query response time Completeness
Gómez-Pérez and Suárez-Figueroa, 2004	RDF(S) and DAML+OIL parsers (Validating RDF parser, RDF Validation Service, DAML Validator, DAML+OIL Ontology Checker)	Ability to detect taxonomic anomalies
Euzenat, 2003	Ontology alignment methods	Distance between alignments Amount of resources consumed (time, memory, user input, etc.)

[21] http://www.ksl.stanford.edu/software/chimaera/
[22] http://www.cse.lehigh.edu/~heflin/research/download/
[23] http://139.91.183.30:9090/RDF/VRP/
[24] http://www.w3.org/RDF/Validator/
[25] http://www.daml.org/validator/
[26] http://potato.cs.man.ac.uk/oil/Checker

4.5.5 Workload generation for ontology tools

Other task concerning the evaluation of ontology tools deals with workload generation for ontology tools, in order to produce input data for experiments.

Magkanaraki and colleagues [2002] performed a structural analysis with quantitative criteria of RDF/S schemas that represented ontologies from various applications. The main conclusions they obtained after the analysis were:

- Most of the RDF schemas define few classes and properties.
- Schema implementation is property-centric or class-centric, depending on whether the designer decides to model concepts as classes or as properties.
- In general, schemas are shallow and they tend to be developed in breadth rather than in depth.
- Multiple inheritance for classes, although not widely used, is more frequent than multiple inheritance for properties.
- Multiple classification of resources is rarely used.
- There is a tight correlation of the notion of semantic depth to the variety of modelling constructs used by the designers.

The work of Tempich and Volz [2003] also belongs to this group of studies; they analysed the ontologies in the DAML ontology library[27] to classify them and derive parameters that can be used for the generation of synthetic ontologies. They classified ontologies in three clusters:

- **Ontologies of taxonomic or terminological nature**. This is the largest cluster. Ontologies in this cluster contain few properties and a large number of classes.
- **Description logic-style ontologies**. Ontologies in this cluster are characterised by a high number of axioms per class and a low number of primitive classes. These ontologies also contain a very high number of restrictions and properties (especially datatype properties), but they scarcely have individuals.
- **Database schema-like ontologies**. This cluster is more heterogeneous. The ontologies are medium size, containing on average 65 class expressions and 25 properties.

Regarding the implementation of workload generators, the description of OntoGenerator[28] appeared in the OntoWeb deliverable 1.3 [OntoWeb, 2002]; this is an OntoEdit

[27] http://www.daml.org/ontologies/

[28] http://www.ontoprise.de/products/ontoedit_plugins_en

plugin that creates synthetic ontologies for performance tests of ontology based tools. Guo and colleagues [2003] presented UBA[29] (Univ-Bench Artificial data generator), a tool that systematically generates DAML+OIL instances based on a realistic ontology. Finally, Corcho and colleagues [2004] generated the workload for their experiments from the definition of the different tests executed in the evaluation of the WebODE ontology management API.

4.5.6 RDF and OWL test suites

The RDF and OWL test suite [Grant and Beckett, 2004, Carroll and Roo, 2004] were created by the W3C RDF Core Working Group and the W3C Web Ontology Working Group, respectively. These tests check the correct usage of the tools that implement RDF and OWL knowledge bases and illustrate the resolution of different issues considered by the Working Groups.

These test suites are intended to provide examples for, and clarification of, the normative definition of the languages, and also to be suitable for use by developers in test harnesses, possibly as part of a test driven development process.

4.5.7 Description Logics systems comparison

The 1998 International Workshop on Description Logics (DL'98) hosted a Description Logics (DL) systems comparison session [Horrocks and Patel-Schneider, 1998]. This comparison was performed by executing a range of test problems that could be used to measure a DL system's performance. The benchmark suite consisted of four types of tests:

- **Concept satisfiability tests**. These tests measure the performance of the DL system when computing the coherence (satisfiability) of large concept expressions without reference to a TBox.
- **Artificial TBox classification tests**. These tests measure the performance of the DL system when classifying an artificially generated TBox.
- **Realistic TBox classification tests**. These tests measure the performance of the DL system when classifying a realistic TBox.
- **Synthetic ABox tests**. These tests measure the performance of the DL system's ABox when realising a synthetic ABox (inferring the most specific concept in the TBox which each individual instantiates).

[29] http://www.lehigh.edu/~ yug2/Research/SemanticWeb/LUBM/LUBM.htm

Based on the DL'98 tests, Haarslev and Möller [1999b] generated a set of nine new ABox test problem sets for evaluating different optimisation strategies for ABox reasoning.

In the 1999 International Workshop on Description Logics (DL'99), the DL systems comparison was based on the different features of the systems [Patel-Schneider, 1999]: logic implemented, availability, future plans, etc.

Regarding workload generation for evaluating Description Logics systems, Elhaik and colleagues [1998] designed a method for randomly generate TBoxes and ABoxes according to probability distributions. Concerning these probability distributions, Ycart and Rousset [2000] defined a natural probability distribution of ABoxes associated to a given TBox.

4.5.8 Modal Logics systems comparison

Heuerding and Schwendimann [1996] presented a set of benchmark formulae for proof search in the propositional modal logics $\mathbf{K}$, $\mathbf{KT}$, and $\mathbf{S4}$. These formulae were divided into nine classes of provable and nine classes of non provable formulae for each logic. They also presented the results of applying these formulae to the Logics Workbench[30] (a system that provides inference mechanisms for different logical formalisms including basic modal logic).

The formulae developed by Heuerding and Schwendimann were also used in the Tableaux 98 conference [Balsiger and Heuerding, 1998], where a comparison of automated theorem provers for the propositional modal logics $\mathbf{K}$, $\mathbf{KT}$, and $\mathbf{S4}$ was performed.

Giunchiglia and Sebastiani [1996a, 1996b] presented a technique for evaluating decision procedures for propositional modal logics. They developed a decision procedure called KSAT and tested its propositional satisfiability in modal $\mathbf{K}_{(\mathbf{m})}$. They compared KSAT with TABLEAU (a decision procedure) and $\mathcal{KRIS}$[31] (a system for modal logics). The workload used for the evaluation consisted of randomly generated $\mathbf{3CNF}_K$ formulae.

Hustadt and Schmidt [Hustadt and Schmidt, 1997, Hustadt and Schmidt, 1999] continued the work of Giunchiglia and Sebastiani, modifying the random formula generator to get more difficult test samples. They evaluated KSAT, $\mathcal{KRIS}$, the Logics Workbench, and TA (a translation approach where the formulae are translated into first order logic and proved with the first-order theorem prover SPASS[32]).

Later Giunchiglia and colleagues [1998, 2000] basing on their previous work and Hustadt and Schmidt's work, compared KSATC, KSATLISP (C and Lisp implementations of the KSAT decision procedure, resp.), KSATLISP(UNSORTED) (Lisp implementation

[30] http://www.lwb.unibe.ch/

[31] http://www.dfki.uni-sb.de/ tacos/kris.html

[32] http://spass.mpi-sb.mpg.de/

without presorting input formulae), and $\mathcal{KRIS}$.

The Tableaux 99 conference [Massacci, 1999] included a comparison of theorem provers for modal systems. The compared were FaCT[33] [Patel-Schneider and Horrocks, 1999], HAM-ALC [Haarslev and Möller, 1999a], and KtSeqC [Boyapati and Gore, 1999]. The formulae used in the evaluation were grouped into four divisions: a modal PSPACE division, a multimodal PSPACE division, a global PSPACE division, and a modal EXPTIME division.

Giunchiglia and colleagues [Giunchiglia *et al.*, 1999, Giunchiglia *et al.*, 2002] performed two theorem prover comparisons. Using the randomly generated formulae developed by Giunchiglia and Sebastiani, they compared *SAT[34] (a platform for the development of decision procedures for modal and description logics), KSATC, DLP[35], and TA. They also compared *SAT, DLP, and TA using the formulae developed by Heuerding and Schwendimann.

Horrocks and Patel-Schneider modified the formula generator, which was developed by Giunchiglia and colleagues, to produce less uniform formulae. They compared DLP with KSATC [Horrocks and Patel-Schneider, 1999b, Horrocks and Patel-Schneider, 2002] and DLP with TA, KSAT, and FaCT [Horrocks and Patel-Schneider, 1999a].

Horrocks and colleagues [2000] proposed a new variant of the random formula generator of Giunchiglia and Sebastiani. The difference with the previous generator relies on a different interpretation of one of the formula generation parameters. They evaluated *SAT with the random generated formulae.

Giunchiglia and Tacchella [2000] evaluated different cache optimisation methods of *SAT using the formulae from the Tableaux 99 conference.

4.5.9 Automated Theorem Proving systems evaluation

In 1993, Suttner and Sutcliffe [Suttner and Sutcliffe, 1997, Sutcliffe and Suttner, 1998] compiled the TPTP[36] library of problems for Automated Theorem Proving (ATP) systems. This library has been widely used and has evolved through the years, and at the moment of writing this text the TPTP library version is 2.6.0.

Since 1996, Automated Theorem Prover comparisons have been performed at the Conference on Automated Deduction (CADE). Pelletier and Sutcliffe [2001] summarised the motivation, history and results of these CADE ATP System Competitions.

Sutcliffe and Suttner [2001] presented an overview of the evaluation criteria, the classification of test problems, and the methods used for evaluating ATP systems and problems.

[33] http://www.cs.man.ac.uk/~horrocks/FaCT/
[34] http://www.mrg.dist.unige.it/~tac/StarSAT.html
[35] http://www.bell-labs.com/user/pfps/dlp/
[36] http://www.cs.miami.edu/~tptp/

Fuchs and Sutcliffe [2002] described a method for measuring the homogeneity of ATP problems with respect to the performance of ATP systems on those problems. This method can also be used to identify homogeneous subsets of a set of problems.

Colton and Sutcliffe [2002, 2002] presented HR, a program that performs automated theory formation in mathematical domains, and showed how it could be used to generate problems for ATP Systems.

Chapter 5

Conclusion

A lot of techniques around scalability are investigated in this deliverable. In general we believe that approximation and modularisation/distribution techniques are the best to achieve scalability for the Semantic Web.

Approximation techniques can be classified if they change the inference technique or if they transform the knowledge base. A more fine-grained classification can be applied to the modularisation/distibution where the techniques can be separated into modularisation (for decomposing a knowledge-/database), integration (where during compile-time different sources are combined) and coordination (for composing several sources during run-time). Most prominent and representative techniques are reviewed.

The survey leads to three interesting points: first to the best of our knowledge there exists no techniques for approximating or modularising Semantic Web techniques or for distributed inferences but there exist some more general techniques. For those few cases where some techniques can be identified (e.g. distributed description logics, instance store) they can be viewed to be at their starting point. Second, the general approaches seem to be applicable for the Semantic Web but need to be examined and adapted to the special needs for the Semantic Web. Third, a interesting relationship between this working package and the heterogeneity working package was discovered which will be tracked and deepened in future.

Especially the second point above needs an expressive and realistic benchmarking for assessing scalability methods. In general the Semantic Web field lacks of benchmarking. For this purpose, we have presented an broader overview of the main research areas involved in benchmarking. Furthermore our review of the existing benchmarking techniques may also serve as a foundation for the benchmark activities in other working packages. We have started with the analysis of benchmarking, experimental studies and measurement in Software Engineering, and then we have moved into the Semantic Web field.

Again the survey on benchmarking allows us to conclude that, although there is no common benchmarking methodology, all the works presented in this section, either

benchmarking, experimentation, or measurement ones, are on the one hand very similar, and on the other hand highly general so as to be easily used in any benchmark study for the ontology and the Semantic Web field. As benchmarking elements (people, goals, scope, resources, etc.) differ across benchmarking studies, the methodologies proposed should be instantiated for each study.

We have also presented the state of the art of ontology technology evaluation. The main conclusions that can be extracted from the studies presented in this chapter are the following:

- Evaluation studies concerning ontology tools have been scarce since these tools appeared. Nevertheless, in the last few years the effort devoted to evaluating ontology technology has significantly grown.
- Most of the evaluation studies concerning ontology tools are performed through case studies or surveys of qualitative nature. Only two of them [Guo *et al.*, 2003, Corcho *et al.*, 2004] involved performing formal experiments that deal with quantitative data.
- In the logics field, where the need for technology evaluation has been broadly adopted, numerous evaluation studies have been performed through the years.

Bibliography

[Aberer *et al.*, 2004a] K. Aberer, T. Catarci, P. Cudre-Mauroux, T. Dillon, S. Grimm, M. Hacid, A. Illarramendi, M. Jarrar, V. Kashyap, M. Mecella, E. Mena, E. Neuhold, A. Ouksel, T. Risse, M. Scannapieco, F. Saltor, L. Santis, S. Spaccapietra, S. Staab, R. Studer, and O. Troyer. Emergent semantics systems. In M. Bouzeghoub, C. Goble, V. Kashyap, and S. Spaccapietra, editors, *Proceeding of the International Conference on Semantics of a Networked World*, LNCS, pages 14 – 44, France, Paris, June 2004. Springer Verlag.

[Aberer *et al.*, 2004b] K. Aberer, P. Cudre-Mauroux, T. Catarci M.Ouksel, M. Hacidand A. Illarramendi, V. Kashyap, M.Mecella, E. Mena, E. Neuhold, O. De Troyer, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, and R. Studer. Emergent semantics principles and issues. In *Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, Jeju Island, Korea, 2004.

[ACE, 2004] ACE. *Annotation Guidelines for Entity Detection and Tracking (EDT)*, Feb 2004. Available at http://www.ldc.upenn.edu/Projects/ACE/docs/.

[Ahmed and Rafiq, 1998] P.K. Ahmed and M. Rafiq. Integrated benchmarking: a holistic examination of select techniques for benchmarking analysis. *Benchmarking for Quality Management and Technology*, 5(3):225–242, 1998.

[Amir and Engelhardt, 2003] E. Amir and B. Engelhardt. Factored planning. In *International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[Amir and McIlraith, 2004] E. Amir and S. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence*, 2004. Accepted for publication.

[Angele and Sure, 2002] J. Angele and Y. Sure, editors. *Proceedings of the 1st International Workshop on Evaluation of Ontology based Tools (EON2002)*, volume 62, Sigüenza, Spain, September 2002. CEUR-WS.

[ARPA, 1993] Advanced Research Projects Agency. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann, California, 1993.

[Attardi and Simi, 1995] G. Attardi and M. Simi. A formalisation of viewpoints. *Fundamenta Informaticae*, 23(2–4):149–174, 1995.

[Bäckström and Jonsson, 1995] Christer Bäckström and Peter Jonsson. Planning with abstraction hierarchies can be exponentially less efficient. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1599–1604, 1995.

[Badard, 1999] T. Badard. On the automatic retrieval of updates in geographic databases based on geographic data matching tools. In *International Cartographic Conference*, pages 47–56, 1999.

[Baker *et al.*, 1999] P.G. Baker, C.A. Goble, S. Bechhofer, N.W. Paton, R. Stevens, and A. Brass. An ontology for bioinformatics applications. *Bioinformatics*, 15(6):510–520, 1999.

[Balsiger and Heuerding, 1998] P. Balsiger and A. Heuerding. Comparison of theorem provers for modal logics - introduction and summary. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 25–26. Springer-Verlag, 1998.

[Basili *et al.*, 1986] V.R. Basili, R.W. Selby, and D.H. Hutchens. Experimentation in software engineering. *IEEE Transactions on Software Engineering*, SE-12(7):733–743, July 1986.

[Basili, 1993] V.R. Basili. The experimental paradigm in software engineering. In *Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions*, pages 3–12. Springer-Verlag, September 1993.

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

[Benerecetti *et al.*, 1998a] M. Benerecetti, P. Bouquet, and C. Ghidini. Formalizing belief reports – the approach and a case study. In *International Conference on Artificial Intelligence, Methodology, Systems, and Applications (AIMSA 98)*, pages 62–75, 1998.

[Benerecetti *et al.*, 1998b] M. Benerecetti, F. Giunchiglia, and L. Serafini. Model checking multiagent systems. *Journal of Logic and Computation*, 8(3):401–423, 1998.

[Bernstein *et al.*, 2002] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos andL. Serafini, and I. Zaihrayeu. Data management for peer-to-peercomputing: A vision. *WebDB*, 2002.

[Boddy and Dean, 1989] M. Boddy and T. Dean. Solving time-dependent planning problems. In *Proceedings IJCAI–89*, Detroit, Michigan USA, August 1989.

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

[Bouquet and Giunchiglia, 1995] P. Bouquet and F. Giunchiglia. Reasoning about theory adequacy. a new solution to the qualification problem. *Fundamenta Informaticae*, 23(2–4):247–262, 1995.

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

[Bouquet *et al.*, 2003b] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In K. Sekara and J. Mylopoulis, editors, *Proceedings of the Second International Semantic Web Conference*, number 2870 in Lecture Notes in Computer Science, pages 164–179. Springer Verlag, October 2003.

[Boyapati and Gore, 1999] V. Boyapati and R. Gore. KtSeqC: System description. In *Proceedings TABLEAUX'99, The 6th International Conference on Theorem Proving with Analytic Tableaux and Related Methods*, pages 29–31, Berlin, 1999. Springer-Verlag.

[Brewington *et al.*, 1999] Brian Brewington, Robert Gray, Katsuhiro Moizumi, David Kotz, George Cybenko, and Daniela Rus. Mobile agents in distributed information retrieval. In Matthias Klusch, editor, *Intelligent Information Agents*. Springer-Verlag: Heidelberg, Germany, 1999.

[Buvač and Fikes, 1995] S. Buvač and R. Fikes. A declarative formalization of knowledge translation. In *Conference on Information and Knowledge Management (CIKM 95)*, 1995.

[Buvač and Mason, 1993] S. Buvač and I. A. Mason. Propositional logic of context. In *National Conference on Artificial Intelligence (AAAI 93)*, 1993.

[Buvač and McCarthy, 1996] S. Buvač and J. McCarthy. Combining planning contexts. In Austin Tate, editor, *Advanced Planning Technology – Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*. AAAI Press, 1996.

[Buvač, 1996a] S. Buvač. Quantificational logic of context. In *National Conference on Artificial Intelligence (AAAI 96)*, 1996.

[Buvač, 1996b] S. Buvač. Resolving lexical ambiguity using a formal theory of context. In Kees van Deemter and Stanley Peters, editors, *Semantic Ambiguity and Underspecification*. CSLI Lecture Notes, 1996.

[Cadoli and Schaerf, 1995] Marco Cadoli and Marco Schaerf. Approximate inference in default reasoning and circumscription. *Fundamenta Informaticae*, 23:123–143, 1995.

[Cadoli *et al.*, 1994] Marco Cadoli, Francesco M. Donini, and Marco Schaerf. Is intractability of non-monotonic reasoning a real drawback? In *National Conference on Artificial Intelligence*, pages 946–951, 1994.

[Cadoli, 1993] Marco Cadoli. A survey of complexity results for planning. In A. Cesta and S. Gaglio, editors, *Italian Planning Workshop*, pages 131–145, Rome, Italy, 1993.

[Cadoli, 1996] M. Cadoli. Panel on "Knowledge Compilation and Approximation": Terminology, Questions, and References. *Fourth International Symposium on Artificial Intelligence and Mathematics (AI/MATH-96)*, pages 183–186, 1996.

[Cali *et al.*, 2003] A. Cali, D. Calvanese, G. De Giacomo, M. Lenzerini, P. Naggar, and F. Vernacotola. Ibis: Semantic data integration at work. In *CAiSE*, pages 79–94, 2003.

[Calvanese *et al.*, 2001] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. A framework for ontology integration. In *Proc. of the 2001 Int. Semantic Web Working Symposium (SWWS 2001)*, pages 303–316, 2001.

[Calvo and Gennari, 2003] F. Calvo and J.H. Gennari. Interoperability of protégé 2.0 beta and OilEd 3.5 in the domain knowledge of osteoporosis. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Florida, USA, October 2003.

[Camp, 1989] R. Camp. *Benchmarking: The Search for Industry Best Practice that Lead to Superior Performance*. ASQC Quality Press, Milwaukee, 1989.

[Carroll and Roo, 2004] J.J. Carroll and J. De Roo. OWL web ontology language test cases. Technical report, W3C, February 2004.

[Chandra and Markowsky, 1978] A. K. Chandra and G. Markowsky. On the number of prime implicants. *Discrete Mathematics*, 24:7–11, 1978.

[Cimatti and Serafini, 1995] A. Cimatti and L. Serafini. Multi-agent reasoning with belief contexts: the approach and a case study. In *Intelligent Agents Conference – Workshop on Agent Theories, Architectures, and Languages*, pages 71–85, 1995.

[Cohen *et al.*, 1998] P. Cohen, R. Schrag, E. Jones, A. Pease, A. Lin, B. Starr, D. Gunning, and M. Burke. The darpa high performance knowledge bases project. *AI Magazine*, 19(4):25–49, 1998.

[Colton and Sutcliffe, 2002] S. Colton and G. Sutcliffe. Automatic generation of benchmark problems for automated theorem proving systems. In *Proceedings of the 7th International Symposium on Artificial Intelligence and Mathematics*, USA, 2002.

[Colton, 2002] S. Colton. The hr program for theorem generation. In *Proceedings of CADE'02*, Copenhagen, Denmark, 2002.

[Consortium, 2000] The MurMur Consortium. Supporting multiple representations in spatio-temporal databases. In *EC-GI and GIS Workshop*, 2000. Available at http://lbdwww.epfl.ch/e/MurMur/.

[Cook, 1971] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computation*, pages 151–158, 1971.

[Corcho *et al.*, 2003] O. Corcho, A. Gómez-Pérez, D.J. Guerrero-Rodríguez, D. Pérez-Rey, A. Ruiz-Cristina, T. Sastre-Toral, and M.C. Suárez-Figueroa. Evaluation experiment of ontology tools' interoperability with the WebODE ontology engineering workbench. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Florida, USA, October 2003.

[Corcho *et al.*, 2004] O. Corcho, R. García-Castro, and A. Gómez-Pérez. Benchmarking ontology tools. a case study for the WebODE platform. To be presented in LREC-2004 May 26th, Lisbon, Portugal, 2004.

[Côté, 1993] R.A. Côté, editor. *Systematized Nomenclature of Medicine - SNOMED International*. College of American Pathologists, 1993.

[Dalal, 1992] Mukesh Dalal. Efficient propositional constraint propagation. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 409–414, San Jose, California, 1992. American Association for Artificial Intelligence.

[Dalal, 1996a] M. Dalal. Semantics of an anytime family of reasoners. In W. Wahlster, editor, *Proceedings of ECAI-96*, pages 360–364, Budapest, Hungary, August 1996. John Wiley & Sons LTD.

[Dalal, 1996b] Mukesh Dalal. Anytime families of tractable propositional reasoners. In *International Symposium on Artifical Intelligence and Mathematics AI/MATH-96*, pages 42–45, 1996. Extended version submitted to Annals of Mathematics and Artifical Intelligence.

[de Kleer, 1990] J. de Kleer. Exploiting locality in a tms. In *Proceedings of the Eight National Conference on Artificial Intelligence (AAAI-90)*, pages 264–271, 1990.

[de Kleer, 1992] J. de Kleer. An improved algorithm for generating prime implicates. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 780–785, 1992.

[Dean and Boddy, 1988] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the seventh National conference on artificial intelligence AAAI–88*, pages 49–54, Saint Paul, Minnesota, 1988.

[del Val, 1994] A. del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 551–561, 1994.

[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 *Proc. of the Twelfth International World Wide Web Conference (WWW)*, 2003.

[Dou *et al.*, 2002] D. Dou, D. McDermott, and P. Qi. Ontology translation by ontology merging and automated reasoning. In *Proceedings of EKAW2002 Workshop on Ontologies for Multi-Agent Systems*, pages 3–18, 2002.

[Duineveld *et al.*, 1999] A.J. Duineveld, R. Stoter, M.R. Weiden, B. Kenepa, and V.R. Benjamins. Wondertools? a comparative study of ontological engineering tools. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*, Banff, Canada, 1999. Kluwer Academic Publishers.

[Dupont, 1994] Y. Dupont. Resolving fragmentation conflicts in schema integration. In *Entity-Relationship Approach - ER'94*, pages 513–532, 1994.

[Elhaik *et al.*, 1998] Q. Elhaik, M.C. Rousset, and B. Ycart. Generating random benchmarks for description logics. In E. Franconi, G. De Giacomo, R.M. MacGregor, W. Nutt, C.A. Welty, and F. Sebastiani, editors, *Proceedings of the International Description Logics Workshop (DL'98)*, volume 11, pages 55–57. CEUR-WS, May 1998.

[Euzenat, 2003] J. Euzenat. Towards composing and benchmarking ontology alignments. ISWC2003 Workshop on Semantic Integration. Sanibel Island, Florida, October 2003.

[Fenton and Neil, 2000] N.E. Fenton and M. Neil. Software metrics: roadmap. In *Proceedings of the conference on The future of Software engineering*, pages 357–370. ACM Press, 2000.

[Fenton, 1991] N.E. Fenton. *Software Metrics - A Rigorous Approach*. Chapman & Hall, London, UK, 1991.

[Fernandez *et al.*, 2001] P. Fernandez, I.P. McCarthy, and T. Rakotobe-Joel. An evolutionary approach to benchmarking. *Benchmarking: An International Journal*, 8(4):281–305, 2001.

[Fikes and Farquhar, 1999] R. Fikes and A. Farquhar. Large-scale repositories of highly expressive reusable knowledge. *IEEE Intelligent Systems*, 14(2), 1999.

[Fillies, 2003] C. Fillies. Semtalk eon2003 semantic web export / import interface test. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Florida, USA, October 2003.

[Fisher and Ghidini, 1999] M. Fisher and C. Ghidini. Programming resource-bounded deliberative agents. In *International Joint Conference on Artificial Intelligence (IJCAI 99)*, pages 200–206, 1999.

[Franconi *et al.*, 2003] E. Franconi, G. Kuper, A. Lopatenko, and L. Serafini. A robust logical and computational characterisation of peer-to-peer database systems. *In International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, 2003.

[Franconi *et al.*, 2004] E. Franconi, G. Kuper, A. Lopatenko, and I. Zaihrayeu. The codb robust peer-to-peer database system. *Proc. of the 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid)*, 2004.

[Frei and Faltings, 2000] Christian Frei and Boi Faltings. Abstraction and constraint satisfaction techniques for planning bandwidth allocation. In *INFOCOM (1)*, pages 235–244, 2000.

[Fuchs and Sutcliffe, 2002] M. Fuchs and G. Sutcliffe. Homogeneous sets of ATP problems. In *Proceedings of the 15th Florida Artificial Intelligence Research Symposium*, pages 57–61, Pensecola, USA, 2002.

[Gee *et al.*, 2001] D. Gee, K. Jones, D. Kreitz, S. Nevell, B. O'Connor, and B. Van Ness. *Using Performance Information to Drive Improvement*, volume 6 of *The Performance-Based Management Handbook*. Performance-Based Management Special Interest Group, 2001.

[Ghidini and Giunchiglia, 2001a] C. Ghidini and F. Giunchiglia. Local model semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[Ghidini and Giunchiglia, 2001b] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, April 2001.

[Ghidini and Giunchiglia, 2001c] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[Ghidini and Serafini, 1998] C. Ghidini and L. Serafini. Distributed First Order Logics. In D. Gabbay and M. de Rijke, editors, *Frontiers Of Combining Systems 2*, Studies in Logic and Computation, pages 121–140. Research Studies Press, 1998.

[Ghidini and Serafini, 2000] C. Ghidini and L. Serafini. Distributed First Order Logics. In *Frontiers of Combining Systems 2*, pages 121–139, 2000.

[Ghidini, 1999] C. Ghidini. Modeling (un)bounded beliefs. In *Modeling and Using Context (CONTEXT 99)*, pages 145–158, 1999.

[Giboin *et al.*, 2002] A. Giboin, F. Gandon, O. Corby, and R. Dieng. Assessment of ontology-based tools: A step towards systemizing the scenario approach. In *Proceedings of the International Workshop on Evaluation of Ontology-based Tools (EON2002)*, Sigüenza, Spain., October 2002.

[Giunchiglia and Giunchiglia, 1996] E. Giunchiglia and F. Giunchiglia. Ideal and real belief about belief. In *International Conference on Formal and Applied Practical Reasoning (FAPR 96)*, pages 261–275, 1996.

[Giunchiglia and Sebastiani, 1996a] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedure - the case study of modal K. In *Conference on Automated Deduction*, pages 583–597, 1996.

[Giunchiglia and Sebastiani, 1996b] F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for ALC. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *KR'96: Principles of Knowledge Representation and Reasoning*, pages 304–314. Morgan Kaufmann, San Francisco, California, 1996.

[Giunchiglia and Sebastiani, 2000] F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures - the case study of modal k_m. *Journal of Information and Computation*, 162(1/2):158–178, 2000.

[Giunchiglia and Serafini, 1994] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70, 1994.

[Giunchiglia and Tacchella, 2000] E. Giunchiglia and A. Tacchella. A subset-matching size-bounded cache for satisfiability in modal logics. In *Analytic Tableaux and Related Methods*, pages 237–251, 2000.

[Giunchiglia and Walsh, 1990] Fausto Giunchiglia and Toby Walsh. Abstract theorem proving: Mapping back. Technical Report 8911-16, IRST, Trento, Italy, 1990.

[Giunchiglia and Walsh, 1992] Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artificial Intelligence*, 57(2-3):323–389, 1992.

[Giunchiglia and Zaihrayeu, 2004] F. Giunchiglia and I. Zaihrayeu. Implementing database coordination in p2p networks. *Proc. of the 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid)*, November 2004.

[Giunchiglia *et al.*, 1993] F. Giunchiglia, L. Serafini, E. Giunchiglia, and M. Frixione. Non-omniscient belief as context-based reasoning. In *International Joint Conference on Artificial Intelligence (IJCAI 93)*, pages 548–554, 1993.

[Giunchiglia *et al.*, 1997] Fausto Giunchiglia, Adolfo Villafiorita, and Toby Walsh. Theories of abstraction. *AI Communications*, 10(3-4):167–176, 1997.

[Giunchiglia *et al.*, 1998] E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 626–635. Morgan Kaufmann, San Francisco, California, 1998.

[Giunchiglia *et al.*, 1999] E. Giunchiglia, F. Giunchiglia, and A. Tacchella. *SAT, KSATC, DLP and TA: a comparative analysis. In P. Lambrix, editor, *International workshop on description logics (DL'99)*, 30 July-1 August 1999.

[Giunchiglia *et al.*, 2000] E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. Sat vs. translation based decision procedures for modal logics: A comparative evaluation. *Journal of applied non classical logics*, 10(2), 2000.

[Giunchiglia *et al.*, 2002] E. Giunchiglia, A. Tacchella, and F. Giunchiglia. SAT-based decision procedures for classical modal logics. *Journal of Automated Reasoning*, 28(2):143–171, 2002.

[Giunchiglia, 1993a] F. Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, XVI:345–364, 1993. Short version in Proceedings IJCAI'93 Workshop on Using Knowledge in its Context, Chambery, France, 1993, pp. 39–49. Also IRST-Technical Report 9211-20, IRST, Trento, Italy.

[Giunchiglia, 1993b] F. Giunchiglia. Contextual reasoning. *Epistemologia*, XVI:345–364, 1993.

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

[Gómez-Pérez and Suárez-Figueroa, 2003] A. Gómez-Pérez and M.C. Suárez-Figueroa. Results of taxonomic evaluation of RDF(S) and DAML+OIL ontologies using RDF(S) and DAML+OIL validation tools and ontology platforms import services. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Florida, USA, October 2003.

[Gómez-Pérez and Suárez-Figueroa, 2004] A. Gómez-Pérez and M.C. Suárez-Figueroa. Evaluation of RDF(S) and DAML+OIL import/export services within ontology platforms. In *Proceedings of the Third Mexican International Conference on Artificial Intelligence*, pages 109 – 118, Mexico City, Mexico, April 2004.

[Goodman, 1993] P. Goodman. *Practical Implementation of Software Metrics*. McGraw Hill, London, 1993.

[Grady and Caswell, 1987] R.B. Grady and D.L. Caswell. *Software Metrics: Establishing a Company-Wide Program*. Prentice-Hall, 1987.

[Grant and Beckett, 2004] J. Grant and D. Beckett. RDF test cases. Technical report, W3C, February 2004.

[Guha, 1991] R.V. Guha. *Contexts: A Formalization and some Applications*. PhD thesis, Stanford University, 1991.

[Guo *et al.*, 2003] Y. Guo, J. Heflin, and Z. Pan. Benchmarking DAML+OIL repositories. In *Proceedings of the 2nd International Semantic Web Conference, (ISWC 2003)*, Florida, USA, October 2003.

[Haarslev and Möller, 1999a] V. Haarslev and R. Möller. Applying an ALC ABox consistency tester to modal logic SAT problems. In *Proceedings TABLEAUX'99, The 6th International Conference on Theorem Proving with Analytic Tableaux and Related Methods*, pages 24–28, Berlin, 1999. Springer-Verlag.

[Haarslev and Möller, 1999b] V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, volume 22, pages 115–119, Linkoeping, Sweden, July-August 1999. CEUR-WS.

[Haarslev and Möller, 2001a] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, 2001.

[Haarslev and Möller, 2001b] Volker Haarslev and Ralf 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.

[Hainaut, 1999] J.L. Hainaut. Methodology and case tools for the development of federated databases. *Journal of Cooperative Information Systems*, 8:169–194, 1999.

[Halevy *et al.*, 2003] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in a peer data management system. *ICDE*, 2003.

[Heuerding and Schwendimann, 1996] A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, and S4. Technical Report IAM-96-015, University of Bern, Switzerland, 1996.

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

[Horrocks and Patel-Schneider, 1998] I. Horrocks and P.F. Patel-Schneider. DL systems comparison. In E. Franconi, G. De Giacomo, R.M. MacGregor, W. Nutt, C.A. Welty, and F. Sebastiani, editors, *Proceedings of the International Description Logics Workshop (DL'98)*, volume 11, pages 55–57. CEUR-WS, May 1998.

[Horrocks and Patel-Schneider, 1999a] I. Horrocks and P.F. Patel-Schneider. Generating hard modal problems for modal decision procedures. In *Proceedings 1st workshop on Methods for Modalities (M4M-1)*. IOS Press, May 1999.

[Horrocks and Patel-Schneider, 1999b] I. Horrocks and P.F. Patel-Schneider. Performance of DLP on random modal formulae. In *International Workshop on Description Logics (DL'99)*, pages 120–124. Springer Verlag, 30 July-1 August 1999.

[Horrocks and Patel-Schneider, 2002] I. Horrocks and P.F. Patel-Schneider. Evaluating optimised decision procedures for propositional modal $\mathbf{K}_{(\mathbf{m})}$ satisfiability. *Journal of Automated Reasoning*, 28(2):173–204, February 2002.

[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 Tessaris, 2000] Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399 – 404, 2000.

[Horrocks *et al.*, 2000] I. Horrocks, P.F. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8(3):293–323, 2000.

[Horrocks, 1998] Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proceedings of the Sixth International Conference 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.

[Horvitz, 1987] E. J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In L. N. Kanal, T. S. Levitt, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pages 301–324. Elsevier, Amsterdam, The Netherlands, 1987.

[Hustadt and Schmidt, 1997] U. Hustadt and R.A. Schmidt. On evaluating decision procedures for modal logic. In *IJCAI (1)*, pages 202–209, 1997.

[Hustadt and Schmidt, 1999] U. Hustadt and R.A. Schmidt. An empirical analysis of modal theorem provers. *Journal of Applied Non-Classical Logics*, 9(4):479–522, 1999.

[IEEE, 1991] IEEE. *IEEE-STD-610 ANSI/IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology*. IEEE, February 1991.

[Isaac *et al.*, 2003] A. Isaac, R. Troncy, and V. Malais. Using XSLT for interoperability: DOE and the travelling domain experiment. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Florida, USA, October 2003.

[Jackson and Pais, 1990] P. Jackson and J. Pais. Computing prime implicants. In *Proceedings of the Tenth International Conference on Automated Deduction (CADE-90)*, pages 543–557, 1990.

[Kautz and Selman, 1994] H. Kautz and B. Selman. An Empirical Evaluation of Knowledge Compilation. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 155–160, 1994.

[Kim,] J. Kim. Moa core - a basic library for owl ontology merging and alignment applications. `http://mknows.etri.re.kr/moa/docs/moacore.html`.

[King, 2003] M. King. Living up to standards. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing*, Budapest, Hungary, 2003.

[Kitchenham *et al.*, 1994] B.A. Kitchenham, S.G. Linkman, and D.T. Law. Critical review of quantitative assessment. *Software Engineering Journal*, 9(2):43–53, 1994.

[Kitchenham *et al.*, 1995] B.A. Kitchenham, S.L. Pfleeger, and N.E. Fenton. Towards a framework for software measurement validation. *IEEE Transactions on Software Engineering*, 21(12):929–944, December 1995.

[Kitchenham *et al.*, 2002] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El-Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on Software Engineering*, 28(8):721–734, 2002.

[Kitchenham, 1996] B. Kitchenham. DESMET: A method for evaluating software engineering methods and tools. Technical Report TR96-09, Department of Computer Science, University of Keele, Staffordshire, UK, 1996.

[Klein, 2001] Michel Klein. Combining and relating ontologies: an analysis of problems and solutions. In Asuncion Gomez-Perez, Michael Gruninger, Heiner Stuckenschmidt, and Michael Uschold, editors, *Workshop on Ontologies and Information Sharing, IJCAI'01*, 2001.

[Klusch, 1999] Matthias Klusch, editor. *Intelligent information agent*. Springer-Verlag, Berlin, 1999.

[Klusch, 2001] Matthias Klusch. Information agent technology for the internet: A survey. *Data & Knowledge Engineering*, 36(3):337–372, 2001.

[Knoblock, 1989] C. A. Knoblock. A theory of abstraction for hierachical planning. In *Proceedings of the Workshop on Change of Representation and Inductive Bias*, 1989.

[Knublauch, 2003] H. Knublauch. Case study: Using protégé to convert the travel ontology to UML and OWL. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Florida, USA, October 2003.

[Korf, 1990] R. E. Korf. Planning as search: A quantitative approach. In J. Allen, J. Hendler, and A. Tate, editors, *Readings in Planning*, pages 566–577. Kaufmann, San Mateo, CA, 1990.

[Kotis and Vouros, 2004] K. Kotis and G. Vouros. HCONE approach to Ontology Merging. In *Proceedings of the 1st European Semantic Web Symposium (ESWS2004)*, 2004.

[Lambrix and Edberg, 2003] P. Lambrix and A. Edberg. Evaluation of ontology merging tools in bioinformatics. In *Proceedings of the Pacific Symposium on Biocomputing (PSB03)*, Kauai, Hawaii, USA, 2003.

[Lambrix *et al.*, 2003] P. Lambrix, M. Habbouche, and M. Pérez. Evaluation of ontology development tools for bioinformatics. *Bioinformatics*, 19(12):1564–1571, 2003.

[Lankford, 2000] W.M. Lankford. Benchmarking: Understanding the basics. *Coastal Business Journal*, (1), 2000.

[Larson *et al.*, 1989] J.A. Larson, S.B. Navathe, and R. Elmasri. A theory of attribute equivalence in databases with application to schema integration. *IEEE Transactions On Software Engineering*, 15:449–463, 1989.

[Lenat and Guha, 1990] D. B. Lenat and R. V. Guha. *Building Large Knowledge Based Systems*. Addison Wesley, 1990.

[Lenzerini, 2002] M. Lenzerini. Tutorial on "Data integration: A theoretical perspective". *ACM SIGMOD-PODS conference, Madison, WI, USA*, June 2002.

[Levesque, 1984] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 198–202, 1984.

[Levesque, 1988] H. J. Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17:355–389, 1988.

[Levesque, 1989] H. J. Levesque. A knowledge-level account of abduction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1061–1067, 1989.

[Li and Horrocks, 2003] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proc. of the Twelfth International World Wide Web Conference (WWW)*, pages 331–339. ACM, 2003.

[MacCartney *et al.*, 2003] B. MacCartney, S. McIlraith, E. Amir, and T. Uribe. Practical partition-based theorem proving for large knowledge bases. In *International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 89–96, 2003.

[Magkanaraki *et al.*, 2002] Aimilia Magkanaraki, Sofia Alexaki, Vassilis Christophides, and Dimitris Plexousakis. Benchmarking rdf schemas for the semantic web. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, pages 132–146. Springer-Verlag, 2002.

[Marquis and Sadaoui, 1996] P. Marquis and S. Sadaoui. A new algorithm for computing theory prime implicates compilations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 504–509, 1996.

[Marquis, 1995] P. Marquis. Knowledge compilation using theory prime implicates. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 837–843, 1995.

[Massacci, 1996] F. Massacci. Contextual reasoning is NP complete. In *National Conference on Artificial Intelligence (AAAI 96)*, pages 621–626, 1996.

[Massacci, 1999] F. Massacci. Design and results of the tableaux-99 non-classical (modal) systems comparison. In *TABLEAUX-99*, volume 1617 of *LNAI*, pages 14–18. Springer-Verlag, 1999.

[Massart, 1997] D. Massart. Complexity management of object schemas. Technical report, Universit catholique de Louvain, 1997.

[McAllester, 1990] David McAllester. Truth maintenance. In *Proceedings AAAI90*, pages 1109–1116. Morgan Kaufmann Publishers, 1990. internet file ftp.ai.mit.edu:/pub/dam/aaai90.ps.

[McCarthy and Buvač, 1998] J. McCarthy and S. Buvač. Formalizing context (expanded notes). In *Computing Natural Language*, volume 81 of *CSLI Lecture Notes*, pages 13–50. 1998.

[McCarthy, 1987] J. McCarthy. Generality in artificial intelligence. *Communications of ACM*, 30(12):1030–1035, 1987.

[Melnik *et al.*, 2003] S. Melnik, E. Rahm, and P. A. Bernstein. Rondo: A programming platform for generic model management. In *SIGMOD*, pages 193–204, 2003.

[Menczer and Monge, 1999] F. Menczer and A. Monge. Scalable web search by adaptive online agents: an infospiders case study. In M. Klusch, editor, *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*, pages 323 – 347. Springer, Berlin, 1999.

[Nayak and Levy, 1995] P. Pandurang Nayak and Alan Levy. A semantic theory of abstractions. In Chris Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 196–203, San Francisco, 1995. Morgan Kaufmann.

[Nelson and Powell, 2002] Stuart J. Nelson and Betsy L. Powell, Tammy andn Humphreys. The unified medical language system (umls) project. In Allen Kent and Carolyn M. Hall, editors, *Encyclopedia of Library and Information Science.*, pages 369–378. Marcel Dekker, Inc., 2002.

[Nelson *et al.*, 2001] Stuart J. Nelson, Douglas Johnston, and Betsy L. Humphreys. Relationships in medical subject headings. In Carol A. Bean and Rebecca Green, editors, *Relationships in the organization of knowledge*, pages 171–184, New York, 2001. Kluwer Academic Publishers.

[Ng *et al.*, 2003] W. Ng, B. Ooi, K. Tan, and A. Zhou. Peerdb: A p2p-based system for distributed data sharing. *ICDE*, 2003.

[Nicolle and Yetongnon, 2001] C. Nicolle and K. Yetongnon. Xml enabled metamodeling and tools for cooperative information systems. In *Conference on Electronic Commerce and Web Technologies (EC-Web 01)*, pages 260–269, 2001.

[Noy and Musen, 2001] N. F. Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, 2001.

[Noy and Musen, 2002] N.F. Noy and M.A. Musen. Evaluating ontology-mapping tools: Requirements and experience. In *Proceedings of the International Workshop on Evaluation of Ontology-based Tools (EON2002)*, Sigenza, Spain, October 2002.

[Omicini *et al.*, 2001] Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors. *Coordination of Internet Agents: Models, Technologies, and Applications*. Springer, 2001.

[OntoWeb, 2002] OntoWeb. Ontoweb deliverable 1.3: A survey on ontology tools. Technical report, IST OntoWeb Thematic Network, May 2002.

[Papazoglou *et al.*, 1992] Mike P. Papazoglou, S. C. Laufmann, and Timos K. Sellis. An organizational framework for cooperating intelligent information systems. *International Journal of Cooperative Information Systems*, 1(1):169–202, 1992.

[Parent and Spaccapietra, 2000] C. Parent and S. Spaccapietra. Database integration: the key to data interoperability. In *Advances in Object-Oriented Data Modeling*, 2000.

[Park *et al.*, 1996] R.E. Park, W.B. Goethert, and W.A. Florac. Goal-driven software measurement - a guidebook. Technical Report CMU/SEI-96-HB-002, Software Engineering Institute, August 1996.

[Patel-Schneider and Horrocks, 1999] P.F. Patel-Schneider and I. Horrocks. DLP and FaCT. In *Proceedings TABLEAUX'99, The 6th International Conference on Theorem Proving with Analytic Tableaux and Related Methods*, pages 19–23, Berlin, 1999. Springer-Verlag.

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

[Patel-Schneider, 1999] P. Patel-Schneider. Systems comparison. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics (DL'99)*, volume 22, pages 115–119, Linkoeping, Sweden, July-August 1999. CEUR-WS.

[Pelletier and Sutcliffe, 2001] J. Pelletier and G. Sutcliffe. CASC: Effective evaluation having an effect. In *Proceedings of the IJCAI'01 Workshop on Empirical Methods in Artificial Intelligence*, pages 33–40, Seattle, USA, 2001.

[Pentland, 1998] A. Pentland. *Wearable Intelligence*, volume 276. Scientific American, 1998.

[Pinto *et al.*, 2002] H. Sofia Pinto, Duarte Nuno Peralta, and Nuno J. Mamede. Using protégé-2000 in reuse processes. In *Proceedings of the International Workshop on Evaluation of Ontology-based Tools (EON2002)*, Sigüenza, Spain, October 2002.

[Plaisted, 1980] David A. Plaisted. Abstraction mappings in mechanical theorem proving. In *Proceedings of the 5th Conference on Automated Deduction*, pages 264–280, July 08-11 1980.

[Plaisted, 1981] David A. Plaisted. Theorem proving with abstraction. *Artificial Intelligence*, 16:47–108, 1981.

[Quine, 1959] W. V. O. Quine. On cores and prime implicants of truth functions. *American Mathematical Monthly*, 66, 1959.

[Rahm and Bernstein, 2001] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334 – 350, 2001.

[Rector and Nowlan, 1993] A.L. Rector and W.A. Nowlan. The galen project. *Computer Methods and Programs in Biomedicine*, 45:75–78, 1993.

[Reiter and de Kleer, 1987] R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-87)*, pages 183–188, 1987.

[Roelofsen and Serafini, 2004] F. Roelofsen and L. Serafini. Complexity of contextual reasoning. In *National Conference on Artificial Intelligence (AAAI 04)*, 2004. Accepted for pulication.

[Russell and Zilberstein, 1991] S. J. Russell and S. Zilberstein. Composing real-time systems. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 212–217, Sydney, Australia, 1991.

[Sacerdoti, 1973] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. In *Proc. of the 3rd IJCAI*, pages 412–422, Stanford, MA, 1973.

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

[Schrag and Crawford, 1996a] R. Schrag and J. Crawford. Compilation for critically constrained knowledge bases. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 510–515, 1996.

[Schrag and Crawford, 1996b] Robert Schrag and James M. Crawford. Implicates and prime implicates in random 3SAT. *Artificial Intelligence*, 81(1-2):199–222, 1996.

[Selman and Kautz, 1991] B. Selman and H. A. Kautz. Knowledge compilation using Horn approximations. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 904–909, 1991.

[Selman and Kautz, 1996] B. Selman and H. A. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43:193–224, 1996.

[Serafini and Bouquet, 2004] L. Serafini and P. Bouquet. Comparing formal theories of context in AI. *Artificial Intelligence*, 155:41–67, 2004.

[Serafini and Roelofsen, 2004] L. Serafini and F. Roelofsen. Satisfiability for propositional contexts. In *Principles of Knowledge Representation and Reasoning (KR 04)*, 2004. Accepted for publication.

[Serafini and Tamilin, 2004] L. Serafini and A. Tamilin. Distributed reasoning services for multiple ontologies. Technical report, ITC-IRST, 2004.

[Serafini *et al.*, 2003] L. Serafini, F. Giunchiglia, J. Mylopoulos, and P. A. Bernstein. Local relational model: a logical formalization of database coordination. *In Modeling and Using Context (CONTEXT 03)*, pages 286–299, June 2003.

[Sheth and Kashyap, 1992] A. Sheth and V. Kashyap. So far (schematically) yet so near (semantically). In *Conference on Semantics of Interoperable Databases Systems*, pages 272–301, 1992.

[Sill, 1996] D. Sill. comp.benchmarks frequently asked questions version 1.0, 1996.

[Sim *et al.*, 2003] S. Sim, S. Easterbrook, and R. Holt. Using benchmarking to advance research: A challenge to software engineering. In *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, pages 74–83, Portland, OR, 2003.

[Simon and del Val, 2001] L. Simon and A. del Val. Efficient consequence finding. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, 2001.

[Sole and Bist, 1995] T.D. Sole and G. Bist. Benchmarking in technical information. *IEEE Transactions on Professional Communication*, 38(2):77–82, June 1995.

[Spendolini, 1992] M.J. Spendolini. *The Benchmarking Book*. AMACOM, New York, NY, 1992.

[Stojanovic and Motik, 2002] L. Stojanovic and B. Motik. Ontology evolution within ontology editors. In *Proceedings of the International Workshop on Evaluation of Ontology-based Tools (EON2002)*, Sigüenza, Spain, October 2002.

[Stuckenschmidt and van Harmelen, 2002] Heiner Stuckenschmidt and Frank van Harmelen. Approximating terminological queries. In Troels Andreasen, Amihai Motro, Henning Christiansen, and Henrik Legind Larsen, editors, *Flexible Query Answering Systems, 5th International Conference, FQAS 2002, Copenhagen, Denmark, October 27-29, 2002, Proceedings*, volume 2522 of *Lecture Notes in Computer Science*, pages 329–343. Springer, 2002.

[Stuckenschmidt *et al.*, 2004] H. Stuckenschmidt, F. van Harmelen, L. Serafini, P. Bouquet, and F. Giunchiglia. Using C-OWL for the alignment and merging of medical ontologies. In *First International Workshop on Formal Biomedical Knowledge Representation Collocated with KR 2004*, February 2004.

[Stumme and Maedche, 2001] G. Stumme and A. Maedche. FCA-Merge: A Bottom-Up Approach for Merging Ontologies. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 2001.

[Subrahmanian *et al.*, 2000] V. S. Subrahmanian, Piero Bonatti, Jrgen Dix, Thomas Eiter, Sarit Kraus, Fatma Ozcan, and Robert Ross. *Heterogeneous Agent Systems*. MIT Press, 2000.

[Sure and Iosif, 2002] Y. Sure and V. Iosif. First results of a semantic web technologies evaluation. In *Proceedings of the Common Industry Program at the federated event co-locating the three international conferences: DOA'02: Distributed Objects and Applications; ODBASE'02: Ontologies, Databases and Applied Semantics; CoopIS'02: Cooperative Information Systems*, University of California, Irvine, USA, October-November 2002.

[Sure *et al.*, 2003] Y. Sure, J. Angele, and O. Corcho, editors. *Proceedings of the Second International Workshop on Evaluation of Ontology based Tools (EON 2003)*, volume 87 of *CEUR Workshop Proceedings*, Florida, USA, 2003. CEUR-WS Publication, available at http://CEUR-WS.org/Vol-87/.

[Sure *et al.*, 2004] Y. Sure, O. Corcho, J. Euzenat, and T. Hughes. Proceedings of the Tird International Workshop on Evaluation of Ontology based Tools (EON 2004), 2004. to appear.

[Sutcliffe and Suttner, 1998] G. Sutcliffe and C. Suttner. The TPTP problem library: CNF release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[Sutcliffe and Suttner, 2001] G. Sutcliffe and C.B. Suttner. Evaluating general purpose automated theorem proving systems. *Artificial Intelligence*, 131(1-2):39–54, 2001.

[Suttner and Sutcliffe, 1997] C.B. Suttner and G. Sutcliffe. The TPTP problem library v2.1.0. Technical report, Technical Report AR-97-04, Institut für Informatik, Technische Universität München, Munich, Germany; Technical Report 97/08, Department of Computer Science, James Cook University, Townsville, Australia, 1997.

[Tempich and Volz, 2003] C. Tempich and R. Volz. Towards a benchmark for semantic web reasoners - an analysis of the DAML ontology library. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, Florida, USA, October 2003.

[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.

[Teorey *et al.*, 1989] T.J. Teorey, G. Wei, D.L. Bolton, and J.A. Koenig. Er model clustering as an aid for user communication and documentation in database design. *Communications of the ACM*, 32:975–987, 1989.

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

[Thiran and Hainaut, 2001] P. Thiran and J.L. Hainaut. Wrapper development for legacy data reuse. In *WCRE'01*, 2001.

[Tison, 1967] P. Tison. Generalized consensus theory and application to the minimization of boolean circuits. *IEEE Transactions on Computers*, EC-16:446–456, 1967.

[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, *Proc. of the 3nd International Semantic Web Conference (ISWC)*, number 2870 in Lecture Notes in Computer Science, pages 882–896. Springer, 2003.

[Wache *et al.*, 2001] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hbner. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117, 2001.

[Weiss, 2002] A.R. Weiss. Dhrystone benchmark: History, analysis, scores and recommendations. White paper, EEMBC Certification Laboratories, LLC, 2002.

[Wiederhold, 1992] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25:38–49, 1992.

[Woelk and Tomlinson, 1995] D. Woelk and C. Tomlinson. Carnot and infosleuth: Database technology and the world wide web. In *ACM SIGMOD Intl. Conf. on the Management of Data*, May 1995.

[Wohlin *et al.*, 2002] C. Wohlin, A. Aurum, H. Petersson, F. Shull, and M. Ciolkowski. Software inspection benchmarking - a qualitative and quantitative comparative opportunity. In *Proceedings of 8th International Software Metrics Symposium*, pages 118–130, June 2002.

[Wooldridge and Jennings, 1995] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 1995.

[Ycart and Rousset, 2000] B. Ycart and M.C. Rousset. A zero-one law for random sentences in description logics. In *Proceedings of the Colloquium on Mathematics and Computer Science: Algorithms, Trees, Combinatorics and Probabilities*, September 2000.

[Zilberstein and Russell, 1996] Shlomo Zilberstein and Stuart J. Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82(1-2):181–213, 1996.

[Zilberstein, 1993] S. Zilberstein. *Operational rationality through compilation of anytime algorithms*. PhD thesis, Computer science division, university of California at Berkley, 1993.

[Zilberstein, 1996] S. Zilberstein. Using anytime algorithms in intelligent systems. *Artificial Intelligence Magazine*, fall:73–83, 1996.

Related deliverables

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

Project	Number	**Title** and relationship
KW	D2.2.1	**Specification of a common framework for characterising alignment** proposes a framework for defining what alignment are. This definition may influence the description of the module relationships.
KW	D2.2.3	For the modularity the alignments analysed in **State of the art on current alignment techniques** are interesting for the relationships between different modules/ontologies because of their ability to reconcile heterogeneity problems.