



LarKC

The Large Knowledge Collider

a platform for large scale integrated reasoning and Web-search

FP7 – 215535

D4.3.1 Strategies and Design for Interleaving Reasoning and Selection of Axioms

Coordinator: [Zhisheng Huang (VUA)]

With contributions from: [Zhisheng Huang (VUA), Yi Zeng (WICI), Stefan Schlobach (VUA), Annette den Teije (VUA), Frank van Harmelen (VUA), Yang Wang (WICI), Ning Zhong (WICI)]

Quality Assessor: [Gulay Unel (STI)]

Quality Controller: [Frank van Harmelen (VUA)]

Document Identifier:	LarKC/2008/D4.3.1/V1.0
Class Deliverable:	LarKC EU-IST-2008-215535
Version:	version 1.0.0
Date:	September 29, 2009
State:	final
Distribution:	public



EXECUTIVE SUMMARY

In this document, we discuss the main features of Web scale reasoning and develop a framework of interleaving reasoning and selection. We examine the framework of interleaving reasoning and selection with the LarKC platform. The framework is explored further from the following three perspectives: i) Query-based selection. We propose various query-based strategies of interleaving selection and reasoning with respect to the LarKC data sets; ii) Granularity-based selection. We investigate the Web scale reasoning from the perspective of granular reasoning, and develop several strategies of Web scale reasoning with granularity; and iii) Language-based selection. We propose an approach of classification with anytime behaviours based on approximate reasoning and report the results of the experiments with several realistic ontologies.



DOCUMENT INFORMATION

IST Project Number	FP7 – 215535	Acronym	LarKC
Full Title	The Large Knowledge Collider: a platform for large scale integrated reasoning and Web-search		
Project URL	http://www.larkc.eu/		
Document URL			
EU Project Officer	Stefano Bertolo		

Deliverable	Number	4.3.1	Title	Strategies and Design for Interleaving Reasoning and Selection of Axioms
Work Package	Number	4	Title	Reasoning and Deciding

Date of Delivery	Contractual	M18	Actual	30-Sept-09
Status	version 1.0.0		final <input checked="" type="checkbox"/>	
Nature	prototype <input type="checkbox"/> report <input checked="" type="checkbox"/> dissemination <input type="checkbox"/>			
Dissemination Level	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			

Authors (Partner)	Zhisheng Huang (VUA), Yi Zeng (WICI), Stefan Schlobach (VUA), Annette den Teije (VUA), Frank van Harmelen (VUA), Yan Wang (WICI), Ning Zhong (WICI)			
Resp. Author	Zhisheng Huang (VUA)		E-mail	huang@cs.vu.nl
	Partner	WICI, MPG	Phone	+31 (20) 5987823

Abstract (for dissemination)	In this document, we discuss the main features of Web scale reasoning and develop a framework of interleaving reasoning and selection. We examine the framework of interleaving reasoning and selection with the LarKC platform. The framework is explored further from the following three perspectives: i) Query-based selection. We propose various query-based strategies of interleaving selection and reasoning with respect to the LarKC data sets; ii) Granularity-based selection. We investigate the Web scale reasoning from the perspective of granular reasoning, and develop several strategies of Web scale reasoning with granularity; and iii) Language-based selection. We propose an approach of classification with anytime behaviours based on approximate reasoning and report the results of the experiments with several realistic ontologies.
Keywords	Reasoning, Selection, Semantic Web, Web scale reasoning



PROJECT CONSORTIUM INFORMATION





Participant's name	Partner	Contact
Semantic Technology Institute Innsbruck, Universitaet Innsbruck	 	Prof. Dr. Dieter Fensel Semantic Technology Institute (STI), Universitaet Innsbruck, Innsbruck, Austria Email: dieter.fensel@sti-innsbruck.at
AstraZeneca AB		Bosse Andersson AstraZeneca Lund, Sweden Email: bo.h.andersson@astrazeneca.com
CEFRIEL - SOCIETA CONSORTILE A RESPONSABILITA LIMITATA		Emanuele Della Valle CEFRIEL - SOCIETA CONSORTILE A RE- SPONSABILITA LIMITATA Milano, Italy Email: emanuele.dellavalle@cefriel.it
CYCORP, RAZISKOVANJE IN EKSPERI- MENTALNI RAZVOJ D.O.O.		Michael Witbrock CYCORP, RAZISKOVANJE IN EKSPERI- MENTALNI RAZVOJ D.O.O., Ljubljana, Slovenia Email: witbrock@cyc.com
Höchstleistungsrechenzentrum, Universitaet Stuttgart		Georgina Gallizo Höchstleistungsrechenzentrum, Universitaet Stuttgart Stuttgart, Germany Email : gallizo@hlrs.de
MAX-PLANCK GESELLSCHAFT ZUR FOERDERUNG DER WISSENSCHAFTEN E.V.		Dr. Lael Schooler, Max-Planck-Institut für Bildungsforschung Berlin, Germany Email: schooler@mpib-berlin.mpg.de
Ontotext AD		Atanas Kiryakov, Ontotext Lab, Sofia, Bulgaria Email: naso@ontotext.com
SALTLUX INC.		Kono Kim SALTLUX INC Seoul, Korea Email: kono@saltlux.com
SIEMENS AKTIENGESELLSCHAFT		Dr. Volker Tresp SIEMENS AKTIENGESELLSCHAFT Muenchen, Germany Email: volker.tresp@siemens.com
THE UNIVERSITY OF SHEFFIELD		Prof. Dr. Hamish Cunningham, THE UNIVERSITY OF SHEFFIELD Sheffield, UK Email: h.cunningham@dcs.shef.ac.uk
VRIJE UNIVERSITEIT AMSTERDAM		Prof. Dr. Frank van Harmelen, VRIJE UNIVERSITEIT AMSTERDAM Amsterdam, Netherlands Email: Frank.van.Harmelen@cs.vu.nl
THE INTERNATIONAL WIC INSTI- TUTE, BEIJING UNIVERSITY OF TECHNOLOGY		Prof. Dr. Ning Zhong, THE INTERNATIONAL WIC INSTITUTE Mabeshi, Japan Email: zhong@maebashi-it.ac.jp
INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER		Dr. Paul Brennan, INTERNATIONAL AGENCY FOR RE- SEARCH ON CANCER Lyon, France Email: brennan@iarc.fr
INFORMATION RETRIEVAL FACILITY		Dr. John Tait, Dr. Paul Brennan, INFORMATION RETRIEVAL FACILITY Vienna, Austria Email: john.tait@ir-facility.org



TABLE OF CONTENTS

LIST OF FIGURES	7
LIST OF ACRONYMS	8
1 INTRODUCTION	9
2 A FRAMEWORK OF INTERLEAVING REASONING AND SELECTION	11
2.1 Web Scale Reasoning	11
2.2 Framework of Web Scale Reasoning by Interleaving Reasoning and Selection	12
3 QUERY-BASED SELECTION STRATEGIES	13
3.1 Selection Functions	13
3.2 Strategies	14
3.3 Relevance based Selection Functions	15
4 INTERLEAVING REASONING AND SELECTION IN THE LARKC PLATFORM	17
4.1 LarKC Platform	17
4.1.1 LarKC Architecture	17
4.1.2 LarKC Plug-ins	18
4.1.3 LarKC Selection Plug-ins	18
4.1.4 LarKC Reasoning Plug-ins	19
4.1.5 LarKC Decider Plug-ins	19
4.2 Interleaving Reasoning and Selection	20
4.3 Syntactic Relevance based Selection Functions	21
4.4 Semantic Relevance based Selection Functions	21
4.5 Strategies	24
4.5.1 Variant Strategies of Interleaving Reasoning and Selection	24
4.5.2 Strategies for Over-determined Processing	25
5 UNIFYING SEARCH AND REASONING FROM THE VIEWPOINT OF GRANULARITY	26
5.1 Introduction	26
5.2 Basic Notions	27
5.2.1 Knowledge Graph	27
5.2.2 Granule	28
5.2.3 Perspective	28
5.2.4 Level	29
5.3 Searching and Reasoning on a Knowledge Graph	29
5.4 Starting Point Strategy	30
5.5 Multilevel Completeness Strategy	31
5.6 Multilevel Specificity Strategy	32
5.7 Multiperspective Strategy	32
5.8 A Case Study on the Semantic Dataset	33
5.8.1 Multilevel Completeness Strategy	33
5.8.2 Starting Point Strategy	34
5.8.3 Multilevel Specificity Strategy	34



5.8.4	Multiperspective Strategy	36
5.9	Related Work	39
5.10	Conclusion and Future Work	40
6	ANYTIME CLASSIFICATION BY ONTOLOGY APPROXIMATION	42
6.1	INTRODUCTION	42
6.2	A SOUND APPROXIMATION FOR CLASSIFICATION	43
6.3	EXPERIMENTAL SETUP	45
6.3.1	Approximation Strategies	46
6.3.2	Datasets	46
6.3.3	Performance Measure	47
6.4	EXPERIMENTAL RESULTS	49
6.5	CONCLUSIONS	52
6.6	Proofs	53
7	CONCLUSION	60
	REFERENCES	60



LIST OF FIGURES

3.1	Linear Extension Strategy.	14
4.1	The LarKC Platform Architecture	17
5.1	Comparison of predicted and actual completeness value.	34
5.2	Normalized edge degree distribution in the SwetoDBLP RDF dataset.	37
5.3	Coauthor number distribution in the SwetoDBLP dataset.	38
5.4	log-log diagram of Figure 5.3.	38
5.5	A zoomed in version of Figure 5.3.	38
5.6	A zoomed in version of coauthor distribution for “Artificial Intelligence”.	38
5.7	Publication number distribution in the SwetoDBLP dataset.	38
5.8	log-log diagram of Figure 5.7.	38
6.1	anytime performance profile from examples 1 and 2	45
6.2	Some properties of the ontologies used in our experiments	47
6.3	Results for the MORE strategy on the DICE ontology	48
6.4	Summary of success and failure of the different strategies.	50
6.5	Interruptible behaviour	51
6.6	52



LIST OF ACRONYMS

Acronym	Description
DL	Description Logics
OWL	Web Ontology Language
PION	The System of Processing Inconsistent Ontologies
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SPARQL	SPARQL Protocol And RDF Query Language



1. Introduction

Web scale reasoning has become a crucial issue for practical applications of the Semantic Web, because of the extremely large scale data on the Web. Web scale semantic data have the following main features:

- **Infiniteness.** There are extremely large amount of semantic data on the Web. Till June 2009, Linked Data have reached the scale of above four billion triples. More linked data are expected to grow rapidly in coming few years. Therefore, Web scale data can be considered to be infinitely scalable.
- **Dynamics.** Web scale data are in flux. They are growing extremely rapidly, so that it is hard to know what the clear border of data is.
- **Inconsistency.** Re-using and combining multiple ontologies on the Web is bound to lead to inconsistencies between the combined vocabularies. Even many of the ontologies that are in use today turn out to be inconsistent once some of their implicit knowledge is made explicit. Therefore, infinitely scaled Web data tend to be semantically inconsistent. Moreover, consistency checking of Web scale data is impossible because of the infinite scale.

Because of those features of Web scale data, many traditional notions of reasoning are not valid any more. Essence of the LarKC project is to go beyond traditional notions of absolute correctness and completeness in reasoning. We are looking for retrieval methods that provide useful responses at a feasible cost of information acquisition and processing. Therefore, generic inference methods need to be extended to non-standard approaches.

In this document, we will explore an approach of Web scale reasoning, in which various strategies of interleaving reasoning and selection are developed, so that the reasoning processing can focus on limited part of data to improve the scalability of Web scale reasoning. This approach is inspired by our previous work on reasoning with inconsistent ontologies[19]. However, in this document we will develop a general framework of interleaving reasoning and selection, so that it can deal with not only reasoning with inconsistent ontologies, but also generic Web scale data.

Collins and Quillian observe that knowledge is stored as a system of propositions organized hierarchically in memory [11], in problem solving, human can focus on appropriate levels to avoid redundant information. Minsky remarks that in order to avoid the failure of understanding knowledge in one way, knowledge should be represented from different viewpoints [28]. As an emerging field of study, Granular Computing extracts the commonality of human and machine intelligence and emphasizes on multilevel and Multiperspective organization of granular structures [47].

In this document, we will develop various strategies under the notion of granular reasoning to solve the problems for Web scale reasoning. Inspired by Cognitive Science, Artificial Intelligence and Granular Computing, we bring the strategies of multilevel, multiperspective, starting point to Web scale reasoning. With user involvement, switching among different levels and perspectives during the process of reasoning is the basic philosophy of granular reasoning. From the multilevel point of view, in order to meet different levels of user needs, we can provide reasoning results with variable completeness and variable specificity. From the multiperspective point of view, reasoning can be based on different perspectives of the knowledge source.



Reasoning based on starting point utilizes the user background and provides most important reasoning results to users. These strategies is aimed at satisfying a wide variety of user needs and removing the scalability barriers.

Anytime algorithms are attractive for Web scale reasoning, because they allow a trade-off between the cost of the algorithm and the quality of the results. Such anytime algorithms have been developed for many AI reasoning tasks, such as planning, diagnosis and search. However, until now no anytime methods exist yet for subsumption-based classification in Description Logics. This is important among the other problems because classification is essential to Semantic Web applications, which require reasoning over large or complex ontologies.

In this document, we will present an algorithm for classification with anytime behaviour based on approximate subsumption. We give *formal definitions* for approximate subsumption, and show soundness and monotonicity. We develop *an algorithm and heuristics* to obtain anytime behaviour for classification reasoning. This anytime behaviour can be realised with classical DL reasoners. We study the computational behaviour of the algorithm on a set of realistic ontologies. Our *experiments* show attractive performance profiles. The most interesting finding is that anytime classification works best where it is most needed: on ontologies where classical subsumption is hardest to compute.

This document is organized as follows: In Chapter 2 we discuss the main features of Web scale reasoning and develop a framework of interleaving reasoning and selection. In Chapter 3 we explore the framework from the perspective of query-based selection. In Chapter 4 we examine the framework of interleaving reasoning and query-based selection with the LarKC platform and propose several strategies of interleaving selection and reasoning with respect to the LarKC data sets. In Chapter 5 we investigate the Web scale reasoning from the perspective of granular reasoning, and develop several strategies of Web scale reasoning with granularity. In Chapter 6 we propose an approach of classification with anytime behaviours based on approximate reasoning and report the results of the experiments with several realistic ontologies. In Chapter 7 we discuss the future work and conclude the document.



2. A Framework of Interleaving Reasoning and Selection

2.1 Web Scale Reasoning

Web scale reasoning is reasoning with Web scale semantic data. As we discussed before, The main features of Web scale semantic data are: i) Infiniteness. There are extremely large amount of semantic data on the Web. They can be considered to be infinitely scalable. ii) Dynamics. Web data are in flux. There is no a clear border of data, and iii) Inconsistency. It is most likely that so large amount of data are semantically inconsistent. However, consistency checking of Web scale data is impossible.

Those features of Web scale data force us to re-examine the traditional notion of reasoning. The classical notion of reasoning is to consider the consequence relation between a knowledge base (i.e. a formula set Σ) and a conclusion (i.e., a formula ϕ), which is defined as follows:

$$\Sigma \models \phi \text{ iff for any model } M \text{ of } \Sigma, M \text{ is a model of } \phi.$$

For Web scale reasoning, Knowledge base Σ can be considered as an infinite formula set. However, when the cardinality $|\Sigma|$ of the knowledge base Σ becomes infinite and Σ is inconsistent, many notions of logic and reasoning in classical logics, including many existing description logics, which are considered to be standard logics for ontology reasoning and the Semantic web, are not valid any more.

It is worthy to mention that classical logics do not limit the cardinality of their knowledge bases to be finite, because the compactness theorem in classical logics would help them to deal with the infiniteness.

The Compactness theorem states that:

(CT) a (possibly infinite) set of first-order formulas has a model, iff every finite subset of it has a model,

Or conversely:

(CT') a (possibly infinite) set of formulas doesn't have a model if there exists its finite subset that doesn't have a model.

That means that given an infinite set of formulas Σ and a formula ϕ , if we can find a finite subset $\Sigma' \subseteq \Sigma$ such that $\Sigma' \cup \{\neg\phi\}$ is unsatisfiable (namely, there exists no model to make the formula set holds), it is sufficient to conclude that ϕ is a conclusion of the infinite Σ . In other words, the compactness theorem means that in the formalisms based on FOL we can positively answer the problems of the form $\Sigma \models \phi$, by showing that $\Sigma \cup \{\neg\phi\} \models$ contradiction. Thus, we have chances to show (even if Σ is infinite) if we are able to identify a finite subset of Σ (call it Σ') such that $\Sigma' \cup \{\neg\phi\} \models$ contradiction.

However, we would like to point out that the compactness theorem would not help for Web scale reasoning because of the following reason.

For Web scale data, Knowledge base Σ may be inconsistent. Now, consider the problem to answer the form $\Sigma \models \phi$ where Σ is inconsistent. When Σ is inconsistent, a finite subset of Σ (call it Σ') such that $\Sigma' \cup \{\neg\phi\} \models$ contradiction would not be sufficient to lead to a conclusion that $\Sigma \models \phi$, because there might exist another subset of Σ (call it Σ'') such that $\Sigma'' \cup \{\phi\} \models$ contradiction.



If we re-examine the classical notions of the complexity in the setting of Web scale reasoning, many those of the notions would also become meaningless. Just take the example of the complexity of finding the answer problems of the form $\Sigma \models \phi$. Consider a polynomial complexity with respect to the complexity of knowledge base Σ , say, a linear complexity $O(|\Sigma|)$. When Σ becomes infinite, a linear complexity would become intractable.

2.2 Framework of Web Scale Reasoning by Interleaving Reasoning and Selection

A way out to solve the infiniteness and inconsistency problems of Web scale reasoning is to introduce a selection procedure so that our reasoning processing can focus on a limited (but meaningful) part of the infinite data. That is the motivation for developing the framework of Web scale reasoning by interleaving reasoning and selection.

Therefore, the procedure of Web scale reasoning by interleaving reasoning and selection consists of the following *selection-reasoning-decision-loop*:

Algorithm 2.1: Selection-Reasoning-Loop

repeat

Selection: Select a (consistent) subset $\Sigma' \subseteq \Sigma$

Reasoning: Reasoning with $\Sigma' \models \phi$ to get answers

Decision: Deciding whether or not to stop the processing

until Answers are returned.

Namely, the framework depends on the following crucial processes: i) How can we select a subset of a knowledge base and check the consistency of selected data, ii) How can we reason with selected data, iii) how can we make the decision whether or not the processing should be stop. That usually depends on the problem how we can evaluate the answer obtained from the process ii), Our framework is inspired by our previous work in reasoning with inconsistent ontologies[19]. Since Web scale data may be inconsistent, we can apply the same framework to deal with the problem of Web scale reasoning.

In the following, we will explore the framework further from the following three perspectives: i) Query-based selection. We propose various query-based strategies of interleaving selection and reasoning; ii) Granularity-based selection. We investigate the Web scale reasoning from the perspective of granular reasoning, and develop several selection strategies of Web scale reasoning with granularity; and iii) Language-based selection. We propose an approach of classification with anytime behaviours based on sub-language selection and report the results of the experiments with several realistic ontologies.



3. Query-based Selection Strategies

3.1 Selection Functions

Selection functions play the main role in the framework of interleaving reasoning and query-based selection. A system of interleaving reasoning and query-based selection uses a selection function to determine which subsets of a knowledge base should be considered in its reasoning process. This general framework is independent of the particular choice of selection function. The selection function can either be based on a syntactic approach, like Chopra, Parikh, and Wassermann’s syntactic relevance [8] and those in PION[19], or based on semantic relevance like for example in computational linguistics as in Wordnet [7] or based on semantic relevance which is measure by the co-occurrence of concepts in search engines like Google[22].

In our framework, selection functions are designed to query-specific, which is different from the traditional approach in belief revision and nonmonotonic reasoning, which assumes that there exists a general preference ordering on formulas for selection. Given a knowledge base Σ and a query ϕ , a selection function s is one which returns a subset of Σ at the step $k > 0$. Let \mathbf{L} be the ontology language, which is denoted as a formula set. A selection function s is a mapping $s : \mathcal{P}(\mathbf{L}) \times \mathbf{L} \times N \rightarrow \mathcal{P}(\mathbf{L})$ such that $s(\Sigma, \phi, k) \subseteq \Sigma$.

A selection function s is called *monotonic* if the subsets it selects monotonically increase or decrease, i.e., $s(\Sigma, \phi, k) \subseteq s(\Sigma, \phi, k + 1)$, or vice versa. For monotonically increasing selection functions, the initial set is either an emptyset, i.e., $s(\Sigma, \phi, 0) = \emptyset$, or a fixed set Σ_0 . For monotonically decreasing selection functions, usually the initial set $s(\Sigma, \phi, 0) = \Sigma$. The decreasing selection functions will reduce some formulas from the inconsistent set step by step until they find a maximally consistent set.

Traditional reasoning methods cannot be used to handle knowledge bases with large scale. Hence, selecting and reasoning on subsets of Σ may be appropriate as an approximation approach with monotonically increasing selection functions. Web scale reasoning on a knowledge base Σ can use different selection strategies to achieve this goal. Generally, they all follow an iterative procedure which consists of the following processing loop, based on the selection-reasoning-decision loop discussed above:

- i) select part of the knowledge base, i.e., find a subset Σ'_i of Σ where i is a positive integer, i.e., $i \in I^+$;
- ii) apply the standard reasoning to check if $\Sigma'_i \models \phi$;
- iii) decide whether or not to stop the reasoning procedure or continue the reasoning with gradually increased selected subgraph of the knowledge graph (Hence, $\Sigma'_1 \subseteq \Sigma'_2 \subseteq \dots \subseteq \Sigma$).

Monotonically increasing selection functions have the advantage that they do not have to return *all* subsets for consideration at the same time. If a query can be answered after considering some consistent subset of the knowledge graph KG for some value of k , then other subsets (for higher values of k) don’t have to be considered any more, because they will not change the answer of the reasoner. In the following, we use $\Sigma \models \phi$ to denote that ϕ is a consequence of Σ in the standard reasoning¹, and use $\Sigma \approx \phi$ to denote that ϕ is a consequence of Σ in the nonstandard reasoning.

¹Namely, for any model M of Σ , $M \models \phi$.

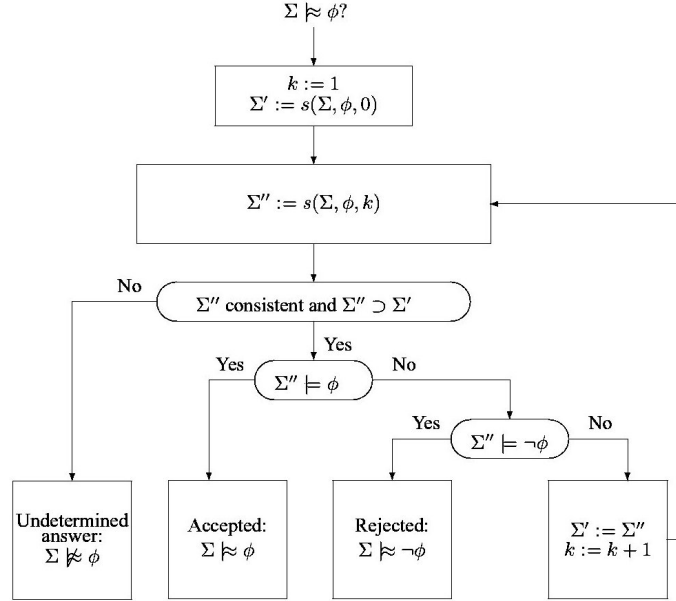


Figure 3.1: Linear Extension Strategy.

3.2 Strategies

A linear extension strategy is carried out as shown in Figure 3.1. Given a query $\Sigma \approx \phi$, the initial consistent subset Σ' is set. Then the selection function is called to return a consistent subset Σ'' , which extends Σ' , i.e., $\Sigma' \subset \Sigma'' \subseteq \Sigma$ for the linear extension strategy. If the selection function cannot find a consistent superset of Σ' , the inconsistency reasoner returns the answer ‘undetermined’ (i.e., unknown) to the query. If the set Σ'' exists, a classical reasoner is used to check if $\Sigma'' \models \phi$ holds. If the answer is ‘yes’, the reasoner returns the ‘accepted’ answer $\Sigma \approx \phi$. If the answer is ‘no’, the reasoner further checks the negation of the query $\Sigma'' \models \neg\phi$. If the answer is ‘yes’, the reasoner returns the ‘rejected’ answer $\Sigma \approx \neg\phi$, otherwise the current result is undetermined, and the whole process is repeated by calling the selection function for the next consistent subset of Σ which extends Σ'' .

It is clear that the linear extension strategy may result in too many ‘undetermined’ answers to queries when the selection function picks the wrong sequence of monotonically increasing subsets. It would therefore be useful to measure the successfulness of (linear) extension strategies. Notice, that this depends on the choice of the monotonic selection function.

In general, one should use an extension strategy that is not over-determined (i.e., the selected set is inconsistent) and not undetermined. For the linear extension strategy, we can prove that a reasoner using a linear extension strategy may be undetermined, always sound, and always meaningful[20]. A reasoner using a linear extension strategy is useful to create meaningful and sound answers to queries. The advantages of the linear strategy is that the reasoner can always focus on the current working set Σ^2 . The reasoner doesn’t need to keep track of the extension chain. The disadvantage of the linear strategy is that it may lead to an inconsistency reasoner that is undetermined. There exists other strategies which can improve the linear extension approach,

²Alternatively it is called the selected set.



for example, by backtracking and heuristics evaluation. We will discuss how it can be achieved in the over-determined processing in Section *Over-determined Processing*.

3.3 Relevance based Selection Functions

[8] proposes a syntactic relevance to measure the relationship between two formulas in belief sets, so that the relevance can be used to guide the belief revision based on Schaerf and Cadoli's method of approximate reasoning[34]. Given a formula set Σ , two atoms p, q are directly relevant, denoted by $R(p, q, \Sigma)$ iff there is a formula $\alpha \in \Sigma$ such that p, q appear in α . A pair of atoms p and q are k -relevant with respect to Σ iff there exist $p_1, p_2, \dots, p_k \in \mathcal{L}$ such that: (a) p, p_1 are directly relevant; (b) p_i, p_{i+1} are directly relevant, $i = 1, \dots, k - 1$; and (c) p_k, q are directly relevant (i.e., directly relevant is k -relevant for $k = 0$).

The notions of relevance above are based on propositional logics. However, ontology languages are usually written in some fragment of the first order logic. We extend the ideas of relevance to ontology language. The Direct relevance between two formulas are defined as a binary relation on formulas, namely $\mathcal{R} \subseteq \mathbf{L} \times \mathbf{L}$. Given a direct relevance relation \mathcal{R} , we can extend it to a relation \mathcal{R}^+ on a formula and a formula set, i.e., $\mathcal{R}^+ \subseteq \mathbf{L} \times \mathcal{P}(\mathbf{L})$ as follows:

$$\langle \phi, \Sigma \rangle \in \mathcal{R}^+ \text{ iff } \exists \psi \in \Sigma \text{ such that } \langle \phi, \psi \rangle \in \mathcal{R}.$$

Namely, a formula ϕ is relevant to a knowledge base Σ iff there exists a formula $\phi' \in \Sigma$ such that ϕ and ϕ' are directly relevant. We can similarly specialize the notion of k -relevance. Two formulas ϕ, ϕ' are k -relevant with respect to a formula Σ iff there exist formulas $\phi_0, \dots, \phi_k \in \Sigma$ such that ϕ and ϕ_0 , ϕ_0 and ϕ_1 , \dots , and ϕ_k and ϕ' are directly relevant. A formula ϕ is k -relevant to a set Σ iff there exists a formula $\phi' \in \Sigma$ such that ϕ and ϕ' are k -relevant with respect to Σ .

We can use a relevance relation to define a selection function s to extend the query ' $\Sigma \approx \phi$?' as follows: We start with the query formula ϕ as a starting point for the selection based on syntactic relevance. Namely, we define:

$$s(\Sigma, \phi, 0) = \emptyset.$$

Then the selection function selects the formulas $\psi \in \Sigma$ which are directly relevant to ϕ as a working set (i.e. $k = 1$) to see whether or not they are sufficient to give an answer to the query. Namely, we define:

$$s(\Sigma, \phi, 1) = \{\psi \in \Sigma \mid \phi \text{ and } \psi \text{ are directly relevant}\}.$$

If the reasoning process can obtain an answer to the query, it stops. Otherwise the selection function increases the relevance degree by 1, thereby adding more formulas that are relevant to the current working set. Namely, we have:

$$s(\Sigma, \phi, k) = \{\psi \in \Sigma \mid \psi \text{ is directly relevant to } s(\Sigma, \phi, k - 1)\},$$

for $k > 1$. This leads to a "fan out" behavior of the selection function: the first selection is the set of all formulae that are directly relevant to the query; then all formulae are selected that are directly relevant to that set, etc. This intuition is formalized in this:



The relevance-based selection function s is monotonically increasing. We observe that if $k \geq 1$, then

$$s(\Sigma, \phi, k) = \{\phi \mid \phi \text{ is } (k-1)\text{-relevant to } \Sigma\}$$

The relevance-based selection functions defined above usually grows up to an inconsistent set rapidly. That may lead to too many undetermined answers. In order to improve it, we can require that the selection function returns a consistent subset Σ'' at the step k when $s(\Sigma, \phi, k)$ is inconsistent such that $s(\Sigma, \phi, k-1) \subset \Sigma'' \subset s(\Sigma, \phi, k)$. It is actually a kind of backtracking strategies which are used to reduce the number of undetermined answers to improve the linear extension strategy. We call the procedure an *over-determined processing*(ODP) of the selection function. Note that the over-determined processing does not need to exhaust the powerset of the set $s(\Sigma, \phi, k) - s(\Sigma, \phi, k-1)$, because of the fact that if a consistent set S cannot prove or disprove a query, then nor can any subset of S . Therefore, one approach of ODP is to return just a maximally consistent subset. Let n be $|\Sigma|$ and k be $n - |S|$, i.e., the cardinality difference between the ontology Σ and its maximal consistent subset S (note that k is usually very small), and let C be the complexity of the consistency checking. The complexity of the over-determined processing is polynomial to the complexity of the consistency checking. Note that ODP introduces a degree of non-determinism: selecting different maximal consistent subsets of $s(\Sigma, \phi, k)$ may yield different answers to the query $\Sigma \approx \phi$. The simplest example of this is $\Sigma = \{\phi, \neg\phi\}$.

4. Interleaving Reasoning and Selection in the LarKC Platform

4.1 LarKC Platform

4.1.1 LarKC Architecture

In this document, we consider the LarKC architecture which has been proposed in [43]. Figure 4.1 shows a detailed view of the LarKC Platform architecture.

The LarKC platform has been designed in a way so that it is as lightweight as possible, but must provide all necessary features to support both users and plug-ins. For this purpose, the following components are distinguished as part of the LarKC platform:

- Plug-in API: it defines interfaces for required behaviour from plug-in and therefore provides support for interoperability between platform and plug-ins and between plug-ins.
- Data Layer API: the Data Layer provides support for data access and management via its API.
- Plug-in Registry: it contains all necessary features for plug-in registration and discovery
- Pipeline Support System: it provides support for plug-in instantiation, through the deployment of plug-in managers, and for monitoring and controlling plug-in execution at pipeline level.

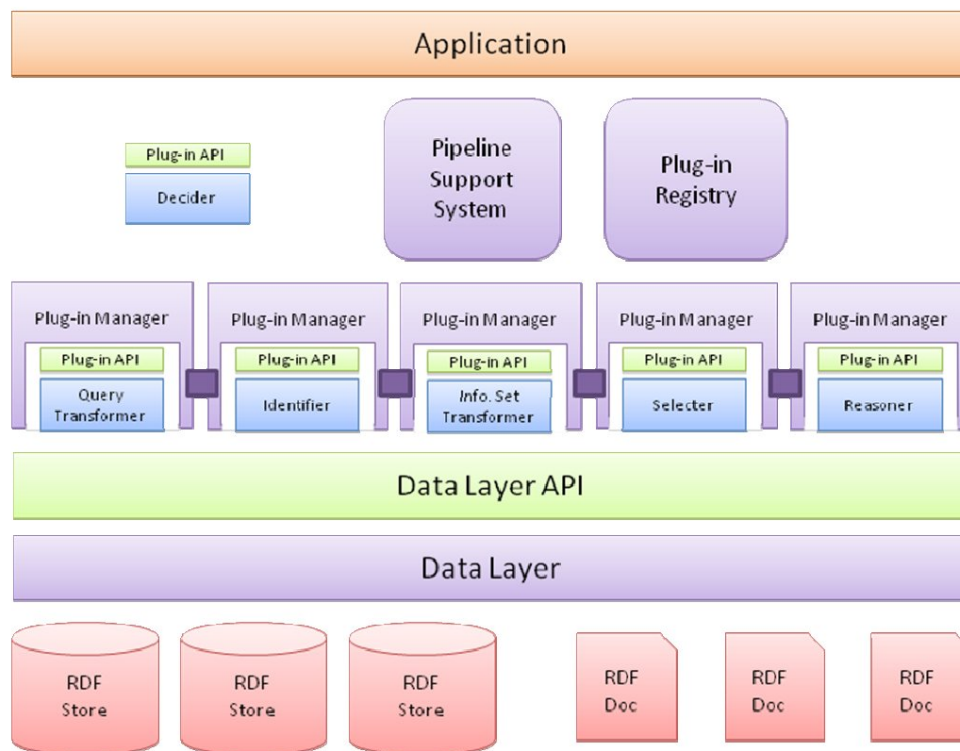


Figure 4.1: The LarKC Platform Architecture



- Plug-in Managers: provide support for monitoring and controlling plug-ins execution, at plugin level. An independent instance of Plug-in Manager is deployed for each plug-in to be executed. This component includes the support for both local and remote deployment and management of plug-ins.
- Queues: provide support for deployment and management of the communication pipes between platform and plug-ins and between plug-ins.

4.1.2 LarKC Plug-ins

All LarKC plug-ins share a common super class, which is the Plugin class. This class provides that functionality which is common to all plug-in types. The interface of the Plugin class can be seen below:

```
public interface Plugin
{
    public String getIdentifier();
    public MetaData getMetaData();
    public QoSInformation getQoSInformation();
    public void setInputQuery(Query theQuery);
}
```

Namely, all plug-ins are identified by a name, which is a string. Plug-ins provide meta data that describes the functionality that they offer. Plug-ins provide Quality of Service (QoS) information regarding how they perform the functionality that they offer. All plug-ins may need access to the initial query (entry query in the LarKC platform) and thus a mutator is provided by specifying this query.

4.1.3 LarKC Selection Plug-ins

The LarKC Selection plug-in are used for taking a selection (or a sample) of the Data Set that has been made available by some previous processes, say an identify plug-in on which reasoning should be performed. The output of a Selection plug-in is a Triple Set, which is essentially a subset of the input Data Set.

The interface of the Selection Plug-in is:

```
public interface Selector extends Plugin
{
    public SetOfStatements select(SetOfStatements theSetOfStatements, Contract contract, Context context);
}
```

Thus the Select plug-in takes a set of statement as input, identifies a selection from this dataset according to its strategy and returns a set of statements according to the contract and the context. The contract is used to define the dimensions of the output. The context defines the special information of the reasoning task. An example of a Selection plug-in would be one that extracts a particular number of triples from each of the RDF graphs within the Data Set to build the Triple Set. The Contract in this case would define the number of triples to be present in the output Triple Set, or the number of triples to extract from the each of the RDF graphs in the Data Set.



Selection plug-in is not necessarily independent of the user query. All plug-ins have a method to accept the user query and this is passed as part of pipeline construction. The query is known beforehand, so there is no need to pass this query to the selector upon every invocation.

4.1.4 LarKC Reasoning Plug-ins

The reasoning plug-in executes a given SPARQL Query against a Triple Set provided by a Selection plug-in. The interface of the reasoning plug-in can be seen below:

```
public interface Reasoner extends Plugin {
    public VariableBinding sparqlSelect(SPARQLQuery theQuery, SetOfStatements
statements, Contract contract, Context context);
    public RdfGraph sparqlConstruct(SPARQLQuery theQuery, SetOfStatements state-
ments, Contract contract, Context context);
    public RdfGraph sparqlDescribe(SPARQLQuery theQuery, SetOfStatements state-
ments, Contract contract, Context context);
    public BooleanInformationSet sparqlAsk(SPARQLQuery theQuery, SetOfStatements
statements, Contract contract, Context context);
}
```

The reasoning plug-in supports the four standard methods for a SPARQL endpoint, namely select, describe, construct and ask. The input to each of the reason methods are the same and includes the query to be executed, the statement set to reason over, the contract, which defines the behavior of the reasoner, and the context, which defines the special information of the reasoning task. The output of these reasoning methods depends on the reasoning task being performed. The select method returns a Variable Binding as output where the variables correspond to those specified in the query. The construct and describe methods return RDF graphs, in the first case this graph is constructed according to the query and in the second the graph contains triples that describe the variable specified in the query. Finally ask returns a Boolean Information Set as output, which is true if the pattern in the query can be found in the Triple Set or false if not.

4.1.5 LarKC Decider Plug-ins

The Decider plug-in is responsible for building and maintaining the pipeline containing the other plug-in types, and managing the control flow between these plug-ins.

The interface of this plug-in can be seen below:

```
public interface Decider extends Plugin {
    public VariableBinding sparqlSelect(SPARQLQuery theQuery, QoSParameters the-
QoSParameters);
    public SetOfStatements sparqlConstruct(SPARQLQuery theQuery, QoSParame-
ters theQoSParameters);
    public SetOfStatements sparqlDescribe(SPARQLQuery theQuery, QoSParameters
theQoSParameters);
    public BooleanInformationSet sparqlAsk(SPARQLQuery theQuery, QoSParame-
ters theQoSParameters);
}
```



}

The interface of the Decider plug-in is very similar to that of the reasoning plug-in. The major difference is that actual data to reason over is not explicitly specified, as the Identify plug-in is responsible for finding the data within the pipeline.

4.2 Interleaving Reasoning and Selection

In the following, we examine a framework of interleaving reasoning and selection in the setting of the LarKC platform. We will propose several selection functions which are based on the LarKC data model, namely, in which a knowledge base is considered to be triple sets.

In the LarKC platform, ontology data are represented as a SetofStatements. Namely, they are a set of RDF statements. We can consider a RDF statement as a triple. Thus, conceptually, ontology data can be considered as a set of triples. Alternatively, it is called a triple set. A triple t has the form $\langle s, p, o \rangle$ where s is called a subject, p is called a predicate, and o is called an object of the triple.

For OWL ontology data, we usually use the popular DL reasoners such as Racer, FACT++, KAON2, and Pellet to obtain the standard DL reasoner support if OWL ontology data is consistent and can be handled by those DL reasoners. All of those popular DL reasoners provide the DIG interface. Thus, those popular DL reasoners can serve as an external reasoner which can be called within the LarKC platform via its DIG interface reasoner plugin. Furthermore, OWL APIs provide OWL-DL/OWL2 reasoning interface, which is considered as a new generation and updated DIG interface. Thus, in the LarKC platform, we will use the OWL APIs reasoner plug-in for reasoning with OWL ontology data. In the following, we will use \models to denote the standard DL reasoning.

In the LarKC platform, a query of a reasoner is represented as a SPARQL query, like `sparqlAsk` and `sparqlSelect`, which have been discussed in the previous section. The SPARQL query language is designed for querying with RDF data originally. Thus, it is too powerful for a DL-based reasoner for reasoning with OWL ontology data. Thus, we will consider only the limited part of the SPARQL language, which is called as SPARQL-DL, i.e., the part of the SPARQL which corresponds with DL expressions semantically. SPARQL-DL can be considered as a special case of conjunctive queries for DL, which provide a facility for databaselike querying with DL data. In the following, we will use a formula ϕ to denote a SPARQL-DL query. Semantically, a formula ϕ corresponds to be a set of triples which is implied by the formula. Namely,

$$[[\phi]] = \{t : \phi \models t\}$$

A `SparqlAsk` query corresponds with a query formula ϕ in which there are no free variables. A `SparqlSelect` query corresponds with a query formula ϕ in which there are some free variables. For a triple set Σ and a query ϕ , we use $\Sigma \models \phi$ to denote that $\Sigma \models t$ for all $t \in [[\phi]]$.

Various selection functions can be defined in the LarKC platform. In the following, we will propose several selection functions for the processing of interleaving reasoning and selection.



4.3 Syntactic Relevance based Selection Functions

Syntactic relevance means that the relevance is measured with respect to symbolic appearance of two triple sets without considering the semantics of the symbols. However, we should ignore the predicates of triples when two triples are examined with respect to their relevance, because many predicates such as "rdf:type" and "rdfs:subClassOf" appear frequently in a RDF/RDFS data set, which suggests nothing on the relevance of two triples. Thus, we can define a syntactic relevance relation *Syn* on two triples as follows.

For any triple $t_1 = \langle s_1, p_1, o_1 \rangle$ and any triple $t_2 = \langle s_2, p_2, o_2 \rangle$,

$$\langle t_1, t_2 \rangle \in Syn \text{ iff } s_1 = s_2 \text{ or } s_1 = o_2 \text{ or } s_2 = o_1 \text{ or } o_1 = o_2.$$

Thus, we can extend this relevance measure to the relevance measure between a triple and a triple set Σ as discussed in the previous chapter. Namely, a triple t is said to be relevant with a triple set Σ if there exists a triple $t' \in \Sigma$ such as t and t' are relevant (with respect to the relation *Syn*). Furthermore, we can define a relevant subset of a triple set Σ with respect to a relevance relation *Syn* and a triple set Σ' , written $Syn(\Sigma, \Sigma')$ as follows:

$Syn(\Sigma, \Sigma') = \{t \in \Sigma : t \text{ is relevant with } \Sigma' \text{ with respect to the relevance relation } Syn.\}$

Now, we can define a selection function s with respect to a relevance relation *Syn* as follows:

- (i) $s(\Sigma, \phi, 0) = \emptyset$;
- (ii) $s(\Sigma, \phi, 1) = Syn(\Sigma, [[\phi]])$;
- (iii) $s(\Sigma, \phi, k) = Syn(\Sigma, s(\Sigma, \phi, k - 1))$, for $k > 1$.

Furthermore, we can define a syntactic relevance relation Syn_C which considers only the relevance with concepts as follows:

For any triple $t_1 = \langle s_1, p_1, o_1 \rangle$ and any triple $t_2 = \langle s_2, p_2, o_2 \rangle$,

$$\langle t_1, t_2 \rangle \in Syn_C \text{ iff } \langle t_1, t_2 \rangle \in Syn \text{ and } ((p_1 = \text{"rdfs:subClassOf"} \text{ and } o_1 \neq \text{"owl:Thing"} \text{ and } s_1 \neq \text{"owl:Nothing"}) \text{ or } (p_2 = \text{"rdfs:subClassOf"} \text{ and } o_2 \neq \text{"owl:Thing"} \text{ and } s_2 \neq \text{"owl:Nothing"})).$$

In the concept relevance measure above, we consider the triples which state the *subClassOf* relation and ignore their relevance of the trivial *subClassOf* relation via the top concept and the bottom concept.

4.4 Semantic Relevance based Selection Functions

The syntactic relevance-based selection functions prefer shorter paths to longer paths in the reasoning. It requires knowledge engineers should carefully design ontologies to avoid unbalanced reasoning path. Naturally we will consider semantic relevance based selection functions as alternatives of syntactic relevance based selection functions. In [22] we propose a semantic relevance based section function that is developed based on Google distances. Namely, we want to take advantage of the vast knowledge on



the web by using Google based relevance measure, by which we can obtain light-weight semantics for selection functions. The basic assumption here is that: more frequently two concepts appear in the same web page, more semantically relevant they are, because most of web pages are meaningful texts. Therefore, information provided by a search engine can be used for the measurement of semantic relevance among concepts. We select Google as the targeted search engine, because it is the most popular search engine nowadays. The second reason why we select Google is that Google distances are well studied in [10, 9].

In [10, 9], Google Distances are used to measure the co-occurrence of two keywords over the Web. Normalized Google Distance (NGD) is introduced to measure semantic distance between two concepts by the following definition:

$$NGD(x, y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x, y)}{\log M - \min\{\log f(x), \log f(y)\}}$$

where

$f(x)$ is the number of Google hits for the search term x ,

$f(y)$ is the number of Google hits for the search term y ,

$f(x, y)$ is the number of Google hits for the tuple of search terms x and y , and,

M is the number of web pages indexed by Google.

$NGD(x, y)$ can be understood intuitively as a measure for the symmetric conditional probability of co-occurrence of the search terms x and y .

$NGD(x, y)$ takes a real number between 0 and 1. $NGD(x, x) = 0$ means that any search item is always the closest to itself. $NGD(x, y)$ is defined for two search items x and y , which measures the semantic dissimilarity, alternatively called *semantic distance*, between them.

The semantic relevance is considered as a reverse relation of the semantic dissimilarity. Namely, more semantically relevant two concepts are, smaller distance between them. Mathematically this relation can be formalized by the following equation if the similarity measurement and the distance measurement take a real number between 0 and 1.

$$Similarity(x, y) = 1 - Distance(x, y).$$

In the following we use the terminologies *semantic dissimilarity* and *semantic distance* interchangeably. To use NGD for reasoning with inconsistent ontologies, we extend this dissimilarity measure on two triples in terms of the dissimilarity measure on the distances between two concepts/roles/individuals from the two triples. Moreover, in the following we consider only concept names $C(t)$ as the symbol set of a triple t to simplify the formal definitions. However, note that the definitions can be easily generalized into ones in which the symbol sets contain roles and individuals. We use $SD(t_1, t_2)$ to denote the semantic distance between two triples. We expect semantic distances between two formulas $SD(t_1, t_2)$ satisfying the following intuitive properties:

- (i) **(Range)** The semantic distances are real numbers between 0 and 1. Namely, $0 \leq SD(t_1, t_2) \leq 1$ for any t_1 and t_2 .
- (ii) **(Reflexivity)** Any triple is always semantically closest to itself. Namely, $SD(t, t) = 0$ for any t .



- (iii) **(Symmetry)** The semantic distances between two triples are symmetric. Namely, $SD(t_1, t_2) = SD(t_2, t_1)$ for any t_1 and t_2 .
- (iv) **(Remoteness)** If all symbols in a triple is semantically most-dissimilar from any symbol of another triple, then these two triples are totally semantic-dissimilar. Namely, if $NGD(C_i, C_j) = 1$ for all $C_i \in C(t_1)$ and $C_j \in C(t_2)$, then $SD(t_1, t_2) = 1$.
- (v) **(Intermediary)** If there are some shared symbols which appear in both triples and some symbols are semantically dissimilar between two triples, then the semantic distance between two triples are neither the closest, nor are the most dissimilar. Namely, if $C(t_1) \cap C(t_2) \neq \emptyset$ and $C(t_1) \neq C(t_2)$, then $0 < SD(t_1, t_2) < 1$.

However, note that the semantic distance does not always satisfy the triangle Inequality

$$SD(t_1, t_2) + SD(t_2, t_3) \geq SD(t_1, t_3),$$

a basic property of distances in a metric topology. [25] provides a counter-example of the Triangle Inequality in semantic similarity measure.

Simple ways to define the semantic distance between two triples is to take the minimal or the maximal or the average NGD values between two concepts/roles/individuals which appear in two triples as follows:

$$\begin{aligned} SD_{min}(t_1, t_2) &= \min\{NGD(C_1, C_2) | C_1 \in C(t_1) \\ &\quad \text{and } C_2 \in C(t_2)\} \\ SD_{max}(t_1, t_2) &= \max\{NGD(C_1, C_2) | C_1 \in C(t_1) \\ &\quad \text{and } C_2 \in C(t_2)\}. \\ SD_{ave}(t_1, t_2) &= \text{sum}\{NGD(C_1, C_2) | C_1 \in C(t_1) \\ &\quad \text{and } C_2 \in C(t_2)\} / (|C(t_1)| * |C(t_2)|) \end{aligned}$$

where $|C(t)|$ means the cardinality of $C(t)$. However, it is easy to see that SD_{min} and SD_{max} do not satisfy the property (v)Intermediary, and SD_{ave} do not satisfy the properties (ii) Reflexivity and (iv) Remoteness.

In the following, we propose a semantic distance which is measured by the ratio of the distance sum of the difference between two formulas to the total distance sum of the symbols between two triples.

Definition 1 (Semantic Distance between two triples)

$$SD(t_1, t_2) = \frac{\text{sum}\{NGD(C_i, C_j) | C_i, C_j \in (C(t_1)/C(t_2)) \cup (C(t_2)/C(t_1))\}}{|C(t_1)| * |C(t_2)|}$$

It is easy to prove that the following proposition holds:

Proposition 4.4.1 *The semantic distance $SD(\phi, \psi)$ satisfies the properties (i)Range, (ii)Reflexivity, (iii)Symmetry, (iv)Remoteness, and (v)Intermediary.*



Using the semantic distance defined above, we can define a relevance relation for selection functions. Naturally, an easy way to define a direct relevance relation between two triples in an ontology Σ is to define them as the semantically closest triples, i.e., there exist no other triples in the ontology that is semantically more close, like this,

$$\langle t_1, t_2 \rangle \in \mathcal{R}_{sd} \text{ iff } \neg \exists t' \in \Sigma (SD(t_1, t') < SD(t_1, t_2)).$$

Based on the semantic relevance above, we can define the selection functions like those defined in the previous section.

Using semantic distances, we propose a specific approach to deal with subsumption queries which have the form like $C_1 \sqsubseteq D$ where C_1 is a concept. In this new approach, C_1 is considered as a center concept of the query, and the newly defined selection function will track along the concept hierarchy in an ontology and always add the closest formulas (to C_1) which have not yet been selected, into the selected set as follows:

$$s(\Sigma, C_1 \sqsubseteq D, 0) = \emptyset.$$

Then the selection function selects the formulas $\phi \in \Sigma$ which is the closest to C_1 as a working set (i.e. $k = 1$) to see whether or not they are sufficient to give an answer to the query. Namely, we define¹

$$s(\Sigma, C_1 \sqsubseteq D, 1) = \{t \in \Sigma \mid \neg \exists t' \in \Sigma (SD(t', C_1) < SD(t, C_1))\}$$

If the reasoning process can obtain an answer to the query, it stops. Otherwise the selection function selects the formulas that are closest to the current working set. Namely, we have:

$$s(\Sigma, C_1 \sqsubseteq D, k) = \{t \in \Sigma \mid \neg \exists t' \in \Sigma (SD(t', C_1) < SD(t, C_1) \wedge \psi \notin s(\Sigma, C_1 \sqsubseteq D, k - 1))\} \cup s(\Sigma, C_1 \sqsubseteq D, k - 1)$$

for $k > 1$.

4.5 Strategies

4.5.1 Variant Strategies of Interleaving Reasoning and Selection

Various strategies can be developed for interleaving reasoning and selection of axioms. In Chapter 5, we will investigate the processing of interleaving reasoning and selection via a relevance measure with respect to the connections among nodes in triples. That would provide an approach of granular reasoning in which various granularity of web scale data can be selected for reasoning to improve the scalability. In Chapter 6, we will propose a different strategies for interleaving reasoning and selection of axioms, by selecting a sub-language of ontology data, namely, by focusing on axioms in which some pre-selected concepts appear.

¹It is easy to see the definition about $SD(t_1, t_2)$ is easily extended into a definition about $SD(t_1, C)$, where t_1, t_2 are triples, and C is a concept.



4.5.2 Strategies for Over-determined Processing

For inconsistent ontology data, reasoning extension procedure usually grows up to an inconsistent set rapidly. That may lead to too many undetermined answers. In order to improve it, over-determined processing (ODP) is introduced, by which we require that the selection function returns a consistent subset Σ'' at the step k when $s(\Sigma, \phi, k)$ is inconsistent such that $s(\Sigma, \phi, k - 1) \subset \Sigma'' \subset s(\Sigma, \phi, k)$. It is actually a kind of backtracking strategies used to reduce the number of undetermined answers to improve the extension strategy. An easy solution to the over-determined processing is to return the first maximal consistent subset (FMC) of $s(\Sigma, \phi, k)$, based on certain search procedure. Query answers which are obtained by this procedure are still sound, because they are supported by a consistent subset of the ontology. However, it does not always provide intuitive answers because it depends on the search procedure of maximal consistent subset in over-determined processing.

One of the improvements for the over-determined processing is to use the semantic relevance information. For example, we can prune semantically less relevant paths to obtain a maximal consistent set. Namely, In the over-determined processing, the reasoning processing will remove the most dissimilar formulas from the set $s(\Sigma, \phi, k) - s(\Sigma, \phi, k - 1)$ first, until it can find a maximal consistent set such that the query ϕ can be proved or disproved.



5. Unifying Search and Reasoning from the Viewpoint of Granularity

5.1 Introduction

The assumption of traditional reasoning methods do not fit very well when facing Web scale data. One of the major problems is that acquiring all the relevant data is very hard when the data goes to Web scale. Hence, unifying reasoning and search is proposed [12]. Under this approach, the search will help to gradually select a small set of data (namely, a subset of the original dataset), and provide the searched results for reasoning. If the users are not satisfied with the reasoning results based on the sub dataset, the search process will help to select other parts or larger sub dataset prepared for producing better reasoning results [12]. One detailed problem is that how to search for a good or more relevant subset of data and do reasoning on it. In addition, the same strategy may not meet the diversity of user needs since their backgrounds and expectations may differ a lot. In this chapter, we aim at solving this problem.

Granular computing, a field of study that aims at extracting the commonality of human and machine intelligence from the viewpoint of granularity [46, 47], emphasizes that human can always focus on appropriate levels of granularity and views, ignoring irrelevant information in order to achieve effective problem solving [47, 49]. This process contains two major steps, namely, the search of relevant data and problem solving based on searched data. As a concrete approach for problem solving based on Web scale data, the unification of search and reasoning also contains these two steps, namely, the search of relevant facts, and reasoning based on rules and searched facts. A granule is a set of elements that are drawn together by their equality, similarities, indistinguishability from some aspects (e.g. parameter values) [45]. Granules can be grouped into multiple levels to form a hierarchical granular structure, and the hierarchy can also be built from multiple perspectives [47]. Following the above inspirations, the web of data can be grouped together as granules in different levels or under different views for searching of subsets and meeting various user needs. From the perspective of granularity, we provide various strategies for unifying user driven search and reasoning under time constraints. From the multilevel point of view, in order to meet user needs in different levels, unifying search and reasoning with multilevel completeness and multilevel specificity are proposed. Furthermore, from the multiperspective point of view, the unifying process can be investigated based on different perspectives of the knowledge source. We also propose unifying search and reasoning with a starting point, which is inspired by the basic level advantage from cognitive psychology [32], to achieve diversity and scalability.

Section 5.2 introduces some basic notions related to this study. Section 5.3 give a very preliminary discussion on the search and reasoning process on a knowledge graph. The rest of this chapter focuses on introducing various strategies for unifying search and reasoning from the viewpoint of granularity: Section 5.4 discusses the starting point strategy. Section 5.5 introduces the multilevel completeness strategy. Section 5.6 introduces unifying strategy with multilevel specificity. Section 5.7 investigates on the multiperspective strategy. In Section 5.8, for each strategy introduced in this chapter, we provide some preliminary experimental results based on a semantic Web dataset SwetoDBLP, an RDF version of the DBLP dataset [3]. Section 5.9 discusses some



related work. Finally, Section 7 makes concluding remarks by highlighting major contributions of this chapter.

5.2 Basic Notions

In this section, we introduce some basic notions for unifying selection and reasoning from the viewpoint of granularity, namely, knowledge graph, granule, level, perspective, which are fundamental thoughts that this study is built upon.

5.2.1 Knowledge Graph

We consider knowledge graphs as a general data model for Web data/knowledge (e.g., RDF/RDFS data and OWL ontologies). Thus, in this chapter, granular reasoning is based on graph representation of knowledge.

Definition 2 (Knowledge Graph) *A knowledge graph (KG) is defined as:*

$$KG = \langle N, E, T \rangle, \quad (5.1)$$

where N is a set of nodes, E is a set of edges, and T is a triple set of $N \times E \times N$.¹

In a knowledge graph, the edges are with directions, and the nodes can be understood as classes in RDF/RDFS modeling. The relationship of two nodes in a knowledge graph is represented as a triple $t = \langle s, p, o \rangle$. The subject (s) and object (o) are nodes from N , and the predicate (p) is from the set of edges (namely $p \in E$). T is a set of triple sets ($t \in T$).²

Definition 3 (Node Degree) *For a node n in the knowledge graph $KG = \langle N, E, T \rangle$, its degree $degree(n)$ is measured by:*

$$\begin{aligned} degree(n) &= degree_{in}(n) + degree_{out}(n), \\ degree_{in}(n) &= |\{ \langle s, p, n \rangle : \langle s, p, n \rangle \in T \}|, \\ degree_{out}(n) &= |\{ \langle n, p, o \rangle : \langle n, p, o \rangle \in T \}|, \end{aligned} \quad (5.2)$$

where $degree_{in}(n)$ denotes the in-degree of n , while $degree_{out}(n)$ denotes the out-degree.

Definition 4 (Edge Degree) *Given a knowledge graph $KG = \langle N, E, T \rangle$, the edge degree e is defined as:*

$$degree(e) = |\{ \langle s, e, o \rangle : \langle s, e, o \rangle \in T \}|, \quad (5.3)$$

Definition 5 (Normalized Degree) *The normalized node degree or edge degree can be represented as:*

$$\begin{aligned} normalized_degree(n) &= \frac{degree(n)}{\max_{n' \in N} \{degree(n')\}}, \\ normalized_degree(e) &= \frac{degree(e)}{\max_{e' \in E} \{degree(e')\}}. \end{aligned} \quad (5.4)$$

The $normalized_degree(n)$ or $normalized_degree(e)$ can give a relative evaluation on the importance of the node or the edge in the KG .

¹The definition of the knowledge graph can be extended to be with weighted edges. Namely, a weighted knowledge graph $WKG = \langle N, E, T, R \rangle$ where $R : T \rightarrow \mathcal{R}$ is a mapping which assigns a triple $t \in T$ a real number $r \in \mathcal{R}$.

²A knowledge graph is said to be a first order one if its node set and its edge set are disjoint (i.e., $N \cap E = \emptyset$). If a knowledge graph G is a second order one, then an edge can be a node.



5.2.2 Granule

In the context of the KG , a granule is a set of nodes that are grouped together by equality, similarity, indistinguishability, etc [14, 47]. A granule can be a singleton, i.e., $\{n\}$. When a granule is a singleton, we can use a node n to denote a granule.

We define a general binary relation “contains” to represent the hierarchical relation among granules³. We assume that this relation satisfy the following rational postulates:

- (1) Reflexivity: $\forall g [g \text{ contains } g]$,
- (2) Transitivity: $\forall g_1, g_2, g_3 [g_1 \text{ contains } g_2, g_2 \text{ contains } g_3 \Rightarrow g_1 \text{ contains } g_3]$,
- (3) Antisymmetry: $\forall g_1, g_2 [(g_1 \text{ contains } g_2) \wedge (g_1 \neq g_2) \Rightarrow \neg(g_2 \text{ contains } g_1)]$,
- (4) Universal container: $\exists \top \forall g [\top \text{ contains } g]$,
- (5) Bottom container: $\exists \perp \forall g [g \text{ contains } \perp]$.

5.2.3 Perspective

In a knowledge graph, there might be various types of edges, if we consider creating a series of subgraphs by a subset of the edge type, several subgraphs reflecting different characteristics of the knowledge graph can be acquired.

Definition 6 (Perspective) *In a knowledge graph KG , a perspective P is a subset of edges (i.e., $P \subseteq E$).*

A perspective is a viewpoint to investigate the KG . It can be a singleton (namely, $P = \{e\}$) or a set of predicates. Different perspectives reflect various characteristics of the graph. The set of all perspectives $P \subseteq E$ collectively describe a graph from multiple viewpoints. Under a specified perspective, a subgraph of KG is generated, the node degree of this subgraph may reflect a unique characteristic of the original KG .

Definition 7 (Node Degree under a Perspective) *The Node Degree under a perspective is defined as:*

$$\begin{aligned}
 \text{degree}(n, P) &= \text{degree}_{in}(n, P) + \text{degree}_{out}(n, P), \\
 \text{degree}_{in}(n, P) &= |\{\langle s, p, n \rangle : \langle s, p, n \rangle \in T \text{ and } p \in P\}|, \\
 \text{degree}_{out}(n, P) &= |\{\langle n, p, o \rangle : \langle s, p, n \rangle \in T \text{ and } p \in P\}|,
 \end{aligned} \tag{5.5}$$

where $\text{degree}_{in}(n, P)$ and $\text{degree}_{out}(n, P)$ denote the indegree and outdegree for the node n under the perspective P respectively.

Proposition 5.2.1 (Formal Properties of Node Degree under a Perspective)

For a knowledge graph $KG = \langle N, E, T \rangle$, the following properties hold:

- (1) Monotonicity: $P' \subseteq P'' \Rightarrow \text{degree}(n, P') \leq \text{degree}(n, P'')$,
- (2) Triviality: $P' = E \Rightarrow \text{degree}(n, P') = \text{degree}(n)$,
- (3) Emptiness: $P' = \emptyset \Rightarrow \text{degree}(n, P') = 0$,
- (4) Union: $\text{degree}(n, P' \cup P'') \leq \text{degree}(n, P') + \text{degree}(n, P'')$,
- (5) Disjointness: $P' \cap P'' = \emptyset \Rightarrow \text{degree}(n, P' \cup P'') = \text{degree}(n, P') + \text{degree}(n, P'')$.

³For ontologies, the contain relation can be understood as the union of the *subClassOf* and the *instanceOf* relation, etc. From the perspective of the set theory, the contain relation can be understood as either the subset relation or the membership relation. Namely, $\text{contains} = \{\subseteq, \subset, \in\}$.



Definition 8 (Normalized Node Degree under a Perspective)

$$normalized_degree(n, P) = \frac{degree(n, P)}{\max_{n' \in N} \{degree(n', P)\}}, \quad (5.6)$$

Normalized node degree under a perspective can be used to evaluate the relative importance of a node in a knowledge graph.

5.2.4 Level

Granularity is the grain size of granules. In a knowledge graph, a level of granularity, denoted as $L_{g(i)}$ (where i is a positive integer, $i \in I^+$), can be considered as a partition/covering over the set of all granules or the set of all nodes if we only consider singleton granules. A level of granularity $L_{g(i)}$ is finer than $L_{g(j)}$ iff the partition/covering in $L_{g(i)}$ is finer than $L_{g(j)}$.

Considering the semantics of the nodes in a knowledge graph, some nodes are more general, while some are more specific than others. Hence, they belong to different level of specificity, denoted as $L_{s(i)}$ (where $i \in I^+$). Let g_m, g_n be two granules, and the level of specificity $L_{s(i)}$ is said to the next level of specificity $L_{s(i-1)}$, written $L_{s(i)} \succ L_{s(i-1)}$ if the following conditions are satisfied:

- **Exclusion.** Any two granules which are located at the same level would not contain each other. $\forall g_m, g_n \in L_{s(i)} [\neg(g_m \text{ contains } g_n) \wedge \neg(g_n \text{ contains } g_m)]$.
- **Neighboring.** There are no middle granules which are located in two neighboring levels. $(g_m \text{ contains } g_n) \wedge (g_m \in L_{s(i)}) \wedge (g_n \in L_{s(i-1)}) \Rightarrow \nexists g' [(g_m \text{ contains } g') \wedge (g' \text{ contains } g_n)]$.

5.3 Searching and Reasoning on a Knowledge Graph

In general, in the context of knowledge graph (KG), we can consider a task of reasoning is to check whether or not a KG entails a triple t , written as $KG \models t^4$. We can extend this entailment relation with a knowledge graph and a triple set as follows:

$$KG \models \{t_1, \dots, t_n\} \text{ iff } KG \models t_1, \dots, KG \models t_n.$$

Traditional reasoning method cannot be used to handle knowledge graph with large scale. Hence, selecting and do reasoning on subgraphs of KG may be appropriate as an approximation approach. Unifying search and reasoning from the viewpoint of granularity provides several strategies to achieve this goal on a KG . Generally, they all follow an (iterative) procedure which consists of the following processing loop :

- i) select part of the knowledge graph, i.e., find a subgraph KG'_i of KG where i is a positive integer, i.e., $i \in I^+$;
- ii) apply the standard reasoning to check if $KG'_i \models t$ for some triple $t \in \{t_1, \dots, t_n\}$ ⁵;
- iii) decide whether or not to stop the reasoning procedure or continue the reasoning with gradually increased selected subgraph of the knowledge graph (Hence, $KG'_1 \subseteq KG'_2 \subseteq \dots \subseteq KG$).

⁴In logics, this entailment relation can be formally defined as $KG \models t$ iff for any model M of $KG, M \models t$.

⁵Here we assume standard reasoning is sound.



From this processing loop, it is easy to see that unifying search and reasoning is with anytime behavior, hence each of the strategies introduced below can be considered as a method for anytime reasoning.

5.4 Starting Point Strategy

Psychological experiments support that during problem solving, in most cases, people try to investigate the problem starting from a “basic level” (where people find convenient to start according to their own background knowledge), in order to solve the the problem more efficiently [32]. In addition, concepts in a basic level are used more frequently than others [42]. Following this idea, we define that during the unification of search and reasoning process on the Web for a specified user, there is a starting point (denoted as SP).

Definition 9 (Starting Point) *A starting point SP consists of a set of nodes N and a (relevant) perspective P . Namely, $SP = \langle N', P' \rangle$, which satisfies the following relevance condition:*

$$\forall p \in P' \exists n \in N' [\exists o (\langle n, p, o \rangle \in T) \vee \exists s (\langle s, p, n \rangle \in T)].$$

The nodes in N' is with orders which are ranked based on the node degree under the specified perspective ($\text{degree}(n, P')$). Among these nodes, there is one node representing the user (e.g. a user name, a URI, etc.), and other nodes are related to this node from the perspective P' which serve as the background for the user (e.g. user interests, friends of the user, or other user familiar or related information).

A starting point SP can be understood as a context or background for reasoning tasks which contains user related information (More specifically, for the LarKC project, a starting point is used to create the context for retrieval and reasoning). It is easy to see that this strategy would make sense only when a starting point should be connected with the knowledge graph. A starting point is used for refining the unification of search and reasoning process in the form that the user may prefer.

Following the idea of starting point, the search of important nodes for reasoning can be based on the following strategies:

- Strategy 1 (Familiarity-Driven): The search process firstly select out the nodes which are directly related to the SP for the later reasoning process, and SP related results are ranked to the front of others.
- Strategy 2 (Novelty-Driven): The search process firstly select out the nodes which are not directly related to the SP , then they are transferred to the reasoning process, and SP related nodes are pushed to the end of others.

Strategy 1 is designed to meet the user needs who want to get more familiar results first. Strategy 2 is designed to meet the needs who want to get unfamiliar results first. One example for strategy 2 is that in news search on the Web, in most cases the users always want to find the relevant news webpages which have not been visited. We will provide an example which uses strategy 1 in Section 5.8.2.



5.5 Multilevel Completeness Strategy

Web scale reasoning is very hard to achieve complete results, since the user may not have time to wait for a reasoning system going through the complete dataset. If the user does not have enough time, a conclusion is made through reasoning based on a searched partial dataset, and the completeness is not very high since there are still some sets of data which remain to be unexplored. If more time is allowed, and the reasoning system can get more sub datasets through search, the completeness can migrate to a new level since the datasets cover wider range.

There are two major issues in this kind of unifying process of search and reasoning: (1) Since under time constraint, a reasoning system may just can handle a sub dataset, methods on how to select an appropriate subset need to be developed. (2) Since this unification process require user judges whether the completeness of reasoning results is good enough for their specific needs, a prediction method for completeness is required. We name this kind of strategy as unifying search and reasoning with multilevel completeness, which provides reasoning results in multiple levels of completeness based on the searched sub dataset under time constraints, meanwhile, provides prediction on the completeness value for user judges. In this chapter, we develop one possible concrete solution.

For issue (1), searching for a more important sub dataset for reasoning may be a practical approach to select the subset effectively [12], and may be an approach to handle the scalability issue, since in most cases, the amount of important data is relatively small. Under the context of the Semantic Web, the semantic dataset can be considered as a graph that contains a set of nodes (subjects and objects in RDF dataset) and a set of relations (predicates in RDF dataset) on these nodes. Hence, in this chapter, we borrow the idea of “pivotal node” from network science [5], we propose a network statistics based data selection strategy. Under this strategy, we use the node degree (denoted as $degree(n)$) to evaluate the importance of a node in a dataset. The nodes with relatively high value of node degree are selected as more important nodes and grouped together as a granule for reasoning tasks. In the context of a knowledge graph, first we choose a perspective (P) from the starting point (SP) of a specified user, then nodes with the same or close node degree under a perspective ($degree(n, P)$) are grouped together as a granule (If the starting point does not provides constraints for this, normally, edges with a relatively high edge degree ($degree(e)$) is suggested.). Nodes are ranked according to $degree(n, P)$ for reasoning. With different number of nodes involved, a subgraph with different scale is produced for reasoning, hence reasoning results with multiple levels of completeness are provided.

For issue (2), here we give a formula to produce the predicted completeness value ($PC(i)$) when the nodes which satisfy $degree(n, P) \geq i$ (i is a nonnegative integer) have been involved.

$$PC(i) = \frac{|N_{rel(i)}| \times (|N_{sub(i)}| - |N_{sub(i')}|)}{|N_{rel(i)}| \times (|N| - |N_{sub(i')}|) + |N_{rel(i')}| \times (|N_{sub(i)}| - |N|)}, \quad (5.7)$$

where $|N_{sub(i)}|$ represents the number of nodes which satisfy $degree(n, P) \geq i$, $|N_{rel(i)}|$ is the number of nodes which are relevant to the reasoning task among the involved nodes $N_{sub(i)}$, and $|N|$ is the total number of nodes in the dataset. The basic idea is that, first we can obtain a linear function which go through $(|N_{sub(i)}|, |N_{rel(i)}|)$ and $(|N_{sub(i')}|, |N_{rel(i')}|)$ (i' is the last assigned value of $degree(n, P)$ for stopping the reasoning process before i). Knowing $|N|$ in the dataset ($|N|$ only needs to be acquired



once and can be calculated offline), by this linear function, we can predict the number of satisfied nodes in the whole dataset, then the predicted completeness value can be acquired.

5.6 Multilevel Specificity Strategy

Reasoning results can be either very general or very specific. If the user has not enough time, the search and reasoning process will just be on a very general level. And if more time is available, this process may go to a more specific level which contains results in a finer level of grain size (granularity). Namely, the unification of search and reasoning can be with multilevel specificity, which provides reasoning results in multiple levels of specificities under time constraints.

The study of the semantic networks emphasizes that knowledge is stored as a system of propositions organized hierarchically in memory [11]. The concepts in various levels are with different levels of specificities. Hence, the hierarchical knowledge structure can be used to supervise the unification of search and reasoning with multilevel specificity.

Definition 10 (Hierarchical Knowledge Structure) *Although as a whole, a knowledge graph does not force a hierarchical organization, some ordered nodes ($n \in N$) and their interrelation “contains” can form a subgraph of KG , which is a hierarchical knowledge structure (HKS), which can be represented as:*

$$HKS = \langle N, \{\text{contains}\}, T \rangle. \quad (5.8)$$

In the HKS , some nodes are with a coarser level of granularity and are more general than others, while some of them are more specific, and with a finer level of granularity. The nodes are well ordered by the “contains” relations. In the unification process of search and reasoning with multilevel specificity strategy, the search of sub datasets is based on the hierarchical relations (e.g. sub class of, sub property of, instance of, etc.) among the nodes (subjects and objects in RDF) in the HKS and is forced to be related with the time allowed. Nodes which are not sub classes, instances or sub properties of other nodes will be searched out as the first level for reasoning. If more time is available, more deeper levels of specificity can be acquired according to the transitive property of these hierarchical relations. The specificity will just go deeper for one level each time before the next checking of available time (Nodes are searched out based on direct hierarchical relations with the nodes from the former direct neighborhood level).

5.7 Multiperspective Strategy

User needs may differ from each other when they expect answers from different perspectives. In order to avoid the failure of understanding in one way, knowledge needs to be represented in different points of view [28]. If the knowledge source is investigated in different perspectives, it is natural that the search and reasoning results might be organized differently. Each perspective satisfies user needs in a unique way. As another key strategy, unifying search and reasoning from multiperspective aims at satisfying user needs in multiple views.



It is possible to choose all the edges in a KG as a perspective, but as mentioned in Proposition 1(2), in this case, it will be too trivial to realize the uniqueness of each type of edge, and may be hard to satisfy various user needs. In a KG , considering each $P \subseteq E$, a subgraph KG_P of the original one is generated. Different subgraphs reflect various characteristics of the original one. Hence, perspectives and the different characteristics reflected from these perspectives can be considered as another attempt to meet the diverse user needs.

For a reasoning task, if the perspective in a starting point is available, the process will take this acquired perspective. If not, the perspectives will be chosen by the normalized edge degree ($normalized_degree(e)$). They can help to judge the importance of a perspective in the KG . Edges with relatively high value of $normalized_degree(e)$ may reflect major characteristics of the KG , but users are not force to accept the recommended perspectives, they can switch perspectives to meet their needs. After the perspective is chosen, the multilevel completeness/specificity strategy can be used.

The multiperspective strategy aims at satisfying various user needs from multiple perspectives. Based on different perspectives, even using the same method to rank nodes for reasoning (e.g., in this report, we use node degree under a perspective ($degree(n, P)$) for the multilevel completeness strategy), the organization of the results are different.

5.8 A Case Study on the Semantic Dataset

In the context of the Semantic Web, an RDF file is composed of triple sets, and it can be considered as a knowledge graph (KG). All the defined statistical parameters for the knowledge graph can be used on the RDF graph. In this section, we provide some illustrative examples of the granular reasoning strategies discussed above. All the examples are developed based on the SwetoDBLP dataset [3].

5.8.1 Multilevel Completeness Strategy

Variable completeness reasoning on the Semantic Web provides reasoning results in multiple levels of completeness under time constraints. A perspective (P) need to be chosen and the nodes in the RDF graph for reasoning will be ordered according to $degree(n, P)$. As an illustrative example, we take the reasoning task “Who are authors in Artificial Intelligence (AI)?” based on the SwetoDBLP dataset. For the most simple case, following rule can be applied for reasoning to find relevant authors:

$$haspaper(X, Y), contains(Y, \text{“Artificial Intelligence”}) \rightarrow author(X, \text{“AI”})$$

where $haspaper(X, Y)$ denotes that the author X has a paper titled Y . $contains(Y, \text{“Artificial Intelligence”})$ denotes that the title Y contains the term “Artificial Intelligence”. $author(X, \text{“AI”})$ denotes that the author X is an author in the field of AI . Since the SwetoDBLP contains too many publications (More than 1,200,000), doing reasoning based on a dataset like this may require an unacceptable period of time, it is better that more important authors could be provided to the user first. Here we assume a starting point that indicate using coauthor number as the chosen perspective (denoted as P_{cn}). Under this perspective, the authors with more coauthors, namely, has a higher value of $degree(n, P_{cn})$, are more important. In order to illustrate the levels of completeness, we randomly choose some $degree(n, P_{cn})$ to stop the reasoning process, as



shown in Table 5.1. The reasoning process will start from the nodes with the biggest value of $degree(n, P_{cn})$, reduce the value gradually as time passed by, and will stop at the chosen $degree(n, P_{cn})$ for user judges. In order to meet users' specific needs on the levels of completeness value, using the proposed completeness prediction method introduced above, the prediction value has also been provided in Figure 5.1. This prediction value serves as a reference for users to judge whether they are satisfied. If more time is allowed and the user has not been satisfied yet, more nodes are involved, one can get reasoning results with higher levels of completeness. In this way, we provide solutions for the various user needs.

$degree(n, P_{cn})$ value to stop	Satisfied authors	AI authors
70	2885	151
30	17121	579
11	78868	1142
4	277417	1704
1	575447	2225
0	615124	2355

Table 5.1: Unifying search and reasoning with multilevel completeness and anytime behavior.

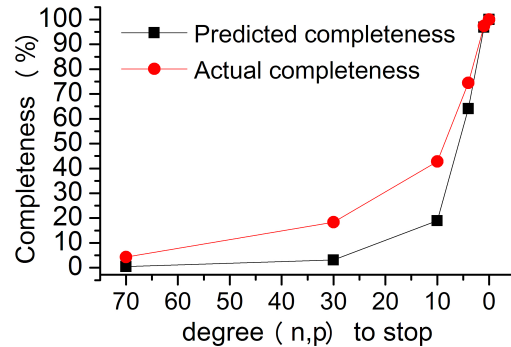


Figure 5.1: Comparison of predicted and actual completeness value.

5.8.2 Starting Point Strategy

We continue the discussion of the above example for the multilevel completeness strategy, and give an example using strategy 1 for the starting point strategy. Notice that this example is a synergy of the multilevel completeness strategy and the starting point strategy.

Following the same reasoning task in the above sections, “John McCarthy”, is taken as a concrete user name in a *SP*, and his coauthors⁶ whom he definitely knows (with * after the names) are ranked into the top ones in every level of the “Artificial Intelligence” author lists when the user tries to stop while an arbitrary $degree(n, P_{cn})$ of the relevant nodes has been involved (Since the coauthors are all persons whom the author should know. These information helps users get more convenient reasoning results.). Some partial output in some levels is shown in Table 5.2. The strategy of multilevel specificity and starting point can also be integrated together, which provide reasoning results based on starting point in every level of specificity to produce a more user-preferred form of results.

5.8.3 Multilevel Specificity Strategy

In the multilevel specificity strategy, if the user has very limited time, we may just use the input keywords as reasoning constraints and do not move to more specific or general levels. As an illustrative example, we use the same reasoning task in the upper

⁶In this study, we represent the coauthor information for each author in an RDF file using the FOAF vocabulary “foaf:knows”. The coauthor network RDF dataset created based on the SwetoD-BLP dataset can be acquired from <http://www.iwici.org/dblp-sse>. One can utilize this dataset to create a starting point for refining the reasoning process.



Table 5.2: A comparative study of the multilevel completeness strategy without and with a starting point. (User name: John McCarthy)

Completeness	Authors (coauthor numbers) without a starting point	Authors (coauthor numbers) with a starting point
Level 1 $degree(n, P_{cn}) \geq 70$	Carl Kesselman (312) Thomas S. Huang (271) Edward A. Fox (269) Lei Wang (250) John Mylopoulos (245) Ewa Deelman (237) ...	Hans W. Guesgen (117) * Carl Kesselman (312) Thomas S. Huang (271) Edward A. Fox (269) Lei Wang (250) John Mylopoulos (245) ...
Level 2 $degree(n, P_{cn}) \in [30, 70)$	Claudio Moraga (69) Virginia Dignum (69) Ralph Grishman (69) Biplav Srivastava (69) Ralph M. Weischedel (69) Andrew Lim (69) ...	Virginia Dignum (69) * John McCarthy (65) * Aaron Sloman (36) * Claudio Moraga (69) Ralph Grishman (69) Biplav Srivastava (69) ...
...

section. For the very general level, the reasoning system will just provide authors whose paper titles contain “Artificial Intelligence”, and the reasoning result is 2355 persons (It seems not too many, which is not reasonable.). Since in many cases, the authors in the field of AI do not write papers whose titles include the exact term “Artificial Intelligence”, they may mention more specific terms such as “Agent”, “Machine Learning”, etc. If more time is given, answers with a finer level of specificity according to a hierarchical domain ontology of “Artificial Intelligence” can be provided. Based on all the AI related conferences section and subsection names in the DBLP, we create a “three-level Artificial Intelligence ontology” automatically (This ontology has a hierarchical structure representing “Artificial Intelligence” related topics. Topic relations among levels are represented with “rdfs:subClassOf”), and we utilize this ontology to demonstrate the unification of search and reasoning with multilevel specificity⁷.

The rule for this reasoning task is:

$$hasResttime, haspaper(X, Y), contains(Y, H), topics(H, “AI”) \rightarrow author(X, “AI”)$$

where *hasResttime* is a dynamic predicate which denotes whether there is some rest time for the reasoning task⁸, *topics(H, “AI”)* denotes that *H* is a related sub topic from the hierarchical ontology of AI. If the user allows more time, based on the “rdfs:subClassOf” relation, the subtopics of AI in Level 2 of the ontology will be used as *H* for reasoning to find more authors in the field of AI. Further, if the user

⁷Here we ignore the soundness of this ontology, which is not the focus of this paper (Supporting materials on how we build the ontology can be found from : <http://www.iwici.org/user-g.>). One can choose other similar ontologies instead.

⁸For implementation, logic programming languages such as Prolog does not allow a dynamic predicate like *hasResttime*. But we can consider *resttime(T)* as a counter which would return a number. Then, we can check the number to know whether there is any rest time left. Namely: $resttime(T), T > 0 \rightarrow hasResttime$.



Table 5.3: Answers to “Who are the authors in Artificial Intelligence?” in multiple levels of specificity according to the hierarchical knowledge structure of Artificial Intelligence.

Specificity	Relevant keywords	Number of authors	
Level 1	Artificial Intelligence	2355	
Level 2	Agents	9157	
	Automated Reasoning	222	
	Cognition	19775	
	Constraints	8744	
	Games	3817	
	Knowledge Representation	1537	
	Natural Language	2939	
	Robot	16425	
	
Level 3	Analogy	374	
	Case-Based Reasoning	1133	
	Cognitive Modeling	76	
	Decision Trees	1112	
	Proof Planning	45	
	Search	32079	
	Translation	4414	
	Web Intelligence	122	
	

Table 5.4: A comparative study on the answers in different levels of specificity.

Specificity	Number of authors	Completeness
Level 1	2355	0.85%
Level 1,2	207468	75.11%
Level 1,2,3	276205	100%

wants to get results finer than Level 2, then the subtopics in Level 3 are used as H to produce an even more complete result list. As shown in Tables 5.3 and 5.4, based on the hierarchy of Artificial Intelligence, Since Levels 2 and 3 contain more specific sub branches, it is not surprising that one can get more authors when deeper levels of terms are considered, hence, the completeness of the reasoning result also goes to higher levels, as shown in Table 5.4.

5.8.4 Multiperspective Strategy

For simplification, here we consider the situation that a perspective is a singleton ($P = \{e\}$). As mentioned in section 5.7, we choose P by the normalized edge degree ($normalized_degree(e)$). Figure 5.2 shows the distribution of $normalized_degree(e)$. According to this figure, we find that among the edges who hold relatively big $normalized_degree(e)$, “rdf:Seq” and “rdfs:label” are very meaningful (“rdf:Seq” can be used to find coauthor numbers, and “rdfs:label” can be used to find publication numbers for each author). Hence, we analyze the distribution of the node degrees under the per-



spective of coauthor numbers (P_{cn}) and publication numbers (P_{pn}). Firstly, We choose the perspective of the number of coauthors. From this perspective, we find following characteristics of the SwetoDBLP dataset: Coauthor number distribution is shown as in Figure 5.3. In the left side of Figure 5.4, there is a peak value in the distribution, and it does not appear at the point of 0 or 1 coauthor number (as shown in Figure 5.5). Hence, the shape of the distribution is very much like a log-normal distribution. These phenomena are not special cases that just happen to all the authors, we also observed the same phenomenon for authors in many sub-fields in computer science, such as Artificial Intelligence (as shown in Figure 5.6, Software Engineering, Data Mining, Machine Learning, the World Wide Web, Quantum Computing, etc. As a comparison of the coauthor number view, we provide some partial results from the view point of publication number. We observe that, different from the perspective of coauthor number distribution, the publication number distribution follows very much like a power law distribution, without a peak value in the middle of the distribution curve, as shown in Figures 5.7 and 5.8.

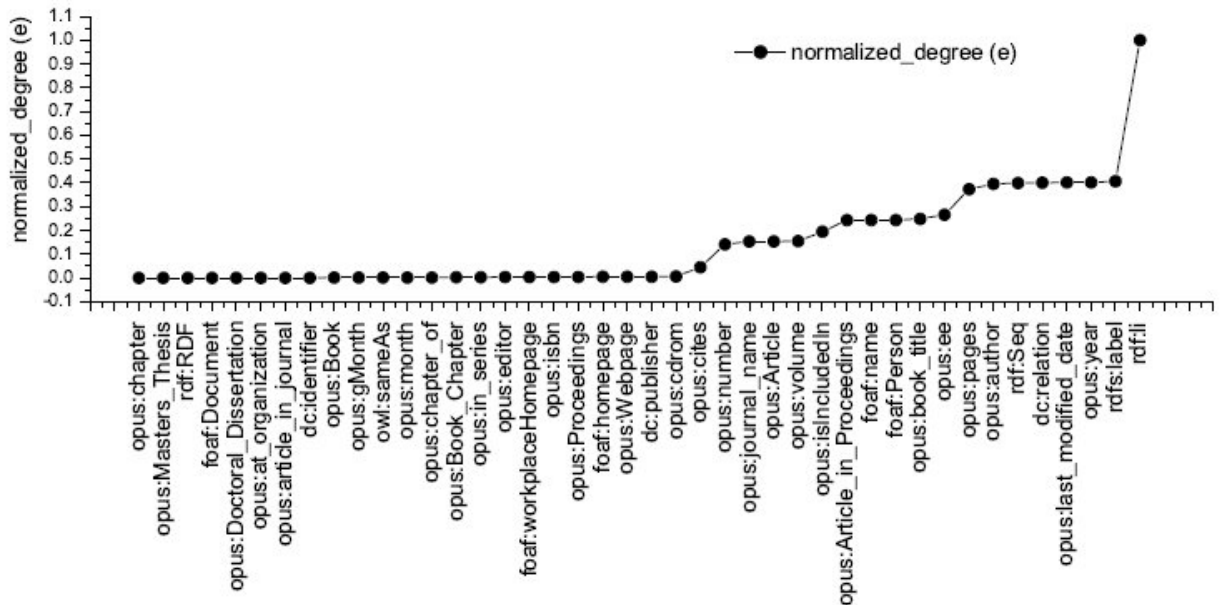


Figure 5.2: Normalized edge degree distribution in the SwetoDBLP RDF dataset.

Table 5.5: A partial result of the variable specificity reasoning task “The list of authors in Artificial Intelligence” in level 1 from two perspectives.

Publication number perspective	Coauthor number perspective
Thomas S. Huang (387)	Carl Kesselman (312)
John Mylopoulos (261)	Thomas S. Huang (271)
Hsinchun Chen (260)	Edward A. Fox (269)
Henri Prade (252)	Lei Wang (250)
Didier Dubois (241)	John Mylopoulos (245)
Thomas Eiter (219)	Ewa Deelman (237)
...	...

Table 5.5 provides a partial result for the experiment in variable specificity reasoning introduced in Section 5.8.3 from two perspectives (namely, publication number and coauthor number). As shown in Figure 5.4, Figure 5.8, and Table 5.5, it is clear that since the distribution of node degree under the above two perspectives are different,

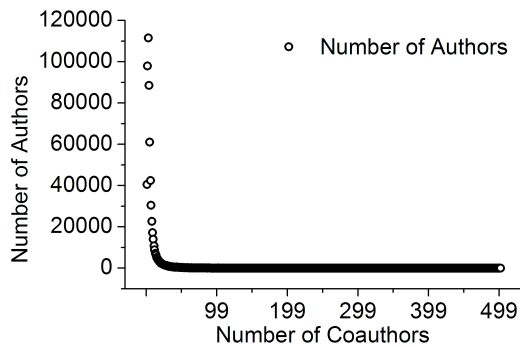


Figure 5.3: Coauthor number distribution in the SwetoDBLP dataset.

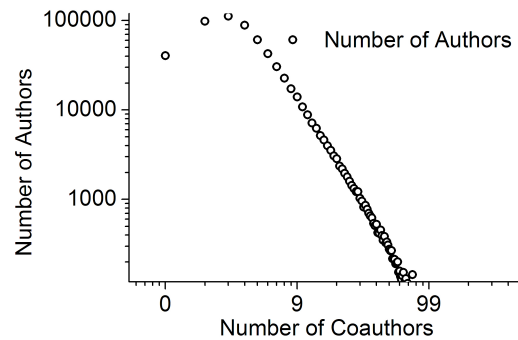


Figure 5.4: log-log diagram of Figure 5.3.

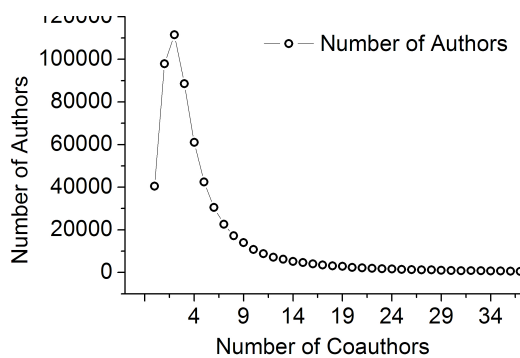


Figure 5.5: A zoomed in version of Figure 5.3.

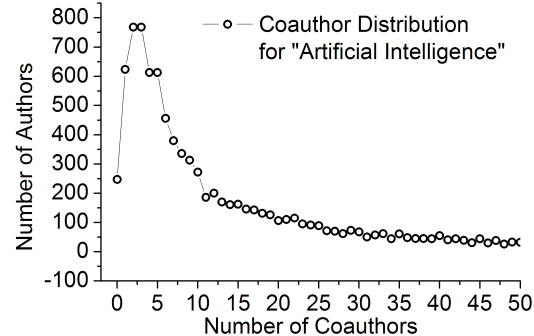


Figure 5.6: A zoomed in version of coauthor distribution for “Artificial Intelligence”.

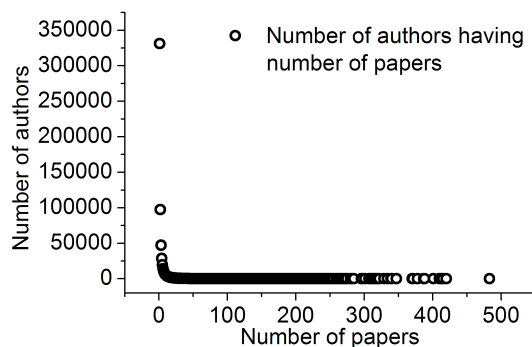


Figure 5.7: Publication number distribution in the SwetoDBLP dataset.

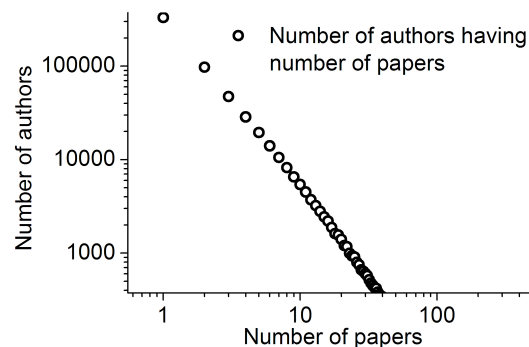


Figure 5.8: log-log diagram of Figure 5.7.

and for the same node, the node degree under these two perspectives are different, we can conclude that using different perspectives, both of the sequence of nodes provided for reasoning and the reasoning results are organized differently. In this way, various user needs can be satisfied.



5.9 Related Work

The study of unifying reasoning and search at Web scale [12] is the framework that this chapter is based on. The strategies introduced in this chapter aim at providing some possible solutions for how the unification can be done in a more user-oriented way from the viewpoint of granularity. They are developed based on many existing studies. Here we introduce some major related areas, namely, variable precision logic and previous studies on reasoning with granularity.

Variable precision logic is a major method for reasoning under time constraints, which provides two reasoning strategies, namely, variable certainty and variable specificity reasoning [27]. Concerning time constraint, given more time, a system with variable specificity can provide a more specific answer, while a system with variable certainty can provide a more certain answer [27]. Some strategies on unifying search and reasoning introduced in this chapter, 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, while multilevel specificity strategy uses hierarchical knowledge structure to supervise the unification process of search and reasoning. In this document, we did not investigate on the idea of variable certainty. Since it belongs to non-monotonic reasoning, and the certainty won’t necessarily go higher as more data is involved (since there might be contradictions [1] or inconsistency [21] on the facts, especially in the dynamic changing context of the Web). How it can be applied to a more user-centric environment still needs further investigations.

The study of reasoning with granularity starts from the logic approaches for granular computing [14, 26, 51], etc. Under the term of granular reasoning, it has also been studied from the perspectives of propositional reasoning [29], Aristotle’s categorical syllogism [30], and granular space [44]. These studies concentrate on the logic foundations for reasoning under multi-granularity (mainly on zoom-in and zoom-out). In this chapter, our focus is on how to unify the search and reasoning process from the viewpoint of granularity, namely, how to search for a good subset of the original dataset, and do reasoning on the selected dataset based on the idea of granularity. Besides the inspiration from granular computing [47, 49], especially granular structures [47]. The strategies proposed in this chapter are also inspired from Cognitive Psychology studies on human problem solving (e.g. starting point) [32, 41]. Further, we concentrate on how granularity related strategies can help to effectively solve Web scale reasoning problems according to different user context and time constraints.

We also need to point out that although the strategies introduced in this chapter are inspired by some basic strategies in granular computing, the granular structures, more specifically granular knowledge structures that are mentioned in this chapter are different from previous studies [47, 48]. In granular computing, granules are organized hierarchically from larger grain sizes to smaller ones (or the other way around), and the granules in coarser levels contain the ones in finer levels. In this study, although granules are still in a hierarchy, the granules does not contain each other. In the multilevel completeness strategy, granules are organized into different levels by the node degree under a perspective, granules with higher value of $degree(n, P)$ do not contain those with lower values. In the multilevel specificity strategy, although the hierarchical knowledge structures of Artificial Intelligence has a typical granular structure (All the subtopics are covered under the terms one level coarser than them.), the granular structure of the reasoning results based on this hierarchy is different from the granular



structures studied previously [47, 48], since the results which were got from the coarser levels cannot cover finer levels (The reason is that if the user does not have enough time, nodes in finer levels, such as authors of “Decision Trees”, will not be selected for the reasoning task whether they are AI authors.).

The idea of network analysis has been introduced from Network Science to the Semantic Web [39, 18, 23], ranging from Social Network analysis [39, 23] to Ontology structure analysis [18] and Ontology partition [37]. Anytime Reasoning has been studied from the perspective of approaches [24, 40, 31], evaluation (in terms of completeness and soundness) [16, 17], and applications [2]. nevertheless, there are not much study that touch the prediction of completeness when the reasoning process is stopped, which will be very useful for users to judge whether they are satisfied or not. In our study, we develop network statistical degree distribution analysis considering the semantics of the edges, and based on this analysis, we provide a concrete type of anytime reasoning method as well as a simple completeness prediction method.

5.10 Conclusion and Future Work

As an approach for incomplete reasoning at Web scale, unifying search and reasoning from the viewpoint of granularity provides some strategies which aim at removing the diversity and scalability barriers for Web reasoning.

For the diversity issue: The strategy of starting point focuses on user specific background and the unification process is familiarity driven or novelty driven, and is obviously user oriented. Multilevel completeness strategy is with anytime behavior [40], and provides predictions of completeness for user judges when the user interact with the system. Multilevel specificity strategy emphasizes on reasoning with multiple levels of specificity and users can choose whether to go into more specific or more general levels. Multiperspective strategy attempts to meet various user needs from multiple perspectives.

For the scalability issue: In the multilevel completeness strategy, although the partial results may have low completeness, more important results have been searched out and ranked to the top ones for reasoning based on their higher values of $degree(n, P)$. In other words, more important results are provided as a possible way to solve the scalability problems. The starting point strategy also provides two methods to select important nodes for reasoning. The multilevel specificity strategy concentrates on the appropriate levels of specificity controlled by the knowledge hierarchy and does not get into unnecessary levels of data. Hence, under limited time, the reasoning task and time is reduced.

Here we provide a preliminary discussion on the relationship of granularity and centrality. In this chapter, we use granularity (for example, in the multilevel specificity strategy) to supervise the hierarchical search and reasoning process, and we use normalized node degree under a perspective $normalized_degree(n, P)$ to select the nodes (and organize them as different granules) for reasoning in the multilevel completeness strategy. Actually, these two concepts are highly related but totally different. The related part for these two concepts is that $normalized_degree(n, P)$ can be considered as a simplified model for calculating centrality. Hence, in other words, here we use centrality to produce granules in different granularity (Nodes with different $normalized_degree(n, P)$ are organized as granules with different granularity.). The different part for these two concepts is that a node with coarser granularity is not



necessarily have high centrality. For example, suppose there is a node A that is a super class of only one direct connected node B , hence, B is with low centrality. But if the direct connected node B is super class of many other nodes, by reasoning, we can conclude that A is a super class of these nodes. Hence, A is with coarser granularity, and more extra links have been produced for A . In this way, we can find that through reasoning, more implicit relationship can be found, and a node with lower centrality explicitly, may has higher centrality implicitly. More deeper discussions and related experiments (currently, the WordNet dataset is considered since it contains many hierarchical relations) is planned for future work to find deeper relationships of these two concepts which are considered to be two approaches to solve the scalability barriers for Web-scale reasoning.

Since user needs are very related to the satisfaction of reasoning results, in future studies, we would provide a comparison from the user perspective on the effects of multiple strategies mentioned in this chapter. We would also like to investigate in great details on how these strategies can be combined together to produce better solutions. Since $normalized_degree(n, P)$ is used to rank nodes for reasoning, by Proposition 1 (4), it is clear that for each node, $normalized_degree(n, P)$ is not hard to get from several knowledge graphs if they come from multiple sources. In order to solve the scalability problem, it is also possible to parallelize the calculation of $normalized_degree(n, P)$ based on multiple knowledge graphs, and we are going to investigate on this problem in our future study⁹. Since the unification of Web scale search and reasoning from the viewpoint of granularity brings many human problem solving strategies to Web reasoning, it can be considered as an effort towards Web intelligence [50].

⁹Further investigations can be tracked through the USer-G (Unifying Search and Reasoning from the viewpoint of Granularity) website <http://www.iwici.org/user-g>



6. Anytime Classification by Ontology Approximation

6.1 INTRODUCTION

Motivation Since the introduction of anytime algorithms in [6] it has become widely accepted that they are attractive for many reasoning tasks in AI [52]. Instead of producing the perfect answer after a long period of computation, they allow a reasoning task to progress gradually, producing output of increasing quality as runtime progresses. This allows to produce meaningful output under time-pressure, and to save time for applications where an approximate answer is already sufficient.

A recent set of reasoning challenges has been posed to AI by Semantic Web applications. These applications typically use very large or complex ontologies for purposes of searching information on the Web, personalising Web sites, matchmaking between web-services, etc. They rely on subsumption reasoning in languages based on Description Logics [15]. Many of these Semantic Web applications are performed under time pressure (e.g. because of user-interaction), and often approximate answers are sufficient. Given the incomplete and noisy nature of the data on the Web, many user-queries do not require exact and complete answers

This raises the question *whether we can develop an anytime algorithm for subsumption-based classification*.

Approach: The basic intuition of our approach will be to select a subset of the vocabulary of an ontology, to perform classification only with this limited vocabulary, and to gradually increase the selected vocabulary during the runtime of the algorithm. This will yield an anytime algorithm that produces sound but incomplete results for classifying a given ontology, with increasing completeness as the algorithm progresses. A key ingredient of this approach is the strategy to select the subset of the vocabulary. We will empirically investigate the behaviour of a number of different strategies.

Related work: The basic intuition of selecting a subset of the vocabulary can be formalised in terms of approximate deduction, given in [35]. This foundation has been used in earlier work on approximate subsumption by [13] and [38], and in fact we will apply a rewrite procedure defined in [38]. The essential difference with our approach is that in [13, 38] the approximation is used to reformulate the *queries*, whereas we use it to approximate the *ontology*. The results in [13] are mostly negative, while [38] does not report any empirical results. Our experiments show that approximating the ontology produces much better anytime behaviour than the results on approximating the query reported.

Experiment and main findings: The main parameter determining the anytime behaviour of our algorithm is the strategy used to select the increasing subset of the vocabulary used in the classification. In our experiments we will measure the performance of four different selection strategies on eight ontologies which have been chosen to cover a range of size and complexity. The performance of the strategies is measured in terms of the recall obtained vs. the runtime required on increasing subsets of the ontology vocabulary.

Our experiments will show that some approximation strategies do indeed give good anytime behaviour, i.e. for a small percentage of the run-time, we already obtain a large percentage of the answers. Most interestingly, it will turn out that anytime classification works best where it is most needed: on ontologies where classical subsumption is hardest to compute.



Contributions and structure of this chapter: The main contributions of this research are (a) *formal definitions* of a sound and incomplete form of terminological reasoning based on vocabulary selection (based on [38], section 6.2) (b) an *algorithm and heuristics* for anytime behaviour for approximation classification reasoning (also section 6.2) (c) *experiments* that show attractive performance profiles for our approach, and that investigate which factors influence the anytime behaviour (sections 6.3 and 6.4).

6.2 A SOUND APPROXIMATION FOR CLASSIFICATION

In this section we will define a sound and incomplete approximation for terminological reasoning. This will then be the basis for an anytime classification algorithm.

Our definitions will be based on the usual syntax and semantics for the \mathcal{ALC} description logic. An ontology is a set of \mathcal{ALC} axioms of the form $A_i \sqsubseteq B_i$, with A_i and B_i built from atomic concepts, conjunction, disjunction, negation, and universally and existentially quantified role expressions. See [4] for details.

The basic intuition of our approximation is that if we rewrite an \mathcal{ALC} theory T to a weaker theory T' , then establishing $T' \models C \sqsubseteq D$ is sufficient to establish $T \models C \sqsubseteq D$ (but not vice versa). In other words, entailment under T' is a sound (but incomplete) approximation of entailment under T .

We will now define a rewrite procedure that we will apply to every axiom $A_i \sqsubseteq B_i$ of T in order to obtain T' . Following the ideas of Cadoli & Schaerf [35] we define an *approximation set* S , consisting of a subset of the atoms from T . The rewrite procedure will restrict the vocabulary of T only to atoms that appear in S . The rewrite procedure constructs a lower approximation $(\cdot)^{S-}$ by replacing atoms not in S with \perp , and an upper approximation $(\cdot)^{S+}$ by replacing atoms not in S with \top :

Definition 1 (Rewrite Procedure [38])

The rewrite procedures $(\cdot)^S$, $(\cdot)^{S+}$ and $(\cdot)^{S-}$ are defined as follows:

$$\begin{aligned}
 (A \sqsubseteq B)^S &= A^{S-} \sqsubseteq B^{S+} \\
 A^{S-} &= A \text{ if } A \in S & A^{S+} &= A \text{ if } A \in S \\
 A^{S-} &= \perp \text{ if } A \notin S & A^{S+} &= \top \text{ if } A \notin S \\
 (\neg C)^{S-} &= \neg C^{S+} & (\neg C)^{S+} &= \neg C^{S-} \\
 (C \sqcap D)^{S-} &= C^{S-} \sqcap D^{S-} & (C \sqcap D)^{S+} &= C^{S+} \sqcap D^{S+} \\
 (C \sqcup D)^{S-} &= C^{S-} \sqcup D^{S-} & (C \sqcup D)^{S+} &= C^{S+} \sqcup D^{S+} \\
 (\exists R.C)^{S-} &= \exists R.C^{S-} & (\exists R.C)^{S+} &= \exists R.C^{S+} \\
 (\forall R.C)^{S-} &= \forall R.C^{S-} & (\forall R.C)^{S+} &= \forall R.C^{S+}
 \end{aligned}$$

It is easy to see that both $(\cdot)^{+S}$ and $(\cdot)^{-S}$ terminate on any concept C in \mathcal{ALC} as the complexity of the formula decreases in any application of the rules¹.

The theory T^S is obtained by applying this rewrite procedure to every axiom in T . This rewrite procedure equals the one proposed in [38]. However the essential difference is that in [38] this procedure is applied to approximate the *queries* (i.e. $T \models \phi^S$), whereas we use it to approximate the *ontology* (i.e. $T^S \models \phi$).

¹The rewrite procedure can be trivially extended to concept abstraction in OWL DL



The following property is crucial to establish that T^S is a sound approximation of T :

Theorem 1 (From [38])

For any formula $A_i \sqsubseteq B_i$: if $A_i \sqsubseteq B_i$ then $A_i^{S^-} \sqsubseteq B_i^{S^+}$.

The intuition behind this is that $A_i^{S^-} \sqsubseteq A_i$, (since the atoms in A_i not listed in S have been replaced by \perp), and $B_i \sqsubseteq B_i^{S^+}$ (since the atoms in B_i not listed in S have been replaced by \top). The full proof of this is given in [36]. From this the following is immediate:

Corrolary 1 (Soundness) *If $T^S \models C \sqsubseteq D$ then $T \models C \sqsubseteq D$*

It is also easy to see that if $S = \emptyset$, T^S is reduced to the empty (trivial) theory, entailing only tautologies. Similarly, if S contains all atoms from T then the rewrite operation is the identity, and the consequences of T^S equal those of T . In general, if S grows, the entailments from T^S become more complete:

Theorem 2 (Monotonicity) *If $S_1 \subseteq S_2$ then*

$T^{S_1} \models C \sqsubseteq D$ entails $T^{S_2} \models C \sqsubseteq D$.

This is because any model for T^{S_2} is necessarily also a model for T^{S_1} . A full proof is given in [36].

Anytime classification algorithm: We can now obtain an anytime algorithm for classifying T by starting out with classifying T^S for an initial (typically small) set S . We then increase S and repeat the procedure until either (a) the quality of the classification is sufficient for our purposes, or (b) we run out of available computing time, or (c) S contains all atoms from T .

Theorem 2 guarantees that the output of this algorithm monotonically improves during the iteration, as is typically required of anytime algorithms [52]

We illustrate all this with a small example.

Example 1 *Let $T = \{A \sqsubseteq B \sqcap C, B \sqsubseteq D\}$.*

If we take as successive values

$S = \emptyset, \{B\}, \{A, B\}, \{A, B, D\}, \{A, B, C, D\}$,

then the rewriting procedure produces approximate theories T^S as follows:

$$\begin{aligned}
 T^\emptyset &= \{\perp \sqsubseteq \top \sqcap \top, \perp \sqsubseteq \top\} \\
 T^{\{B\}} &= \{\perp \sqsubseteq B \sqcap \top, B \sqsubseteq \top\} \\
 T^{\{A,B\}} &= \{A \sqsubseteq B \sqcap \top, B \sqsubseteq \top\} \\
 T^{\{A,B,D\}} &= \{A \sqsubseteq B \sqcap \top, B \sqsubseteq D\} \\
 T^{\{A,B,C,D\}} &= T
 \end{aligned}$$

To show that the theories T^S for increasing S are a sequence of sound and increasingly less incomplete approximations for T , we list all their atomic non-tautological entailments.

$$\begin{aligned}
 T^\emptyset &: \emptyset \\
 T^{\{B\}} &: \emptyset \\
 T^{\{A,B\}} &: \{A \sqsubseteq B\} \\
 T^{\{A,B,D\}} &: \{A \sqsubseteq B, A \sqsubseteq D, B \sqsubseteq D\} \\
 T^{\{A,B,C,D\}} &: \{A \sqsubseteq B, A \sqsubseteq D, B \sqsubseteq D, A \sqsubseteq C\}
 \end{aligned}$$

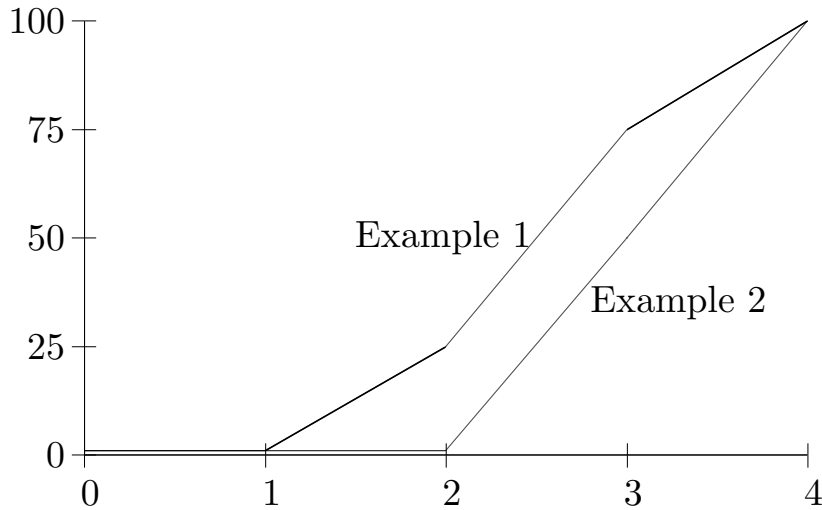


Figure 6.1: anytime performance profile from examples 1 and 2

This example illustrates that for small values of S , T^S is a very incomplete approximation of T ; with increasing S , T^S becomes a less incomplete approximation; and when S contains all atoms from T , T^S is no longer an approximation, but simply equals T .

The anytime classification algorithm given above increases S in successive iterations. The choice of how to increment S determines how quickly the approximation approaches the classical result. This is shown in the following example:

Example 2 Let T be the same as in ex. 1, but now with the sequence

$$S = \emptyset, \{D\}, \{C, D\}, \{A, C, D\}, \{A, B, C, D\}.$$

This yields the following set of atomic non-tautological entailments:

$$\begin{aligned} T^\emptyset & : \emptyset \\ T^{\{D\}} & : \emptyset \\ T^{\{C,D\}} & : \emptyset \\ T^{\{A,C,D\}} & : \{A \sqsubseteq C, A \sqsubseteq D\} \\ T^{\{A,B,C,D\}} & : \{A \sqsubseteq C, A \sqsubseteq D, A \sqsubseteq B, B \sqsubseteq D\} \end{aligned}$$

Figure 6.1 plots the anytime performance profiles for both approximation strategies. For each size of S it plots the percentage of non-tautological atomic subsumptions entailed by T^S . The first strategy has a more attractive anytime behaviour than the second, since it already obtains a higher degree of completeness for smaller values of S . This shows that the strategy for choosing successive values of S is crucial in determining the behaviour of the anytime algorithm.

6.3 EXPERIMENTAL SETUP

The example from the previous section illustrates that different approximation strategies for the set S result in different anytime behaviours of the classification algorithm. This raises the question on what would be a good approximation strategy. In this section, we will define three different strategies and we will investigate their resulting anytime behaviour on a number of realistic datasets.



6.3.1 Approximation Strategies

In our experimentation, we tested eight selection functions, namely:

- **RANDOM**: select concept-names in a random order. This strategy is included as a baseline measurement.
- **MORE, LESS**: select the most or least often occurring concept-name first. The intuition behind MORE is that by choosing the most frequently occurring concept names first we can quickly get a “general” view of the overall classification hierarchy, which can then be successively refined by using less frequently occurring symbols in later iterations. To test this hypothesis, we compare MORE with the exact opposite strategy (LESS).
- **BOTTOM, TOP, MIDDLE**: **BOTTOM** selects the most specific and **TOP** the most general concepts first. Of course, in general we do not precisely know which concepts are most specific or most general, since that is exactly what we want to compute through classification). One would instead need a heuristic oracle that is able to order concepts based on their *heuristically expected* generality. In our experiments, we use the actual classification of the ontologies as the perfect version of such an oracle, allowing us to test the perfect version of these selection functions.
- **MAXLABEL, MINLABEL**: select concept names with the longest or shortest concept-label first. The intuition behind these selection functions is that they are an implementation of the **BOTTOM** and **TOP** heuristics: more specific concepts can be expected to have longer descriptive labels.

6.3.2 Datasets

For our experiments we used the following well known ontologies: DICE, a medical terminology used for registering diagnoses in Intensive Care units²; MGED³, an ontology for microarray experiments; UNSPSC⁴, an ontology version of a coding system to classify products and services; FMA⁵, the foundational model of anatomy; and the pizza ontology⁶, used as training example for the OWL language. We also used three ontologies (Kpoly5, Kt4p13 and Kphp5) taken from the DL benchmark⁷.

Because of technical limitations, all ontologies were simplified from their original OWL versions into corresponding *ALC* versions. Consequently, our experiments only give insight into the approximation properties of *ALC* ontologies. The majority of practical OWL ontologies do not go much beyond *ALC*, and running experiments on OWL DL ontologies is planned for future work.

Table 6.2 summarises some properties of these ontologies: the number of axioms, their respective classification time⁸, as well as the number of occurrences of operators.

This table shows that we have chosen a dataset of realistic ontologies of different size (ranging from hundreds of axioms to tens of thousands of axioms), of different

²kik.amc.uva.nl/dice/home.jsp

³mged.sourceforge.net/ontologies/index.php

⁴www.unspsc.org

⁵sig.biostr.washington.edu/projects/fm

⁶www.co-ode.org/ontologies/pizza

⁷dl.kr.org/dl98/comparison/data.html

⁸using RACER Version 1.7.24



	secs.	#Axioms	<i>secs/kAx</i>	# \forall	# \exists	# \sqcap	# \sqcup	# \neg
DICE	60	4859	12.3	3734	5606	1951	784	0
MGED	0.2	792	0.3	0	171	12	5	12
UNSPSC	7	19590	0.4	0	0	0	0	0
FMA	50	3824	13.1	9348	17280	2654	2527	0
PIZZA	0.2	1052	0.2	23	148	796	26	796
Kpoly5	4	317	12.6	0	202	114	0	163
Kt4p13	5	410	12.2	0	289	120	0	224
Kphp5	8	242	33.1	0	62	179	0	213

Figure 6.2: Some properties of the ontologies used in our experiments

logical expressivity (with very different use of the logical connectives), and of different cost (with classification times ranging from tenths of seconds to tens of seconds).

Notice that classification time does not depend only on the number of axioms, but is also greatly determined by the logical complexity of the ontology: DICE has 4 times fewer axioms than UNSPSC, but requires an order of magnitude more classification time. This is caused by the differences in logical expressivity. We use the ratio of secs/axioms as a rough measure of the complexity of an ontology (scaled to seconds per 1000 axioms for readability).

6.3.3 Performance Measure

Quality Measure: To study the effect of our approximation method, and of the choice of the approximation set, we need an appropriate performance measure. Since our algorithm is a sound but incomplete approximation we need to somehow measure the “degree of completeness” of an answer⁹. As in the examples 1 and 2 from section 6.2 we will use the number of entailed atomic subsumption relations and take this as a percentage of the number of atomic subsumptions entailed classically.¹⁰

Intuitively, this measures the percentage of atomic subsumption queries that can be answered correctly using the approximated ontology.

The advantage of this performance metric is that it puts a small penalty on mistakes in the details lower down in the ontology, and a high penalty on mistakes in the important top-level categorisations. Subsumptions higher in the hierarchy are considered more important, since they are involved in entailing more pairwise subsumptions (because of the tree-structure of most ontologies). This is illustrated in the following toy example:

Example 3

⁹Following standard accepted terminology, we will use the shorter term “recall” for “degree of completeness”. Notice that since our approximation is only incomplete, but always sound (theorem 1), our precision will always be 100% and need not be measured.

¹⁰From here on, we include tautological atomic entailments in our recall measure. This constant factor (the two entailments $\perp \sqsubseteq A$ and $A \sqsubseteq \top$ for every atom $A \in S$) will not influence our measurements in an essential way.

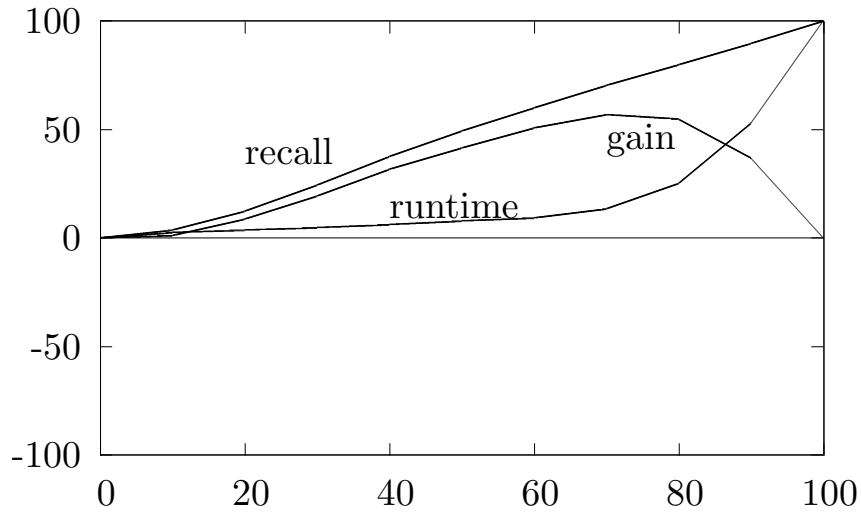
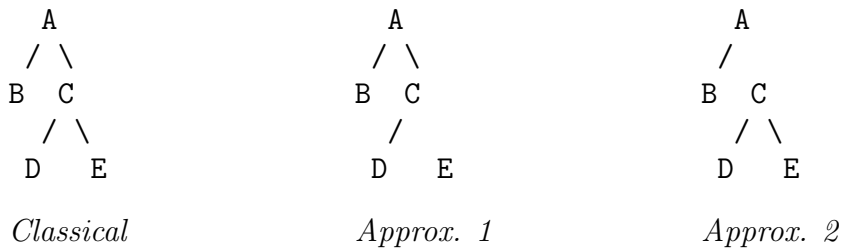


Figure 6.3: Results for the MORE strategy on the DICE ontology



Both Approximation 1 and Approximation 2 miss to compute a single link in the hierarchy. However, because the missing link in Approximation 2 is higher in the hierarchy, the recall of Approximation 2, measured by the number of entailed atomic subsumption is lower: Classically, the following 6 atomic subsumption are entailed: $B \sqsubseteq A, C \sqsubseteq A, D \sqsubseteq A, E \sqsubseteq A, D \sqsubseteq C, E \sqsubseteq C$. All of these are also entailed by Approximation 1, with the exception of $E \sqsubseteq C$ and $E \sqsubseteq A$, giving a recall of $4/6=66\%$. Approximation 2 however only entails 3 atomic subsumptions ($B \sqsubseteq A, D \sqsubseteq C, E \sqsubseteq C$), giving a recall of only $3/6=50\%$, showing that a single missing subsumption higher in the hierarchy leads to a lower recall than a single missing subsumption lower in the hierarchy.

Cost Measure: As our cost measure, we will simply take the run-time of the approximate classification task, taken as a percentage of the computation time of the classical algorithm.

Performance Measure: As overall performance measure we will take the difference between the quality measure (“gain”) and cost measure (“pain”). These will be plotted in a “pain/gain diagram” as a function of the increasing size of the approximation set S (again taken as percentage of the total vocabulary of T).

Figure 6.3 illustrates these measures¹¹: as the size of S increases along the x-axis from 0-100%¹², both the recall (gain) and the runtime (pain) also increase from 0-100%,

¹¹In fact, this figure plots the curves for one of the experiments to be discussed in section 6.4, namely running the MORE strategy on the DICE ontology

¹²In all our experiments we used fixed increments of S with 10%. This somewhat arbitrary step-size is a pragmatic trade-off between the resolution of our plots and the costs of running the experiments



plotted on the y -axis. The combined performance measure (gain-curve) is calculated as the difference between these two.

The *ideal recall* curve rises sharply at small values of S (convex), while the *ideal runtime* curve only starts to increase significantly at large values of S (concave). Together, these would produce an *ideal gain* curve that rises sharply at small values of S representing the desired outcome of a high recall and low runtime in the early stages of the algorithms.

Although such a convex gain-curve is the most ideal, even a flat gain curve at $y = 0$ is already attractive, because it indicates that the gains grow proportionally with costs, giving still an attractive anytime behaviour.

In figure 6.3 the recall-function is everywhere above the runtime-function hence there is a positive gain everywhere. If the anytime algorithm would cost more relative runtime than it would yield in relative recall, the gain curve would fall below the $y = 0$ mark, indicating that the pain is greater than the gain, resulting in negative gains.

Notice that gain-curves always start in $(0,0)$, since for the empty vocabulary both runtime and recall are 0, hence their difference is 0. Gain curves always ends in $(100,0)$, since for the complete vocabulary both recall and runtime are 100%, hence their difference is again 0.

In our plots, the runtimes are not summed over all previous value for S . Therefore, the gain curve is not a performance profile of an interruptible algorithm, but shows the optimal point for a contract-algorithm (namely the value of S where the gain is maximal). However, [33] has shown that contract algorithms can be converted into interruptible algorithms with a constant overhead of at most a factor 4. Since this is constant factor, this difference can be ignored

Since the gain curve of a strategy summarises how successful the strategy is in obtaining attractive anytime behaviour, we will report the full set of experiments showing only the gain curves for every strategy, and not the separate recall and runtime curves.

6.4 EXPERIMENTAL RESULTS

In this section, we will investigate (1) in which cases anytime classification is effective, and (2) which of our strategies is most effective.

Figure 6.4 summarises the results of our experiments: for each ontology it indicates how attractive the anytime behaviour is under the various strategies. The amounts of '+'s summarise the length of the interval where the gain curve is positive and how high this value is. The full gain curves for all experiments can be found in the appendix. From these results we can observe the following:

Anytime classification benefits some cases: The plots in the appendix show a number of cases with very attractive anytime behaviour: on FMA, all strategies have an always-positive gain curve. This confirms our hope that anytime classification is a useful reasoning method for large and complex ontologies. Other examples are Kphp5, where again all strategies have positive gains everywhere; kt4p13, where all strategies except LESS have positive gains; and kpoly5 where again all strategies have positive gains everywhere, although the gains are smaller. Less positive but still attractive is the behaviour on DICE, where all strategies are positive up to the 90% point. Even strategies that score around the 0 gain mark (e.g. most of the other strategies on



ontology	secs/ax	R	M	L	B	C	T	MAX	MIN
Kphp5	33.1	+	+	+++	+++	+	+	+	+
FMA	13.1	+	+	+	+++	+	+	+	+
Kpoly5	12.6	+++	+	+	+++	+	+	+	+
DICE	12.3	+	+++	+	+	+	+	+	+
Kt4p13	12.2	+	+++	-	---	-	+	---	+
UNSPSC	0.4	--	-	---	---	---	---	---	---
MGED	0.3	--	-	---	---	---	---	---	---
PIZZA	0.2	--	---	-	---	---	---	---	---

R= RANDOM, M= MORE, L= LESS, B=BOTTOM, C=CENTER, T=TOP,
 MAX=MAXLABEL, MIN=MINLABEL

Figure 6.4: Summary of success and failure of the different strategies.

FMA) are still attractive because this means that the recall grows proportionally with the runtime while increasing S , hence still giving a nice anytime behaviour.

Anytime classification doesn't benefit all cases: However, we also see cases with less attractive anytime behaviour. On UNSPSC and MGED all strategies have negative gains everywhere. This means that more time is lost than correct answers are found.

When does anytime classification benefit? Figure 6.4 is sorted by the classification-complexity of the ontologies (as rated by their secs/axiom score, see figure 6.2). From this it is immediately apparent that approximate classification has most success on complex ontologies. This is of course very good news: *Approximate classification works best in exactly those cases when it is needed most*, namely for those ontologies that are expensive to classify with classical algorithms.

Which strategy performs best? Our data is inconclusive on the question which strategy performs best. Figure 6.4 shows weak evidence to suggest that the MORE and BOTTOM strategies perform best, although each of them is outperformed by others on some of the ontologies. Furthermore, it is noticeable that these two winning strategies are somehow complementary, performing best on different cases. Finally, even though MAXLABEL and MINLABEL were introduced as heuristic estimates for the TOP and BOTTOM strategies respectively, figure 6.4 shows almost no correlation between these, suggesting that label-length is not a good estimator of concept specificity.

Contract algorithms vs. interruptible algorithms. An important distinction between different types of anytime algorithms is that between contract and interruptible algorithms [52]. Interruptible algorithms are anytime because they can be interrupted at any point during their execution, at which point they give the best answer available at the time. Contract algorithms instead must be given *in advance* what their maximal allowed resources are, and then aim to compute the best possible result with those available resources.

If we regard our iterative algorithm as an interruptible algorithm, the performance must be measured against the *cumulated* values of the runtimes of all iterations over increasing S before the interrupt. If we aim for a contract algorithm, it is sufficient to set an appropriate size for S at the start of the algorithm, and measure only the runtime of that single computation.

All the results above are based on non-cumulative runtimes, and hence are valid



ontology	MORE			BOTTOM		
	TP	recall at TP	max. runtime	TP	recall at TP	max. runtime
Kphp5	90%	63%	137%	90%	85%	123%
FMA	60%	46%	336%	70%	82%	287%
Kpoly5	70%	45%	304%	90%	76%	148%
DICE	80%	80%	225%	80%	30%	211%
Kt4p13	80%	65%	263%	90%	6%	143%

Figure 6.5: Interruptible behaviour

for a contract algorithm. A contract algorithm can be trivially turned into an interruptible algorithm by simply iterating the contract algorithm for increasing sizes of S , and simply accumulating the runtimes of all iterations. Of course, this only makes sense up to the point where the accumulated iterative runtime begins to exceed the classical runtime (at which point we would be better off with the classical algorithm, giving us full recall at $S=100\%$). We will use the term *tipping point* for the size of S at which the accumulated iterative runtime starts to exceed the classical runtime. The tipping point is the maximal S at which the iterated contract algorithm can be used as an interruptible algorithm. Obviously, higher values for this tipping point are more attractive. Besides knowing the size of S at which the tipping point occurs, one would like to know the recall that can be achieved at this tipping point (this being the maximal recall obtainable by a naive interruptible algorithm, since beyond this tipping point, we would be better off running the classical algorithm just once). The table of figure 6.5 shows for the various experiments the values of the tipping point (as percentage of S), and the corresponding recall obtained at this tipping point. The results in this table are again very encouraging: in general, the tipping point is reached only at large sizes of S : 60% and upwards, with an average of 80%. This means that for a considerable range of S , it is still useful to deploy the iterated contract algorithm as an interruptible algorithm. Furthermore, at this tipping point, a reasonable recall is already obtained: almost always upwards of 45%, with an average of 57%. (The single exception to this is the very low tipping-point recall of the BOTTOM strategy on Kt4p13, which is consistent with the corresponding entry in figure 6.4). The third column for each strategy in figure 6.5 measures the costs of the “worst case” scenario: comparing the total accumulated runtime of all iterations up to $S = 100\%$ against a single run at $S = 100\%$ (= the runtime of the classical algorithm). Again, this shows rather favourable numbers: even by applying the naive iterative algorithm up to the worst case (i.e. when no intermediate interrupts justified the iterative computation), the costs are never high than a factor of ± 3 , with an average of a factor of 2.3.

Choice of the control parameter. The design of every anytime algorithm is characterised by the choice of three parameters: the *cost parameter* (“pain”, in our case: runtime), the *benefit parameter* (“gain”, in our case: recall), and a *control parameter* which is used as the dial to set the trade-off between cost and gains. Zilberstein [52] formulates two properties that should hold for the cost-benefit trade-off of every anytime algorithm:

- *monotonicity*: benefits should increase monotonically with cost
- *diminishing returns*: early increases of cost should yield higher increases in benefit than later increases.

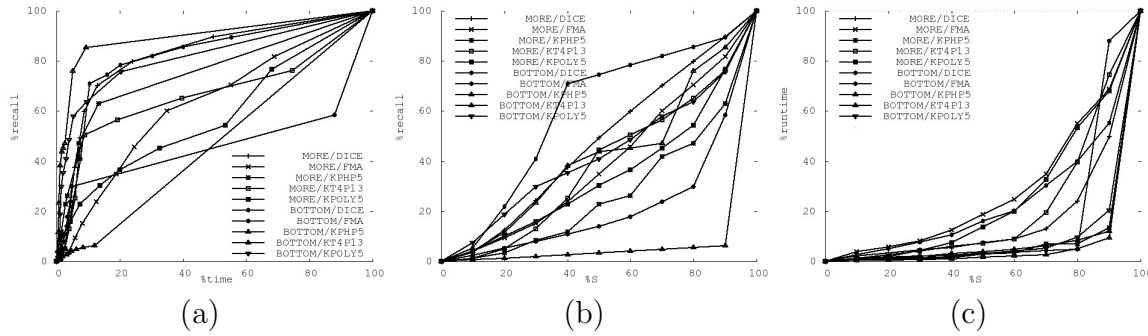


Figure 6.6:

Figure 6.6(a) shows these properties do indeed hold for all the “interesting cases”: our two most promising strategies (MORE and BOTTOM) on the 5 complex ontologies from the top half of figure 6.4) all display a nicely convex curve.

In many anytime settings, runtime is used as the cost parameter. However, in a practical algorithm, one can often not control the runtime explicitly, but instead another quantity (the control parameter) is used as a substitute measure for costs. This control parameter must be chosen in such a way that (1) it can be explicitly controlled in the algorithm, and (2) it reflects the actual runtime costs of the algorithm. In our algorithm the set S plays the role of this control parameter, the intuition being that runtime increases with S . The question is then whether the desired properties of monotonicity and diminishing returns still hold for the relation between control-parameter and gains (instead of the relation between cost parameter and gains, shown in fig. 6.6(a)). Figure 6.6(b) shows that the relation S vs. recall is still monotonic (as already guaranteed by theorem 1), but that it does not satisfy diminishing returns: most of the curves are concave. Figure 6.6(c) gives us the explanation for this: it shows that runtime does not grow linearly with S ; in fact the runtime only starts to increase at very high values of S .

This is actually an attractive property, since apparently recall already increases even for low values of S (fig. 6.6(b)), while runtime only starts to increase at high values of S (fig. 6.6(c)). This behaviour suggests that our current linear iteration over S (in steps of 10%) is not the optimal approach. Instead, we can use a greedy approach, where early iterations choose larger slices of S : 50%, 75%, 87.5%, etc. This has two benefits: (1) recall becomes higher in the earlier iterations, while figure 6.6(c) ensures us that costs will still remain low; and (2) the steep jump in recall at high values of S shown in figure 6.6(b) is more equally spread out over multiple iterations. Together, these two effects cause the graph to become convex, hence satisfying the requirement of diminishing returns. We have experimentally confirmed these hypothesis.

6.5 CONCLUSIONS

In this chapter we have presented a method for approximate classification based on selecting only part of the set of atoms in an ontology. By incrementally increasing this set, we can construct an anytime classification algorithm, which yields sound and increasingly complete answers w.r.t. classical reasoning.

Theorems and algorithm: We have given theorems that establish the soundness



and monotonicity of the algorithm, and we have shown how a classical inference procedure can be used to compute the approximate classification, by applying the classical inference procedure to an approximated version of the ontology. The behaviour of this algorithm is dependent on the strategy for selecting the subset of atoms taken into account by the approximation.

Experiments: We have tested our anytime classification method with a rich set of heuristics on a set of 8 realistic benchmark ontologies, comparing its performance against classical reasoning in terms of runtime and recall (since the approximation is sound, precision is always guaranteed).

Main Findings: Our experiments show that the algorithm is indeed a well-behaved anytime algorithm with monotonic gains and diminishing returns. Furthermore, significant gains can indeed be obtained. In many cases, the algorithm already achieves a high recall against only a low runtime in its first few approximations. We have also argued that the performance of our contract algorithm is still attractive when deployed as an interruptible algorithm, even if this is done in the naive way. The approximation works best on very complex ontologies, namely those that are expensive to classify with traditional methods. *This shows that our algorithm works best in the cases where it is needed most.*

Future work: This research has shown that it is indeed possible to obtain attractive anytime behaviour for classification reasoning. Future work should aim at the following. First, our rewrite procedure can be trivially extended from \mathcal{ALC} to OWL as long as only concepts are being approximated. The rewrite procedure should be extended to include role approximation, and experiments should be performed with OWL ontologies. Second, we should develop a truly incremental algorithm, where results of the previous iterations are used to compute the results of later iterations. This would result in a non-naive interruptible algorithm, improving over our results in figure 6.5. A major future task is the quest for even more effective strategies. In particular, we would expect that strategies which exploit properties of the task for which the ontology is deployed would bring major benefits. Finally, it remains to be investigated how these strategies behave in realistic Semantic Web applications.

6.6 Proofs

In this section we will sketch the formal justification for the soundness and incompleteness of our approximation, as well as for the monotonicity of our approximation with increasing size of S .

The general intuition will be that we will define approximate models for a theory T , and a corresponding notion of approximate entailment (\models_S) using these approximate models. Since (as we will show) any approximate model is also a classical model (but not vice versa), establishing approximate entailment \models_S implies classical entailment \models , giving a sound but incomplete approximation of classical entailment.

In a second step, we will then show that establishing \models_S for a theory (T) is equivalent to establishing \models for an approximation of the theory (T^S). This result will enable us to compute approximate entailment \models_S on a theory T by calling a classical reasoner on the approximated version T^S of the theory (corollary 1).

First, we must then define the notion of an approximate interpretation, using these to define approximate models, and using this to define approximate entailment. The following definition of lower and upper S -approximation closely follows [35]:



Definition 2 (Approximate Interpretations)

The lower approximate interpretation $(.)^{\mathcal{I}_S^-}$ and the upper approximate interpretation $(.)^{\mathcal{I}_S^+}$ are defined as follows:

$$\begin{aligned}
 A^{\mathcal{I}_S^-} &= A^{\mathcal{I}_S^+} = A^{\mathcal{I}} \text{ if } A \in S \\
 A^{\mathcal{I}_S^-} &= \emptyset \text{ and } A^{\mathcal{I}_S^+} = \mathcal{U} \text{ if } A \notin S \\
 (\neg C)^{\mathcal{I}_S^-} &= \mathcal{U} \setminus C^{\mathcal{I}_S^+} \\
 (\neg C)^{\mathcal{I}_S^+} &= \mathcal{U} \setminus C^{\mathcal{I}_S^-} \\
 (C \sqcap D)^{\mathcal{I}_S^-} &= C^{\mathcal{I}_S^-} \cap D^{\mathcal{I}_S^-} \\
 (C \sqcap D)^{\mathcal{I}_S^+} &= C^{\mathcal{I}_S^+} \cap D^{\mathcal{I}_S^+} \\
 (C \sqcup D)^{\mathcal{I}_S^-} &= C^{\mathcal{I}_S^-} \cup D^{\mathcal{I}_S^-} \\
 (C \sqcup D)^{\mathcal{I}_S^+} &= C^{\mathcal{I}_S^+} \cup D^{\mathcal{I}_S^+} \\
 (\exists R.C)^{\mathcal{I}_S^-} &= \{d \in \mathcal{U} \mid \exists e \in \mathcal{U} : (d, e) \in R^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}_S^-}\} \\
 (\forall R.C)^{\mathcal{I}_S^-} &= \{d \in \mathcal{U} \mid \forall e \in \mathcal{U} : (d, e) \in R^{\mathcal{I}} \text{ implies } e \in C^{\mathcal{I}_S^-}\} \\
 (\exists R.C)^{\mathcal{I}_S^+} &= \{d \in \mathcal{U} \mid \exists e \in \mathcal{U} : (d, e) \in R^{\mathcal{I}} \text{ and } e \in C^{\mathcal{I}_S^+}\} \\
 (\forall R.C)^{\mathcal{I}_S^+} &= \{d \in \mathcal{U} \mid \forall e \in \mathcal{U} : (d, e) \in R^{\mathcal{I}} \text{ implies } e \in C^{\mathcal{I}_S^+}\}
 \end{aligned}$$

The crucial property of these approximate interpretations is that the lower approximation interpretations grow, and upper approximation interpretations shrink with increasing size of S :

Lemma 1 (Generalised Monotonicity) *Given a lower \mathcal{I}_S^- and an upper S -approximate interpretation \mathcal{I}_S^+ , and two sub-vocabularies $S_1 \subseteq S_2$, the following equations hold for all concept expressions C .*

$$1) \ C^{\mathcal{I}_{S_1}^-} \subseteq C^{\mathcal{I}_{S_2}^-} \qquad 2) \ C^{\mathcal{I}_{S_2}^+} \subseteq C^{\mathcal{I}_{S_1}^+}$$

Proof: The proof is by induction over the structure of C . \square

Based on these approximations of an interpretation we can now define the notion of an approximate model for a terminology. The basic intuition is that we slightly release the constraints on the terminology by forcing the left-hand sides (lhs) of axioms to be more specific, and the right-hand side (rhs) of the axioms to be more general, than in the classical case. This can be achieved by considering also interpretations to be models of an axioms, in which the lower approximation of the lhs is a subset of the upper approximation of the rhs.

Definition 3 (S -approximate models) *Let C and D be Description Logic concepts, and T a TBox. An interpretation \mathcal{I} is an S -approximate model of an axiom $C \sqsubseteq D$ if, and only if, $C^{\mathcal{I}_S^-} \subseteq D^{\mathcal{I}_S^+}$. An interpretation \mathcal{I} is an S -approximate model of the TBox T if, and only if, it is an S -approximate model of all axioms $ax \in T$. In this case we write $\mathcal{I} \models_S T$.*

The crucial property of approximate models is that the approximation preserve models, i.e. if an interpretation is a model for an axiom or a TBox, it is also an S -approximate model for any subset S of the vocabulary.



Lemma 2 *If \mathcal{I} is a (classical) model for an axioms ax , it is an S -approximate model for ax for any S . Similarly, if \mathcal{I} is a classical model for a TBox T , it is an S -approximate model for T .*

Proof: As $C^{\mathcal{I}_S^-} \subseteq C^{\mathcal{I}} \subseteq C^{\mathcal{I}_S^+}$ we also have that $C^{\mathcal{I}_S^-} \subseteq C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \subseteq D^{\mathcal{I}_S^+}$ for any axiom $C \sqsubseteq D$ with model \mathcal{I} . \square

Finally we need to consider monotonicity for increasing size of set S to be able to apply an anytime algorithm to approximate classification.

Lemma 3 *Let $S_1 \subseteq S_2$, T be a TBox. Then, $\mathcal{I} \models_{S_2} T$ implies that $\mathcal{I} \models_{S_1} T$.*

Proof: From Lemma 1 we know that $C^{\mathcal{I}_{S_1}^-} \subseteq C^{\mathcal{I}_{S_2}^-}$ and $D^{\mathcal{I}_{S_2}^+} \subseteq D^{\mathcal{I}_{S_1}^+}$, from which Lemma 3 follows immediately. \square

Note that this lemma shows inverse monotonicity for models, i.e. with increasing size of S there are lesser models for any TBox. From this (proper) monotonicity of terminological reasoning follows for classical (not approximated) subsumption'

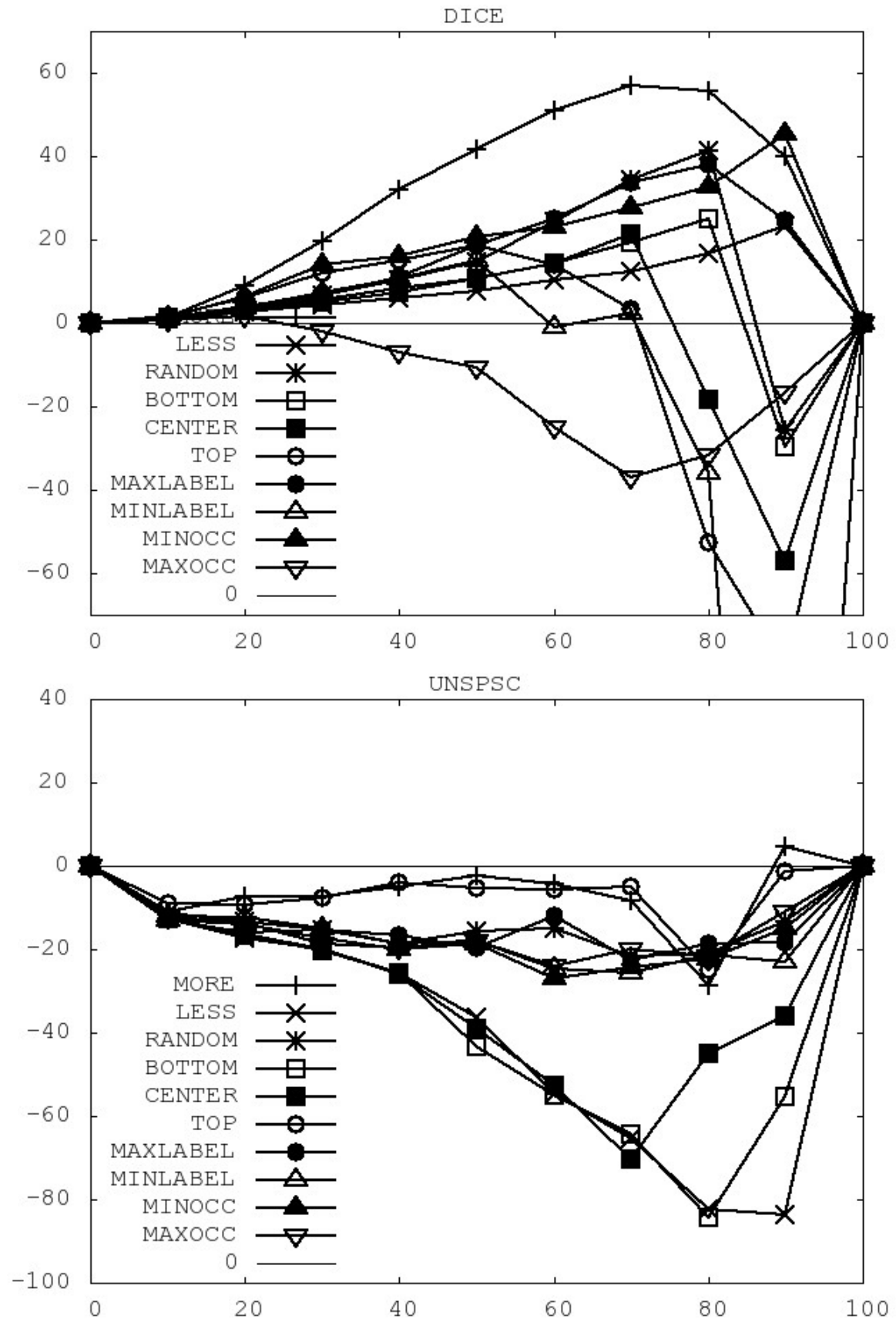
Theorem 3 (Monotonicity of \models_S) *Let $S_1 \subseteq S_2$ and T be a TBox, and C and D DL concepts. We then have that $T \models_{S_1} C \sqsubseteq D$ implies that $T \models_{S_2} C \sqsubseteq D$.*

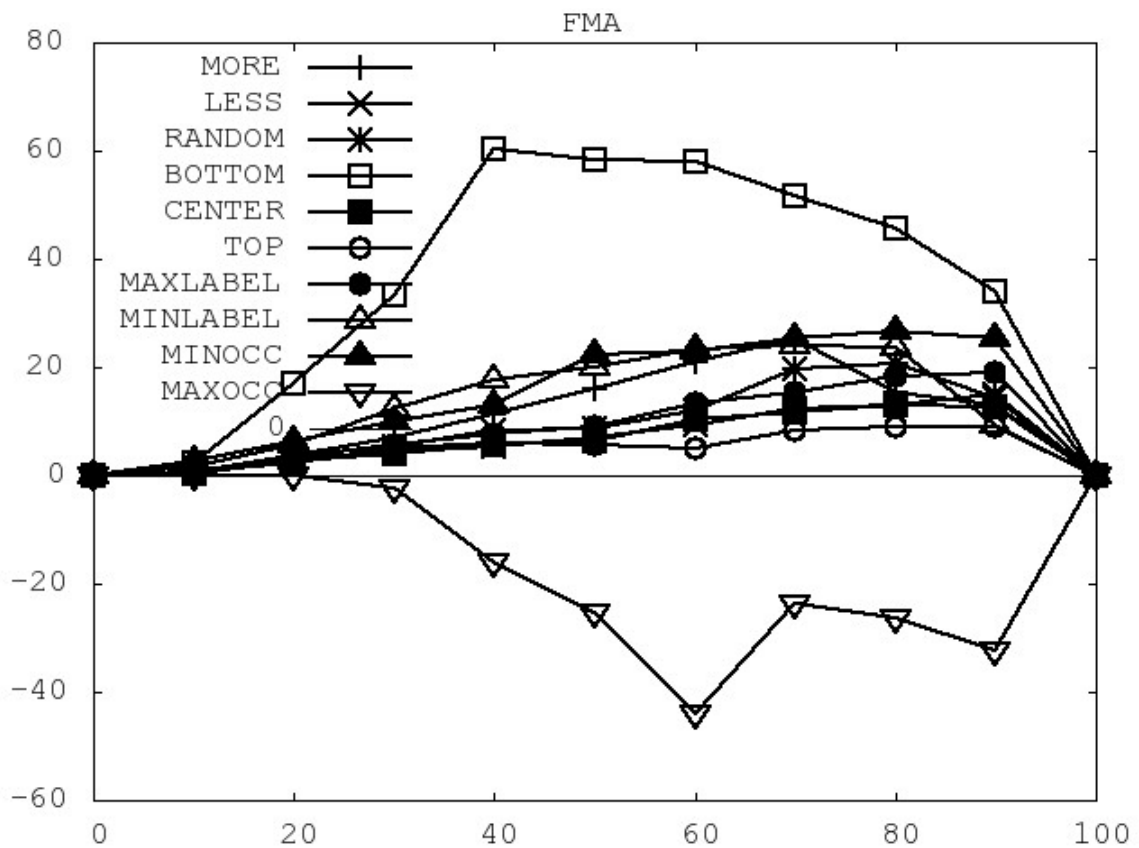
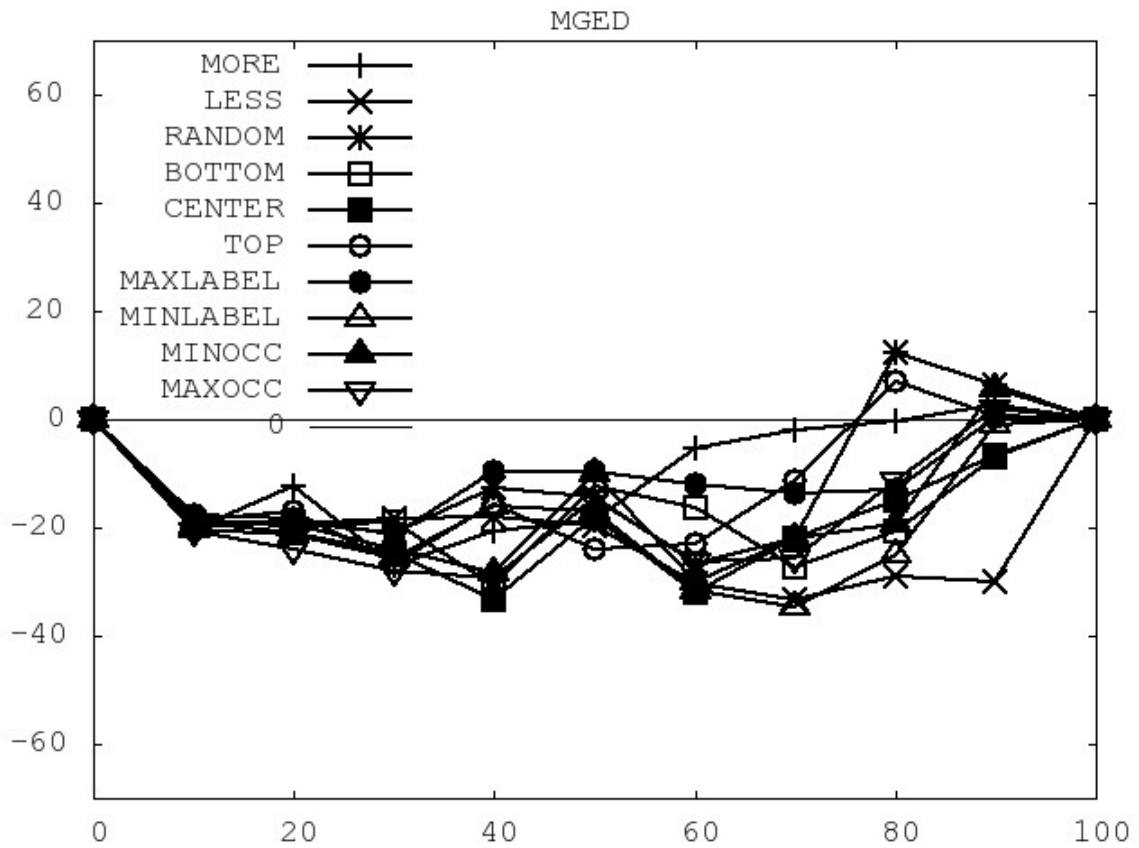
The theorem follows immediately from Lemma 3. This theorem also immediately implies the soundness of approximate subsumption (simply take in the theorem for S_2 the set of all atoms in T):

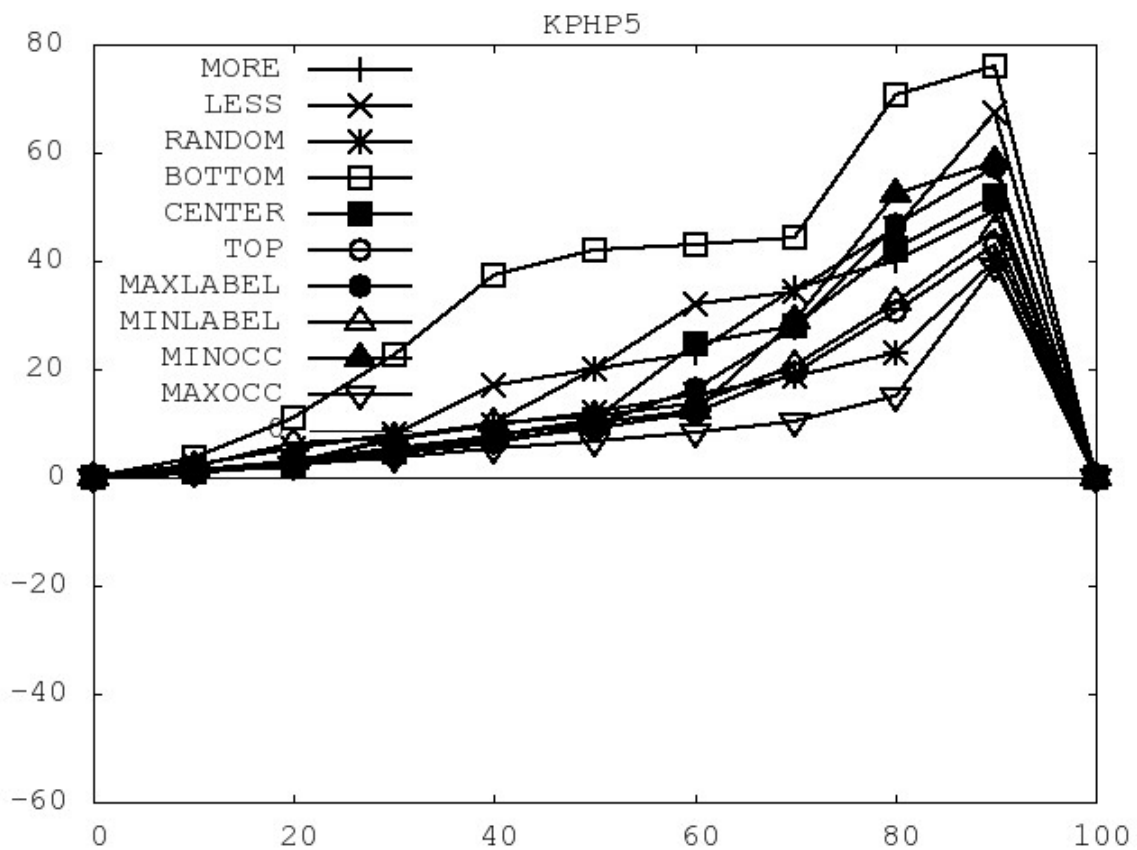
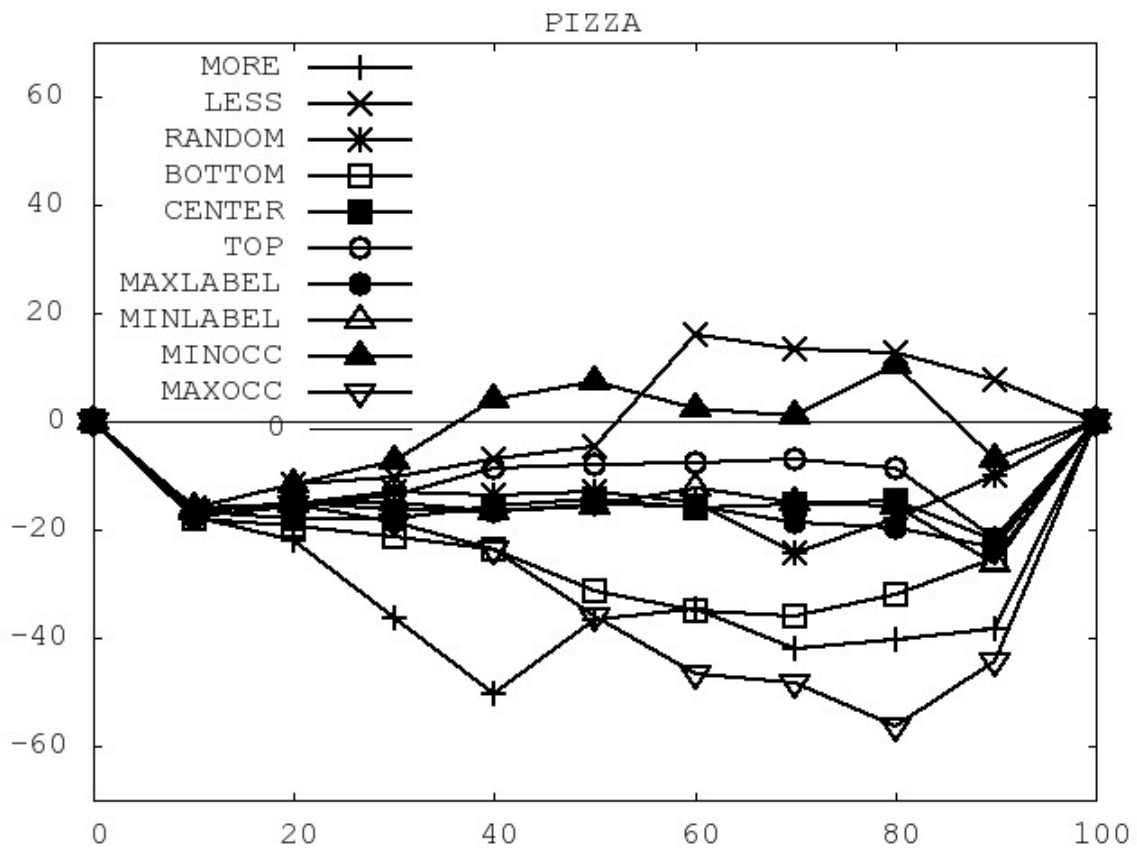
Corrolary 2 (Soundness) *For any S_i , $T \models_{S_i} C \sqsubseteq D$ implies $T \models C \sqsubseteq D$*

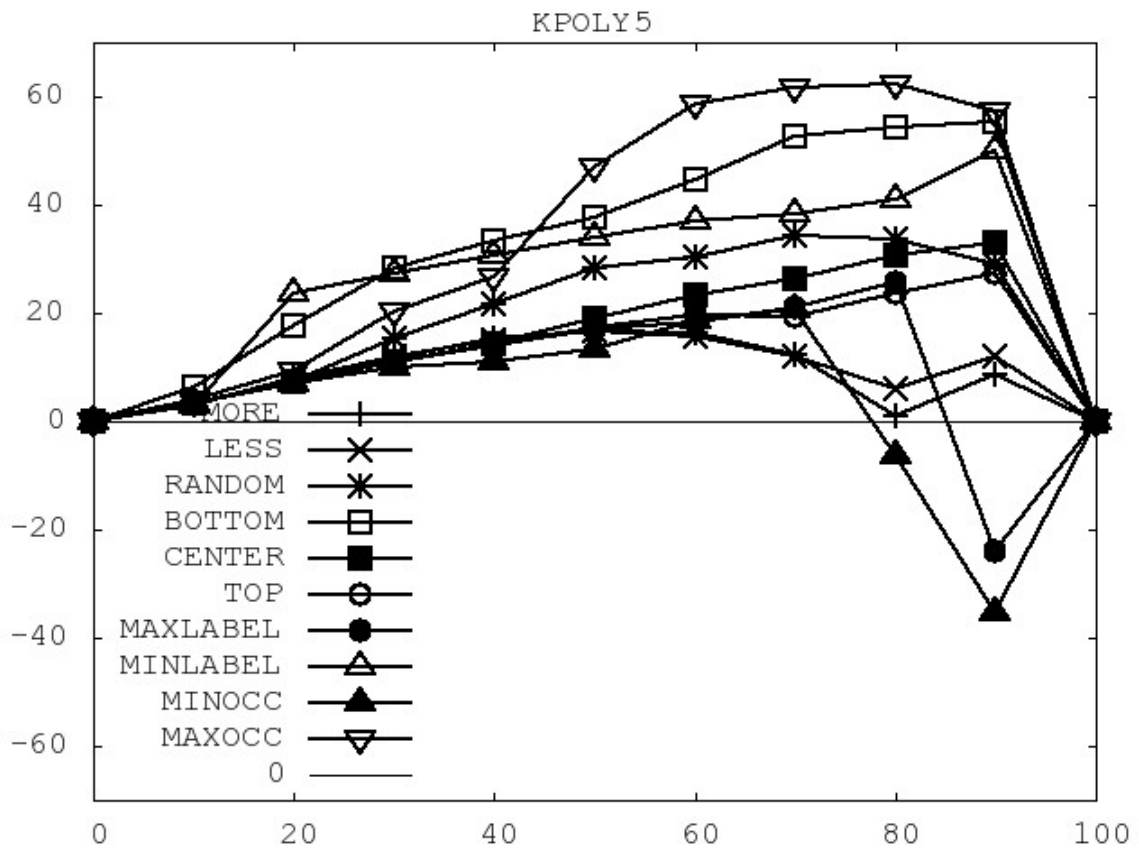
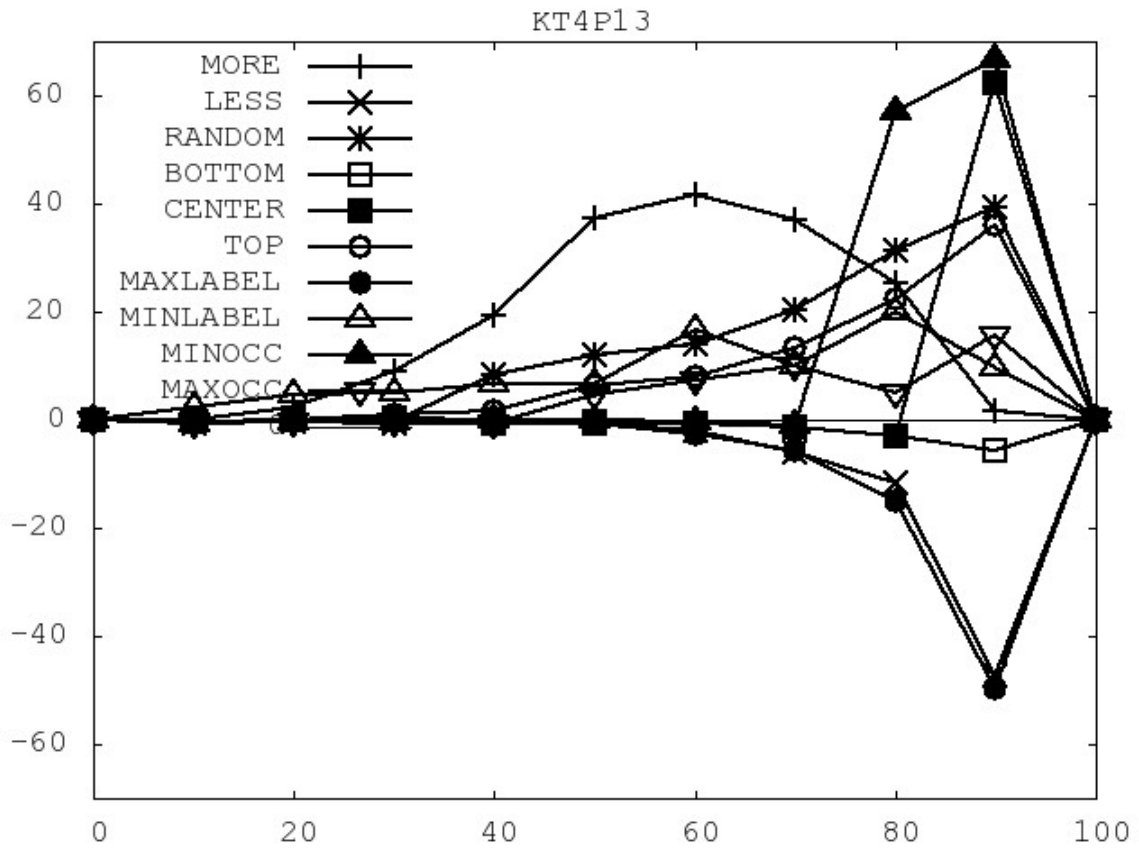
This gives us all the results we need on approximate entailment \models_S . However, we will show that it is possible to rewrite T in such a way that computing approximate entailments on T is equivalent to computing classical entailments \models on a rewritten theory T^S :

Theorem 4 *Let T be a TBox. $T \models_S C \sqsubseteq D$ if, and only if, $T^S \models C \sqsubseteq D$.*











7. Conclusion

In this document, we have developed a general framework of Web scale reasoning, in which various strategies of interleaving reasoning and selection are developed, so that the reasoning processing can focus on limited part of data to improve the scalability of Web scale reasoning. That provides a general approach for interleaving reasoning and selection of axioms. This framework is explored further with three specific selection approaches: an approach of query-based selection and reasoning, an approach of granular reasoning with selection of different perspectives and multiple views from the users, and an approach of approximate reasoning with with selection of different sub-languages.

For the approach of query-based selection and reasoning, we have proposed various selection strategies, which include syntactic-relevance-based selection functions and semantic-relevance-based selection functions. We also explore various strategies of over-determined processing.

For the approach of granular reasoning, we have developed various strategies under the notion of granular reasoning to solve the problems for Web scale reasoning. We bring the strategies of multilevel, multiperspective, starting point to Web scale reasoning. From the multilevel point of view, in order to meet different levels of user needs, we have provided reasoning results with variable completeness and variable specificity. From the multiperspective point of view, reasoning are based on different perspectives of the knowledge source. Reasoning based on starting point utilizes the user background and provides most important reasoning results to users. These strategies and their experiments show that it can provide a satisfying and wide variety of user needs and removing the scalability barriers.

For the approach of approximate reasoning, we have presented an algorithm for classification with anytime behaviour based on approximate subsumption. We give *formal definitions* for approximate subsumption, and show soundness and monotonicity. We have developed *an algorithm and heuristics* to obtain anytime behaviour for classification reasoning. We have explored further the computational behaviour of the algorithm on a set of realistic ontologies. Our *experiments* show attractive performance profiles.



REFERENCES

- [1] W. ACarnielli, 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.
- [2] P. Adjiman, P. Chatalic, F. Goasdoue, M. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *Journal of Artificial Intelligence Research*, 25:269–314, 2006.
- [3] B. Aleman-Meza, F. Hakimpour, I.B. Arpinar, and A.P. Sheth. Swetodblp ontology of computer science publications. *Journal of Web Semantics*, 5(3):151–155, 2007.
- [4] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [5] A. Barabási. *Linked: The New Science of Networks*. Perseus Publishing, 2002.
- [6] Mark S. Boddy and Thomas Dean. Solving time-dependent planning problems. In *IJCAI*, pages 979–984, 1989.
- [7] Alexander Budanitsky and Graeme Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources, 2nd meeting of the North American Chapter of the Association for Computational Linguistics*. Pittsburgh, PA., 2001.
- [8] Samir Chopra, Rohit Parikh, and Renata Wassermann. Approximate belief revision preliminary report. *Journal of IGPL*, 2000.
- [9] R. Cilibrasi and P. Vitány. The Google similarity distance. *IEEE/ACM Transactions on Knowledge and Data Engineering*, 19:3:370–383, 2007.
- [10] Rudi Cilibrasi and Paul Vitányi. Automatic meaning discovery using Google. Technical report, Centre for Mathematics and Computer Science, CWI, 2004.
- [11] A. M. Collins and M. R. Quillian. Retrieval time from semantic memory. *Journal of Verbal Learning & Verbal Behavior*, 8:240–247, 1969.
- [12] D. Fensel and F. van Harmelen. Unifying reasoning and search to web scale. *IEEE Internet Computing*, 11(2):96, 94–95, 2007.
- [13] Perry Groot, Heiner Stuckenschmidt, and Holger Wache. Approximating description logic classification for semantic web reasoning. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2005.
- [14] J. R. Hobbs. Granularity. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 432–435, 1985.
- [15] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.



- [16] H. Horst. Extending the rdfs entailment lemma. In *Proceedings of the 3rd International Semantic Web Conference*, pages 77–91, 2004.
- [17] H. Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Journal of Web Semantics*, 3(2-3):79–115, 2005.
- [18] B. Hoser, A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Semantic network analysis of ontologies. In *Proceedings of the 3rd European Semantic Web Conference*, pages 514–529, 2006.
- [19] 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.
- [20] Z. Huang, F. van Harmelen, A. ten Teije, P. Groot, and C. Visser. Reasoning with inconsistent ontologies: a general framework. Project Report D3.4.1, SEKT, 2004.
- [21] 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.
- [22] Zhisheng Huang and Frank van Harmelen. Using semantic distances for reasoning with inconsistent ontologies. In *Proceedings of the 7th International Semantic Web Conference (ISWC2008)*, 2008.
- [23] J. Jung and J. Euzenat. Towards semantic social networks. In *Proceedings of the 4th European conference on The Semantic Web*, pages 267–280, 2007.
- [24] L. Khreisat and K. Dalal. Anytime reasoning with probabilistic inequalities. In *Proceedings of the 9th International Conference on Tools with Artificial Intelligence*, pages 60–66, 1997.
- [25] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of International Conference on Machine Learning, Madison, Wisconsin, July, 1998*.
- [26] 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.
- [27] R.S. Michalski and P.H. Winston. Variable precision logic. *Artificial Intelligence*, 29(2):121–146, 1986.
- [28] M. Minsky. *The Emotion Machine : commonsense thinking, artificial intelligence, and the future of the human mind*. Simon & Schuster, 2006.
- [29] 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.
- [30] 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.



- [31] E. Oren, C. Gu ereret, and S. Schlobach. Anytime query answering in rdf through evolutionary algorithms. In *Proceedings of the 7th International Semantic Web Conference*, pages 98–113, 2008.
- [32] 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.
- [33] Stuart J. Russell and Shlomo Zilberstein. Composing real-time systems. In *IJCAI*, pages 212–217, 1991.
- [34] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74:249–310, 1995.
- [35] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.
- [36] S. Schlobach, E. Blaauw, M. El Kebir, A. ten Teije, F. van Harmelen, S. Bortoli, M. Hobbelman, K. Millian, Y. Ren, S. Stam, P. Thomassen, R. van het Schip, and W. van Willigem. Anytime classification by ontology approximation. In Ruzica Piskac et al., editor, *Proceedings of the workshop on new forms of reasoning for the Semantic Web: scalable, tolerant and dynamic*, pages 60–74, 2007.
- [37] Heiner Stuckenschmidt. Network analysis as a basis for partitioning class hierarchies. In *Proceedings of the ISWC-2005 Workshop on Semantic Network Analysis*, pages 43–54, 2005.
- [38] Heiner Stuckenschmidt. Partial matchmaking using approximate subsumption. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, 2007.
- [39] Gerd Stumme, Bettina Hoser, Christoph Schmitz, and Harith Alani. *Proceedings of the ISWC 2005 Workshop on Semantic Network Analysis (SNA2005)*. 2005.
- [40] K. Vanderveen and C. Ramamoorthy. Anytime reasoning in first-order logic. In *Proceedings of the 9th International Conference on Tools with Artificial Intelligence*, pages 142–148, 1997.
- [41] W.A. Wickelgren. *Handbook of learning and cognitive processes*, chapter Memory storage dynamics, pages 321–361. Hillsdale, NJ: Lawrence Erlbaum Associates, 1976.
- [42] E.J. Wisniewski and G.L. Murphy. Superordinate and basic category names in discourse: A textual analysis. *Discourse Processing*, 12:245–261, 1989.
- [43] Michael Witbrock, Blaz Fortuna, Luka Bradesko, Mick Kerrigan, Barry Bishop, Frank van Harmelen, Annete ten Teije, Eyal Oren, Vassil Momtchev, Axel Tenschert, Alexey Cheptsov, Sabine Roller, and Georgina Gallizo. D5.3.1 - requirements analysis and report on lessons learned during prototyping, Joune 2009. Available from: <http://www.larkc.eu/deliverables/>.



- [44] 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.
- [45] Y. Y. Yao. Perspectives of granular computing. In *Proceedings of 2005 IEEE International Conference on Granular Computing*, volume 1, pages 85–90, 2005.
- [46] Y. Y. Yao. The art of granular computing. *Lecture Notes in Artificial Intelligence*, 4585:101–112, 2007.
- [47] Y. Y. Yao. *Handbook of Granular Computing*, chapter A unified framework of granular computing, pages 401–410. Wiley, 2008.
- [48] Y. Zeng and N. Zhong. On granular knowledge structures. In *Proceedings of the first International Conference on Advanced Intelligence*, pages 28–33, 2008.
- [49] B. Zhang and L. Zhang. *Theory and Applications of Problem Solving*. Elsevier Science Inc., 1 edition, 1992.
- [50] N. Zhong, J. M. Liu, and Y. Y. Yao, editors. *Web Intelligence*. Springer, 1 edition, 2003.
- [51] 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.
- [52] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.