LarKC

The Large Knowledge Collider
a platform for large scale integrated reasoning and Web-search

FP7 – 215535

D4.1

A Survey of Web Scale Reasoning

Zhisheng Huang (Coordinator, VUA)
Barry Bishop(STI Innsbruck), Daniele Braga (CEFIREL),
Florian Fischer (STI Innsbruck), Frank van Harmelen (VUA),
Yi Zeng(WICI), Haiyan Zhou (WICI),
Jose Quesada (MPG), Jörg Rieskamp (MPG),
Lael Schooler (MPG), Annette ten Teije (VUA),
Axel Tenschert (HLRS), Emanuele Della Valle (CEFRIEL),
Holger Wache (University of Applied Sciences Northwestern Switzerland, FHNW)
Michael Witbrock (CycEuro), Ning Zhong (WICI)

Document Identifier:	LarKC/2008/D4.1
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.1.0
Date:	September 3, 2009
State:	final
Distribution:	public

Executive Summary

In this document, we make a survey of the relations between Web scale reasoning and various reasoning approaches which have been developed in computer science, artificial intelligence, logics, cognitive science and other relevant areas. The investigated reasoning approaches are: approximate reasoning, resource bounded reasoning, rule-based reasoning, contextual and modular reasoning, and distributed and parallel reasoning. All of the reasoning approaches are examined with respect to the five aspects of web scale reasoning: Scalability, Heterogeneity, Dynamcis, Inconsistency, and Parallelism.

In this document we also present a systematic analysis of the various use cases and tasks of semantic web reasoning, and discuss these tasks and use cases of with respect to the aspects of web scale reasoning.

Document Information

IST Project Number	FP7 – 215535	**Acronym**	LarKC
Full Title	Large Knowledge Collider		
Project URL	`http://www.larkc.eu/`		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	**Number**	4.1	**Title**	A Survey of Web Scale Reasoning
Work Package	**Number**	4	**Title**	Reasoning and Deciding

Date of Delivery	**Contractual**	M6	**Actual**	30-Sept-08
Status	version 1.1.0		final ☒	
Nature	prototype □ report ☒ dissemination □			
Dissemination Level	public ☒ consortium □			

Authors (Partner)	Zhisheng Huang (VUA), Barry Bishop(STI Innsbruck), Daniele Braga (CEFIREL), Florian Fischer (STI Innsbruck), Frank van Harmelen (VUA), Yi Zeng(WICI), Haiyan Zhou (WICI), Jose Quesada (MPG), Jörg Rieskamp (MPG), Lael Schooler (MPG), Annette ten Teije (VUA), Axel Tenschert (HLRS), Emanuele Della Valle (CEFRIEL), Holger Wache (FHNW), Michael Witbrock (CycEuro), Ning Zhong (WICI)			
Resp. Author	Zhisheng Huang (VUA)		**E-mail**	huang@cs.vu.nl
	Partner	VUA	**Phone**	+31 (20) 598-7823

Abstract (for dissemination)	In this document, we make a survey of the relations between Web scale reasoning and various reasoning approaches which have been developed in computer science, artificial intelligence, logics, cognitive science and other relevant areas. The investigated reasoning approaches are: approximate reasoning, resource bounded reasoning, rule-based reasoning, contextual and modular reasoning, and distributed and parallel reasoning. All of the reasoning approaches are examined with respect to the five aspects of web scale reasoning: Scalability, Heterogeneity, Dynamcis, Inconsistency, and Parallelism. In this document we also present a systematic analysis of the various use cases and tasks of semantic web reasoning, and discuss these tasks and use cases of with respect to the aspects of web scale reasoning.
Keywords	ontology reasoning, web scale reasoning, the Semantic Web

Project Consortium Information

Acronym	Partner	Contact
Semantic Technology Institute Innsbruck `http://www.sti-innsbruck.at`	STI	Prof. Dr. Dieter Fensel Semantic Technology Institute (STI) Innsbruck, Austria E-mail: dieter.fensel@sti-innsbruck.at
AstraZeneca AB `http://www.astrazeneca.com/`	ASTRAZENECA	Bosse Andersson AstraZeneca Lund, Sweden E-mail: bo.h.andersson@astrazeneca.com
CEFRIEL SCRL. `http://www.cefriel.it/`	CEFRIEL	Emanuele Della Valle CEFRIEL SCRL. Milano, Italy E-mail: emanuele.dellavalle@cefriel.it
CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O. `http://cyceurope.com/`	CYCORP	Michael Witbrock CYCORP, RAZISKOVANJE IN EKSPERIMENTALNI RAZVOJ D.O.O., Ljubljana, Slovenia E-mail: witbrock@cyc.com
Hchstleistungsrechenzentrum, Universitaet Stuttgart `http://www.hlrs.de/`	HLRS	Georgina Gallizo Hchstleistungsrechenzentrum, Universitaet Stuttgart Stuttgart, Germany E-mail : gallizo@hlrs.de
Max-Planck-Institut fr Bildungsforschung `http://www.mpib-berlin.mpg.de/index_js.en.htm`	MAXPLANCKGESELLSCH	Dr. Lael Schooler, Max-Planck-Institut fr Bildungsforschung Berlin, Germany E-mail: schooler@mpib-berlin.mpg.de
Ontotext Lab, Sirma Group Corp. `http://www.ontotext.com/`	ONTO	Atanas Kiryakov, Ontotext Lab, Sirma Group Corp. Sofia, Bulgaria E-mail: atanas.kiryakov@sirma.bg
SALTLUX INC. `http://www.saltlux.com/EN/main.asp`	Saltlux	Kono Kim SALTLUX INC Seoul, Korea E-mail: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT `http://www.siemens.de/`	Siemens	Dr. Volker Tresp SIEMENS AKTIENGESELLSCHAFT Muenchen, Germany E-mail: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD `http://www.shef.ac.uk/`	Sheffield	Prof. Dr. Hamish Cunningham THE UNIVERSITY OF SHEFFIELD Sheffield, UK E-mail: h.cunningham@dcs.shef.ac.uk
VRIJE UNIVERSITEIT AMSTERDAM `http://www.vu.nl/`	Amsterdam	Prof. Dr. Frank van Harmelen VRIJE UNIVERSITEIT AMSTERDAM Amsterdam, Netherlands E-mail: Frank.van.Harmelen@cs.vu.nl
THE INTERNATIONAL WIC INSTITUTE, BEIJING UNIVERSITY OF TECHNOLOGY `http://www.iwici.org/`	WICI	Prof. Dr. Ning Zhong THE INTERNATIONAL WIC INSTITUTE Mabeshi, Japan E-mail: zhong@maebashi-it.ac.jp

INTERNATIONAL AGENCY FOR RESEARCH ON CANCER `http://www.iarc.fr/`	IARC2	Dr. Paul Brennan INTERNATIONAL AGENCY FOR RESEARCH ON CANCER Lyon, France E-mail: brennan@iarc.fr

Table of Contents

1 Introduction

General background. In recent years (since roughly 1999), an important challenge for Knowledge Representation and Reasoning has been raised by a research area that is known as the "Semantic Web". The essence of the Semantic Web is the idea that information on the Web can be made available in machine processable form, in order to allow techniques from Knowledge Representation and Reasoning to be applied, and hence allow for more intelligent support for Web users. This would enable novel uses of the Web such as semantic search, data integration, personalisation and others.

Most of today's Web content is suitable for human consumption only, and is not amenable to intelligent processing by machines. Even Web content that is generated automatically from databases is usually presented without the original structural information found in databases. Typical uses of the Web today involve people seeking and combining information, or reviewing catalogues of on-line stores and ordering products by filling out forms. Much of this work must be done by the users themselves, without machine support: search-engines only search for strings without understanding the concepts denoted by these strings, and hence suffer from homonym- and synonym-problems, information from different catalogues cannot be combined automatically (except by ad hoc tools such as the current shop-bots and price-comparison sites), and web-pages cannot be personalised to match the user's interests.

The main obstacle to providing better support to Web users is that, at present, the meaning of Web content is not machine-accessible. The Semantic Web vision proposes to make (at least some aspects of) information on the Web machine processable by representing it in a structured form with a clear logical semantics. In other words: the Semantic Web proposes to apply Knowledge Representation techniques on a world-wide scale.

Researchers have developed reasoning methods for rather small, closed, trustworthy, consistent, and static domains. They usually provide a small set of axioms and facts. For many DL Logics, their reasoners can deal with about 10^5 axioms (concept definitions), but they scale poorly for large instance sets. Logic programming engines can deal with similar-sized rule sets as well as larger instance sets (say, 10^6), but they can draw only simple logical conclusions from these theories. Both streams are highly interesting areas of research, and topics such as how to combine them attract a lot of attention. Still, there is a deep mismatch between reasoning on a Web scale and efficient reasoning algorithms over restricted subsets of first-order logic.

The goals of LarKC. The LarKC project is focused on this large scale reasoning. It offers a platform that exists (see deliverable D1.2.1) of several types of plug-ins: Identify plug-ins, Transform plug-inn, Select plug-inn, Reasoning plug-in and Decide plug-in. In this document we are considering the Reasoning plug-in. The functional behaviour of a reasoning plug-in is that it draw conclusions/hypotheses from selected information. We will survey different reasoning approaches, theories, and technologies, which have been developed in computer science, artificial intelligence, logics, cognitive science and other relevant areas. By which, we want to know to what extent they can be used or adapted to deal with similar problems for web scale reasoning. More exactly, we will survey how those approaches, theories, and technologies can be used for web scale reasoning with respect to the *following goals* for the LarKC platform:

- **Scalability**. Web data are growing rapidly. The same problem also occurs for ontologies and other semantic web data. We want to know how the scalability problem can be solved by the inspiration from those approaches which have been developed in relevant areas.
- **Heterogeneity**. Web data come from very different data resources. They are represented with different data formats and can be very different from one another. Dealing with web data with various data formats or integration of various reasoning approaches is an important issue in web-scale reasoning.
- **Dynamics**. Web data are not static. They are always changed with various application scenarios and context. We will survey how relevant approaches may be introduced or similar approaches may be developed to deal with dynamic web data.
- **Inconsistency**. Web data may consist of significantly large amounts of inconsistent, incoherent, and noisy data. Dealing with inconsistency becomes an important issue for web scale reasoning.
- **Parallelism**. Whether or not those approaches can be achieved by parallel processes or distributed systems.

Different approaches towards the goals. In the sequel chapters we will make a survey on the following families of approaches. Each chapter discusses a number of techniques, which we will relate to the desired goals of the LarKC platform like scalability, heterogeneity, inconsistency.

The families of approaches are:

- **Approximate Reasoning**. Approximate reasoning has been developed in logics and artificial intelligence to deal with the scalability problem. The idea behind various approximate reasoning approaches is to obtain approximate reasoning answers, rather than sound and complete answers. We will survey anytime availability which trade-offs between computation time and answer quality. We will explore various query re-writting technologies, which allow many possibilities to facilite various database technology for Web scale reasoning.
- **Resource Bounded Reasoning**. Intuitions on the value of resource bounded reasoning are of course present in many different areas of formal investigation. We will survey real time reasoning in which time is critical. Heuristics are simple decision strategies that exploit informational structures in the environment. Heuristics can not only perform as well as more complex algorithms, they can also perform better. The surprisingly high performance of heuristics results from their ecological rationality, exploiting the statistical structure of its environment. We will exploit such cognitively inspired heuristics for web scale reasoning.

- **Rule-based Reasoning for dynamic and incomplete knowledge**. We will survey rule-based systems, logic programming, non-monotonic reasoning, and default reasoning, and examine to what extent various rule-based technologies can be used for Web scale reasoning.

- **Contextual and Modular Reasoning**. We discuss different approaches to exploit modularity and context as a way of dealing with scale.

- **Distributed and Parallel Reasoning**. We will take a closer look at various approaches of distribution which should be considered for Web Scale reasoning, which include P2P architectures and various approaches such as a Web Service.

Linking reasoning tasks to types of use-cases. In the final chapter of this document we will also investigate what are the typical use cases for semantic web reasoning. We categorise those use-cases, and analyse them by decomposing them in a number of basic reasoning tasks. Subsequently we can discuss whether the different approaches can play a role in these basic reasoning tasks. We will give concrete techniques from a particular family (e.g. approximation techniques) and show how these techniques can be applied for a basic reasoning task such that it contributes to some large scale goals (like scalability, heterogeneity etc.) of the LarKC platform.

Structure of this document. In the Chapters 2-6 we discuss a particular family of reasoning methods. All the different techniques inside a family will be scored against the large scale reasoning goals. In Chapter 2 we overview various methods of approximate reasoning and discuss approximate reasoning from the perspective of human intelligence. In Chapter 3 we survey the theories of resource bounded reasoning and various approaches of bounded rationality. In Chapter 4 we explore rule-based reasoning, both from the perspectives of computer science and that of human intelligence. In Chapter 5, we discuss the contextual reasoning and modular reasoning. In Chapter 6 we investigate various approaches of distributed reasoning and parallel reasoning. In Chapter 7 we present a categorisation of various use cases and identify a number of basic reasoning tasks that are used in web scale reasoning. In Chapter 8 we conclude the document.

2 Approximate Reasoning

2.1 A brief overview of approximation techniques

Approximate reasoning is nonstanard reasoning which is based on the idea of sacrificing soundness or completeness for a significant speed-up reasoning. This is to be done in such a way that the number of introduced mistakes is at least outweighed by the obtained speed-up[124]. In this section we aim to give a brief overview of various techniques for approximate reasoning *in general*, i.e. without any specific application to the Semantic Web tasks.

A full survey of the extensive literature on approximate reasoning (with contributions from logic, from Computer Science and from AI) is beyond the scope of this deliverable. An extensive survey can be found in [63]. Both that survey and this section are based on a classification of techniques originally given in [61].

The typical problem solving architecture of a reasoning method has two inputs, namely (i) a request and (ii) knowledge to be used in answering that request, and as output (iii) the answer of the problem. This is depicted in Figure 2.1. In [61] this architecture is used for classifing approximation methods, namely methods that approximate the request, or the knowledge base or the reasoning method. We will discuss a number of approximation techniques by following this structure.

Approximation of the reasoning method

In [130], Schaerf and Cadoli define two approximations of classical propositional entailment, named $\vdash_1$ and $\vdash_3$ which are either unsound but complete ($\vdash_1$) or sound but incomplete ($\vdash_3$). Both of these approximations are parameterised over a set of predicate letters S (written $\vdash_1^S$ and $\vdash_3^S$). This set S determines the accuracy of the approximate entailment relations. The idea is that predicate letters in S are given classical truth assignments, while for letters outside S all literals are false (i.e. both x and $\neg x$ are false, so called 1-S-assignments), while in 3-S-assignments we allow either or both x and $\neg x$ to be true. These non-standard truth-assignments

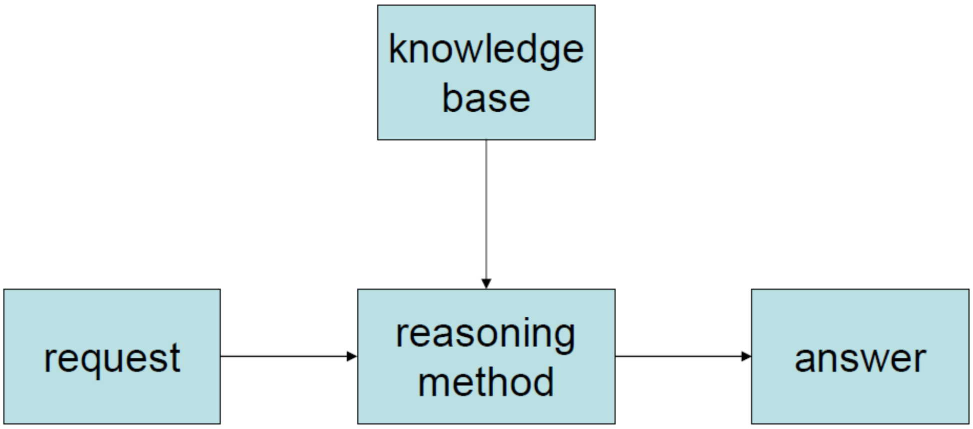

Figure 2.1: Typical knowledge-based architecture

result in non-standard entailment relations. With increasing S, these are closer approximations of the classical entailment.

In [38] and [39], Dalal and Yang define the family of approximate deduction relations BCP_k, which are variants of Boolean Constraint Propagation (BCP). BCP [95] is a limited form of deduction which is a variant of unit resolution, which performs limited deduction in linear time, at the price of incompleteness. Given a theory T, BCP monotonically expands T by adding literals as follows: in each step, if any single clause in T and all the literals in T taken together entail any other literal (or true), then this literal (or false) is added to the theory T. Allowing BCP to chain on arbitrary clauses would make it sound and complete and would therefore render it uninteresting as an approximate deduction method. This is the motivation for defining BCP_k [38], where chaining is allowed, but only on formulae of limited length: when k increases, BCP_k is allowed to chain on ever more formulae, and will become ever more complete (notice that for $k = 1$, BCP_k reduces to BCP).

Another family of approximate reasoning methods is the abstraction theory proposed by [58], and later refined by [107]. The main idea is to create an abstract, simpler form of the original problem, then solve this abstract simpler problem, and finally refine the abstract solution to the concrete problem in order to obtain a concrete solution.

Approximation of the request

The most obvious example of approximation the request box in Figure 2.1 is query-relaxation. In [149], Stuckenschmidt proposes a query-relaxation based on the assumption that less complex queries can be answered in shorter time. Following this idea, they propose to compute a sequence of queries $Q_1, \ldots, Q_n$ starting with very simple ones and gradually increasing their complexity. In order to use these simpler queries as approximations for the original query the quality of the results of the sequence of queries must be non-decreasing.

An alternative approach is formed by so-called "top-k query algorithms" (e.g. [13]). Such algorithms do not change the query, but rather limit the number of answers returned on the original query to k answers, with k increasing number over time. In other words, the original query Q is changed to a series of queries $Q(k)$, where k limits the size of the answer set. The assumption is that a smaller answer set is cheaper to compute than a large one.

In Section 2.2, We will explore various query re-writing technologies in DL reasoning and the Semantic Web, which allow many possibilities to facilitate various database technology to improve the scalability.

Approximation of the knowledge base

Knowledge compilation is the area that deals with translating the knowledge base such that the computational complexity of reasoning decreases. 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 it by) another knowledge base with better computational properties. This "compiled" knowledge base can be re-used to solve many problem instances, thereby saving time and computational resources when compared to solving the problem instances with the original knowledge base. Two classical approaches have been developed for addressing the computational hardness of reasoning problems, which are language restriction and theory approximation. Both can be classified in the general scheme 2.1 as approximating the knowledge base.

Language restriction: We may restrict the language used to represent knowledge, so that knowledge is compiled into the restricted language. An example of language restriction is the method developed in [136], where a knowledge base is approximated by a set of Horn formulae. 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).

Theory approximation: Using theory approximation, one can compile the theory into another "easier" theory (although still expressed in the same language). During the compilation one can give up the soundness or the completeness in the answer to the original reasoning problem. An example of theory approximation is using an exact knowledge compilation method that can be turned into an approximate method by stopping it before it is completed, because the compilation method is an anytime algorithm. Examples are methods for computing prime implicants and prime implicates like [94, 42]. Another example is the approach of Instance Store[1], where a retrieval engine is proposed that is sound and complete for role-free A-boxes in Description Logics. In other words: the theory is approximated by removing all roles from the A-box. For query answering, Pan and Thomas [113] apply the idea of knowledge compilation on semantically approximating a source ontology Os in a more expressive DL Ls (source language) with its (least) upper-bound Ot in a less expressive DL Lt (target language). The authors show that, by applying their approach to approximating OWL DL ontologies to OWL 2 QL (DL-Lite$_R$) ones, their system (Quill) could provide scalable query answering for OWL DL ontologies: the reasoning service is soundness preserving in general; when there are no non-distinguished variables, the reasoning service is sound and complete.

2.2 Query Rewriting Technologies

SPARQL denotes the SPARQL Protocol and RDF Query Laguage[2], which is designed for querying RDF-based information and for representing the query answers. For Web scale reasoning, we ususally consider querying knowledge bases that use advanced ontology languages like OWL-DL and various fragments of OWL2[3]. SPARQ-DL is a substantial subset of SPARQL, which is designed for quering certain OWL-DL based semantics[144].

Conjunctive queries (CQ) are a fundamental mechanism for querying DL-based knowledge bases. A conjunctive query q is of the form $q(X) \leftarrow \exists Y.conj(X,Y,Z)$ or simply $q(X) \leftarrow conj(X,Y,Z)$, where $q(X)$ is called the head, $conj(X,Y,Z)$ is called the body, X are called the distinguished variables, Y are existentially

[1] http://semanticweb.org/wiki/Instance_Store

[2] http://www.w3.org/TR/rdf-sparql-query/

[3] http://www.w3.org/TR/owl2-profiles/

quantified variables called the non-distinguished variables, Z are individual names, and $conj(X, Y, Z)$ is a conjunction of atoms of the form $A(v)$, $R(v_1, v_2)$, where A, R are respectively named classes and named properties, v, v_1 and v_2 are individual variables in X and Y or individual names in Z.

In DLs, a TBox is used to represent conceptual information, whereas Abox is used to store instance data. For Web scale reasoning, the size of instance data is usually extremely large. Efficient and scalable approaches to deal with conjuctive querys on DL-based knowledge bases are very important for Web scale reasoning. In [29], Calvanese et al. argue that true scalability of conjunctive query answering over DL-based knowledge bases can only be achieved by making use of standard relational database management systems. Conjunctive query answering generalizes instance retrieval by admitting also queries whose relational structure is not tree-shaped. Calvanese et al. show that the logics of the DL-Lite family are the maximal DLs supporting efficient query answering over large amounts of instances.

Calvanese et al. propose an algorithm which takes a conjunctive query and a DL-Lite$_R$ Tbox as input and computes a union of conjunctive queries that is rewritting of the query with respect to the Tbox. In [114], Perez-Urbina, Motik, and Horrocks propose an algorithm which takes a conjunctive query and an $\mathcal{ELHI}$ Tbox as input, and returns a re-writting query of the conjunctive query, and shows that their algorithm will often produce smaller rewrittingd that Calvanese's algorithm.

In [84], Krötzsch, Rudolph, and Hitzler propose an algorithm for answering conjunctive queries in the tractable $\mathcal{EL}^{++}$-fragment of $\mathcal{SROIQ}$ and thus of OWL2. The algorithm is based on an automata-theoretic formulation of complex role inclusion axioms. The algorithm allows one to derive a number of complexity results related to conjunctive query answering in $\mathcal{EL}^{++}$. They show that the conjunctive queries in $\mathcal{EL}^{++}$ are undecidable in general and identify the $\mathcal{EL}^{++}$-fragment of $\mathcal{SROIQ}$ as an appropriate decidable sub-DL.

2.3 Variable Precision Logics and Granular Reasoning

Variable precision logic is a major method for approximate reasoning based on incomplete information under time constraints, which provides two reasoning strategies, namely, variable certainty and variable specificity reasoning [99]. Given more time, a system with variable certainty can provide a more certain answer, while a system with variable specificity can provide a more specific answer [99].

Variable certainty reasoning belongs to non-monotonic reasoning, and the certainty won't necessarily go higher as more data is involved (since there might be contradictions [31] or inconsistency [74] on the facts, especially in the dynamic changing context of the Web). How it can be applied to a more user-centric environment at Web scale still needs further investigations. In our study, we find a bridge for variable specificity reasoning and granular reasoning.

The study of granular reasoning emphasizes on reasoning with facts under different levels of granularity (grain size), which is also aimed at dealing with the problem of reasoning under incomplete information. It starts from the logic approaches for granular computing, such as some logic foundations for granularity [69], a logic approach of granular computing which emphasizes on multilevel granular structures [161], granular logic based on rough sets [91], etc. Under the term of granular reasoning, it has also been studied from some perspectives. Granular reasoning using zooming-in and zooming-out has been studied from the perspectives of propositional reasoning [105], Aristotle's categorial syllogism [106], etc. Granular reasoning based on granular space has also been proposed [159]. These studies concentrate on the logic foundations for reasoning under multi-granularity. In LarKC, we plan to unify search and reasoning from the viewpoint of granularity and extend the strategies of granular reasoning.

In future research for LarKC, we plan to design various strategies to unify search and reasoning. For example, the multilevel specificity strategy is inspired by variable specificity reasoning. The major difference is that: variable specificity reasoning uses "if-then-unless" rule [99], while multilevel specificity strategy will use hierarchical knowledge structure to supervise the unification process of search and reasoning. Since for LarKC, a subset of the original dataset is selected for the later reasoning process. A strategy called "Multilevel completeness reasoning" is proposed. It can be used to provide reasoning results with multiple levels of completeness. Meanwhile, it can produce a method to predict the completeness value for user judges whether the result is good enough. Plus, we also plan to seek for inspirations from Cognitive Science (e.g. the basic level advantage for problem solving [123] etc. which might be relevant to unifying search and reasoning from the viewpoint of granularity).

In conclusion, we concentrate on how proposed strategies for variable precision logic, current studies of granular reasoning, and new strategies proposed by us can help to effectively solve Web scale incomplete reasoning problems under time constraints and in an interactive way.

2.4 Approximate Reasoning and Web Scale Reasoning

In this chapter, we will make an analysis on the relation between approaches of approximate reasoning and the requirement of web scale reasoning. As discussed in Chapter Introduction, Web scale reasoning would consider the following problems: Scalability, Heterogeneity, Dynamics, Inconsistency, and Parallelism.

Scalability As analyzed above, almost all of the approximate reasoning approaches in logics and Artificial Intelligence aim to deal with the scalability problem. Although those approaches have not yet provided a perfect solution for the scalability, they can be still classified as ones for the scalability.

The query-rewriting techniques involve DL-Lite family can use relational database systems have proved to be very efficient for the scalablity. The query-rewriting techniques which involve $\mathcal{EL}^{++}$-family can deal with large scale T-box. They can be still classified as ones for the scalability.

Variable precision logic and granular reasoning achieves scalability by providing reasoning results with multiple levels of details. If there is not much time, a more general result will be provided. If more time is allowed, more specific results will be produced [99].

Heterogeneity Actually Heterogeneity has multiple meanings. It may mean support for multiple data formats, or mean support for different reasoners, or

Classification	Method	Scalability	Heterogeneity	Dynamics	Inconsistency	Parallelism
Approximation (method)	Schaerf and Cadoli	yes	no	no	yes	no
Approximation (method)	Dalal	yes	no	?	?	no
Approximation (request)	simple queries	yes	no	?	?	no
Approximation (request)	top-k queries	yes	no	?	?	no
Approximation (KB)	Language restriction	yes	maybe	no	maybe	yes
Approximation (KB)	theory approximation	yes	maybe	no	maybe	yes
Approximation (KB)	Pan and Thomas et al.	yes	yes	yes	?	maybe
Approximation (query-rewriting)	Calvanese et al.	yes	no	?	no	maybe
Approximation (query-rewriting)	Perez-Urbina et al.	yes	no	?	no	maybe
Approximation (query-rewriting)	Krötzsch et al.	yes	no	?	no	?
Approximation (Human)	Granular Reasoning	yes	yes	yes	yes	no
Approximation (Human)	VPL	yes	yes	yes	yes	no

VPL= Variable Precision Logics

Figure 2.2: Approximate Reasoning and Web Scale Reasoning

others. In this section we would consider only the heterogeneity which allows for different specification languages. Most approaches of approximate reasoning such as Schaerf and Cadoli's approach, Dalal's approach, the approach of simple queries, and the approach of top-k queries, would not support different specification languages. Therefore, their answers to this heterogeneity is "no". The approach of the language restriction and that of theory approximation allows for some kinds of converting specification. Thus, we can consider them as ones with non-restricted sense of heterogeneity.

For the approaches of approximate reasoning from the perspectives of human intelligence, we would consider only the heterogeneity which allows for different specification languages" because the heterogeneity can be defined as the reasoning method can process different specification languages. Granular reasoning has not developed a demo to deal with this situation yet. However, from the perspectives of human intelligence, people can deal with heterogeneity.

Dynamics None of the approximate reasoning approaches in logics and Artificial Intelligence claim to support reasoning in a dynamic environment. Thus, the answers to this requirement is "no" in general. However, it is still a question whether or not the approach of simple queries can deal with rapidly changed data efficiently.

Inconsistency Schaerf and Cadoli's approach can be extended to deal with inconsistency, as reported in Huang et al 2005[73]. Therefore, it can be classified into one for inconsistency. It is still an open question whether or not the other approaches of approximate reasoning can be extended into ones which can be used for reasoning with inconsistencies or repairing inconsistency. As analysed above, Granular reasoning can deal with the inconsistency problem.

Parallelism The approach of the language restriction and that of theory approximation allows for some kinds of converting specification. That may be achieved by parallel processing. Thus, the answers of those two approaches to parallelism can be considered to be positive. None of the other approaches report to support parallel processing or distributed processing. Therefore, the answers to this are negative.

The table 2.2 is a summary of various approximate reasoning methods and their relation with the requirement imposed by web scale reasoning.

3 Resource Bounded Reasoning

Perhaps more than anybody else in economic theory, Herbert A. Simon stressed that individual decision makers have no choice but to make decisions under the constraints of limited cognitive resources (e.g., [142]). On the basis of this indisputable truth about the human cognitive system, he challenged classical economic theory, which in his view projected an omniscient rationality assuming unbounded knowledge, computational capacities, and time. Simon argued that psychologically plausible theories of decision making, which assume realistic limits on the knowledge and computational abilities of the human agent, also lead to conclusions at the level of aggregate phenomena. Importantly, these conclusions are not always the same as those suggested by neoclassical theory, thus rendering possible crucial tests [142]. Simons vision of a different rationality of economic behavior, bounded rationality [141, 143], has not only posed a challenge to economic theory but has also suggested a new research agenda revolving around the following key question: how rational are people, given their limited computational capabilities and their incomplete knowledge?

In psychology, two research programs have worked toward answering this question. One program is the heuristics and biases program instigated in the early 1970s by Daniel Kahneman and Amos Tversky; the other is the program on fast and frugal heuristics initiated by Gerd Gigerenzer and colleagues (e.g., [57, 37]). The goal of this chapter is to introduce the view of bounded rationality as exposed by the fast and frugal heuristics program.

3.1 Bounded Rationality as Ecological Rationality

The research program on fast and frugal heuristics advocates a different interpretation of bounded rationality from that of heuristics and biases research one that does not uncritically accept the normative standard of logic, statistics, and probability theory. In this view, psychological mechanisms such as heuristics are adapted to particular task environments, and this match between particular environment structures and heuristics can enable the reasoner to behave adaptively, that is, in a computationally fast, information-frugal, and comparatively accurate way in the face of environmental challenges. These real-world requirements fulfilled by the match between environment and cognition lead to a new conception of what proper reasoning is: ecological rationality. In other words, a heuristic is not just rational or irrational. Instead, it can be judged as rational only with respect to the environment in which it is used.

The notion of ecological rationality, that is, the tandem of cognition and environment, is highlighted in Simons analogy between bounded rationality and a pair of scissors: Human rational behavior ... is shaped by a scissors whose two blades are the structure of task environments and the computational capabilities of the actor[143]. Just as one cannot understand the function of scissors by looking at a single blade, one cannot understand humans cognition by studying either the environment or cognition alone.

The research rationale of the fast and frugal heuristics program begins by analyzing the structure of a specific task environment people face, and then based on the analysis derives attributes of the cognitive models of reasoning that could fare well within the environment. This program thus moves from the environmental structures to the psychological structures, or in terms of Simons metaphor from the environmental blade to the cognitive blade. In the context of web search this implies that people could use different search heuristics depending on the search context. In a context in which lots of information is available search heuristic are necessary that substantially reduce the amount of retrieved information. In contrast, in a context in which little information is retrieved by a search query, heuristics need to be applied that generate more information to produce satisfying answers to the problem. The point is that to understand which reasoning processes people might follow, and when and why these processes work well, one needs to explore the characteristics of the environment. This point was made over sixty years ago by Egon Brunswik [27] and has been made by various psychologists since (e.g., [7, 55, 140]).

The cognitive blade of Simons scissors analogy implies that models of peoples reasoning should be psychologically plausible. That is, the cognitive processes proposed need to be realistic insofar as people need to be able to execute them given their computational capabilities. Simon [141, 143] repeatedly insisted that the cognitive constraints under which people make judgments and decisions have to be taken into account when modeling human behavior. Due to cognitive limitations, people cannot help but use approximate methods to handle most tasks [143]. The strategy of taking both environmental structures and cognitive constraints into account winnows down the set of possible cognitive models that describe human decision making. Fast and frugal heuristics are suggested as one of the remaining candidates for how humans actually make decisions.

Owing to the focus on the task environment and the match between cognition and environment, the fast and frugal heuristics program does not compare peoples judgments to mathematically defined norms of rationality with the explicit purpose of abstracting away from particular content domains and thus holding across a wide range of environments. Instead, it conceptualizes and measures human rationality in terms of performance criteria in the real world. Successful performance includes making accurate decisions in a minimal amount of time and using a minimal amount of information. In other words, this program of heuristics replaces the multiple coherence criteria (i.e., measures that evaluate the logical and mathematical consistency of decisions, stemming from the laws of probability theory and logic) with multiple correspondence criteria (i.e., measures that relate decision-making strategies to the external world and to success therein)[67].

Gigerenzer and colleagues [37] define heuristics as precise computational models that specify the steps of information gathering and processing involved in generating a decision. A heuristic consists of a search rule, a stopping rule, and a decision rule. Search rules specify how decision options are generated and how information about available options is gathered. Stopping rules define when the process of searching for options or information is terminated. These stopping principles themselves need to be simple (owing to the computational limitations of the human mind), such as the aspiration levels in Simons notion of satisficing[141, 143]. Finally, decision rules specify how decisions are made based on the information obtained. They, in turn, must also be simple and computationally bounded. For instance, a decision could be made based on only one reason or cue, regardless of the total number of cues retrieved during the information search.

In many settings, Gigerenzer and colleagues have found that the one-reason decision-making approach of heuristics performs surprisingly well when compared to more complex strategies (see [37]). There are two main reasons why simple heuristics often perform so well in comparison with more complex strategies. First, many problems have flat maxima, meaning the best solution does not differ substantially from other (e.g., heuristic) solutions. Second, heuristics can outperform other strategies in terms of generalization, that is, when applied to new situations. Dawes and Corrigan ([41]) showed that if one uses a linear model for predicting a criterion, many sets of weights will lead to predictions that are similar to those made by optimal weights. In other words, around the model with maximum performance there are many other solutions with near-maximum performance, an example of the flat maximum phenomena. The second advantage of these heuristics is their ability to generalize well. In many situations, the parameters of models are estimated on the basis of a sample. In principle, the data of a sample can be thought of as consisting of two components: structure and noise. Although the structure is of primary interest, a relatively complex model will also increase its fit by adapting to (and trying to predict) the samples noise. This leads to the problem of overfitting, that is, a model that fits the data well but does worse at predicting new data (see [26]). When a complex model increases its sample accuracy by fitting noise, its performance will be relatively poor when applied to new independent data. In contrast, a relatively parsimonious, less flexible model such as a simple heuristic might perform worse when fitting the data but will often outperform the more general model when making predictions for a new sample. Thus, the greater the complexity of a strategy, the greater the risk that it might perform poorly when applied to a new problem.

In sum, the decision maker can gain important advantages by selecting a simple heuristic for solving a new problem. First, a heuristic requires only a small amount of information, which is processed easily. Second, due to the flat maximum phenomenon and heuristics and robustness, heuristics can perform as well as, or even better than, more complex strategies, undermining the complexity accuracy relationship that is usually assumed. Hence the belief that basing a decision on more information and computation will always lead to more accurate decisions, a belief that has dominated much research is a fiction.

3.2 Heuristic reasoning with bounded resources

Heuristic may enhance the ability to solve a problem efficiently. Researchers in many domains such as psychology and AI have conducted many researches on this topic. Specifically, Newell and Simon [109] proposed the theory of information processing based on behavioral experiments. They elaborated problem state space theory in ”Human Problem Solving”, in which problem solving is defined as the process of searching the possible path in space of mind with restriction and teleology. They emphasized that representation and searching are the two important steps to solve the unfamiliar problems; the representation is the process of describing the problem and the searching is to find a feasible way to settle it according to the description. Simon introduced the concept of limited rationality to better model these processes. Based on the concept of limited rationality, the heuristic methods are proposed to improve the efficiency of problem solving.

Many researchers have used this theory to explain and to simulate the processes of solving the problems such as the Hanoi tower, the savage and the missionary, and so on.

With the application of new technology of brain functional imaging such as fMRI and the new understanding of the cognitive process, many results have been reported about the brain mechanism during problem solving. Many problem solving tasks, such as Tower of London, Water Jug task, algebraic equations, etc., require heuristic reasoning to help to attain the answer more efficiently. For example, recent researches have confirmed that the prefrontal cortex plays an important role in problem solving, reasoning. It is proved that the prefrontal region is the physiological foundation of controlling and integrating all kinds of complex descriptions when carrying out the different forms of reasoning, problem solving and planning [46, 66, 77]. Through Tower of London experiment, researchers [138, 70, 103] found that the performance of patients with lesions to the prefrontal cortex is significantly worse than that of controls. Patients' poor performance is a result of deficits in "planning" or "look-ahead" abilities. Colvin, Dunbar, and Grafman [36] predicted that frontal patients may also perform more poorly than normal participants in Water Jug task. In a series of studies, researchers have identified two left cortical regions that are intimately involved in various symbolic tasks. These studies [9, 146] involved solving of algebraic equations, isomorphs of algebra, and memory retrieval. Across these experiments, researchers have found the involvement of two regions reflecting abstract information processing. One is a left parietal region whose activity reflects changes to the problem representation and the second is a left prefrontal region whose activity reflects retrieval of stored information.

Some cognitive models have been proposed based on results of cognitive psychology to simulate/model heuristic reasoning during problem solving. For example, Langley proposed the BACON model that used the data-oriented inductive methods to find the scientific rules and successfully simulate the creative thought of human being. SOAR (State Operator And Result) model, proposed by Newell [108], used production rules to remember how to solve the problem and use the same rules when the problem occurs again. ACT-R (Adaptive Control Thought-Rational), proposed by Anderson et al [4], is a platform to build computational models to simulate/predict human cognitive behavior, and fMRI BOLD (Blood Oxygen Level Dynamic) effect.

The key of Web problem solving system is how to study and use the heuristic rules efficiently in Web environment. The study of brain mechanism of heuristic problem solving and strategies of searching in problem space can provide the cognitive evidence for next generation web search engine and web problem solving system.

3.3 Streaming Reasoning and Real Time Reasoning

Data streams are unbounded sequences of time-varying data elements; they occur in a variety of modern applications, such as network monitoring, traffic engineering, sensor networks, RFID tags applications, telecom call records, financial applications, Web logs, click-streams, etc.

Processing of data streams has been largely investigated in the last decade [53], a continuous query language (named CQL [12]) was defined, specialized Data Stream Management Systems (DSMSs) have been developed, and features of DSMSs are becoming supported by major database products, such as Oracle and DB2.

The combination of reasoning techniques with data streams gives rise to **Stream Reasoning**, an unexplored, yet high impact, research area. To understand the potential impact of Stream Reasoning, we can consider the emblematic case of Urban Computing [80, 10, 119, 15] (i.e., the application of pervasive computing to urban environments). The very nature of Urban Computing can be explained by means of data streams, representing real objects that are monitored at given locations: cars, trains, crowds, ambulances, parking spaces, and so on. Reasoning about such streams can be very effective in reducing costs: for instance, looking for parking lots in large cities may cost up to 40% of the daily fuel consumption. Problems dramatically increase when big events, involving lots of people, take place; a typical Urban Computing problem is to help citizens willing to participate to such events in finding a parking lot and reaching the event locations in time, while globally limiting the occurrences of traffic congestions.

A new generation of reasoners is clearly needed in order to simultaneously instruct the car GPS of numerous citizens with the fastest route to the most convenient parking lot during exceptional events. Time constraints for such a reasoner are very demanding (i.e., few ms per query) because citizens are continuously making driving decisions and the traffic keeps evolving; therefore, continuous inference is required. In this work, we define Stream Reasoning as a new paradigm, based upon the state of the art in DSMS and reasoning, in order to enable such applications. By coupling reasoners with powerful, reactive, throughput-efficient stream management systems, we expect to enable reasoning in real time, at a throughput and with a reactivity not obtained in previous works.

In order to move the first steps toward Stream Reasoning we have investigated how to combine Stream Databases and Reasoning within the pluggable algorithmic framework of LarKC. In [43], we propose a conceptual architecture for a Stream Reasoner based on the LarKC framework and we envision two alternative ways to realize such architecture with plug-ins for the selection, abstraction, and reasoning steps.

Moreover, we realized that the first research issue to be addressed concerned the specification of a continuous query language for the Semantic Web extending SPARQL with data streams, aggregating operators, and a continuous semantics. In Appendix to LarKC D3.1 [155] we defined C-SPARQL, i.e., its syntax, its semantics, and exemplification of its usage for urban computing.

3.4 Resource Bounded Reasoning and Web Scale Reasoning

We will focus on what can be learned from theories on how the mind does similar reasoning and comprehension tasks to the ones WP4 plug-ins will do. As section 3.1 explained, theories of bounded rationality assume that the human mind routinely faces situations of incomplete and inconsistent knowledge. The key question ”How rational are people, given their limited computational capabilities and their incomplete knowledge?” has spawned a lot of research that is relevant for web scale reasoning in the following areas: Scalability, heterogeneity, dynamics, and inconsistency.

Scalability The web is growing rapidly. The same problem also occurs for ontologies and other semantic web data. The mind routinely copes with potential information overload. Mental representations must accommodate a constant stream of information coming from the senses. Note that the issue of vast amounts of potentially relevant and time-sensitive information about environmental contingencies is analogous to the LarKC Urban Computing use case (WP6). So the question arises: if the mind has a constant stream of information as input (filtered only by our attention capacity) and storage is virtually unlimited, how does it deal with scalability problems?

The work of MPI on simple heuristics is directly related to how the human mind deals with problems in which there are large amounts of knowledge. There are basically two answers: First one can assume that the mind does access all knowledge available to make an inference. Long-term memory is essentially limitless; no known boundaries have been calculated. This contrasts with the limited capacity of working memory, estimated to be 7+-2 chunks. However, it is clear that humans do not always bring all their relevant background knowledge (and all logical entailments of their beliefs) to bear when facing a decision or stopping to think and starting to act.

An alternative answer is that the mind uses fast and frugal heuristics to make adaptive decisions that depend on minimal information. As Gigerenzer puts it: ”simple heuristics such as take-the-best can often predict more accurately than neural networks and other complex linear and non-linear regression techniques. Thus, a mind that would engage in these complex computations, as it is still commonly assumed, would actually be slower, need more information, and would do worse” [56](p. 7). In this tradition, the solution to scalability problems is to avoid them altogether, by always using a small subset of all the available information. LarKC could implement this approach in the form of selection plug-ins (WP2).

A second way the mind deals with large amounts of information is forgetting. If all information in an essentially limitless mind gets activated at the same time, there could be a lot of noise. The mechanisms of forgetting could help heuristic inference [135]. Forgetting makes relevant information easier to access since the signal to noise ratio increases when dropping non-relevant items. The LarKc project will implement forgetting in its ACT-R activation based plug-ins described and motivated in D2.1.2 (Selection and Retrieval: D2.1.2 Apparatus). For example, a decaying amount of activation associated to each RDF triple could make rarely-used triples idle at a low activation level. Using some triples in a certain context would make them more active, and thus more available, for future occasions. The statistics of file access on one www repository suggest that activation methods that forget should help LarKC hone in on relevant information [116, 115].

The main argument for computing a relevant subset of triples before going to the inference mechanisms is scalability. Considering the size of knowledge bases available, it would be hard to operate on the entire repository of data. But a second reason is that functional forgetting, i.e., the gradual decay of more remote and less frequently needed information-may help produce better results overall. Thus, a reasoning mechanism that operates upon the entire knowledge base would not necessarily do better than the same reasoner under a restricted dataset. Of course, these predictions need empirical testing.

Heterogeneity Web data originate from very different data resources. They are represented in different data formats and can be very different from one another. Dealing with web data in various data formats or integrating various reasoning approaches is an important issue in web-scale reasoning. Psychologists have proposed that there are many representational formats for different kinds of information. In no particular order: rules [6, 108], images[14], schemas [133], semantic networks [132], neural networks[125], and combinations thereof [126]. The equivalent in the web world would be heterogeneous data formats. But it is not only data formats and representations that are heterogeneous. The question, "Do humans use two modes of [cognition, representation, processing, learning, etc] or just one?" is central to cognitive science and has been discussed by numerous theorists (e.g.,[65, 145, 139, 117]). In the domain of expert chess, the question has been formulated as How much of expert problem-solving behavior is explained by real-time search through the task problem space and how much is explained by pattern recognition? ([59], p. 291). Evans and Over ([51], chapter 7) proposed that almost all reasoning tasks show evidence of a logical and non-logical component of performance. They argued that the two modes of performing are interactive and complementary rather than competing and mutually exclusive.

Stanovich [147] reviews a collection of theories in which theorists have proposed two different systems of reasoning. He grouped them as system1 and system2. System1 is characterized as automatic, unconscious, and relatively undemanding of computational capacity. System2 is controlled, reflected in IQ tests, rule-based, and involved in analytical cognition. The tasks that pertain to system1 are highly contextualized, whereas system2 tends to decontextualize (abstract) problems. The two systems can produce contrary solutions to the same situation. Stanovich [147] seems to be very concerned with the evolutionary interpretation of the two systems and its relevance to set the arguments about rational behavior. Sloman [145] tagged the two components "associative system" and "rule-based system." He used similarity-based thought and temporal similarity relations to draw inferences and make predictions that resemble those of a sophisticated statistician: "Rather than trying to reason on the basis of an underlying causal or mechanical structure, it constructs estimates based on underlying statistic structure. Lacking highly predictive causal models, this is the preferred mode for many forecasters, such as weather and economic ones ([145], p. 4). In airline pilots, for example, the underlying causal or mechanical structure can be present, but, after extensive experience, it can be easier for them to operate using the statistical structure that they have extracted during practice.

In this view, not only do human use heterogeneous representation systems, but they also use different reasoning systems to accommodate different situations. Thus, cognitive science has to deal with integration of heterogeneous proposed data formats. Most theorists bypass the problem of integration due to the ubiquitous piecemeal approach to cognition (one phenomenon, one theory). Nonetheless, there are two groups of theories that face and confront the problem of integration: (1) Comprehensive theories that try to explain cognition as a whole (in the form of cognitive architectures). (2) Theories that address an area of cognition in which multiple types of representation interact (e.g., the argument between propositional vs. spatial representations of visual objects, see [5], or the rules vs. exemplars argument [65]).

RDF and RDF schema constitute big steps towards interoperability. Data format differences in the RDF family are not that great compared to the differences in cognitive representations since it seems that in the case of the semantic web they are always structured representations. In this sense, LarKC is in a better position than most cognitive scientists to solve heterogeneity problems.

Dynamics Web data are not static. They are always changing with various application scenarios and context. We will survey how those relevant approaches may be introduced or similar approaches may be developed to deal with dynamic web data. Google updates their algorithm every few days, producing changes in the ranking of web pages (what is popularly called Google slap). In contrast, cognitive scientists test algorithms on corpora that are static. For example, all testing corpora and datasets from TREC tracks are indeed static[1]. How does this affect our theories? And this all comes from traditional Information retrieval paradigms (i.e., based on keyword matching and variations of the vector space models). Updates are a bit more difficult on structured representations such as the one the semantic web advances. For illustrative purposes, imagine that at some point, most of the web is described in some structured format. Imagine that the Ontology Alignment Evaluation Initiative [49] succeeds massively and there is a single ontology that covers all knowledge areas. That is, somebody solves the entire ontology alignment problem. This is still not the end of ontology work: of course, this ontology must be updated every time that the web grows, i.e., when somebody creates a new page. Ontologies, just like any structure holding knowledge, need to be updated for several reasons, including a change in the world being modeled, a change in users' needs, and the discovery of knowledge previously unknown. This need for perpetual updates introduces further issues concerning ontology versioning. Imagine that some reasoning plug-in or agent produces a deduction path that is sound and complete on one version of an ontology, but this path is rendered invalid when the ontology is updated.

When working with structured representations, there are two situations when adding knowledge. (1) new knowledge may fit perfectly in the existing structure, i.e., it could add a leaf to the tree. (2) new knowledge may be disruptive, i.e., necessitate new branches or a different kind of tree. Accommodating new facts may require abandoning the existing structure for a new one (example: Galileos solar system). In large ontologies, this is statistically unlikely, as large ontologies rarely require traumatic changes [32]. Looking for a contribution from cognitive psychology on how the mind deals with dynamics we found that ontology work is not very popular because ontologies are hand-built, not learnt. A theory using a hand-built ontology must explain how the mind acquires the ontology in the first place. The only viable alternative is to propose that we are born with that ontology. This is proposed for things like language learning [34].

Chomsky believed that there is not enough information in the environment to learn the knowledge needed for making syntactic judgments (i.e., is this sentence

[1] Even though in some occasions there are successive versions of a particular test corpus or knowledge base, this is the exception to the rule.

	Text	Inference
(a)	Jack missed his class because he went to play golf. He told his teacher he was sick.	Jack lied.
(b)	John bought a ticket, went to the airport in Denver, got on a plane, ordered some drinks and dozed off, and finally got off the plane in Chicago.	Jack flew to Chicago.
(c)	Superconductivity is the disappearance of resistance to the flow of electric current. **Until now it has been obtained only by cooling certain materials to low temperatures near to zero.** That made its technical application very difficult. Many laboratories now are trying to produce superconducting alloys. Many materials with immediate technical applicability have recently been discovered. Until now, **superconductivity has been achieved by considerably increasing the temperature of certain materials** [112]	Flag as conflicting information.

Figure 3.1: Comprehension examples from [81]

grammatically correct or not?). This is the classical poverty of the stimulus argument. So in this sense, the input from psychology to this discussion is slim. There are of course treatments on how mental representations are dynamic and how this should be considered in the construction of theories (see for example [44]), but at this point they are hard to integrate with the current problem.

Inconsistency Web data may consist of a substantial amount of inconsistent, incoherent, and noisy data. Dealing with inconsistency becomes an important issue for web scale reasoning. The mind deals with inconsistencies on an everyday basis. Reading, a very common human activity, implies comprehension, which is an extremely complex process that deals with incomplete, noisy and incoherent streams of data that must be interpreted in real time and connected with all existing relevant knowledge. That is, a human who is reading must perform similar tasks to those LarKC may face in certain reasoning problems. A common inference task while reading is dealing with polysemy. Many words have different meanings, and an agent has to infer the adequate meaning from the context. Thus meaning is not static, but dynamic. But resolving polysemy is just one example. Other examples of reasoning while reading are described in Figure 3.1.

The proposition inferred in (a) and (b) was not in the text, but must be created for successful comprehension of the meaning of the text. Thus, this could well be the objective of some reasoning plugins. The text in (c) is inconsistent, and should be marked as such. While this could be trivial for a machine, a large proportion of participants did not notice the contradictory claims. Note that the comprehension literature does not approach inconsistencies as a logical problem. Instead, it is mostly a soft constraint resolution approach based on spreading activation. The dominant model is the construction-integration model (CI model, [81]). This model has been proposed as a hybrid cognitive architecture [158] and is compatible with other architectures such as ACT-R [8]. According to the evidence, humans are

Classification	Method	Scalability	Heterogeneity	Dynamics	Inconsistency	Parallelism
Bounded Rationality	Simon	yes	maybe	maybe	maybe	?
Heuristic Computation	Gigerenzer	maybe	yes	maybe	maybe	?
Heuristic Computation	Dawes and Corrigan	no	yes	maybe	maybe	?
Heuristic reasoning	Langley	no	yes	maybe	maybe	?
Heuristic reasoning	SOAR	maybe	yes	maybe	maybe	?
ACT-R	Anderson	yes	yes	yes	yes	yes
Stream Reasoning	Emanuele et al.	yes	no	yes	maybe	yes

Figure 3.2: Resource Bounded Reasoning and Web Scale Reasoning

well-equipped to deal with inconsistencies when in fact, it has been known for quite a while that humans perform poorly in highly formal tasks, such as decide what information is necessary in order to test the truth of an abstract logical-reasoning problem (e. g., [157]). So for humans, inconsistencies are not only not a problem, but a staple in their daily lives. However, we will not review these studies here. The interested reader can see [111] and [101].

The LarKC project faces noisy data, as data in real environments are rarely clean. In fact, sensory systems evolved to not only use noisy data but to take advantage of them. For example, human hearing is enhanced by noise [160], and it is likely this phenomenon holds true for higher levels of cognition. For example, information compression seems to be a general way for the mind to achieve generalization, and noise is an important prerequisite for generalization. Current models of semantics in psychology all use some form of compression (e.g.,[78, 85, 86, 60]). This reduces spurious correlations and enhances co-occurrence detection. Looking at the mechanisms proposed in those models could be useful to tackle noise in web data; we will look into how compression helps generalization in future experiments.

From the point of view of a fast and frugal agent using small samples instead of using all the information available may help producing easier generalizations, since the probability of sampling two conflicting propositions is lower the smaller the sample. For example, if an agent reading text (c) in the table stopped sampling before the last sentence, the contradictory information would not be taken into account. This has obvious disadvantages too (maybe the last sentence was the most important) but of course reading is just one of the many circumstances in which fast and frugal heuristics have not been particularly studied, so we have no empirical evidence

Parallelism It is still an open question how those approaches can be measured with respect to the parallelism.

The table 3.2 is a summary of various resource bounded reasoning methods and their relation with the requirement of web scale reasoning.

4 Rule-based Reasoning for dynamic and incomplete knowledge

In this chapter we will focus on several main topics. First we intend to give a short overview of nonmonotonic logics in general and the motivation underlying them as well as their main properties and the resulting close relationships between various such logics. Then we will examine the relevance of Logic Programming (LP) systems in the scope of nonmonotonic and rule-based reasoning in Section 4.2. Based on the previous chapters we then show applications of rule-based reasoning on the Web. Section **??** describes how humans reason according to rules and draws parallels to logical frameworks in order to show how rule-based reasoning can reflect "common sense". Finally Section 4.3 concerns the applicability of rule-based reasoning in the scope of Web scale reasoning and its distinct challenges. The relevance of nonmonotinic reasoning and Logic Programming for LarKC is actually twofold.

Nonmonotonic inference in a very broad sense allows us to derive useful conclusions (based on common sense assumptions) on the basis of incomplete information – a feature that can be deemed as highly attractive in a Web setting and also i.e. for future LarKC plug-ins. This kind of reasoning often requires to make assumptions which are not bound to be correct – and maybe retract them based on new evidence. In this sense nonmonotonic reasoning bears similarities with belief revision [129], which adds a metalevel for extending or revising the axioms, and as such can be the basis for approximation methods. For example negation-as-failure can be regarded as a belief revision kind of rule, which effectively adds a new axiom. Thus nonmonotonic reasoning opens up new possibilities, i.e. for evaluating the relevance of certain axioms, etc. during the reasoning process.

Secondly, Logic Programming as basis, and rule engines are relevant to LarKC, because it is possible to employ them to reason with several tractable fragments of standard formalisms, i.e. OWL 2 RL, ELP [83], etc. This allows to reuse research results, i.e. from the deductive database area for reasoning tasks in a Web environment. Moreover standardization efforts within the RIF working group[1] draw heavily and are directly grounded in existing research on Logic Programming.

4.1 Nonmonotonic Reasoning

Classical logic has played an important role in the development of computer science and logicians have argued that classical logic can also be used to model human thinking. However, classical logic has limitations in this regard:

1. Consequents cannot be revised on the basis of new information.
2. New information can only lead to new consequences.

These two statements are another way of saying that the logical consequences of any theory cannot be amended; only incrementally added to. Hence, such a logical formalism is described monotonic. The classical problem to illustrate that this is the conclusion that a certain individual can fly, based on the assertions i) that birds

[1] http://www.w3.org/2005/rules/wiki/RIF_Working_Group

can fly and ii) that the individual is a bird. Adding the further information that the specific individual is a penguin either leads to a logical inconsistency (if the inability of penguins to fly is also asserted) or the counter-intuitive conclusion of a flying penguin. An obvious, albeit cumbersome and highly impractical, solution is to extend the assertion that "all birds fly" by listing all the exceptions explicitly, i.e. "all birds that are not penguins and not ostriches and . . . fly". Aside from that, nothing we can add to our knowledge will enable us to change existing conclusions, given that we hold to our premises. Thus, classical deductive reasoning within a theory is "local" in the sense that, having arrived at a conclusion from premises, we need not worry about any other sentences in the theory.

This notion of monotonicity (a fact entailed by a theory is also entailed by a larger theory) can be formally expressed in the following way: In classical logic the meaning of a formula (a set of formulae) S is the set of interpretations (depending on the particular type of logic, i.e. truth assignments, a first-order interpretation, a Kripke-interpretation / world pair for modal logics, . . .) that satisfy it, or its set of models. However, all models of $S \cup \{\psi\}$ are also models of S or more precisely:

$$\text{If } S \vdash \varphi \text{ then } S \cup \{\psi\} \vdash \varphi$$

Minsky [100] explains the two problems above in terms of the permissiveness of axiomatic knowledge, i.e. that axioms permit new inferences, but never add information about which conclusions should be withdrawn. Further, even if axiomatic knowledge is added to indicate the relevancy of conclusions, then an equal amount of information about their irrelevancy is also inferred. At the core these limitations have two consequences according to Minsky: Namely that "logical" reasoning would not be flexible enough to reflect human common sense and that it could not handle inconsistent data.

The development of nonmonotonic reasoning has its origins in the field of Artificial Intelligence (AI), where for the above reasons, classical logic is not considered a suitable basis for "thinking". The assumption in this case is that commonsense reasoning is different to the behavior of classical logic outlined above because a large part of it is carried out based on certain expectations of the world working in a certain predictable way, as this is often the best that can be done due to a lack of complete information. In that sense nonmonotonic logic as well allows to jump to conclusions quicker than traditional logic but on the other hand retract them again if counter evidence is found. Further motivations for the development of nonmonotonic logic can be found in [25].

Nonmonotonic reasoning as a form of reasoning that extends first-order logic was first introduced in [128]. It contrasts with classical deductive reasoning in the following way. Instead of considering a set of interpretations that satisfy a set of formulae (models) it focuses on a subset of preferable models (often called minimal models). Hence, $S \vdash \varphi$ because φ is true in all preferred models of S. However $S \cup \{\psi\}$ might have different preferred models than S and thus $S \cup \{\psi\} \vdash \varphi$ does *not* necessarily hold anymore.

Following, we will subsequently present the most well known examples of nonmonotonic logics.

4.1.1 Default Logic

Default reasoning, first proposed by Reiter in [120], tries to capture the most common expected outcome of situations and is a classical example of nonmonotonic inference. It applies inference rules that admit exceptions – inference rules that only apply under certain, usually "normal" circumstances. A very basic example of this is the Closed-World Assumption (CWA), used to derive negative information in the absence of positive evidence.

Default logic is expressed in terms of a default theory, which is a pair $\langle D, W\rangle$, where W is a set of first-order logical formulae, called the background theory and denoting the facts that are known to be certainly true. D is a set of *default rules*, each expressed as following:

$$\frac{\varphi : \psi_1, \ldots, \psi_n}{\gamma}$$

where φ, ψ and γ denote (first-order) formulae. In contrast to classical inference rules, such default rules require *consistency conditions* or *justifications* $(\psi_1, \ldots, \psi_n)$ to hold in addition to the *prerequisite* φ of the default rule, in order to admit the derivation of some formula γ. Such a default can be read as: if φ holds and for all $\psi_1 \ldots \psi_n$, $\neg\psi_i$ does not hold (non of the consistency conditions is violated), then derive γ. Defaults that contain formulae with free variables (*open defaults*) are usually interpreted as schemata defining all of their closed instances. The simplest approach in this regard is to therefore see open defaults as a convenient notational variant and a placeholder for all these closed instances. More complicated approaches towards open defaults are possible and investigated in detail in [11], [79] and [88].

Semantically, default logic is build on so-called *extensions* which describe different belief sets resulting from the known information. The notion of an extension E of a default theory (D, W) can be defined in the following way:

1. $W \subseteq E$. E is a deductively-closed super-set of W, as W contains knowledge that we are certain to be true.

2. E is closed under all defaults from D. That means that all the defaults have been applied where possible, in the sense that if there is some prerequisite $\varphi \in E$ in E and non of its consistency conditions are in conflict with other knowledge in E, then the consequence γ must also be in E.

3. Every formulae in E has a derivation from known to be true facts in W and a set of defaults. In other words extensions need to be grounded in (D, W): If $\varphi \in E$ then there is a default proof from (D, W) for φ in E.

At this point it becomes obvious that multiple different extensions are possible (particularly due to point two above), depending on the chosen order in which default rules are applied.

Various precise definitions of an extension are possible. The classical definition due to Reiter defines extensions as fixed points of an operator Γ, which has the advantage of making the above mentioned notion of grounding explicitly visible. Let (D, W) be a default theory and S a set of formulae. Then $\Gamma(S)$ is the smallest set that fulfills the following properties:

1. $W \subseteq \Gamma(S)$

2. $Cn(\Gamma(S)) = \Gamma(S)$
3. If $\frac{\varphi:\psi_1,\ldots,\psi_n}{\gamma} \in D, \varphi \in \Gamma(S), \neg\psi_i \notin S (1 \leq i \leq n)$, then $\gamma \in \Gamma(S)$

Entailment of a default theory can be defined in various ways. *Skeptical entailment* requires a formula to be contained in *all* extensions of a default theory. *Credulous entailment* means that a formula is entailed by a default theory if it is contained in *at least one* extension of a default theory.

The well known Closed-World Assumption (CWA), first set down by Reiter in [121] as well, can be regarded as a special, restricted case of default logic, called *normal default theories.* In normal default theories defaults are generally restricted to the following form:

$$\frac{\varphi : \psi}{\psi}$$

Normal default theories are special in two ways: First of all they are guaranteed to have extensions and secondly extensions are simply different enumerations of the defaults. The CWA states that if the truth of an atomic formula cannot be proved then it can be assumed to be false. This idea can be reformulated in the definition of "negation as failure" , i.e. in the absence of any knowledge to the contrary, assumes that any atomic formula is, by default, false. In this sense the CWA can serve as a basic example of a nonmonotonic logic by applying a strong nonmonotonic closure operation.

$$\text{CWA}\,(W) = Cn\,(W \cup \{\neg p : W \nvdash p\})$$

where W denotes a set of ground formulae and p denotes a ground predicate. Practically this means that if it is not possible to infer a certain ground predicate, its negation will be added to the computed closure. If p was added to W at a later stage, then obviously the inference drawn according to the CWA would need to be revisited again. As a default theory this can be easily expressed as

$$\frac{: \neg p}{\neg p}$$

for every fact p.

4.1.2 Circumscription

Circumscription is a nonmonotonic logic originally presented in [98], with the aim to formalize the common sense assumption of certain default values unless otherwise stated. In the original first-order formulation, circumscription defines the closure of a theory by restricting its models to a set of *preferred models*, those that have minimal extensions of certain predicates.

In this sense circumscription gives up the classical point of view that all models have to be regarded as equal possibilities, based on the underlying idea that it would be desirable to stick to normal and expected situations and default values as much as possible – that means to minimize the extension of "abnormal" predicates, which in turn introduces a preference relation on the set of all models.

A formula φ is in turn *preferentially entailed* by a set of other formulae S if and only if φ is true in all *maximally preferred* models of S. As mentioned preferred models in this regard are models which the minimal occurrence of a certain predicate / by forcing a certain predicate to be false if possible. This notion of entailment is realized as a set of syntactic transformations on the initial set of formulae S that "strengthen" it to arrive at a new version S^*. Formulae preferentially entailed by S are then exactly those that are entailed classically by S^*. As this syntactic restrictions required to express the minimality of a predicate (or more precisely its extension) usually require to quantify over this particular predicate a second-order sequence is required.

More particularly given a formula φ and a predicate P, circumscription of P in φ in its simplest form can be achieved by the following second-order formula:

$$\varphi(P) \wedge \neg\exists p(\varphi(p) \wedge p < P)$$

In this formula p is a predicate variable of the same arity as P. This is a second-order formula because it contains a quantification over a predicate. Actually, $p < P$ is shorthand for:

$$\forall x(p(x) \rightarrow P(x)) \wedge \neg\forall x(P(x) \rightarrow p(x))$$

In this formula, x is a n-tuple of terms, where n is the arity of P. This formula accomplished the above mentioned extension minimization. No other predicate p can assign to false every tuple that P assigns to false and be different from P at the same time, in order for a specific model to be considered as a preferred model.

Several different variants of circumscription exist but in general they can all be regarded as model preference theories that place a preference relation on models. A comprehensive overview is available in [89]. More particularly a version that allows to minimize for several predicates in parallel is presented in[90] and [97].

4.1.3 Autoepistemic Logic

Modal logics [33] are extensively studied as a basis for formalizing nonmonotonic reasoning. The first proposal in this direction was made in [45]and extended and formalized as autoepistemic logic in [102].

In these attempts nonmonotonic reasoning is expressed in terms of an ideally rational agent that employs a certain degree of *introspection* to reason about its own beliefs. For that, a modal language $L_\Box$ is defined which extends a classical first-order language with a new modal operator $\Box$. The basic meaning of attaching that operator to a formula ϕ is to express the notion that the agent "believes" ϕ. Nonmonotonic behavior can be expressed by the *negative introspection* the agent is capable of and is expressed as:

If ϕ is not believed, then $\neg\Box\phi$ is.

Furthermore, *positive introspection* is expressed as:

If ϕ is believed, then $\Box\phi$ is.

Taken together these two rules express the perfect and full introspection capabilities of an ideal agent, as belief is also defined in terms of what is actually *not* believed. Classically, [102] more precisely defines belief in that regard as *potential belief sets* of this agent given a certain set f premises. These potential belief sets are in turn called stable expansions, and (due to the circular nature of the two rules above) expressed as a fixed point equitation as following (note that slight variations exist, i.e. see [45]). Let T be a a theory in a modal Language $L_\Box$. A set of formulae S is a stable expansion of T if and only if

$$S = \{\varphi | T \cup \ \{\Box\varphi | \varphi \in S\} \cup \{\neg\Box\varphi | \varphi \in L_\Box \backslash S\} \vdash \varphi\}$$

In that formula φ denotes the classical consequence relation, $\{\Box\varphi|\varphi \in S\}$ expresses the agent's positive introspection capabilities and $\{\neg\Box\varphi|\varphi \in L_\Box\backslash S\}$ its negative introspection capabilities. In turn a stable expansion S is the logical closure of (1) an agent's premises and (2) the results of positive and negative introspection.

Variants of autoepistemic logics turned out to be useful to define the semantics of logic programming with negation as failure in form of the stable model semantics (see [118], [24]), which can be regarded as a simple and restricted form of autoepistemic logic. Additionally default logic can be seen as a weaker form of autoepistemic logic, with the main difference being groundedness. [152] outlines a basic strategy for embedding default rules into autoepistemic formulae and shows that in many cases the notions of extensions (in the sense of default logic) and expansions (in the sense of autoepistemic logic coincided).

4.2 Logic Programming

4.2.1 Semantics of Logic Programs

Logic Programming [92] is interesting since it supports nonmonotonic inference in a very natural way. Its most basic variant is based on *definite* logic programs and on Horn Logic [71], where logical statements (clauses) in propositional or first order logic are formed as a disjunction of literals with at most one positive literal. Thus, a basic logic program is defined by a finite number of definite Horn clauses such as:

$$\neg B_1 \vee \cdots \vee \neg B_m \vee H$$

A Horn clause with exactly one positive atoms is also called a definite clause. Horn Clauses are interesting for two reasons:

1. They permit efficient automated reasoning, because the resolution [122] of two Horn Clauses is another Horn Clause.

2. A Horn Clause can be re-written to be naturally interpreted as a rule of the form if (something) then (something else).

Such a rule can naturally be written as following:

$$H \leftarrow B_1, \cdots, B_m$$

where $H, B_1, \cdots, B_m$ are positive atoms. H is called the *head* of a rule, while $B_1, \cdots, B_m$ are called the *body*. Such a rule can be understood as: to make the statement true when $B_1, \cdots, B_m$ true, H must be also true. In other words, if $B_1, \cdots, B_m$ then H. The implication symbol is often also replaced with $: -$, i.e.: $H : -B_1, \cdots, B_m$. This allows us to state knowledge (1) as known facts by omitting the body of a rule, (2) formulate rules that tell us how to derive new knowledge. Furthermore it is possible to compose queries such as:

$$? \leftarrow B_1, \cdots, B_n$$

where $B_1, \cdots, B_n$ denote positive atoms. Thus queries are simply Horn clauses with no positive literals.

$$\neg B_1 \vee \cdots \vee \neg B_m$$

Already definite programs give rise to nonmonotonic semantics, full non-monotonicity arises when definite programs are extended with negation and/or disjunction. The motivation to include negation in logic problems is roughly the same as in non-monotonic logics in general, that is to express "default" statements that hold unless special conditions hold, i.e.:

$$H \leftarrow B, \text{ not } C$$

where C denotes some unexpected condition. Thus definite logic programs are extended to *normal logic programs* by introducing negation:

$$H \leftarrow B_1, \cdots, B_m, \text{ not } C_1, \cdots, \text{ not } C_n,$$

The two main approaches to defining the nonmonotonic semantics of logic programs with negation (and disjunction) are the *well-founded semantics* and *stable semantics* (and their extensions to accommodate disjunction).

The well-founded semantics (WFS), introduced in [154], associates with every model a unique 3-valued model. In that sense the well-founded semantics does not always decide all atoms – it can occur that the body of a rule is left as *undefined*. Practically, the WFS can be described as the weakest form of semantics that satisfy four basic principles, that each characterize a basic transformation which can also be used to compute the WFS:

1. Elimination of tautologies
2. Reduction
3. Subsumption
4. Generalized Partial Evaluation

More details on these four principles, including a precise definition can be found in [23] and [22].

The stable model semantics in addition introduce the principle of *elimination of contradictions* and thus defines a (classical) 2-valued model for a program (see [21], [54]). The stable model semantics assigns to each program P a set of *stable models*, that is, it computes different sets of models in which each atom has a clear reason to have a specific truth value assigned. As such, it is possible to interpret the stable model model semantics in a similar way as stable extensions in autoepistemic logic. This approach generally forms the basis for *answer set programming.*

A further important observation is that the stable model semantics generally extend the well-founded semantics, that is $\varphi \in WFS(P)$ implies $M \models \varphi$, where M is some stable model. Furthermore, it can be shown that if the well-founded model of a program is two-valued, then it coincides with the unique stable model. The two main differences between the approaches are that (1) the WFS are always consistent, while the stable semantics can derive inconsistencies, (2) that the stable model semantics do not allow a goal-driven, top-down evaluation of a program.

4.2.2 Practical Applications

Prolog [35] is an example of an early programming language based on logic (logic programming [92]) that introduced a nonmonotonic component. The "not" operator allowed for the creation of rules that operated on data that does not explicitly appear in the program. The combination of Horn logic and negation-as-failure allows Prolog to use negation in its rule definitions, e.g.

$$fly(X) : -bird(X), \neg penguin(X)$$

In other words, if X is a bird and we do not know if it is a penguin, then X can fly.

Another well studied Logic programming formalism is Datalog [153], which is a restricted syntactic subset of Prolog. In its original form (which has meanwhile bean extended in various directions) Datalog disallows function symbols, imposes stratification restrictions on negations and recursion and only allows range restricted variables. Datalog's restrictions result in a decidable fragment of Prolog which guarantees query termination. Furthermore Datalog was originally developed as a database query and rule language and a wealth of results and optimizations from the area of deductive database research are available. Combined with its tractable complexity Datalog based systems are thus able to perform query answering efficiently even for large instance sets.

A fundamentally more expressive (and more complex) extension of the Logic Programming paradigm is Disjunctive Logic Programming, which also allows disjunctions in the head of a rule. Based on this Disjunctive Datalog [153] can be derived and applied for Description Logic reasoning [75]. More particularly this approaches covers a large subset of OWL DL and shows very favorable performance results in regard to large instance sets.

4.3 Rule-based Reasoning and Web Scale Reasoning

Heterogeneity Rules on the Web, and thus also rule based reasoning, are central points in the Semantic Web architecture stack, sitting just beside ontologies. Complementary to ontologies, which describe multiple objects in a machine understandable manner, rules in this sense can be seen as means to steer the inference process and ways to combine existing information in order to derive new knowledge. Consequently, a lot of work has been done to fill this place in the stack and on combining Description Logics and rules in a fruitful manner, like SWRL [48], Description Logic Programming (DLP) [64], Al-log [47], and others. Furthermore the Rule Interchange Format (RIF) is a W3C effort towards a rule interchange format for various rule-based languages. Thus rule-based efforts are able to deal with a lot of different formalisms.

Some Web information sources are only able to be represented as cases, and therefore case-base reasoning needs to be used with rule-based reasoning for problem solving. In the granular reasoning platform, rule-based reasoning can work on the knowledge level and case-based reasoning can work on the information level based on the granular structure. In different situations, different methods can be employed or combined for different queries.

Scalability The complexity of nonmonotonic reasoning is generally higher than in equivalent cases in classical logic. This is intuitively clear as there are two separate sources involved in nonmonotonic reasoning: (1) The classical formalism on which nonmonotonic constructs are added, and (2) the complexity of the nonmonotonic extensions themselves. Even when ignoring the classical components nonmonotonic reasoning is usually still NP-complete. Furthermore, both Default Logic as well as Autoepistemic Logic are undecidable (see [120], [82]) and circumscription is uncomputable (see [134]).

In order to achieve subclasses of nonmonotonic logics that can be handled practically there are two possible approaches. First, as usual in classical logics, it is possible to syntactically limit the expressiveness of the formulae (i.e. to Horn clauses). Furthermore it is also necessary to restrict the nonmonotonic constructs in the formalism, which is also possible by syntactic restrictions, for example by stratification in logic programs [110], or by switching to a weaker semantics (i.e. the well-founded semantics, which have only polynomial complexity). In general Logic Programming is interesting in this regard, since it focuses on practically implementable and yet expressive fragments of nonmonotonic reasoning.

In terms of Web standards, SWRL is roughly based on combining OWL DL with the Datalog dialects of the RuleML language. Thus Horn-like rules can be used in conjunction with existing OWL knowledge bases. This approach maintains the full power of OWL DL based on an extension of the model-theoretic semantics of OWL (and SWRL also is based on the open world assumption). In fact it could be said that every SWRL rule is an OWL axiom, albeit in a more generalized and less restricted form (thus SWRL also supports monotonic inference only). This expressivity however leads to an undecidable (in fact semi-decidable) formalism for key inference tasks and thus practical implementations rarely support the full specification. Decidability can be regained by placing restrictions on the SWRL rules, resulting in DL-safe rules [104]. These restrictions mandate that each variable in rules can only bind to explicitly named individuals or more precisely that variables that occur in a rule also need to occur in a non-DL-atom in the rule body.

In contrast DLP (Description Logic Programs) focus on the intersection of Horn-logic and OWL. This implies particular limitations regarding its expressivity, more precisely concerning the use existentials in the rule head and of universals in the rule body, which as a consequence also concerns minimum and maximum cardinality restrictions.

Furthermore an interesting notion is that tractable language fragments, that are either rule based or can be dealt with by rule engines, are explicitly defined as profiles in the current OWL 2 working draft : DL-Lite, OWL-R DL, OWL-R Full. All of these sublanguages have very promising complexity results and are therefore highly relevant for reasoning with instance sets at Web scale.

Classification	Method	Scalability	Heterogeneity	Dynamics	Inconsistency	Parallelism
Logic Programming	Prolog	Yes	Yes	No	Yes	Yes
Logic Programming	Datalog	Yes	Yes	No	Yes	Yes
Nonmonotonic Logic	Circumscription	No	No	NA	Partly	NA
Nonmonotonic Logic	Default Logic	No	No	NA	Partly	NA
Nonmonotonic Logic	Autoepistemic Logic	No	No	NA	Partly	NA

Figure 4.1: Rule-based Reasoning and Web Scale Reasoning

Dynamics It is unclear how changing knowledge bases affect individual formalisms. For some cases results are available, e.g. it is known that Datalog only retains its favorable complexity results for static knowledge bases [40]. However, in many cases no uniform results are available.

Inconsistency While nonmonotonic reasoning cannot prevent inconsistency in the classical parts of the formalism that is rooted i.e. in first-order logic, it can naturally deal with incomplete information – nonmonotonicity is used naturally to jump to conclusions based on the lack of complete information and withdraw it later if required. Furthermore there is active research , for example in the area of paraconsistent logic programming [19] that aims to resolve inconsistencies in logic programs.

Parallelism Logic Programming lends itself to parallelizations in various ways, depending on the particular evaluation technique, a practical outline is available for example in [52]. Rule-based reasoning is also a method in granular reasoning, but it is not enough for Web scale reasoning. The core of granular reasoning is that it connects the data processing and reasoning process together and performing reasoning on multi-level granularity based on the organization of data sources. The aim of this design is inspired from human intelligence for reasoning on the Web scale.

In this stage, we will consider the method to combine rule-based reasoning and case-based reasoning together as the data on the Web is heterogeneous, dynamic and inconsist.

The table 4.1 is a summary of various rule-based reasoning methods and their relations with the requirement imposed by web scale reasoning.

5 Contextual and Modular Reasoning

5.1 Contextual Reasoning

Because reasoning based on mathematical logic produces inference by manipulating discrete symbols according to a very small set of rules, any ambiguity in the meaning of the symbols, or imprecision in specifying how the symbols may be applied, will almost inevitably lead to incorrect inferences. The effects of this characteristic of logic are only exacerbated in large (let alone web-scale) knowledge bases, in which the number of ways in which assertions may be incorrectly combined increases exponentially with the size of the KB. One method for controlling this exponential aspect is to place limits on which elements of KB content may be combined, by placing knowledge within a context, and enforcing or preferring combinations only within that context. Because of this, reasoning with context has long been considered one of the most important issues in AI [96, 17, 93]. Recently, other approaches to limiting, or even taking advantage of, the effects of representational imprecision have been receiving more research attention, perhaps most notably evidenced in the growth of work on probabilistic inference. These approaches will be explored elsewhere in the LarKC project, but even in with probabilistic reasoning, context can help to reduce computational and representational complexity.

Perhaps the most thoroughgoing application of context representations has been within the Cyc system [1] based work by R.V.Guha [127] and extended in [87] In the following, we provide an overview of contextual reasoning in Cyc, as an example of large-scale context reasoning that may be extensible to web-scale context reasoning in LarKC.

5.1.1 Context in Cyc

Contexts are first-class object in Cyc, and in CycL, the logical language in which the Cyc KB is represented. A context (or "microtheory" in Cyc) is a set of assertions, representing a particular set of surrounding circumstances, relevant facts, IF-THEN rules, and background assumptions. Cyc's enormous knowledge base of assertions (including its IF-THEN rules) is divided into many different contexts; every fact or rule is in some particular context (in fact, in some set of contexts).

Whenever Cyc is asked a question, or has to do some reasoning, the task is always done in some particular context, which restricts inference to relevant facts and rules. For example, if you just want to ask Cyc about brushing your teeth, Cyc doesn't need to know every kind of fact in the Cyc Knowledge Base, such as how much a toothbrush costs. The ManualHumanActivitiesMt context, for typical human activities (like tooth-brushing), doesn't make the cost of the toothbrush available. It ıis available, however in a different context, ShoppingGMt, to which it is highly relevant.

In Cyc, a context is a way of talking about a group of related assertions by representing this grouping explicitly as a Cyc object. Each context should therefore be designed as a coherent set of relevant material.

[1] http://research.cyc.com/

5.1.2 How Contexts are Used in Cyc?

- In general, naming a theory (i.e., a set of sentences) make it possible to make statements about it. For example, it may be useful to state explicitly that for answering a certain kind of question, a certain theory is more relevant: calling the group of assertions dealing with ailments $AilmentMt$, and the set of statements about jobs $JobMt$, allows one to state that the former group is more relevant to answering questions about illnesses than the latter.

- Cyc's "Microtheories" provide a mechanism for focusing on relevant information during problem solving. For example, in answering a query about heart attacks, the system can find all the relevant information in the $AnimalPhysiologyMt$, the $HumanAilmentMt$, and the $AilmentMt$, and group all of this information into a single, temporary, "problem-solving" bundle created for the sole purpose of answering the question. This introduces an important point: contexts in Cyc can be referred to by logical functions, including functions that combine other contexts, produce contexts for particular intervals or regions, or produce contexts that describe the propositional content of a literary or other work, under some mode of interpretation [87].

- Contexts allow for the creation of limited domains that rest on certain assumptions. This allows the statement of a theory to be simplified, since it is possible to enumerate the assumptions once for the entire theory, rather than having to repeat them for every assertion in the domain. For instance, the $JobMt$ as a whole assumes that every employed person mentioned in that microtheory has just one job; having two jobs is an exception. Therefore, it is possible to state knowledge with sufficient precision to avoid unwarrented inference, without conditioning every sentence in that microtheory with a logical sentence to the effect of: "Assuming the person has only one job at a time ...".

- By keeping different theories based on different assumptions distinct, contexts allow the same phenomenon to be represented in different ways without inconsistency. For example, two different microtheories may take two different perspectives on the same event (e.g. viewing an earthquake as a human catastrophe versus a morally-neutral tectonic adjustment), or may make different approximations or allow for different margins of error (e.g., by using Newtonian as opposed to relativistic mechanics). Knowledge entry is made easier since the knowledge enterer doesn't have to worry about creating "the" single correct representation.

5.1.3 Context and Microtheory

There are several kinds of context in Cyc; two common sorts are General Microtheories and Problem Solving Contexts (abbreviated as GMts and PSCs). Problem Solving Contexts are specialized contexts created to solve a particular problem; these are usually created "on the fly" and they evaporate after use. General Microtheories have a more permanent status, and are extremely widely used in Cyc for knowledge representation.

A General Microtheory (GMt) is a generally useful context that is not application- or problem-specific. For example, when talking about shopping for food, one assumes various things, such as that the food items are for sale in a store, that they have prices in money, etc. These assumptions hold throughout our modern culture. Axioms (assertions in Cyc) about shopping that rest on those assumptions are collected in in the microtheory called *RetailBuyingMt*. This general microtheory is useful in a number of applications (like directed marketing, automatic personal assistants, simulation games, etc.).

One can also introduce new Microtheories for fictional situations like those in fairy tales, books, television shows, etc. Various things (like the time, place and culture, and the existence and features of the characters, etc.) are assumed in a *FictionalMt*. A story character like Paddington Bear has certain definite characteristics, for example: he wears a yellow hat. This is true in the *PaddingtonBearMt* (an instance of *FictionalMt*) even though in the real world Paddington Bear never existed.

Similarly, a treaty, agreement or law code has a microtheory that describes a deontic theory – the way things are supposed to be, as opposed to how they actually are. For this purpose, Cyc has a kind of Microtheory called a *SupposedToBeMt* – an *Agreement* is one example that contains the facts needed to represent this situation, namely the facts recited, anticipated and agreed-to in the agreement.

Microtheories are also used to represent information about "information-bearing things" like articles, movies, messages, etc. If we were to represent chapter in Cyc, for example, its information content would be in a microtheory; the assertions of the microtheory would at least partially convey the propositional content of the article.

Most Microtheories, and more generally, most contexts have underlying assumptions about space and time. If one asks, for example, about travelling a long distance, the likely means of travel depends on the historical period that is assumed. Using temporal contexts makes it possible for Cyc to avoid the mistake of assuming that Abraham Lincoln travelled to Washington by airplane or car, but to suppose the possibly of their use in a trip by a modern businessperson.

A Microtheory has a set of relevant assertions (including simple facts and rules of implication) and it may have a set of Domain Assumptions that always apply within the Microtheory. In addition, it has a certain set of other Microtheories to which it has access; if Cyc is in that Microtheory, Cyc can access all of the assertions in the other Microtheories from which it inherits, and it is subject to all of the Domain Assumptions in those other Microtheories. Frequently, the microtheory makes basic assumptions which simplify the axioms within it.

A Microtheory is linked to its other, accessible Microtheories by using an explicit microtheory-relating predicate: *genlMt*. If there are two microtheories ?MT1 and ?MT2, one asserts

(*genlMt* ?MT1 ?MT2)

to say that Microtheory ?MT2 is accessible from Microtheory ?MT1. All assertions and Domain Assumptions in ?MT2 then become available for inference by Cyc if it is operating in ?MT1.

Microtheories also allow two different parts of the Knowledge Base to contain conflicting information without causing logical inconsistency. It would present no problem for Cyc if *TeethCleaning* normally happens one to three times a year according to the *JobMt* (which may be describing professional cleanings by a dentist), but one to three times a day according to the *PersonalActivitiesMt*.

In Cyc, Knowledge Entry usually aims for logical consistency within a Microtheory, but not global consistency throughout the whole Cyc Knowledge Base.

5.2 Modular Reasoning

Modular reasoning is considered to be an approach which is introduced to deal with large scale data, i.e., scalability. In the Semantic Web, ontology partitioning and ontology modularization have been explored and investigated in [148, 150, 151].

In [148], Stuckenschmidtand and Klein propose a framework of ontology modularization. They state the motivation of ontology modularization as follows:

Distributed Systems. In distributed environments like the semantic web, the question for modularization arises naturally. Ontologies in different places are built independent of each other and can be assumed to be highly heterogeneous. Unrestricted referencing to concepts in a remote ontology can therefore lead to serious semantic problems as the domain of interpretation may differ even if concepts appear to be the same on a conceptual level. The introduction of modules with local semantics and clearly specified interfaces can help to overcome this problem.

Large Ontologies. Modularization is not only desirable in distributed environments, it also helps to manage very large ontologies we find for example in medicine or biology. These ontologies that sometimes contain more than a hundred thousand concepts are hard to maintain as changes are not contained locally but can affect large parts of the model. Another argument for modularization in the presence of large ontologies is reuse as in most cases, we are not interested in the complete ontology when building a new system, but only in a specific part. Experiences from software engineering shows that modules provide a good level of abstraction to support maintenance and re-use.

Efficient Reasoning. A specific problem that occurs in the case of distributed ontologies as well as very large models is the problem of efficient reasoning. While the pure size of the ontologies causes problems in the latter case, in a distributed setting, hidden dependencies and cyclic references can cause serious problems. The introduction of modules with local semantics and clear interfaces will help to analyze distributed systems and provides a basis for the development of methods for localizing inference.

They discuss further that the requirements of ontology modularization are: i) Loose Coupling: mappings between modules have to be distinguished from local definitions on the semantic as well as the conceptual level, ii)Self-Containment: the result of certain reasoning tasks such as subsumption or query answering within a single module has to be possible without having to access other modules and

Classification	Method	Scalability	Heterogeneity	Dynamics	Inconsistency	Parallelism
Context Reasoning	Cyc	yes	no	no	maybe	maybe
Ontology Modularization	Stuckenschmidt and Klein	yes	no	no	maybe	maybe
Ontology Partitioning	Stuckenschmidt and Klein	yes	no	no	maybe	maybe

Figure 5.1: Contextual and Module Reasoning and Web Scale Reasoning

ensure correctness and whenever possible completeness of local reasoning for obvious reasons, and iii) Integrity: provide mechanisms for checking whether relevant knowledge in other systems has changed and for adapting the reasoning process if needed to ensure correctness.

In [151], Stuckenschmidt and Klein propose a method for structure-based ontology partitioning. This approach supports automatically partitioning large ontologies into smaller modules based on the structure of the class hierarchy. They show that the structure-based performs surprisingly well on real world ontologies. The claim is supported by experiments carried out on real world ontologies including SUMO and the NCI cancer ontology.

5.3 Relation with Web Scale Reasoning

In this seciton, we will summarize the analysis on the relation between approaches of contextual and module reasoning and the requirement of web scale reasoning.

Scalability All of the approaches in contextual reasoning and modular reasoning are motivated to deal with the problem of the scalability. Thus, the answers to the scalability are ”yes”.

Heterogeneity None of the approaches discussed above allow for the semantic data with different formalisms. Thus, the answers to the heterogeneity are ”no”.

Dynamics None of the approaches reported their support of real time data. Thus, the answers to the dynamics are ”no”.

Inconsistency In Cyc, Knowledge Entry usually aims for logical consistency within a Microtheory, but not global consistency throughout the whole Cyc Knowledge Base. Thus, the answer to the inconsistency is ”maybe”. The same stories occur in the approaches of ontology modularization and ontology partitioning.

Parallelism All of the approaches discussed above can be considered as one which supports some kinds of parallelization, because large scale data have been partitioned. However, their implementation with distributed computing or parallel computing have not yet been reported. Thus, their answers to the parallelism are ”maybe”.

The table 5.1 is a summary of contextual and module reasoning methods and their relations with the requirement imposed by web scale reasoning.

6 Distributed Reasoning and Parallel Reasoning

6.1 Distributed Reasoning

There are several ways of distributed computing. One very common method for distributed computing is a P2P architecture which is able to connect several computers in order to ensure the development of a large network consisting of arbitrary different computers. Other methods of developing a distributed system are Grid-Computing, Cloud-Computing, Thinking@Home or the development of a SOA[1]. We have to expose which distributed computing approaches are strongly related to reasoning tasks.

A distributed system consists of different modules with communication paths between the modules. However, there are many ways of distributing a system, e.g. a centralized system with shared knowledge or a totally distributed control and knowledge or a system somewhere between these two directions. But no matter which way of distributing we select, there are some important aspects which are relevant for all kinds of distributed reasoning systems:

1. A responsible coordination between the modules
2. An effectively communication structure
3. Techniques of distributed reasoning

In the following we will present two approaches of distribution, both should be considered for LarKC. The first approach is a P2P architecture and the second approach is based on a Web Service. First of all we will take a closer look to P2P architectures. The idea is to develop a P2P system in which each peer can answer queries by reasoning from its local theory but can also ask queries to some other peers with which it is semantically related by sharing part of its vocabulary[2]. For reasoning tasks such a P2P system acts as an inference system in which every peer is enabled to reason locally and from known peers in the system. Adjiman et al. are describing a peer-to-peer inference system as a network where each peer can reason locally but also solicit some of its acquaintances [1]. All peers share the same vocabulary. Adjiman et al. are stressing this approach by presenting the "Somewhere Peer-to-peer Data Management System". Further, Calvanese et al. are describing a P2P data integration system for managing data [30]. In this P2P system each peer manages a set of local data sources sementically connected. Calvanese are descirbing a semantic data integration which assumes not a global view but a local view of ech peer [28]. Through this, each peer exports data and an integration of information is ensured by a mapping between among all available peers. However, there are several approaches for developing a P2P system.

[1] Service Oriented Architecture

6.1.1 P2P systems

Super P2P

A super P2P architecture consists of a set of super-peers which are connected to other super-peers as well as to a set of clients. A super peer is a node that operates both as a server to a set of clients and as a peer in a network of super-peers. The main advantages are the scalability and the easiness of the discovery process. Furthermore, a super-peer network benefits from a balance between the inherent efficiency of centralized search, and the autonomy, load balancing and robustness to attacks provided by distributed search. Since there are relatively many super-peers in a system, no single super-peer need to handle a very large load, nor will one peer become a bottleneck or single point of failure for the entire system. Moreover, they take advantage of the heterogeneity of capabilities (e.g. bandwidth, processing power) across peers. However, a super-peer architecture may become a point of bottleneck for its clients. Therefore, the design of a super-peer network involves performance tradeoffs and questions difficult to answer, such as, ratio clients/super-peer, topology of the super-peer network, reliability of the peers towards their clients and so on.

Fully decentralized P2P

A fully decentralized P2P architecture consists of a set of peers which are connected to all other peers. A decentralized P2P architecture does not distinguish between different steps of authorization for the different peers because all peers have equal rights. This approach is a rudimentary step of developing a more specified P2P architecture or to get a simple approach. Pure P2P systems tend to be inefficient in different aspects such as the search protocol that usually flood the network with query messages. Furthermore, due to the limited capabilities of some peers there is always the risk of a bottleneck. An efficient P2P system must be designed taking advantage of the peers heterogeneity, assigning greater responsibility to those who are more capable of handling it.

Decentralized P2P with central server

A decentralized P2P architecture with central server consists of a server which is connected with several peers. It is a mixture between a client-server architecture and a P2P architecture, a hybrid approach. In this case we have a server with indexes to peers and resources but the data exchange or file download is decentralized. Such a decentralized P2P architecture can often result in a better trade-off between the ease of software maintenance, scalability, and reliability. One of the key advantages of the decentralized P2P architecture with central server is that it makes the discovery process more simple. This enables the deployment of a large number of clients and a high degree of scalability. Decentralized P2P systems with central server also have drawbacks, in the sense that the costs resulting from the single node housing the decentralized index are very high. Unless the index is distributed across several nodes, this single node becomes a performance and scalability bottleneck. Furthermore, decentralized P2P systems with central server are more vulnerable to attack, as there are few highly-visible targets that would bring down the entire system if they failed.

P2P and Reasoning

In the ”Somewhere Peer-to-peer Data Management System” (Adjiman et al.), every peer provides its own ontology. The ontologies of the peers are not imposed to the other peers but logical mappings of the ontologies. In this view, the Web is a huge peer-to-peer data management system based on simple distributed ontologies and mappings[2]. Serafini and Tamilin [137] have described an approach for handling distributed ontologies as well. They present DRAGO[2] that implements distributed decision procedures. It envisages a web of ontolgies which are distributed over a peer-to-peer network of DRAGO Reasoning Peers (DRP). The idea of DRAGO is to provide reasoning services for ontologies with mappings registered to it and to request reasoning services of other DRPs. Therefore, an ontology can be registered to a DRP with its URI giving it a physical location on the web. It is possible to assign semantic mappings on the ontology. Through the subsumption propagation mechanism it becomes possible to enrich ontologies by attaching mappings. For security cases only registered users are allowed to add mappings. If a user requires reasoning services he is enabled to correspondent a registered ontology by a DRP using the URI of the selected ontology. Furthermore, Calvanese et al. are describing a P2P network for a semantic data integration. This network is based on several peers with a local view and a mapping procedure which ensuires a mapping of the several data of the peers with all the other peers.

6.1.2 Web Service approach

Web Services are one possible way of realizing the technical aspects of SOA. These services can be new applications or just wrapped around existing legacy systems to make them SOA-enabled. Common technologies for developing web services are WSRF[3], SOAP[4], UDDI[5] and WSDL[6]. Furthermore, when using these technologies XML is a basic technology for developing web services this way. For reasoning aspects a Web Service is interesting if several reasoning components are available and accessible through the use of indexes possibly managed by other entity (a broker). A user is able to request a specific reasoning component by checking the indexes of the storage. For this three instances can be identified, a service consumer, a service provider and a Service Broker (storage of indexes). The service consumer is able to request the service provider regarding to the service which is supported. UDDI can be used to register existing services with a service broker. To describe a service WSDL is used and to ensure the communication between service consumer and service provider SOAP is an adequate technology. To demonstrate the use of the mentioned technologies the basic SOA example is presented with some modifications. Figure 6.1 presents how a Web Service can take place.

For a distributed reasoning both approaches (P2P and Web Services) have to be considered. However, the decision for or against an approach depends strongly on the Use-Cases.

[2] Distributed Reasoning Architecture for a Galaxy of Ontology
[3] Web Service Resource Framework
[4] Simple Object Access Protocol
[5] Universal Description, Discovery and Integration
[6] Web Service Description Language

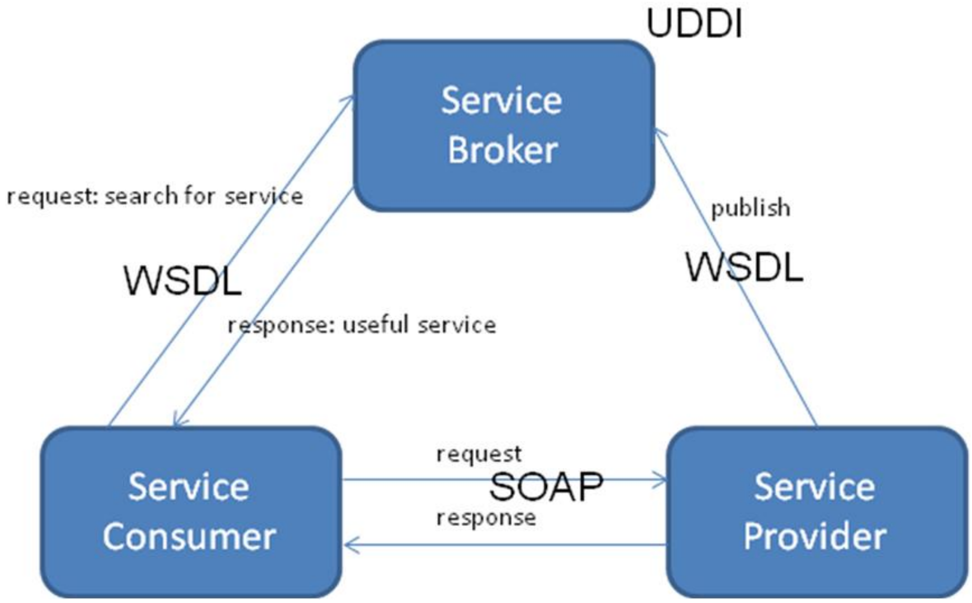

Figure 6.1: Web Service

6.2 Parallel Reasoning

When thinking about parallel reasoning we have to define what we want to parallelize. For reasoning aspects it is advisable to run several reasoning tasks in parallel in order to speed up the processing of all required reasoning tasks. The idea is to run several tasks at the same time. However, some tasks are connected to other tasks. For this tasks it is important to manage the processing of the tasks at similar time to ensure the communication of both tasks. Otherwise, if one task has to wait for the response of another task the whole process is slowed down. Figure 6.2 presents how parallel reasoning can take place.

A user accesses a task storage which stores all required tasks. In this case two tasks (task1 and task2) are running at same time in parallel. A communication between task1 and task2 is possible. Furthermore, there might be a loop between the result of one task and the task itself. This loop goes on until the final result is achieved. After the actual tasks finish the next tasks are processed. If there are no further tasks the whole reasoning process finishes. This parallel processing of tasks allows for a faster process execution. However, it shares the same drawbacks. When running two tasks in parallel both tasks are not processed in double speed because of three main limiting factors.

1. Latency
   The latency is the time when each process waits for another. If there are a lot of tasks running at same time the management of the tasks becomes important to ensure that tasks which are running at same time do not have to wait long time for the other tasks.

2. Bandwidth
   The bandwidth might become crucial if there are just a very few but extensive tasks running at same time. A good bandwidth is needed for such cases.

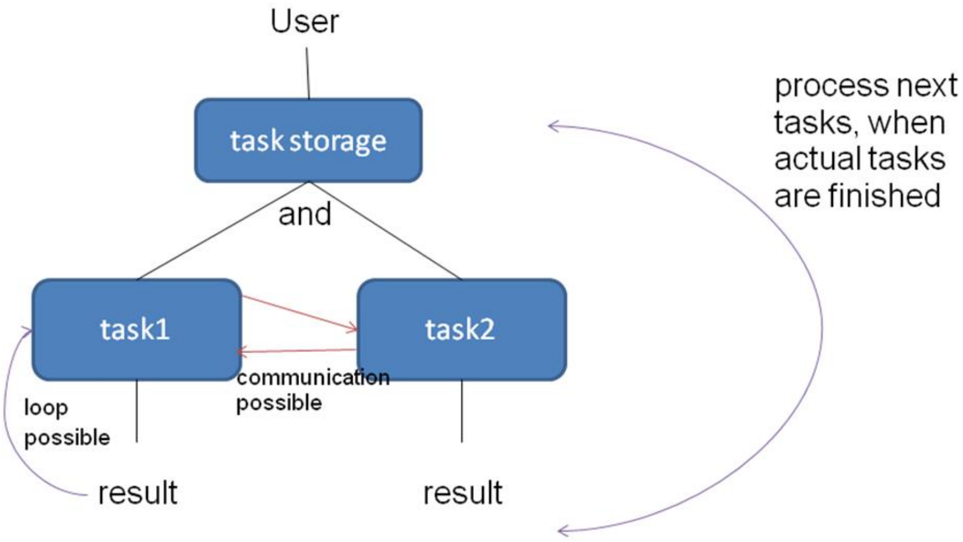

Figure 6.2: Parallel Reasoning

3. Additional work
   For all kinds of parallelization there is additional work to do. If tasks run in parallel it has to be managed which task is running at which time on which compute resources and so on.

However, a parallelization of reasoning tasks makes the processing faster than it is processed sequential one by one in a queue. In general there are some aspects which have to be considered regarding distribution and parallelization. It has to be considered in which way distribution and parallelization takes place, i.e., what is it that we want to distribute, tasks or data, and which tasks should run in parallel. In order to define how parallel reasoning takes place Bergmann and Quantz are describing a method for parallelizing Description Logic[18]. For this three main inference components of normalized-comparing based on DL systems are described, along with the potential for parallelization.

1. Subsumption Checking
   For testing subsumption between two terms they have to be normalized and then structurally compared. The normalization depends on the expressiveness of the DL.

2. Classification
   The classification method classifies each term name and compares it with all previously introduced names. The result is a subsumption hierarchy for concepts and roles as a DAG.

3. Propagation
   Subsumption Checking and Classification describes terminological reasoning whereas Propagation describes an assertional reasoning approach. It is reasoning on the object level by which we have to distinguish between a local and a non-local phase.
   In the local phase the most specific concept which is instantiated is determined. This is done by the use of standard normalization and comparison

of predicates as well as the search for direct supers in the classification component. Hereby, an object becomes normalized and it becomes possible to compare it with other objects in the hierarchy. Additionaly, for processing, DL rules have to be applied by applying all rules whose left-hand sides subsumes the object's normal form. Afterwards again the normal form is normalized and classified unitl no new rules are applicable.
In the non-local phase information to other objects are propagated.

Bergmann and Quantz [18] describe all three inference components as components with potential for parallelization but their favorite is the Propagation.

> The reason for this is that the basic operations in normalization, comparison, and classification are rather fine-grain, compared with the message passing overhead of MIMD systems. Object-level propagation, on the other hand, is an ideal candidate for our parallelization strategy. Each propagation is rather time-consuming and causes additional propagations where two can be straightforwardly parallelized since they are both independent and monotonic.

Regarding to the demonstrated idea for running several tasks in parallel the Propagation method seems to be very promising because of the potential to run tasks (additional propagations) in parallel. Furhtermore,the advantages of parallel strategies for deduction are described by Bonacina in "A taxonomy of parallel strategies for deduction" [20]. A parallelization can take place for inferences at the clause level or at the search level so that multiple deductive processes search the problem space in parallel. Each search process executes a strategy, develops a derivation and builts its own set of data. Regarding to a parallel search the master-slave philosophy becomes interesting (separate control of parallelism and control of deduction). For the most methods this means that each slave executes a sequential search plan to generate its derivation and all other control issues are managed centralized by the master. Hereby, a parallel search plan for such strategies is a collection of sequential search plans (one per slave).

> If $C =< I, ? >$ is a sequential strategy, and $C' =< I, ?' >$ is a parallel search master-slaves parallelization of C, we have $?' =<?1, , ?n >$, if there are n slaves. In the case of distributed search with no multi-search, it is $?1 = \ =?n$, and the activities of the slaves are differentiated only by subdivision decided by the master.

Parallelization is advisable for search. This is essential if we think of matching terms for reasoning tasks. In general such a matching of terms is always related to search algorithms. Therefore, for reasoning aspects in the way of search through knowledge bases to match terms, parallelization is a strong instrument to fasten these processes.

In LarKC, parallelized algorithms may be deployed and executed in, at least, two different ways:

- P2P
- Cluster parallelization

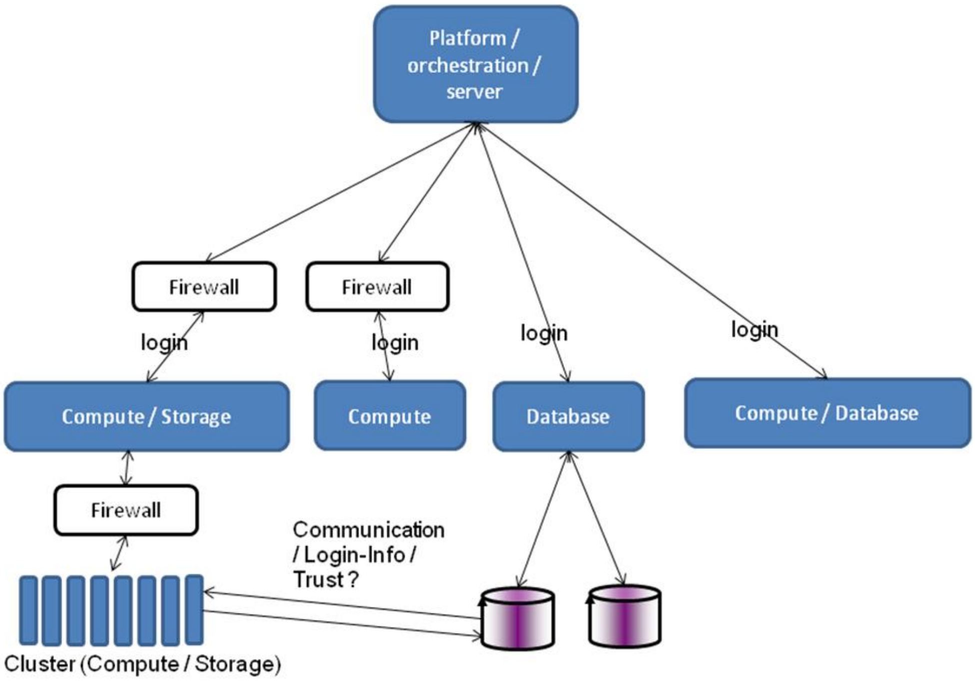
Platform / orchestration / server
Firewall
Firewall
login
login
login
login
Compute / Storage
Compute
Database
Compute / Database
Firewall
Communication / Login-Info / Trust ?
Cluster (Compute / Storage)

Figure 6.3: Accessing the Cluster

The P2P approach has been described in previous sections, when talking about distributed reasoning.

In the case of cluster parallelization the security aspect must be taken into account

Figure 6.3 gives an overview about accessing the Cluster.

The platform (server / orchestration) is connected to different compute and storage nodes (that can be located either in independent machines or in the same machine). This connection is monitored by a firewall which requires a login. The login is necessary because of security aspects of the cluster. Every user who wants to access the compute or storage nodes within the cluster has to identify him/herself.

From a distribution viewpoint it is necessary that every component outside the cluster that wants to access it is identified and known by the cluster.

If for efficiency reasons it is necessary to place the storage nodes inside the cluster's barriers a possible solution would be to use a temporary storage node/module. Such temporary node could be maintained during the processing period. Once the process in finalized, the data will be removed. In case data needs to be permanently stored it is necessary to define in which way the communication takes place. If there is a direct communication between the database and the cluster, trust and security aspects must be considered.

6.3 Distributed Reasoning and Web Scale Reasoning

This section provides an overview of issues about Distribution and Parallelism and the requirement of Web Scale Reasoning. The issues are Scalability, Heterogeneity, Dynamics, Inconsistency and Parallelism.

Classification	Method	Scalability	Heterogeneity	Dynamics	Inconsistency	Parallelism
P2P	Super	yes	yes	maybe	yes	yes
P2P	Fully decentralized	maybe	yes	maybe	maybe	yes
P2P	Decentralized with central server	yes	yes	yes	yes	yes
Web Service	standard approach	yes	yes	yes	yes	yes
Web Service	connected to a P2P network	maybe	yes	maybe	maybe	yes

Figure 6.4: Distributed Reasoning and Web Scale Reasoning

Scalability P2P networks are operating in unreliable environments where peers are continuously connecting and disconnecting. Hence, in a P2P network there are a lot of uncertainties which have to be handled. For this, there are three main factors to consider: communication, searching and load-balancing peers. Regarding to the communication between peers there is the risk of a bottleneck if one peer in a fully decentralized network slows down the whole communication. This problem might be handled by the use of a few super peers which ensure a good communication or by a decentralized P2P system with a central server where the central server is strong enough to guarantee a fast communication. When thinking about searching we have to consider that in the case of a P2P network with lots of peers processing a search might become a critical factor and might slow down the whole system. If the search is done by a central unit where no other connections to other units are necessary this problem is not as problematic as for a P2P network. Furthermore, regarding to load-balancing peers there might be some problems in scalability, too. In theory, fully decentralized P2P networks consist entirely of equal peers. This means that each peer potentially shares as much as it consumes. This in turn can slow down the whole processing. However, P2P systems aim to deal with the problem of scalability, sophisticated approaches are needed. When thinking about P2P networks we determine that those approaches are available, e.g. Napster or other P2P architectures which handle these problems.

Web Services which do not refer to P2P networks do not have the scalability problem. If there is a central unit with enough compute resources for the given task there are no problems in communication, searching and load-bandwidth peers as mentioned before regarding to P2P networks. In this case a communication between several units is not necessary.

Another idea is to distribute a Web Service like a P2P network, e.g. a Web Service accesses a P2P network. For this case there might be problems in scalability if the Web Service also uses a P2P network. At this it is necessary to know where which peers are and if they are usable or not at certain time.

Heterogeneity Web Services support dynamic discovery and composition in heterogeneous environments by using WSDL to describe the Web service in such a way that it is independent of any specific implementation of that service. Furthermore, there has been a widespread adoption of Web Services technology. There are a number of tools that already exist, e.g. WSDL to language convertors. Moreover, Web services are built on standard technology, which is necessary for such widespread adoption.

Regarding to P2P networks there are the same issues about heterogeneity as for Web Services if we use a P2P network in combination with a Web Service. If the communication between the peers is ensured heterogeneity becomes a solvable issue. Furthermore, heterogeneity in P2P networks depends on the used standards but in general it is possible to develop a heterogeneous P2P network.

Dynamics P2P networks and Web Services are applicable in a dynamic environment with rapidly changing dynamic data sets. However, it has to be considered if a high data exchange takes place. In this case the P2P network has to support this massively exchange of data. Therefore, a high bandwidth is required. When thinking about a Web Service with a central server the data exchange is not as critical because data have not to be distributed by several peers. For this case a P2P network with a central server might be another solution which should be considered. It seems to be more comfortable to provide rapidly changing data sets with a P2P network with a central server than using a fully decentralized P2P network. A fully decentralized P2P network is able to handle rapidly changing data but if the data is distributed over lots of peers it has to be ensured that the data is updated when it changes. If the data changes very fast we have to do lots of updates. It depends on the number of peers, bandwidth and availability of the peers if such a P2P network is able to handle these issues in an adequate time. When thinking about super P2P networks these issues might be easier to solve if there is a small number of super peers which is available with much compute resources.

Inconsistency A P2P network or a Web Service do not have problems with inconsistencies normally. In a fully distributed P2P network it might be a problem if some peers are not achievable or the location of a peer is not known by other peers which try to access. This can slow down the processing time and can course inconsistencies as well. But such problems are handleable today. However, a Web Service with a central unit does not have such problems in general. Another idea is to use a P2P network with a central server where this server has the compute resources to solve given tasks or a super peer where a small number of super peers is available and the locations of the super peers are known.

Parallelism Regarding to parallel reasoning tasks we determine that P2P networks and Web Services are adequate for parallel reasoning tasks. Handling such requirements in a P2P network means to run several tasks at same time on several peers or a Web Service which is connected to a P2P network. Whereas, for a Web Service it is not necessary to connect it to a P2P network. However, P2P approaches and Web Service approaches are suitable to run processes in parallel.

The table 6.4 is a summary of various distributed reasoning methods and their relations with the requirement imposed by web scale reasoning.

6.4 Conclusions

For LarKC we have to consider about how to distribute and to run processes in parallel. It has to be defined what entities have to be distributed. However, there are two main aspects of distribution. On the one hand it is possible to distribute the compute resources on the other hand a distribution of data is possible as well. Serafini and Tamilin [137] are presenting an approach for distribution of ontologies and processing them with DRAGO. The peer-to-peer system which is proposed from Adjiman et al. In [1]. Adijman et al. describes distributed reasoning in the way that every peer is able to reason locally and from other known peers. Calvanese et al. are describing a similar approach where the local view is enhanced by a mapping between the peers.

When thinking about distributing compute resources it is rather important to manage the distribution effectively. Therefore, it has to be clarified where which compute resources are available and how to distribute them. The interaction of the compute resources must be managed. We have to know where these resources are ant to ensure a secure connection between these resources. Furthermore, a distribution of data storage requires a secure connection between the data storage and as well. Also, we have to know where to find the data storages. The difference between distributing compute resources and storage of data is a basic difference in architecture. If the storage of data is distributed we have to know where which data is stored. Otherwise, if the compute resources are distributed we have to know which jobs are running on which compute resources.

However, for distributed reasoning we have to define what to distribute and how such an architecture might look like and how to access the Cluster or how to manage the distribution among peers, depending on the chosen approach for such a distribution.

Through the definition of parallel tasks it becomes possible to run reasoning tasks faster. However, a good management of the limitations is needed. Furthermore, we have to define which processes are running in parallel (inference processes, search processes). A parallel reasoning search is described by Bonacina [20] and a parallel processing of inferences by Bergmann and Quantz[18].

7 Use Cases and Reasoning Tasks

7.1 Introduction

Previous chapters are all about very general approaches to scalability (approximation, resource-bounds, modularity, distribution and parallelisation), but these approaches can in principle be applied to many different reasoning tasks.

In this chapter we first identify which reasoning tasks are relevant for Semantic Web reasoning (we do this by inspecting a large number of prominent use-cases, points 1-3 below), and then we discuss which scalability approaches have been applied to which Semantic Web reasoning tasks (point 4 below).

The structure is as follows:

1. make categories of use-cases
2. survey the most prominent Semantic Web system applications and prototypes of the past few years, and classify each into one or more of these use-case categories
3. link each of the use-case categories to different basic reasoning tasks
4. discuss the relation between different reasoning tasks and the large scale reasoning approaches from Chapters 2-6 .
   - approximate reasoning
   - resource-bounded reasoning
   - rule-based reasoning for dynamic and incomplete knowledge
   - modularity and contextual reasoning
   - distributed reasoning and parallel reasoning

7.2 Categorisation of Semantic Web Use-cases

In this section, we characterise a number of Semantic Web use-cases in terms of the requests the user makes, the answer the system gives, and the knowledge that it uses for computing this answer. This categorisation of use-cases is based on use-cases that are currently wide spread available on the web.

We will use the terms terminology, ontology, class, concept and instance as follows: a terminology is set of class-definitions (a.k.a. concepts) organised in a subsumption hierarchy; instances are member of such concepts. Ontologies consist of terminologies and instance sets.

More formally we will consider an ontology as a set of triples:

$$O = \{< s, p, o >: s \in Subject, p \in Predicate, o \in Object\}, where\{\in, \subseteq\} \subseteq Predicate$$

Namely, we consider two specific predicates: $\subseteq$ for the subsumption relation, and $\in$ for the membership relation. We use $< C1, \subseteq, C_2 >$ to denote that a class C_1 is subsumed by a class C_2[1]. We use $< i, \in, C >$ to denote that an individual I is

[1] We use the same notation to denote the role hierarchies, like those in Description Logics

a member of a class C. Thus, a terminology T is a set of triples which predicate is the subsumption $\subseteq$ only, and an instant set I is a set of triples which predicate is the membership relation $\in$ only. Triple sets can be used to represent various semantic data, which includes rules and meta rules. For web scale reasoning, we usually use the notion of ontology in a boarder sense. Namely, ontologies and knowledge (ontologies and their rules) are interchangeable. Thus, a triple set can be always considered one which corresponds with an ontology.

Ontologies may have different versions and different authorship. They may change as the time passes. They may be context dependent. Semantic Web data may consist of a set of ontologies. Thus, it is useful to introduce an additional parameter on triple sets to represent additional information of ontologies,like provenance, version, reliability, preference, context, module, etc. Formally we define semantic web data as a quadruple set.

$$WebData = \{< s, p, o, a >: s \in Subject, p \in Predicate, o \in Object, a \in Thing\}.$$

For example, we use $\{< s, p, o, O_1 >, < O_1, hasVersion, 1.0.0, \_ >\}$ to represent that triple $< s, p, o >$ belongs to the ontology O_1 with the version 1.0.0.

Search:

Perhaps the most prototypical Semantic Web use-case is search, motivated by the low precision of current search engines. Search is a mapping which assigns a set of individuals for each concept c and each ontology $O = (T, A)$.

$$search : Concepts \times Terminologies \times InstanceSets \rightarrow InstanceSets.$$

Search can be characterised as follows:

request: concept c

knowledge:

- Terminology T: set of $< c_i, \subseteq, c_j >$
- Instance set A: set of $< i, \in, c_j >$

answer: members of the instance set:

$$search(c, (T, A)) = \{< i, \in, c > | \exists c_j (< c, \subseteq, c_j > \in T \wedge < i, \in, c_j > \in A)\}$$

Traditional search engines take as their inputs a query (usually in the form of a set of keywords) plus data-set of instances (usually a very large set, $O(10^9)$) of web-pages, and return a subset of those web-pages. A Semantic Web search engine would take a query in the form of a concept description, this concept description would be matched against an ontology that was used to organise the instance set, and members of the data-set matching the query-concept would be returned.

Browse:

Browsing is very similar to searching (and often mentioned in the same breadth), but has as crucial difference that its output can either be a set of instances (as in search), or a set of concepts, that can be used for repeating the same action (i.e. further browsing). Thus, browse is a mapping which assigns a set of individuals or a set of concepts for each concept c and each ontology $O = (T, A)$.

$$browse : Concepts \times Terminologies \times InstanceSets \rightarrow InstanceSets \cup Terminologies.$$

request: concept c

knowledge:

- Terminology T: set of $< c_i, \subseteq, c_j >$
- Instance set A: set of $< i, \in, c_j >$

answer: members of the instance set or concepts of the ontology:

$$\begin{aligned} brows(c, (T, A)) = \; & search(c, (T, A)) \cup \\ & \{< c_j, \subseteq, c > \mid < c_j, \subseteq, c > \in T\} \cup \\ & \{< c, \subseteq, c_j > \mid < c, \subseteq, c_j > \in T\} \end{aligned}$$

Data integration:

The goal of data-integration is to take multiple instance sets, each organised in their own ontology, and to construct a single, merged instance set, organised in a single, merged ontology. Data integration is a mapping which assigns a (new) ontology to a set of ontologies. Thus,

$$DataIntegration : \mathcal{P}(Ontologies) \rightarrow Ontologies.$$

request: multiple ontologies with their instance sets:

$$\{(T, A) | < i, \in, c > \in A \wedge (< c, \subseteq, c_j > \in T\}$$

knowledge:

answer: a single ontology and instance set

$$dataintegration(\{(T_1, A_1 >, .., < T_n, I_n >\}) = (T_{ck}, A_{ck}).$$

Personalisation and recommending:

Personalisation consists of taking a (typically large) data set plus a personal profile, and to return a (typically much smaller) data set based on this user profile. The profile which characterises the interests of the user can be in the form of a set of concepts, or a set of instances (e.g. typical recommend services at on-line shops use previously bought items, which are instances, while news-casting sites typically use general categories of interest, which are concepts). Thus, a profile can be characterised as a preference order on ontologies:

request:

- instance set A: set of $< i, \in, c_j >$
- profile $P \subseteq Ontologies \times Ontologies$

knowledge: terminology T: set of $< c_i, \subseteq, c_j >$

answer: reduced instance set:

$$personalisation(P, A, T) \subseteq A$$

Approximate methods can be employed to determine how closely the resulting instance set should resemble the profile: only instances that are exactly members of the profile-concepts, or also instances that "almost" members of the profile-concepts.

Web-service selection:

Rather than only searching for static material such as text and images, the aim of semantic web services is to allow searching for active components, using semantic descriptions of web-services. At the level of the signature ,the characterisation of this task is roughly the same as that of general search:

request: functionality f

knowledge:

- ontology O_f: set of $< f_i, p, f_j >$
- instance set O_i: set of $< service_i, "implement", f_j >$

answer: members of the instance set:

$$\begin{array}{l} service\_selection(f, (O_f, O_i)) = \\ \{< s_i, "implement", f > \mid < f, p, f_j > \in O_f \vee < f_j, p, f > \in O_f\} \end{array}$$

Differences are that the query describes functionality (rather than content), and the instance set consists of services (and their locations). The opportunities for approximation here are also similar as to the search use-cvase: use approximate sub- and super-concepts of the query-concept, and find instances that are almost members of the search concept.

Web-service composition:

An even more ambitious goal than web-service selection is to compose a given number of candidate services into a single composite service with a specific functionality:

request: functionality f

knowledge:

- ontology O_f: set of $< f_i, p, f_j >$
- instance set O_i: set of $< service_i, "implement", f_j >$

answer: composition of service instances:

$$\begin{array}{l} \mathit{service\_composition}(f,(O_f,O_i)) = \\ \text{control flow of } s_i\text{'s that satisfy } f\text{, where for all } s_i\text{: } < s_i, "\mathit{implement}", f_j > \in O_i \end{array}$$

In many cases it will not be possible to obtain exactly the required functionality, and a composition has to be constructed that has "almost" the required functionality, under some suitable definition of "almost". It is already difficult to define suitable distance measures on static resources (such as in the search and browse use-cases), it is in general unknown how to define such measures on functional objects such as web-services.

Question answering

This use-case could also have been called "deducing implicit information:

request: concept c

knowledge:

- terminology T: set of $< c_i, \subseteq, c_j >$
- instance set A: set of $< i, \in, c_j >$
- background knowledge BK: rules about the concepts in T

answer: answer a

$$\begin{array}{l} \mathit{Question\_answering}(c,(T,A)) = \\ \{< i, \in, c > \in A | \exists c_j(< c, \subseteq, c_j > \in T \wedge < c_j, \subseteq, c > \in T\} \cup BK \vdash a)\} \end{array}$$

the difference is that we expect that the answer to the query is not simply a member of the instance set, but is derived from the ontology and instance set using further background knowledge. Various forms of approximate deduction can of course be used as part of this use-case.

Semantic Enrichment

This use case concerns about annotation of objects, which can be for instance images, documents, etc. Based on these added meta-data other use-case like search, browse increase in their quality of answers.

request:

- instance i
- meta data of an instance: set of $< i, p, c >$

knowledge: ontology $O : \{< s, p, o >: s \in \mathit{Subject}, p \in \mathit{Predicate}, o \in \mathit{Object}\}$

answer:

$$\mathit{Semantic\_enrichment}(i, O) \supseteq O$$

In the next section we will see whether these use-cases correspond with the use-cases that we find nowadays on the web.

Figure 7.1: Classification of Semantic web challenges (http://challenge.semanticweb.org) in our use-case categories.

7.3 Coverage of use-case categorisation

"The Semantic Web Challenge" is an annual event of the Semantic Web community that offers participants the possibility to illustrate to society what the Semantic Web can provide, gives researchers an opportunity to showcase their work and compare it to others and somehow stimulates current research by showing the state-of-the-art every year. Currently there have been organised six Semantic Web Challenge events [2]. We have conducted a study to see whether the semantic web challenges (from 2005, 2006, 2007) fit to our identified use-cases. See figure 7.1 for the results. Our main observations are the following:

- Most applications [3] can be classified in our category of use-cases. Figure 7.1 shows the classification of semantic web challenges applications in terms of the use-cases given.
- One application belongs often to more than one use-case. See for instance the application "MultimediaN E-Culture" which does searching, browsing, and semantical enrichment.
- In the web challenges from 2005-2007 there were no applications implementing web-service selection, although there were some implementing Web service configuration.
- There is very little work on "question answering" based on semantic web techniques. SMART is the only question answering application, and Dbin has a flavor of question answering in the sense that they do not give the answer to the question but give a piece of text.
- Search and browse are use-cases that are often together in an application.

7.4 Semantic Web reasoning tasks

In this section we argue how the use-cases from the previous section can all be implemented using a limited number of reasoning tasks. We give a formal definition of these reasoning tasks, including its signature, and show which use cases above can be implemented using which combination of reasoning tasks. We also discuss possible approximate version of these reasoning tasks. In our notation, o is an ontology (typically expressed in some description logic such as OWL), c is a concept defined in such an ontology, and i is an instance belonging to such a concept.

[2] See `http://challenge.semanticweb.org/`

[3] We were not able to find information about some applications, and from one application we could not understand its functionality.

Realisation

Realisation is the task of determining which concepts a given instance is a member of.

Signature: $i : instance \times o : ontology \mapsto c : concept$

Definition: Find a c such that $o \models i \in c$

Approximate realisation finds concepts c such that i *is almost a member* of c, under some definition of "almost": i might have a small number of missing properties that stop from being a proper member of c.

Subsumption

Subsumption is the task of determining whether one concept is a subset of another:

Signature: $c_1 : concept \times c_2 : concept \mapsto bool$

Definition: Determine whether $c_1 \sqsubseteq c_2$

Approximate subsumption determines if c_1 is *almost a subset of* c_2, under some definition of "almost", for example if some small modification $\delta(c_1)$ is needed before $\delta(c_1) \sqsubseteq c_2$ (or similarly for c_2).

Mapping

Mapping is the task of finding a correspondence relation between two concepts. We follow the definition proposed in the survey report [50]:

Signature: $c_1 : concept \times c_2 : concept \mapsto r : relation$
where any *relation* r is either an equivalence ($=$), subsumption ($\sqsubseteq$,$\sqsupseteq$), or disjointness ($\perp$).

Definition: find an r such that $c_1 \ r \ c_2$

Approximate subsumption determines an r such that $c_1 \ r \ c_2$ *almost holds*, using for example approximate notions of subsumption described above.

Retrieval

Retrieval is the inverse of realisation: determining which instances belong to a given concept:

Signature: $c : concept \mapsto i : instances$

Definition: find i such that $i \in c$

Approximate realisation finds instances i that are "almost a member of c", using similar definitions of "almost" as under approximate realisation.

Classification

Classification is the task of determining where a given class should be placed in a subsumption hierarchy:

Signature: $c : \mathit{concept} \times h : \mathit{hierarchy} \mapsto < c_l, c_h >$

Definition: Find a highest subclass c_l and a lowest superclass s_h such that $c_l \sqsubseteq c \sqsubseteq c_h$

Since classification is defined in terms of subsumption, approximate classification can be defined in terms of approximate subsumption.

Our choice of these five basic reasoning tasks is not the only choice possible. For instance, both classification and mapping can be reduced to repeated subsumption checks, and are hence not strictly speaking required as separate tasks. Similarly, it is well known that subsumption in turn can be reduced to satisfiability. However, we have chosen the above four reasoning tasks because they seem to constitute a conceptually coherent (although not formally minimal) set.

The following table shows how each of the use-cases from section 7.2 can be implemented using the reasoning tasks described in figure 7.2.

	realisation	subsumption classification	mapping	retrieval
search		x		x
browse	x	x		x
data integration	x	x	x	
personalisation	x	x		x
service selection		x		x
service composition		x		
question answering				x
semantic enrichment				

Figure 7.2: Use-cases in terms of reasoning tasks.

We lack the space to discuss all of these decompositions in detail, but will discuss only two examples:

Search: the description of the Search use-case above shows that it is a combination of *classification* (to locate the query-concept in the ontology in order to find its direct sub- or super-concepts) followed by *retrieval* (to determine the instances of those concepts, which form the answers to the query).

Personalisation: If the personal profile in the personalisation use-case consists of a set of instances (e.g. previously bought items), then personalisation is a composition of *realisation* (to obtain the concepts that describe these instances), *classification* (to find closely related concepts), and *retrieval* (to obtain instances of such related concepts, since these instances might of be interest to the user).

In a similar way, all of the prototypical use-cases we described in section 7.2 can be implemented in terms of the small set of elementary reasoning tasks described in this section.

Notice that the table above only displays the minimally required reasoning tasks for each use-case. For example, it is well possible to equip the search use-case with a mapping component in order to map the vocabulary of a user-query to the vocabulary of the ontology used during search. Similar additions could have been made for many other use-cases. These optional reasoning tasks are not listed in the table.

Notice too that semantic enrichment can not be defined based on these reasoning tasks. The reason for this is that usually in reasoning the input facts are assumed to be given, while semantic enrichment deals with *constructing* these input facts. Hence, semantic enrichment cannot be seen in terms of reasoning tasks.

7.5 Using the approaches to web-scalability for the reasoning tasks

In principle, each of the approaches from Chapters 2-6 can be applied to each of the reasoning tasks we identified above, in order to achieve web-scalability. This would produce the cross-product of all approaches in Chapters 2-6 with the four reasoning tasks. The results of this (e.g. approximate subsumption, or parallelised realisation, etc) can then be deployed in the different categories of use-cases, according to table 7.2. From the viewpoint of the use-cases it is clear that such approximate versions of reasoning tasks are useful. For instance opportunities for approximation exist in the search use-case to find approximate sub- and super-concepts of the query-concept, or to find instances that are almost members of such sub- and super-concepts instead of finding the exact sub- and super-concepts of the query concept (see Figure 7.2). Another example is the use-case of data integration (see Figure 7.2): obvious opportunities for approximation are the way in which the merged data-set is constructed (which may be less than the union of the original data sets), or the way in which the merged ontology is constructed (which may entail either more or less consequences than the union of the two original ontologies).

As mentioned, all these use-cases can be described in terms of the four basic reasoning tasks, and each of the approaches from Chapters 2-6 can be applied to each of these reasoning tasks. An exhaustive analysis of all these opportunities would go beyond the scope of the current deliverable. Instead, below we show for two examples what the results of such an analysis could be, namely applying approximation techniques to the subsumption and retrieval tasks. This results in concrete techniques that can be used directly in the relevant use-cases that rely on these tasks.

7.5.1 Example 1: Approximation for Subsumption

Subsumption is one of the main reasoning tasks in the Semantic Web. Fortunately this reasoning method is approximated by several approaches in different ways.

First, the inference engine can be adopted in order to approximate the subsumption query $C \sqsubseteq D$ where C and D are concept expressions. An instantiation of the approximation of the reasoning method is for example Cadoli and Schaerf [131] method (see Section 2.1). Because their method is able to deal with first order logic (FOL) — and the OWL ontologies underlying description logics [72] is a subset of FOL — their method provide already a S-1/S-3 interpretation for description logics. Intuitively spoken parts of the concept expressions C and D which are not a member of S are interpreted as true or false in correspondence to the S-1/S-3-approximation for propositional logics.

Furthermore Cadoli and Schaerf also show in [131] that the effect of approximating the inference can be achieved with simplifying the original query $C \sqsubseteq D$. Initially they propose to replace some sub parts in the concept expressions C and D with $\perp$ or $\top$ resulting in C' and D'. The request $C' \sqsubseteq D'$ approximate the original request $C \sqsubseteq D$. Obviously the replaced parts are those which are not member of S where the S-1/S-3 interpretation interprets everything outside of S per definition with false resp. true. Cadoli and Schaerf prove in [131] that traditional satisfiability of the simplified query $C' \sqsubseteq D'$ is equal to the satisfiability based on S-1/S-3 interpretation. In generalized words the effect of some approximating inferences can be "simulated" by language weakening. In fact these approaches constitute the second class of approximating subsumption methods which approximate the request and/or the knowledge base.

Several approaches was developed for approximating the request; they differ in what they replace. Cadoli and Schaerf themselves propose to replace subexpressions of syntactic form $\exists R.C$ in concept expressions where R is a relation name and C is a concept expression. Furthermore they also discuss to consider additionally some concept names A (and their negation $\neg A$) as sources for replacements [131]. Stuckenschmidt [149] generalize previous approach in that way that he additionally allows to replace any syntactic form of relation expressions to be replaced (like $\forall R.C$ or number restrictions like $\leq n.R$). Furthermore Stuckenschmidt introduced the notion of "subsumption wrt. a sub-vocabulary" S, i.e. $C \sqsubseteq_S D$, and reason with respect to different subsets, e.g. $C \sqsubseteq_{S_1} D \Rightarrow C \sqsubseteq_{S_2} D$ if $S_2 \subseteq S_1$.

However experiments have shown that the success of Cadoli-Schaerf approximation correlates with which part is replaced [62, 156]. Some replacements are worth and do not provide any benefit but some parts can speed up the reasoning. Unfortunately the successful parts differ from request to request. Therefore a static approach which selects the part by its syntactical structure (e.g. all appearance of $\exists R.C$) do not provide the required flexibility; it should be decided depending on the concrete request and knowledge base.

In [156] a flexible replacement approach is presented which ranks the conjuncts in a conjunctive concept expression (i.e. a conjunctive boolean query Q) with the help of a heuristic function Ψ. High ranked conjunctive parts are more less replaced than low ranked parts. Then the ranking function Ψ decides about the success of the approximation. In [156] Ψ estimates the computation time needed for reason with that part. As high the computation time is estimated as low the part is ranked. But further ranking function are imaginable which for example favor those parts which drastically prune the search space.

A different example which still rely on subsumption reasoning was originally developed for mapping of two concepts of different ontologies. Aleksovski et. al.

[3] uses the logical representation of concepts in order to determine if $C \sqsubseteq D$. The left-hand formula C is transformed into disjunctive normal form and the right-hand side D into conjunctive normal form. By doing this, the subclass check can be split into a set of subproblems; each checking if one (left) disjunct is a subclass of a (right) conjunct. If all the subproblems are satisfied, the original problem is satisfied. In approximation of Aleksovski et. al. [3] they allow a few of the subproblems to be unsatisfiable, while still declaring the original problem satisfiable. The (relative) number of satisfiable subproblems is a measure of how strongly the subclass relation between the two given formulas hold. For the matching problem those concept with the highest measure value are used to determine subsumption.

7.5.2 Example 2: Approximate Retrieval

Instance Store [16], Semantic Approximation [113] and Screech [68] are techniques that do approximate retrieval by approximation of the knowledge base (see Section 2.1). Instance Store uses a retrieval engine that is sound and complete only for role-free A-boxes in Description Logics. In other words: the theory is approximated by removing all roles form the A-box. Quill (the OWL 2 QL engine in the TrOWL ontology reasoning infrastructure) [113] semantically approximate OWL DL ontologies to OWL 2 QL (DL-Lite$_R$) ones. The scalable query answering service is soundness preserving in general; when there are no non-distinguished variables, the reasoning service is sound and complete.

In Screech the approach is to compile the ontology into a disjunctive datalog (DDL) program [76] that can be used to retrieve instances for a query. Screech extends the compiled ontology and translates the disjunctive datalog program to normal datalog. Simply spoken any clause with n disjunctions in the head is replaced by n horn clauses, each with one of the disjuncts in its head. Obviously this results in an approximated (unsound) method. It turns out that from practical viewpoint the retrieval time is greatly improved with only a small loss of correctness [68]. In a first experiment with the Galen ontology reported in [68] the retrieval time is improved by 39 percent while the 92 percent of the answers still was correct. From the theoretical point of view the reasoning has been shifted to a more appropriate complexity class. Because now normal optimized datalog engines can be used the retrieval is in P instead of co-NP for the disjunctive datalog engine. But although Screech has some relationship to brave reasoning with well-supported models which are known in non-monotonic reasoning the theory behind SCREECH is still unclear and needs further investigation.

7.6 Summary

In this chapter we have given a categorisation of use-cases, and showed that the web-challenges of the last three years fit very well in this categorisation. Furthermore we have identified the basic reasonings tasks that are used in these use-cases. We have also discussed how the different approaches from earlier chapters could be studied in the context of the identified reasoning tasks.

8 Concluding Remarks

In this document we distinguish a number of approaches that enable large scale reasoning: approximation and anytime behaviour (Chapter 2), resource bounded (Chapter 3), rule-based reasoning for dynamic and incomplete knowledge (Chapter 4), modularity and context (Chapter 5) and distribution and parallelisation (Chapter 6). We have seen that these approaches have the potential to enable one or several of the desired goals of the LarKC platform, as shown in the tables in every chapter where a member of a particular family is scored against its contribution of enabling scalability, heterogeneity, etc. In the final chapter we have identified 8 types of prototypical use-cases, and we decomposed each of those into a total of 4 basic reasoning tasks. This decomposition then allows (in principle) to apply the approaches from the earlier chapters to all these reasoning tasks (and hence to all use-cases).

This document has focused on the Reasoning plug-in, and has thereby laid a broad foundation for work to be done during the development of such Reasoning plug-ins. The ultimate utility of this analysis must become clear during the development of actual Reasoning plug-ins, where many of the approaches discussed in this chapter will be exploited (and the reasoning plug-ins themselves be in turn deployed in the use-cases).

References

[1] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Scalability study of peer-to-peer consequence finding. In *Proceedings of IJCAI05*, pages 351–356. HAL - CCSd - CNRS, 2005.

[2] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *Journal of Artificial Intelligence Research, Vol. 25*, pages 269–314, 2006.

[3] Zharko Aleksovski, Warner ten Kate, and Frank van Harmelen. Approximate semantic matching of music classes on the internet. In *Second Philips Symposium on Intelligent Algorithms (SOIA)*, 2004.

[4] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological Review*, 111:1036–1060, 2004.

[5] J.R. Anderson. Arguments concerning representations for mental imagery. *Psychol Rev*, 85:249–277, 1978.

[6] J.R. Anderson. *The adaptive character of thought.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1990.

[7] J.R. Anderson. Is human cognition adaptive. *Behavioral And Brain Sciences*, 14(3):471–484, 1991.

[8] J.R. Anderson and C. Lebiere. *The atomic components of thought.* Lawrence Erlbaum associates, Mahwah, New Jersey, 1998.

[9] J.R. Anderson, Y.L. Qin, K.J. Jung, and C.S. Carter. Information-processing modules and their relative modality specificity. *Cognitive Psychology*, 54:185–217, 2007.

[10] Masatoshi Arikawa, Shin'ichi Konomi, and Keisuke Ohnishi. Navitime: Supporting pedestrian navigation in the real world. *IEEE Pervasive Computing*, 6(3):21–29, 2007.

[11] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14(1):149–180, 1995.

[12] Shivnath Babu and Jennifer Widom. Continuous queries over data streams. *SIGMOD Rec.*, 30(3):109–120, 2001.

[13] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. Progressive distributed top-k retrieval in peer-to-peer networks. In *21st International Conference on Data Engineering (ICDE'05)*, pages 174–185, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[14] L. W. Barsalou. Perceptual symbol systems. *Behav Brain Sci*, 22:577–595, 1999.

[15] Arianna Bassoli, Johanna Brewer, Karen Martin, Paul Dourish, and Scott Mainwaring. Underground aesthetics: Rethinking urban computing. *IEEE Pervasive Computing*, 6(3):39–45, 2007.

[16] S. Bechhofer, I. Horrocks, and D. Turi. The owl instance store: System description. In *Proceedings CADE*, pages 177–181, 2005.

[17] M. Benerecetti, P. Bouquet P.1, and C. Ghidini. Contextual reasoning distilled. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(3):279–305, 2000.

[18] F. W. Bergmann and J. J. Quantz. Parallelizing description logics. In I. Wachsmuth, C.-R. Rollinger, and W. Brauer, editors, *KI-95: Advances in Artificial Intelligence, Bielefeld, Germany*, page 137148, Berlin, 1995. Springer-Verlag.

[19] H.A. Blair and V.S. Subrahmanian. Paraconsistent logic programming. *Theoretical computer science*, 68(2):135–154, 1989.

[20] M. P. Bonacina. A taxonomy of parallel strategies for deduction. *Annals of Mathematics and Artificial Intelligence, Vol. 29*, pages 223–257, 2001.

[21] S. Brass and J. Dix. Characterizations of the disjunctive stable semantics by partial evaluation. *The Journal of Logic Programming*, 32(3):207–228, 1997.

[22] S. Brass and J. Dix. Characterizations of the Disjunctive Well-Founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, 20(1-2):143–165, 1998.

[23] S. Brass and J. Dix. Semantics of (disjunctive) logic programs based on partial evaluation. *The Journal of Logic Programming*, 40(1):1–46, 1999.

[24] S. Brass, J. Dix, and T.C. Przymusinski. Computation of the semantics of autoepistemic belief theories. *Artificial Intelligence*, 112(1-2):233–250, 1999.

[25] G. Brewka, J. Dix, and K. Konolige. *Nonmonotonic reasoning: an overview.* CSLI Publications, Stanford, CA, 1997.

[26] M.W. Browne. Cross-validation methods. *Journal of Mathematical Psychology*, 44:108–132, 2000.

[27] E. Brunswik. Organismic achievement and environmental probability. *Psychological Review*, 50:255–272, 1943.

[28] D. Calvanese, E. Damaggio, G. De Giacomo, M. Lenzerini, and R. Rosati. Semantic data integration in p2p systems. In *Revised papers of the Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2003)*, volume 2944, pages 77–90, 2003.

[29] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in dls: the dl-lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[30] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data management in peer-to-peer data integration systems. In R. Baldoni, G. Cortese, F. Davide, and A. Melpignano, editors, *Global Data Management*. IOS Press, 2006.

[31] W.A. Carnielli, L.F. del Cerro, and M. Lima-Marques. Contextual negations and reasoning with contradictions. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 532–537.

[32] S. Castano, S. Espinosa, A. Ferrara, V. Karkaletsis, A. Kaya, S. Melzer, R. Moller, S. Montanelli, and G Petasis. Ontology dynamics with multimedia information: The boemie evolution methodology? In *International Workshop on Ontology Dynamics (IWOD-07)*, 2007.

[33] B.F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[34] N. Chomsky. *Aspects of the theory of syntax*. MIT press, Boston, MA, 1965.

[35] A. Colmerauer and P. Roussel. The birth of prolog. In *History of programming languages—II table of contents*, pages 331–367, 1996.

[36] M.K. Colvin, K. Dunbar, and J. Grafman. The effects of frontal lobe lesions on goal achievement in the water jug task. *Journal of Cognitive Neuroscience*, 13:1129–1147, 2001.

[37] J. Czerlinski, G. Gigerenzer, and D.G. Goldstein. How good are simple heuristics? In P. M. T. G. Gigerenzer, editor, *Simple heuristics that make us smart*, pages 97–118. New York: Oxford University Press, 1999.

[38] M. Dalal. Semantics of an anytime family of reasoners. In W. Wahlster, editor, *Proceedings of ECAI-96*, page 360364. Budapest, Hungary, John Wiley and Sons LTD, 1996.

[39] M. Dalal and Yang. Tractable reasoning in knowledge representation systems(??). 1997.

[40] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys (CSUR)*, 33(3):374–425, 2001.

[41] R.M. Dawes and B. Corrigan. Linear models in decision making. *Psychological Bulletin*, 81:95–106, 1974.

[42] Alvaro del Val. Approximate knowledge compilation: The first order case. In *AAAI/IAAI, Vol. 1*, pages 498–503, 1996.

[43] Emanuele Della Valle, Stefano Ceri, Davide F. Barbieri, Daniele Braga, and Alessandro Campi. A first step towards stream reasoning. In *Proceedings of the Future Internet Symposium*, 2008.

[44] E. Dietrich and A. Markman. *Cognitive dynamics: Computation and representation regained. Cognitive dynamics: Conceptual and representational change in humans and machines*. Mawah NJ: Erlbaum, 2000.

[45] J. Doyle and D. McDermott. Nonmonotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[46] J. Duncan, H. Emslie, P. Williams, R. Johnson, and C. Freer. Intelligence and the frontal lobe: The organization of goal-directed behavior. *Cognitive Psychology*, 30:257–303, 1996.

[47] F.M. Donini et al. Al-log: Integrating datalog and description logics. *Journal of Intelligent Information Systems*, 10:227–252, 1998.

[48] I. Horrocks et al. Swrl: A semantic web rule language combining owl and ruleml. In *W3C Member Submission*, volume 21, 2004.

[49] J. Euzenat. ontology alignment evaluation initiative. Technical report.

[50] Jerome Euzenat. State of the art on ontology alignment. Technical report, 2004.

[51] J.S.B.T. Evans. *Rationality and reasoning.* Psychology Press, Hove, England, 1996.

[52] S. Ganguly, A. Silberschatz, and S. Tsur. A framework for the parallel processing of Datalog queries. *ACM SIGMOD Record*, 19(2):143–152, 1990.

[53] Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. *Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[54] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080, 1988.

[55] J.J. Gibson. *The ecological approach to visual perception.* Houghton: Mifflin, 1979.

[56] G. Gigerenzer. 10 questions, life. *Max Planck institute magazine*, 2, 2008.

[57] G. Gigerenzer and D. G. Goldstein. Reasoning the fast and frugal way: Models of bounded rationality. *Psychological Review*, 103(4):650–669, 1996.

[58] Fausto Giunchiglia and Toby Walsh. Abstract theorem proving. In *IJCAI 1989*, pages 372–37, 1989.

[59] F. Gobet. A pattern-recognition theory of search in expert problem solving. *Thinking and Reasoning*, 3:291–333, 1997.

[60] T.L. Griffiths, M. Steyvers, and J. Tenenbaum. Topics in semantic representation. *Psychol Rev*, 2007.

[61] P. Groot. *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving.* PhD thesis, SIKS, Vrije Universiteit, Amsterdam, 2003.

[62] P. Groot, H. Stuckenschmidt, and H. Wache. Approximating Description Logic Classification for Semantice Web Reasoning. In A. Gómez-Pérez and J. Euzenat, editors, *Proceedings of the European Semantic Web Conference (ESWC)*, pages 318–332. Springer-Verlag, 2005.

[63] Perry Groot, Pascal Hitzler, Ian Horrocks, Boris Motik, Jeff Z. Pan, Heiner Stuckenschmidt, Daniele Turi, and HolgerWache. Methods for approximate reasoning. Technical Report D2.1.2, 2004.

[64] B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. 2003.

[65] U. Hahn and N. Chater. Similarity and rules: distinct? exhaustive? empirically distinguishable? *Cognition*, 65:197–230, 1999.

[66] G.S. Halford, W.H. Wilson, and S. Phillips. Processing capacity defined by relational complexity: Implications for comparative, developmental and cognitive psychology. *Behavioral and Brain Sciences*, 21:803–829, 1998.

[67] K.R. Hammond. *Human judgment and social policy.* New York: Oxford University Press, 1996.

[68] Pascal Hitzler and Denny Vrandecic. Resolution-based approximate reasoning for owl dl. In Y. Gil et al., editor, *Proceedings of the 4th International Semantic Web Conference (ISWC)*, number 3729 in Lecture Notes in Computer Science, pages 383–397, Galway, Ireland, 2005. Springer, Berlin.

[69] J.R. Hobbs. Granularity. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 432–435, 1985.

[70] K.J. Holyoak and J.K. Kroger. *Forms of reasoning. Insight into prefrontal functions? Structure and functions of the human prefrontal cortex.* New York: New York Academy of Sciences, 1995.

[71] A. Horn. On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic*, 16:14–21, 1951.

[72] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.

[73] Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI'05*, 2005.

[74] Z.S. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 454–459, 2005.

[75] U. Hustadt, B. Motik, and U. Sattler. Reducing shiq- description logic to disjunctive datalog programs. In *Proc. KR*, page 152?62, 2004.

[76] U. Hustadt, B. Motik, and U. Sattler. Reducing shiq? description logic to disjunctive datalog programs. In *Proc. KR 2004*, page 152162. AAAPress, 2004.

[77] P.N. Johnson-Laird. *Mental models, deductive reasoning and the brain*. Cambridge, MA: MIT Press, 1995.

[78] M.N. Jones and D. J. K. Mewhort. Representing word meaning and order information in a composite holographic lexicon. *Psychol Rev*, pages 1–37, 2007.

[79] M. Kaminski. A comparative study of open default theories. *Artificial Intelligence*, 77(2):285–319, 1995.

[80] Tim Kindberg, Matthew Chalmers, and Eric Paulos. Guest editors' introduction: Urban computing. *IEEE Pervasive Computing*, 6(3):18–20, 2007.

[81] W. Kintsch. *Comprehension: a paradigm for cognition*. Cambridge University Press, 1998.

[82] K. Konolige. Quantification in Autoepistemic Logic, 1991.

[83] M. Kr
"otzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In *Proceedings of the 7th International Semantic Web Conference (ISWC2008)*. Springer, 2008.

[84] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Conjunctive queries for a tractable fragment of owl 1.1. In *ISWC/ASWC*, pages 310–323, 2007.

[85] T. Landauer, D. McNamara, S. Dennis, and W. Kintsch. *LSA: A road to meaning*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc, 2007.

[86] T.K. Landauer and S.T. Dumais. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction,and representation of knowledge. 104:211–240, 1997.

[87] D. Lenat. The dimensions of context space. Technical report, 1999.

[88] V. Lifschitz. On open defaults. *Computational Logic: Symposium Proceedings*, pages 80–95, 1990.

[89] V. Lifschitz. Circumscription. *Handbook of Logic in Artificial Intelligence and Logic Programming*, 3:297–352, 1994.

[90] Vladimir Lifschitz. Computing circumscription. In *Readings in Nonmonotonic Reasoning*, pages 121–127. Morgan Kaufmann, 1985.

[91] Q. Liu and Q.Y. Wang. Granular logic with closeness relation λ and its reasoning. In *Lecture Notes in Computer Science*, volume 3641, pages 709–717, 2005.

[92] J.W. Lloyd. *Foundations of logic programming.* Springer-Verlag New York, Inc. New York, NY, USA, 1987.

[93] S. Makarios. *A Model Theory for a Quantified Generalized Logic of Contexts.* 2006.

[94] Pierre Marquis. Knowledge compilation using theory prime implicates. In *IJCAI*, pages 837–845, 1995.

[95] David A. McAllester. Truth maintenance. *AAAI*, pages 1109–1116, 1990.

[96] J. McCarthy. Notes on formalizing context. In *Proceedings of the thirteenth international Joint Conference in artificial intelligence*, pages 555–560, 1993.

[97] J.L. McCarthy. Applications of Circumscription to Formalizing Common-Sense Knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.

[98] John McCarthy. Circumscription: a form of non-monotonic reasoning. pages 555–566, 1995.

[99] R.S. Michalski and P.H. Winston. Variable precision logic. *Artificial Intelligence, Vol. 29*, pages 121–146, 1986.

[100] Marvin Minsky. A framework for representing knowledge. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[101] M.Oaksford and N. Chater. *Rationality In An Uncertain World: Essays on the Cognitive Science of Human Reasoning.* Psychology Press, 1998.

[102] R.C. Moore. Semantical Considerations on Nonmonotonic Logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[103] R.G. Morris, E.C. Moitto, J.D. Feigenbaum, P. Bullock, and C.E. Polkey. Planning ability after frontal and temporal lobe lesions: The effects of selection equivocation and working memory load. *Cognitive Neuropsychological*, 14:1007–1027, 1997.

[104] B. Motik, U. Sattler, and R. Studer. Query answering for owl-dl with rules. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3:41–60, 2005.

[105] T. Murai, G. Resconi, M. Nakata, and Y. Sato. Granular reasoning using zooming in & out: Propositional reasoning. In *Lecture Notes in Artificial Intelligence*, volume 2639, pages 421–424, 2003.

[106] T. Murai and Y. Sato. Granular reasoning using zooming in & out: Aristotle's categorical syllogism. *Electronic Notes in Theoretical Computer Science*, 82(4):186–197, 2003.

[107] P.P. Nayak and A.Levy. A semantic theory of abstractions. In C. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, page 196203. San Francisco, Morgan Kaufmann, 1995.

[108] A. Newell. *Unified theories of cognition*. Cambridge, Massachusetts: Harvard University Press, 1990.

[109] A. Newell and H.A. Simon. *Human problem solving*. Prentice-Hall, Englewood Cliffs, N.J.: Prentice-Hall, 1972.

[110] I. Niemelä and J. Rintanen. On the Impact of Stratification on the Complexity of Nonmonotonic Reasoning. *Lecture Notes In Computer Science; Vol. 810*, pages 275–295, 1994.

[111] M. Oaksford and N. Chater. A rational analysis of the selection task as optimal data selection. *Psychol Rev*, 101:608–631, 1994.

[112] J. Otero and W. Kintsch. Failures to detect contraditions in a text: what readers believe versus what they read. *Psycological Science*, 3:229–235, 1992.

[113] Jeff Z. Pan and Edward Thomas. Approximating OWL-DL Ontologies. In *the Proc. of the 22nd National Conference on Artificial Intelligence (AAAI-07)*, pages 1434–1439, 2007.

[114] Héctor Pérez-Urbina, Boris Motik, and Ian Horrocks. Rewriting conjunctive queries over description logic knowledge bases. In *SDKB*, pages 199–214, 2008.

[115] P. Pirolli, J. Pitkow, and R. Rao. System for predicting documents relevant to focus documents by spreading activation through network representations of a linked collection of documents. Technical report, 1998.

[116] M. Reckerand J. Pitkow. Predicting document access in large multimedia repositories. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3:352–375, 1996.

[117] E. Pothos. The rules versus similarity distinction. *Behavioral and Brain Sciences(in press)*.

[118] T.C. Przymusinski. Autoepistemic logic of knowledge and beliefs. *Artificial Intelligence*, 95(1):115–154, 1997.

[119] Jonathan Reades, Francesco Calabrese, Andres Sevtsuk, and Carlo Ratti. Cellular census: Explorations in urban data collection. *IEEE Pervasive Computing*, 6(3):30–38, 2007.

[120] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, pages 81–132, 1980.

[121] R. Reiter. On closed world data bases. In *Readings in nonmonotonic reasoning table of contents*, pages 300–310, 1987.

[122] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

[123] T. Rogers and K. Patterson. Object categorization: Reversals and explanations of the basic-level advantage. *Journal of Experimental Psychology: General*, 136(3):451–469, 2007.

[124] Sebastian Rudolph, Tuvshintur Tserendorj, and Pascal Hitzler. What is approximate reasoning? In *Proceedings of RR2008, LNCS 5341*, pages 150–164, 2008.

[125] D.E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT press, Cambridge, MA, 1986.

[126] D.E. Rumelhart, P. Smolensky, J. L. McClelland, and E. Hinton. Schemata and sequential thought processes in pdp models. MIT Press, Cambridge, MA, 1986.

[127] R.V.Guha. *Contexts: a formalization and some applications.* Stanford University PhD thesis report, 1991.

[128] E. Sandewall. An Approach to the Frame Problem, and its Implementation. *Proc. 2nd Congress on Medical Informatics, Europe*, 7:159–166.

[129] K. Satoh. Nonmonotonic reasoning by minimal belief revision. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, volume 2, pages 455–462, 1988.

[130] M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249310, 1995.

[131] M Schaerf and M Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.

[132] R.C. Schank. Conceptual dependency: A theory of natural language processing. *Cognitive Psychol*, 3:552–631, 1972.

[133] R.C. Schank. *Script, plans and Explanation patterns.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[134] J.S. Schlipf. How Uncomputable is General Circumscription? (Extended Abstract). *LICS*, pages 92–95, 1986.

[135] L.J. Schooler and R. Hertwig. How forgetting aids heuristic inference. *Psychol Rev*, 112:610–628, 2005.

[136] Bart Selman and Henry Kautz. Knowledge compilation using horn approximations. In *In Proceedings of AAAI-91*, pages 904–909. MIT Press, 1991.

[137] L. Serafini and A. Tamilin. Drago: Distributed reasoning architecture for semantic web. In *The Semantic Web: Research and Applications*, pages 361–376. Springer Berlin / Heidelberg, 2005.

[138] T. Shallice. *From neuropsychology to mental structure.* Cambridge: Cambridge University Press, 1988.

[139] D.R. Shanks and St. M.F. John. Characteristics of dissociable human learning-systems. *Behav Brain Sci*, 17:367–395, 1994.

[140] R.N. Shepard. *Mind sights: Original visual illusions, ambiguities, and other anomalies, with a commentary on the play of mind in perception and art.* New York: W. H. Freeman, 1990.

[141] H.A. Simon. Rational choice and the structure of the environment. *Psychological Review*, 63:129C138, 1956.

[142] H.A. Simon. Rationality as process and as product of thought. *American Economic Review*, 68:1C16, 1978.

[143] H.A. Simon. Invariants of human behavior. *Annual Review of Psychology*, 41:1C19, 1990.

[144] Evren Sirin and Bijan Parsia. Sparql-dl: Sparql query for owl-dl. In *the Proceedings of OWLED 2007*, 2007. `http://pellet.owldl.com/papers/sirin07sparqldl.pdf`.

[145] S.A. Sloman. *Psychol Bull*, 119:3–22, 1996.

[146] M.H. Sohn, A. Goode, V.A. Stenger, C.S. Carter, and J.R. Anderson. Competition and representation during memory retrieval: Roles of the prefrontal cortex and the posterior parietal cortex. *Proceedings of the National Academy of Sciences, U.S.A.*, 100:7412–7417, 2003.

[147] E. Stanovich and A. E. Cunningham. Reading as constrained reasoning. In J. R. Sternberg and P. A. Frensch, editors, *Complex problem Solving: principles and mechanisms*, pages 3–61. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.

[148] H. Stuckenschmidt. Modularization of ontologies - wonderweb: Ontology infrastructure for the semantic web, 2003.

[149] Heiner Stuckenschmidt. Toward multi-viewpoint reasoning with owl ontologies. In York-Sure and John Domingue, editors, *Proceedings of the European Semantic Web Conference (ESWC)*, volume 4011 of *LNCS*, pages 259–272. Springer-Verlag, 2006.

[150] Heiner Stuckenschmidt and Michel Klein. Integrity and change in modular ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence - IJCAI03*, pages 900–905. Morgan Kaufmann, 2003.

[151] Heiner Stuckenschmidt and Michel Klein. Structure-based partitioning of large concept hierarchies. In *In: International Semantic Web Conference*, pages 289–303, 2004.

[152] M. Truszczynski. Embedding default logic into modal. *Logic Programming and Non-Monotonic Reasoning: Proceedings of the First International Workshop*, 1991.

[153] J.D. Ullman. *Principles of Database Systems.* WH Freeman and Co. New York, NY, USA, 1983.

[154] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM (JACM)*, 38(3):619–649, 1991.

[155] Volker Tresp et al. Survey of literature on abstraction and learning. Technical report, 2008.

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

[157] P.C. Wason. Reasoning about a rule. *Quarterly Journal of Experimental Psychology*, 20:273–281, 1968.

[158] C. Wharton and W. Kintsch. An overview of construction-integration model: a theory of comprehension as a foundation for a new cognitive architecture. *ACM SIGART Bulletin*, 2:169–173, 1991.

[159] L. Yan and Q. Liu. Researches on granular reasoning based on granular space. In *Proceedings of the 2008 International Conference on Granular Computing*, volume 1, pages 706–711, 2008.

[160] F. Zeng, Q. Fu, and R. Morse. Human hearing enhanced by noise. *Brain Research*, 869:251–255, 2000.

[161] B. Zhou and Y.Y. Yao. A logic approach to granular computing. *The International Journal of Cognitive Informatics & Natural Intelligence*, 2(2):63–79, 2008.