NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 — "Semantic-based knowledge and content systems"

D1.2.5 Inconsistency-tolerant Reasoning with Networked Ontologies

Deliverable Co-ordinator: **Guilin Qi**

Deliverable Co-ordinating Institution: **University of Karlsruhe**

Other Authors: **Simon Schenk (UKoblenz)**

In deliverable D1.2.4, we discussed the problem of reasoning with inconsistent networked ontologies and proposed several inconsistency tolerant semantics. In this deliverable, we consider the bilattice-based semantics and propose a segmentation-based approach for approximate query over distributed inconsistent ontologies based on the bilattice-based semantics. We implemented this approach and provide some interesting evaluation results.

Document Identifier:	NEON/2010/D1.2.5/v1.0	Date due:	January 31, 2010
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	January 31, 2010
Project start date	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier, France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield, United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Kolenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

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

- UKarl
- UKoblenz

Change Log

Version	Date	Amended by	Changes
0.1	10-01-2010	Guilin Qi	Create the deliverable
0.2	20-01-2010	Simon Schenk	Add chapter on paraconsistent and bi-lattice based reasoning
0.3	25-01-2008	Guilin Qi	Complete the deliverable
0.4	02-02-2008	Guilin Qi	Address reviewer's comments

Executive Summary

Next generation semantic applications are characterized by a large number of ontologies, some of them constantly evolving. As the complexity of semantic applications increases, more and more knowledge are embedded in applications, typically drawn from a wide variety of sources. This new generation of applications thus likely rely on ontologies embedded in a network of already existing ontologies. Ontologies and metadata have to be kept up to date when application environments and users' needs change. One of the major challenges in managing these networked and dynamic ontologies is to handle potential inconsistencies.

In deliverable D1.2.4, we discussed the problem of reasoning with inconsistent networked ontologies. We proposed a four-valued semantics for description logics $\mathcal{ALC}$ which was further extended to a bilattice-based semantics that extends OWL2 to bilattice. We propose another approach for reasoning with distributed ontologies which is based on concept forgetting. This approach is theoretical interesting but is hard to implement in practice.

In D1.5.4 be have proposed reasoning with meta-knowledge, e.g. trust and uncertainty in OWL2. This approach is on the one hand more general than bilattice based reasoning but on the other hand does not allow for paraconsistent reasoning. In this deliverable, we reduce bilattice based reasoning to meta-knowledge reasoning by adjusting the technique developed for $\mathcal{ALC}\triangle$ in D1.2.4. We provide an efficient algorithm for meta-knowledge reasoning.

With the popularity of semantic information systems distributed on the Web, there is an arising challenge to provide efficient query answering support for these systems. However, common approaches for distributed query answering either exhibit performance disadvantages or loss of completeness in an unbalanced way. In this deliverable, we introduce a novel approach for *segment-based conjunctive query answering* over distributed ontologies which can be extended to deal with inconsistency. Our approach balances the trade-off between performance and completeness by introducing segmentation-based distributed ontology integration. We define the notions of segment and approximate conjunctive query answering. Corresponding algorithms are designed and implemented.

We implemented both approaches and provide some interesting evaluation results.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The NeOn Big Picture

Next generation semantic applications will be characterized by a large number of ontologies, some of them constantly evolving. As the complexity of semantic applications increases, more and more knowledge will be embedded in applications, typically drawn from a wide variety of sources. This new generation of applications will thus likely rely on ontologies embedded in a network of already existing ontologies. Ontologies and metadata will have to be kept up to date when application environments and users' needs change. We argue that in this scenario it will become prohibitively expensive for people to directly adopt the current approach to semantic integration, where the expectation is to produce a single, globally consistent semantic model that serves the needs of application developers and fully integrates a number of pre-existing ontologies. In contrast to the current model, future applications will very likely rely on networks of contextualized ontologies, which are usually locally, but not globally consistent.

This report is part of the work performed in WP 1 on "Dynamics of Networked Ontologies". The goal of this work package is to develop an integrated approach for the evolution process of networked ontologies and related metadata. As shown in Figure 1.1, WP1 belongs to the central part of the research and development WPs in NeOn. The tasks of WP1 are heavily inter-related with other work packages. For the individual phases of the process we will develop new methods that consider the complex relationships in a network of ontologies. These include dependencies, mappings, different versions and also take possible inconsistencies into account.

Specific goals in this work package include support for:

1. representing, managing and interpreting dependencies between multiple networked ontologies
2. evolution of networked ontologies in exploiting various models of change propagation, which have different applicabilities depending on the model of coordination and control
3. maintaining partial/local consistency of a set of networked ontologies, which might not be globally consistent
4. evolving metadata along with changing ontologies and predicting future structural changes in ontologies.

1.2 Motivation and Goals of this Deliverable

Real knowledge bases and data for Semantic Web applications will rarely be perfect. They will be distributed and multi-authored. They will be assembled from different sources and reused. It is unreasonable to expect such realistic knowledge bases to be always logically consistent, and it is therefore important to study ways of

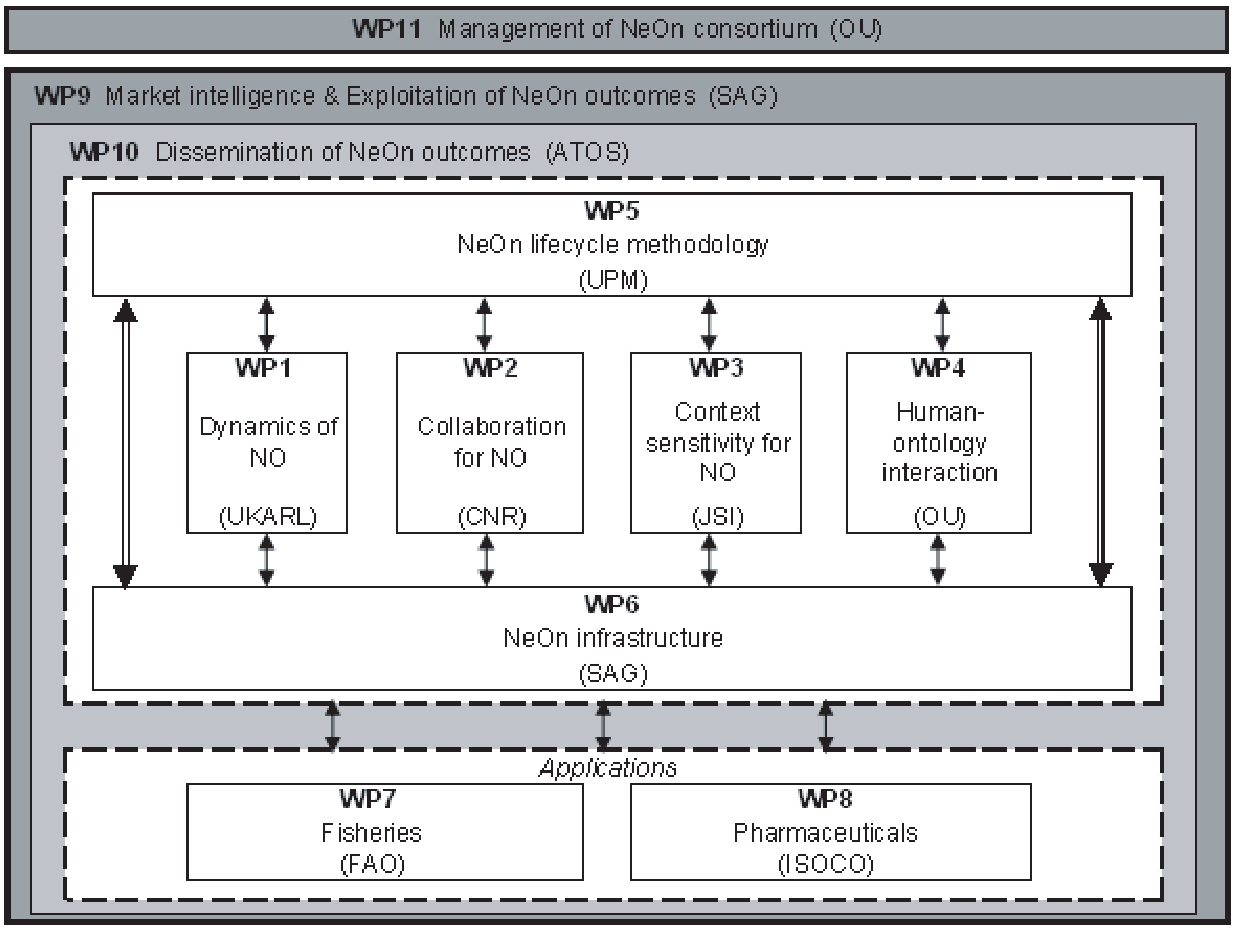

WP11 Management of NeOn consortium (OU)
WP9 Market intelligence & Exploitation of NeOn outcomes (SAG)
WP10 Dissemination of NeOn outcomes (ATOS)
WP5 NeOn lifecycle methodology (UPM)
WP1 Dynamics of NO (UKARL)
WP2 Collaboration for NO (CNR)
WP3 Context sensitivity for NO (JSI)
WP4 Human-ontology interaction (OU)
WP6 NeOn infrastructure (SAG)
Applications
WP7 Fisheries (FAO)
WP8 Pharmaceuticals (ISOCO)

Figure 1.1: Relationships between different work packages in NeOn

dealing with inconsistent knowledge. This is particularly important if the full power of logic-based approaches like the Web Ontology Language OWL shall be employed, as classical logic breaks down in the presence of inconsistent knowledge. The study of inconsistency handling in Artificial Intelligence has a long tradition, and corresponding results are recently being transferred to description logics, which underly OWL. On the other hand, today, many semantic information systems on the Web are going beyond a centralized setting and working in a distributed scenario. In those systems, ontologies are increasingly applied as the data schemata, sources, mediators, etc. [CGL01, CWW$^+$06, GR03], thus, querying the distributed ontologies is one major task in distributed semantic information systems.

There are two extreme situations for query answering over distributed ontologies: (1) We can integrate the distributed ontologies into a single local node and perform query answering in a centralized way (e.g. [CWW$^+$06, GR03]). This approach apparently lacks the optimization for query answering in the distributed scenario, because queries are not executed in a distributed way. (2) On the other hand, we can also query over distributed ontologies without integration. In this case, the ontologies are queried in a pure distributed way on an individual ontology in parallel, so overall execution time is reduced. However, by using this approach, we may lose significant information that is inferred by considering the interrelationships (e.g. in the form of mappings) between the ontologies. The first extreme keeps completeness by losing performance, while the second one pursues performance but loses completeness. Our aim is to find a reasonable balance between the two extremes: We argue that performance is a critical issue on today's Web, while there is often a tradeoff between the completeness and the performance when querying distributed ontologies.

Because it's difficult to achieve complete answers and performance advances at the same time, approximate query answering based on a *proper* integration of distributed ontologies but provides quick response times is an important issue to be addressed. This issue has been widely recognized and discussed in the traditional database community in order to improve query answering performance [AGPR99]. However, there is little existing work on approximate query answering over distributed ontologies in this field. The popular description logics (DL) reasoners, such as FaCT++[1], Pellet[2] and KAON2[3], only support exact query answering over ontologies and are not particularly optimized for a distributed scenario.

In this deliverable, we introduce a novel approach for approximate conjunctive query answering over distributed and inconsistent ontologies in semantic information systems on the Web. We focus on improving the overall performance by sacrificing part of the completeness. We integrate ontologies in a segmentation-based approach, where distributed ontologies are divided into several groups that are connected via mappings. We also find an appropriate way to identify the segments of the integrated global ontology. As central contribution of our approach, we have designed three algorithms for segmentation, query distribution, and termination and results collection, respectively. We implemented this approach and the bilattice-based semantics and provide some interesting evaluation results.

The work presented in this deliverable is related to some activities in WP1 and WP3. This work is closely related to the work presented in NeOn deliverable D1.4 where KAONp2p is proposed. The work presented at Task 3.3 of NeOn project has discussed how to generate mappings connecting distributed and heterogenous ontologies. This work is also related to the work on contextualization presented in Task 3.1 of NeOn project. For example, the work on bilattice-based semantics is closely related to the work on reasoning with meta-knowledge given in D3.1.4.

1.3 Overview of the Deliverable

This deliverable is structured as follows. We first present the bilattice-based semantics in Chapter 2. We then give our segmentation-based approach in Chapter 3. The evaluation results are given in Chapter 4. Finally, we conclude this deliverable and provide the roadmap in Chapter 5.

[1]`http://owl.man.ac.uk/factplusplus/`
[2]`http://www.mindswap.org/2003/pellet/`
[3]`http://kaon2.semanticweb.org`

Chapter 2

Practical Reasoning with Trust, Uncertainty and other Meta-Knowledge

In D1.2.4 we have generalized $\mathcal{ALC}\triangle$ to reasoning with arbitrary bilattices in OWL and applied this generalization to reasoning with trust. While the approach presented there is very flexible and theoretically interesting, it is difficult to implement directly. In D1.5.4 we discussed reasoning with meta-knowledge e.g. trust, uncertainty or edit histories. The approach presented there has a straight forward naive implementation, but lacks support for paraconsistent reasoning provided by bilattices based reasoning. In this chapter, we briefly recapitulate the foundations of both approaches. Then we discuss, how bilattice based reasoning with trust can be reduced to meta-knowledge reasoning. Finally, we provide an optimized algorithm for reasoning with meta-knowledge, which is inspired from pinpointing algorithms. In chapter 4 we evaluate the optimized algorithm and show that it performs up to several orders of magnitude faster than the baseline.

2.1 Reasoning Based on Logical Bilattices

A logical bilattice [Gin92] is a set of truth values, on which two partial orders are defined, which we call the truth order $\leq_t$ and the knowledge order $\leq_k$. Both $\leq_t$ and $\leq_k$ are complete lattices, i.e. they have a maximal and a minimal element and every two elements have exactly one supremum and infimum.

In logical bilattices, the operators $\vee$ and $\wedge$ are defined as supremum and infimum wrt. $\leq_t$. Analogously join ($\oplus$) and meet ($\otimes$) are defined as supremum and infimum wrt. $\leq_k$. As a result, we have multiple distributive and commutative laws, which all hold. Negation ($\neg$) simply is an inversion of the truth order. Hence, we can also define material implication ($a \rightarrow b = \neg a \vee b$) as usual.

The smallest non trivial logical bilattice is $\mathcal{FOUR}$, shown in figure 2.1. In addition to the truth values t and f, $\mathcal{FOUR}$ includes $\top$ and $\bot$. $\bot$ means "unknown", i.e. a fact is neither true or false. $\top$ means "overspecified" or "inconsistent", i.e. a fact is both true and false.

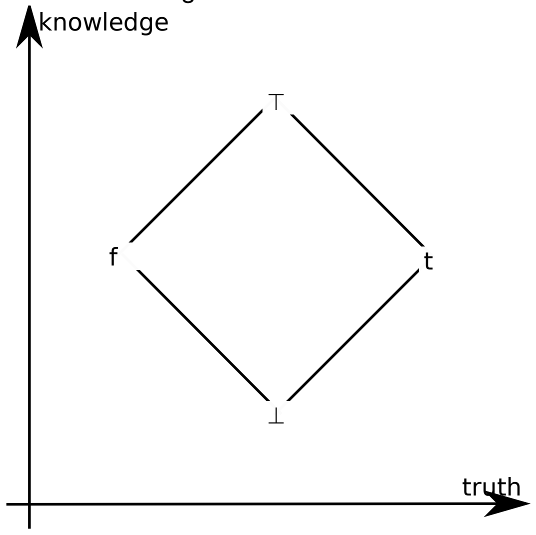
2.1.1 Obtaining bilattices

Ginsberg [Gin92] describes how we can obtain a logical bilattice: Given two distributive lattices $\mathcal{L}_1$ and $\mathcal{L}_2$, create a bilattice $\mathcal{L}$, where the nodes have values from $\mathcal{L}_1 \times \mathcal{L}_2$, such that the following orders hold:

- $\langle a, b\rangle \leq_k \langle x, y\rangle$ iff $a \leq_{\mathcal{L}_1} x \wedge b \leq_{\mathcal{L}_2} y$ and
- $\langle a, b\rangle \leq_t \langle x, y\rangle$ iff $a \leq_{\mathcal{L}_1} x \wedge y \leq_{\mathcal{L}_2} b$

If $\mathcal{L}_1$ and $\mathcal{L}_2$ are infinitely distributive— that means distributive and commutative laws hold for infinite combinations of the operators $\vee_1, \wedge_1$ and $\vee_2, \wedge_2$ respectively — then $\mathcal{L}$ will be as well, i.e. arbitrary combinations

Figure 2.1: $\mathcal{FOUR}$



of $\vee, \wedge, \oplus$, and $\otimes$ are possible in the resulting bilattice. In other words this means that the language of logical connectives over bilattices is complete.

Next, we discuss how a bilattice representing trust ratings can be obtained.

2.1.2 Application to Reasoning with Trust

In addition to inconsistencies, which will almost certainly occur in the Semantic Web, be will also be faced with information sources, which are of varying trustworthiness. Hence, we define a logical bilattice, which allows for reasoning with trust orders. In the previous chapter we have focused on two levels of reliability of information. More generally, we would like to be able to infer multiple levels of trust in a distributed setting.

Definition 1 (Trust Order) *A trust order $\mathcal{T}$ is a partial order over a finite set of information sources with a maximal element, called ∞.*

∞ is the information source with the highest trust level, assigned to local data. For any two information sources a and b comparable wrt. $\mathcal{T}$, we have a $\mathcal{FOUR} - \mathcal{C}$ lattice as described above, with the less trusted information source corresponding to the inner part of the lattice. Extended to multiple information sources, this results in a situation as depicted in Figure 2.2, where the outermost bilattice corresponds to local, fully trusted information.

If a and b are not comparable we introduce virtual information sources inf_{ab} and sup_{ab}, such that

- $inf_{ab} < a < sup_{ab}$ and $inf_{ab} < b < sup_{ab}$;
- $\forall_{c<a,c<b} : c < inf_{ab}$ and
- $\forall_{d>a,d>b} : d > sup_{ab}$

To understand the importance of this last step, assume that $c > a > d$ and $c > b > d$ and a, b are incomparable. Then the truth value of $a \vee b$ would have a trust level of c, as c is the supremum in the trust order. Obviously this escaping to a higher trust level is not desirable. Instead, the virtual information sources represent that we need to trust both, a and b, if we believe in the computed truth value. We illustrate this situation in Figure 2.3 (We abbreviate inf_{ab} by $<$ and sup_{ab} by $>$).

In the general case (≥ 3 incomparable sources) such a trust order results in a non-distributive lattice. This can be fixed, however, by introducing additional virtual nodes. The basic idea here is to create a virtual node

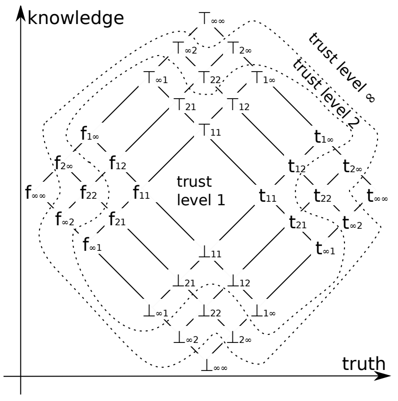


Figure 2.2: $\mathcal{FOUR}-\mathcal{T}$

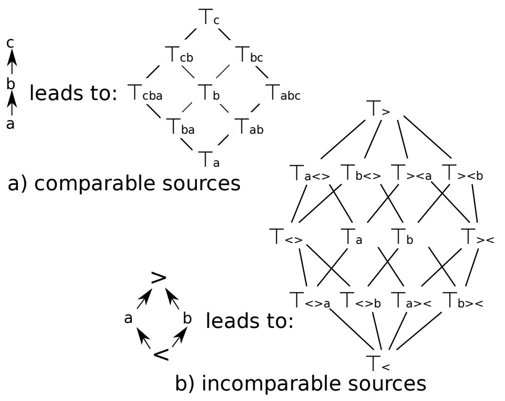


Figure 2.3: Virtual Sources

for each element in the powerset of the incomparable sources, with set inclusion as the order. We will call this modified trust order *completed*. We can again derive a complete lattice from the completed trust order. As it can become quite large, we only show for $\top$ how a fragment of the logical bilattice is derived from the trust order for the case of two incomparable information sources in Figure 2.3.

Now we can construct the corresponding bilattice from a given completed truth order as follows: Given a trust order $\mathcal{T}$, generate a lattice $\mathcal{L}_1$, such that

- $f_a <_{\mathcal{L}_1} f_b$, iff $a >_{\mathcal{T}} b$;
- $t_b <_{\mathcal{L}_1} t_a$, iff $a >_{\mathcal{T}} b$ and
- $\forall a : f_a <_{\mathcal{L}_1} t_a$.

The result is a lattice with t_∞ and f_∞ as maximal and minimal elements. Now create the logical bilattice $\mathcal{L}$ from $\mathcal{L}_1 \times \mathcal{L}_1$ as described in the previous section.

In Figure 2.2 we see a bilattice resulting from a strict trust ordering, i.e. every pair of information sources in the trust order is comparable. In general, it will not be possible to come up with such a strict ordering. For example, interpretations of news may vary also among very trustworthy newspapers, depending on their political background. Hence, we might have a trust ordering with incomparable information sources. This case is demonstrated in Figure 2.3. Case a) shows the $\top$ part of the logical bilattice from Figure 2.2. Case b) shows the same part for a bilattice resulting from a trust ordering, in which sources a and b are incomparable. Their supremum and infimum are denoted by $<$ and $>$ respectively. As we can see, trust-bilattices can grow quite complex. As users of reasoning services will be interested in trust levels only, they need not be faced with all these values, however. Further, the bilattice needs not be materialized for reasoning, as many truth values will not even be used. However, to be complete, all these values must be defined.

2.1.3 Extending OWL2 to logical bilattices

In this section we recapitulate the extension of $\mathcal{SROIQ}$, the description logic underlying the proposed OWL2 [GM08], to $\mathcal{SROIQ}-\mathcal{T}$ evaluated on a logical bilattice. The extension towards logical bilattices works analogously to the extension of $\mathcal{SHOIN}$ towards a fuzzy logic as proposed in [Str06]. Operators marked with a dot, e.g. $\dot{\geq}$ are the lattice operators described above, all other operators are the usual (two valued) boolean operators. For two valued operators and a logical bilattice $\mathcal{L}$ we map t to $max_t(\mathcal{L})$ – i.e. the maximal value with respect to the truth order of $\mathcal{L}$ (usually denoted by $\top$) – and f to $min_t(\mathcal{L})$, in order to model that these truth values are absolutely trusted[1]. Please note that while we limit ourselves to $\mathcal{SROIQ}$ here, analogous extensions are possible for $\mathcal{SROIQ}(\mathcal{D})$ to support datatypes. Please also note that we

[1] In $\mathcal{FOUR}-\mathcal{T}$ these would be f_∞ and t_∞, but we start with the general case.

do not include language constructs, which can be expressed by a combination of other constructs defined below. In particular, $Sym(R) = R^- \sqsubseteq R$ and $Tra(R) = R \circ R \sqsubseteq R$.

Definition 2 (Vocabulary) *A vocabulary $V = (N_C, N_P, N_I)$ is a triple where*

- *N_C is a set of OWL classes,*
- *N_P is a set of properties and*
- *N_I is a set of individuals.*

N_C, N_P, N_I need not be disjoint.

A first generalization is that interpretations assign truth values from any given bilattice. In contrast, $\mathcal{SROIQ}$ is defined via set membership of (tuples of) individuals in classes (properties) and uses two truth values only.

Definition 3 (Interpretation) *Given a vocabulary V an* interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \mathcal{L}, \cdot^{\mathcal{I}_C}, \cdot^{\mathcal{I}_P}, \cdot^{\mathcal{I}_i})$ *is a 5-tuple where*

- *$\Delta^{\mathcal{I}}$ is a nonempty set called the object domain;*
- *$\mathcal{L}$ is a logical bilattice and Λ is the set of truth values in $\mathcal{L}$*
- *$\cdot^{\mathcal{I}_C}$ is the class interpretation function, which assigns to each OWL class $A \in N_C$ a function: $A^{\mathcal{I}_C} : \Delta^{\mathcal{I}} \to \Lambda$;*
- *$\cdot^{\mathcal{I}_P}$ is the property interpretation function, which assigns to each property $R \in N_P$ a function $R^{\mathcal{I}_P} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to \Lambda$;*
- *$\cdot^{\mathcal{I}_i}$ is the individual interpretation function, which assigns to each individual $a \in N_I$ an element $a^{\mathcal{I}_i}$ from $\Delta^{\mathcal{I}}$.*

$\mathcal{I}$ is called a complete interpretation*, if the domain of every class is $\Delta^{\mathcal{I}}$ and the domain of every property is $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$.*

The notion of a complete interpretation is needed, because interpretation functions assign a truth value instead of just defining a set membership. In two valued description logics, set membership of an individual in a class corresponds to a truth value of *true* and the default is *false*. In four valued description logics, analogously two sets are used for each class. Hence, by listing only the membership of some individuals, the truth value of class membership is still well defined for all individuals. Here, such a simple convention can not be applied, as we have multiple possible truth values. Instead, truth values must explicitly be assigned for each individual.
We extend the property interpretation function $\cdot^{\mathcal{I}_P}$ to property expressions:

$$(R^-)^{\mathcal{I}_P} \quad = \quad \{(\langle x, y\rangle, u) | (\langle y, x\rangle, u) \in R^{\mathcal{I}_P}\}$$

The second generalization over $\mathcal{SROIQ}$ is the replacement of all quantifiers over set memberships with conjunctions and disjunctions over Λ. We extend the class interpretation function $\cdot^{\mathcal{I}_C}$ to descriptions as shown in table. 2.1.

Satisfaction of axioms in an interpretation $\mathcal{I}$ is defined in table 2.2. With $\circ$ we denote the composition of binary relations. For any function f, $dom(f)$ returns the domain of f. The generalization is analogous to that of $\cdot^{\mathcal{I}_C}$. Note that for equality of individuals, we only need two valued equality.

Satisfiability in $\mathcal{SROIQ} - \mathcal{T}$ is a bit unusual, because when using a logical bilattice we can always come up with interpretations satisfying all axioms by assigning $\top$ and $\bot$. Therefore, we define satisfiability wrt. a truth value:

$$
\begin{aligned}
\top^{\mathcal{I}}(x) &= t_{yy}, \text{where y is the information source, defining} \top^{\mathcal{I}}(x) \\
\bot^{\mathcal{I}}(x) &= f_{yy}, \text{where y is the information source, defining} \bot^{\mathcal{I}}(x) \\
(C_1 \sqcap C_2)^{\mathcal{I}}(x) &= C_1^{\mathcal{I}}(x) \dot{\wedge} C_2^{\mathcal{I}}(x) \\
(C_1 \sqcup C_2)^{\mathcal{I}}(x) &= C_1^{\mathcal{I}}(x) \dot{\vee} C_2^{\mathcal{I}}(x) \\
(\neg C)^{\mathcal{I}}(x) &= \dot{\neg} C^{\mathcal{I}}(x) \\
(S^-)^{\mathcal{I}}(x,y) &= S^{\mathcal{I}}(y,x) \\
(\forall R.C)^{\mathcal{I}}(x) &= \dot{\bigwedge}_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\rightarrow} C^{\mathcal{I}}(y) \\
(\exists R.C)^{\mathcal{I}}(x) &= \dot{\bigvee}_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\wedge} C^{\mathcal{I}}(y) \\
(\exists R.\mathsf{Self})^{\mathcal{I}}(x) &= R^{\mathcal{I}}(x,x) \\
(\geq nS)^{\mathcal{I}}(x) &= \dot{\bigvee}_{\{y_1,...,y_m\} \subseteq \Delta^{\mathcal{I}}, m \geq n} \dot{\bigwedge}_{i=1}^{n} S^{\mathcal{I}}(x,y_i) \\
(\leq nS)^{\mathcal{I}}(x) &= \dot{\neg} \dot{\bigvee}_{\{y_1,...,y_{n+1}\} \subseteq \Delta^{\mathcal{I}}} \dot{\bigwedge}_{i=1}^{n+1} S^{\mathcal{I}}(x,y_i) \\
\{a_1,...,a_n\}^{\mathcal{I}}(x) &= \dot{\bigvee}_{i=1}^{n} a_i^{\mathcal{I}} = x
\end{aligned}
$$

Table 2.1: Extended Class Interpretation Function

$$
\begin{aligned}
(R \sqsubseteq S)^{\mathcal{I}} &= \dot{\bigwedge}_{x,y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\rightarrow} S^{\mathcal{I}}(x,y) \\
(R = S)^{\mathcal{I}} &= \dot{\bigwedge}_{x,y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\leftrightarrow} S^{\mathcal{I}}(x,y) \\
(R_1 \circ ... \circ R_n \sqsubseteq S)^{\mathcal{I}} &= \dot{\bigwedge}_{\langle x_1, x_{n+1} \rangle \in dom(S^{\mathcal{I}})} \dot{\bigvee}_{\{x_2,...,x_n\}} \dot{\bigwedge}_{i=1}^{n} R_i^{\mathcal{I}}(x_i, x_{i+1}) \\
(Asy(R))^{\mathcal{I}} &= \dot{\bigwedge}_{x,y \in \Delta^{\mathcal{I}}} \dot{\neg}(R^{\mathcal{I}}(x,y) \dot{\wedge} R^{\mathcal{I}}(y,x)) \\
(Ref(R))^{\mathcal{I}} &= \dot{\bigwedge}_{x \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,x) \\
(Irr(R))^{\mathcal{I}} &= \dot{\bigwedge}_{x \in \Delta^{\mathcal{I}}} \dot{\neg} R^{\mathcal{I}}(x,x) \\
(Dis(R,S))^{\mathcal{I}} &= \dot{\bigwedge}_{x,y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \dot{\rightarrow} \dot{\neg} S^{\mathcal{I}}(x,y) \\
(C \sqsubseteq D)^{\mathcal{I}} &= \dot{\bigwedge}_{x \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(x) \dot{\rightarrow} D^{\mathcal{I}}(x) \\
(a : C)^{\mathcal{I}} &= C^{\mathcal{I}}(a^{\mathcal{I}}) \\
((a,b) : R)^{\mathcal{I}} &= R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \\
a \approx b &= a^{\mathcal{I}} = b^{\mathcal{I}} \\
a \not\approx b &= a^{\mathcal{I}} \neq b^{\mathcal{I}}.
\end{aligned}
$$

Table 2.2: Satisfaction of Axioms

Definition 4 (Satisfiability) *We say an axiom E is* u-satisfiable *in an ontology O wrt. a bilattice $\mathcal{L}$, if there exists a complete interpretation $\mathcal{I}$ of O wrt. $\mathcal{L}$, which assigns a truth value $val(E,\mathcal{I})$ to E, such that $val(E,\mathcal{I}) \geq_k u$.*

We say an ontology O is u-satisfiable, if there exist a complete interpretation $\mathcal{I}$, which u-satisfies all axioms in O and for each class C we have $|\{a|\langle a,v\rangle \in C \wedge v \geq_t u\}| > 0$, that means no class is empty.

Analogously we define consistency wrt. the knowledge order. O is an ontology with axioms collected from multiple sources S_i. Please note that we use a single ontology here to follow the definition of $\mathcal{SROIQ}$, but the composition could happen through imports, or mappings in a network of ontologies. The S_i differ with respect to their trustworthiness. Hence, a partial trust order $\mathcal{T}$ models their trustworthiness. Also note that an analogous is possible for partial interpretations for parts of an ontology. Again we focus on complete interpretations analogous to models in $\mathcal{SROIQ}$.

Definition 5 (Consistency) *Let $\mathcal{I}$ be a complete interpretation, O an ontology, which is composed from multiple data sources $\{S_1, ..., S_n\}$ and $\mathcal{T}$ a trust order over $\{S_1, ..., S_n\}$. Let $source(E)$ denote the $\mathcal{T}$-maximal datasource, from which axiom E originates.*

We say O is u-consistent, if there exists an $\mathcal{I}$, which assigns a truth value $val(E,\mathcal{I})$ to all axioms E in O, such that $u \leq_k val(E,\mathcal{I})$. $\mathcal{I}$ is called a u-model *of O.*

We say O is consistent, if there exist an $\mathcal{I}$, which assigns a truth value $val(E,\mathcal{I})$ to all axioms E in O, such that $\forall x : val(E,\mathcal{I}) \notin \{\top_x, \bot_x\}$. We say $\mathcal{I}$ is a model of O.

Finally, we define entailment:

Definition 6 (Entailment) *O entails a $\mathcal{SROIQ} - \mathcal{T}$ ontology O' $(O \models O')$, if every model of O is also a model of O'. O and O' are equivalent if O entails O' and O' entails O.*

2.2 Reasoning with Meta-Knowledge

We summarize the foundations of meta-knowledge reasoning before reducing trust based reasoning with bilattices to meta-knowledge reasoning.

2.2.1 Running Example

When answering queries in the semantic web, one must not only handle the knowledge itself, but also characterizations of this knowledge, e.g.: (i) where did a knowledge item come from (i.e. provenance), (ii) what level of trust can be assigned to a knowledge item, or (iii) what degree of certainty is associated with it. We refer to all such kinds of characterizations as *meta-knowledge*. On the semantic web, meta-knowledge needs to be computed along with each reasoning task.

As a motivation consider a serious faux pas, which happened to the German Press Agency DPA on 10 September 2009: A person called DPA and notified them, that a terrorist attack had just taken place in Bluewater, CA. DPA did a short internet recherche and found a website for Bluewater's local TV station, vpk-tv, and the town itself, and also Wikipedia entries for both the town and vpk-tv. DPA announced the attack as breaking news. However, Bluewater, CA does not exist and all this was guerilla marketing for the new movie "Shortcut to Hollywood" set up by Neverest, a professional marketing agency.

DPA did not take into account meta-knowledge when doing their background recherche. If they had done so, they would have noticed, that all related websites as well as the Wikipedia article had been set up by the same marketing agency at roughly the same time. On the non-semantic Web such a faux pas can only be avoided by a most careful professional analysis of data provenance. We show here, how the analysis may be automated on the Semantic Web - even when using sophisticated reasoning machinery such as OWL-2.

ID	Axiom	source	modified	conf.d.
#1	$RealCity \sqsubseteq City \sqcap \exists hasCompany.Broadcaster$	DPA	2009-09-10	0.2
#2	$Broadcaster \sqsubseteq \exists hqIn.City$	DPA	2009-09-09	
#3	$inverseProperty(hasCompany, hqIn)$	DPA	2009-09-10	0.6
#4	$bluewater : City$	Neverest	2009-09-09	
#5	$vpktv : Broadcaster$	NeverestWP	2009-09-10	0.4
#6	$hqIn(vpktv, bluewater)$	Neverest	2009-09-09	

Table 2.3: Example Ontology and Meta-Knowledge

We will use the DPA scenario to illustrate our method in a concise fashion. In the DPA example, the data from Wikipedia would have come from an open, Wiki-like system, while Bluewater's and the TV broadcaster's webpages would correspond to ontologies from a single source. In both cases, change histories or modification dates and authors respectively are available. For the running example we will use the ontology fragment listed in Table 2.3, which allows to answer the question, whether *bluewater* is a *RealCity*. A *RealCity* is a *City* with at least one *Broadcaster*, who has its headquarters in the *City*. Associated with each axiom is meta-knowledge describing, who has created the axiom (DPA, Neverest or NeverestWP, their Wikipedia user), when it has been modified, and a confidence degree for those axioms coming from Wikipedia. Such a confidence degree can be computed for Wikipedia using for example the approach described in [AdA07].

2.2.2 Pinpointing

As we use pinpointing as a vehicle for computing meta-knowledge, we introduce pinpointing as a foundation for the rest of the paper and give some information of existing algorithms for finding pinpoints.

The term pinpointing has been coined for the process of finding explanations for concluded axioms or for a discovered inconsistency. An explanation is a minimal set of axioms, which makes the concluded axiom true (or the theory inconsistent, respectively). Such an explanation is called a pinpoint. While there may be multiple ways to establish the truth or falsity of an axiom, a pinpoint describes exactly one such way.

Definition 7 *Pinpoint.*
A pinpoint for an entailed axiom A wrt. an ontology O is a set of axioms $\{A_1, ..., A_n\}$ from O, such that $\{A_1, ..., A_n\} \models A$ and
$\forall A_i \in \{A_1, ..., A_n\} : \{A_1, ...A_{i-1}, A_{i+1}, ..., A_n\} \not\models A$. Analogously, a pinpoint for a refuted axiom A wrt. an ontology O is a set of axioms $\{A_1, ..., A_n\}$ from O, such that $\{A, A_1, ..., A_n\}$ is inconsistent and
$\forall A_i \in \{A_1, ..., A_n\} : \{A, A_1, ...A_{i-1}, A_{i+1}, ..., A_n\}$ is not.

Hence, finding pinpoints for a refuted axiom corresponds to finding the Minimum Unsatisfiable Subontologies (MUPS) for this axiom [KPCGS06]. Pinpointing is the computation of all pinpoints for a given axiom and ontology. The truth of the axiom can then be computed using the *pinpointing formula* [BP07].

Definition 8 *Pinpointing Formula.*
Let A be an axiom, O an ontology and $P_1, ..., P_n$ with $P_i = \{A_{i,1}, ..., A_{i,m_i}\}$ the pinpoints of A wrt. O. Let lab be a function assigning a unique label to an axiom. Then $\bigvee_{i=1}^{n} \bigwedge_{j=i}^{m_i} lab(A_{i,j})$ is a pinpointing formula of A wrt. O.

A pinpointing formula of an axiom A describes, which (combination of) axioms need to be true in order to make A true or inconsistent respectively.

2.2.3 Semantics of Meta Knowledge

Meta knowledge can have multiple dimensions, e.g. uncertainty, a least recently modified date or a trust metric. For this paper, we assume that these (and possible further) dimensions are independent of each other. At the end of this section we will sketch, how this assumption can be generalized such that multiple dimensions are combined into one new integrating dimension.

Definition 9 *Knowledge dimension. A knowledge dimension D is an algebraic structure $(B_D, \vee_D, \wedge_D)$, such that $(B_D, \vee_D)$ and $(B_D, \wedge_D)$ are complete semilattices.*

B_D represents the values the meta-knowledge can take, e.g. all valid dates for the least recently modified date or a set of knowledge sources for provenance. As $(B_D, \vee_D)$ and $(B_D, \wedge_D)$ are *complete* semilattices, they are, in fact, also lattices. Hence, there are minimal elements in the corresponding orders.

As an example, let I be the meta-knowledge interpretation that is a partial function mapping axioms into the allowed value range of a meta-knowledge dimension, and A and B be axioms of an ontology such that $A \neq B$. Provenance, i.e. the set of knowledge sources a piece of knowledge is derived from, can be modeled as:

- $I(A \vee B) = I(A) \cup I(B)$
- $I(A \wedge B) = I(A) \cup I(B)$

The least recently modified date could be modeled as:

- $I(A \vee B) = min(I(A), I(B))$
- $I(A \wedge B) = max(I(A), I(B))$

Axioms can be assigned meta-knowledge from any of the meta-knowledge dimensions. Within a single assignment, the meta-knowledge must be uniquely defined.

Definition 10 *Meta Knowledge Assignment.*
A meta-knowledge assignment M is a set $\{(D_1, d_1 \in D_1), ..., (D_n, d_n \in D_n)\}$ of pairs of meta-knowledge dimensions and corresponding truth values, such that $D_i = D_j \Rightarrow d_i = d_j$.

In our running example, the meta-knowledge assignment for
bluewater: City is *{(source, Neverest), (date, 2009-09-09)}*

Without loss of generality we assume a fixed number of meta-knowledge dimensions. As a default value for D_n in a meta-knowledge assignment we choose the minimal element $\perp_D$.

Syntactically meta-knowledge assignments are expressed using axiom annotations. We have provided a detailed grammar for meta-knowledge annotations in [SDS09].

To allow for reasoning with meta-knowledge, we need to formalize, how meta-knowledge assignments are combined. *How provenance* [GKT07] is a strategy, which describes how an axiom A can be inferred from a set of axioms $\{A_1, ..., A_n\}$, i.e. it is a boolean formula connecting the A_i. We call a logical formula expressing how provenance a *meta-knowledge formula*. For example the following query for each city finds all companies:

x:City ∧ hasCompany(x,y).

The result of this query and the corresponding meta-knowledge formulas are:

x	y	meta knowlege formula
bluewater	vpktv	#3 ∧ #4 ∧ #6

The operators for meta-knowledge dimensions extend to meta-knowledge assignments, allowing us to compute meta-knowledge for entailed knowledge by evaluating the corresponding meta-knowledge formula.

In contrast to [SST08] we omit the $\neg$ operator in our formalization, as description logics are monotonic and $\neg$ in [SST08] allows for default negation. While axioms in the underlying description logic may contain negation, this negation is not visible and not needed on the level of the meta-knowledge.

Definition 11 *Operations on Meta Knowledge Assignments.*
Let A, B be axioms and meta(A) = $\{(D_1, x_1), ..., (D_n, x_n)\}$ and meta(B) = $\{(E_1, y_1), ..., (E_m, y_m)\}$ be meta-knowledge assignments. Let dim(A) be the set of meta-knowledge dimensions of A. Then meta$(A) \vee$ meta$(B) = \{(D, x \vee_D y) | (D, x) \in$ meta(A) and $(D, y) \in$ meta$(B)\}$. $\wedge$ is defined analogously.

Having defined the operations on meta-knowledge assignments, we can define how the meta-knowledge of an axiom A within a meta-knowledge dimension is obtained. The meta-knowledge of an axiom A within a meta-knowledge dimension is obtained by evaluating the corresponding meta-knowledge formula in the dimension under consideration.

Definition 12 *Meta Knowledge of an Axiom.*
Let meta *be a function mapping from an axiom to a meta-knowledge assignment in dimension D. The meta-knowledge of an axiom A wrt. O in D is obtained by computing a pinpointing formula ϕ of A wrt. O and obtaining ψ by replacing each axiom in ϕ with its meta-knowledge assignment in D. Then* meta*(A) is computed by evaluating ψ.*

In our running example, if we model the information source dimension as provenance, the meta-knowledge of the query result for *bluewater, vpktv* is: *(source, {DPA})* $\wedge$ *(source, {Neverest}) = (source, {DPA}* $\cup$ *{Neverest}) = (source, {DPA, Neverest})*.

2.3 Bilattice Based Reasoning using Meta Knowledge

We now describe how reasoning with certain classes of bilattices can be reduced to reasoning with meta-knowledge, which can be implemented efficiently as we will see in the next chapter. More precisely, we will show how reasoning with the kind of regular bilattices, which result from multiplying lattices according to section 2.1.2, can be done. The approach uses to the operationalization of paraconsistent reasoning with expressive DLs.

In D1.2.4 [QS09], we have provided a reduction of paraconsistent reasoning in $\mathcal{ALC}\triangle$ to classical reasoning in $\mathcal{ALC}$. This approach has been extended to $\mathcal{SHIQ}$ in [MHL08] and to $\mathcal{SROIQ}$ in [MH09]. In this deliverable, we use a two step process for reasoning with regular bilattices: First, we assume, we are dealing with the bilattice $\mathcal{FOUR}$ and reduce the ontology to a classical one using the approach described in [MH09]. This allows for paraconsistent reasoning.

Second, instead of assigning trust values from the bilattice we assign a value from the *input lattice* used to compute the bilattice in a new meta-knowledge dimension. For example, instead of assigning a truth value from $\mathcal{FOUR} - \mathcal{T}$, we add a meta-knowledge assignment to a value from the underlying trust order. Of cause, this order lacks some of the truth values available in $\mathcal{FOUR} - \mathcal{T}$. However, the truth values, which combine multiple trust levels, are only relevant for inferences. Axioms always originate from one sources and hence can be assigned a corresponding meta-knowledge degree. In case multiple sources claim the same axioms, the corresponding meta-knowledge assignments can be combined into a single one as described in [THD$^+$09] and [SS09].

As a second step, we perform normal meta-knowledge reasoning with the reduced ontology. The subscripts in $\mathcal{FOUR} - \mathcal{T}$ truth values represent the most and least trusted sources used for computing the answer. Meta knowledge reasoning as described in section 2.2 only computes the maximum. We can easily obtain the minimum by inverting the order of the meta-knowledge dimension and repeating the meta-knowledge reasoning step.

The following algorithm summarizes these steps:

We observe an interesting property: The translation from four valued to classical reasoning is linear in the size of the input ontology [MH09]. Furthermore the rewriting of truth values from $\mathcal{FOUR} - \mathcal{T}$ to meta-knowledge assignments using the input trust order is a purely syntactical transformation and also linear.

Algorithm 1: ParaconsistentTrust

```
Data: Trust Bilattice $B$, Ontology O using $B$, trust order $T$ underlying $B$, Query Axiom $Q$
$O' \leftarrow \pi(O)$ ;
foreach ((Axiom $A$, Truth Value $t$) $\in O$, $A' = \pi(A)$) do
    choose $t' \in T$ such that $t = t't'$ ;
    assignMetaK($A'$, $t'$) ;
max $\leftarrow$ MetaPartial($O', Q, (T, \vee_T, \wedge_T)$) ;
min $\leftarrow$ MetaPartial($O', Q, (T, \wedge_T, \vee_T)$) ;
return $\oplus_B$(min $\times$ min, max $\times$ max) ;
```

Finally, we have to repeat the meta-knowledge reasoning step in order to compute both, minimum and maximum. Meta knowledge reasoning has a worst case complexity, which is exponential in the complexity of the underlying DL. Hence, the complexity of bilattice based reasoning using meta-knowledge also is of worst case exponential complexity. In the next section we will show that in the average case, meta-knowledge reasoning can be optimized to perform much better. The evaluation in section 4 will show, that in fact, reasoning can be done fast enough for interactive applications even for large and expressive ontologies.

2.4 An Optimized Algorithm for Reasoning with Meta Knowledge

We have defined meta-knowledge based on pinpointing formulas. For this reason, when reasoning in a logic, where a pinpointing algorithm is known, we can compute a pinpointing formula and then immediately derive meta-knowledge. However, finding all pinpoints is a very expensive operation. Hence, we provide an optimized algorithm for computing meta-knowledge, which does not need to compute all pinpoints, and in fact not even precisely a pinpoint but a sufficiently precise module around a pinpoint.

The optimization is based on the assumption that the meta-knowledge dimension is a lattice $(B_D, \vee_D, \wedge_D)$ such that $a \vee_D b = \sup(a, b)$ and $a \wedge_D b = \inf(a, b)$ for $a, b \in B_D$. This is true for all meta-knowledge dimensions which are total orders and for all dimensions discussed above such as trust, modification date and confidence degrees. In this case, the meta-knowledge formula in disjunctive normal form has the structure of a supremum of infima.

A naive evaluation of meta-knowledge for an axiom might compute all pinpoints and then evaluate the pinpointing formula. We make this evaluation more efficient, by exploiting monotonicity properties of our meta-knowledge dimensions where applicable. For instance, in the simple case of a meta-knowledge dimension with a total order, the greatest supremum indicates the final meta-knowledge value. Thus, in many cases, we do not need all pinpoints and we may restrict the computation of the pinpointing formula to those parts that are relevant for the meta-knowledge computation given its particular lattice structure. Therefore, the optimized algorithm for computing meta-knowledge searches for the pinpoint with the greatest meta-knowledge value. If the meta-knowledge dimension does not have a total order, several pinpoints with incomparable meta-knowledge values may be found. Thus, we need to find all pinpoints with maximal meta-knowledge values and merge the corresponding meta-knowledge values to determine the resulting meta-knowledge value.

The algorithm starts by adding the query axiom to the ontology and then iteratively growing a subontology around it based on the syntactic relevance as selection function.

Definition 13 *Syntactic Relevance [JQH08]. An axiom B is directly syntactic relevant for an axiom A, if their signatures overlap, i.e. if they share a concept, role or individual.*

Using the syntactic relevance selection function, `MetaA(O, A, D)` computes the meta-knowledge degree of $O \models A$ in dimension D, if $O \models A$. We start by determining the set of syntactically relevant axioms for A in line 3. Then we add the ones with the greatest meta-knowledge degree to the module in line 5-8. In the inner loop we now recursively add all syntactically relevant axioms for the new module, which have meta-knowledge degrees greater than or equal the current infimum stored in *degree*, as these degrees will not influence the final result. Then in each iteration of the outer loop we lower *degree* (lines 5 and 6), until we have found a module, which contains a pinpoint (line 13). We work with a set for *degree* to account for the fact that in the presence of partial orders, minimum and maximum are not unique. For total orders, *degree* always is a singleton.

Algorithm 2: MetaA(O, A, D)

```
degree ← $\{\top_D\}$ ;
module ← $\{\}$ ;
syn ← $\{B \in O | B$ is syntactically relevant for $A\}$ ;
repeat
    add ← $\{B \in$ syn $| \nexists C \in$ syn: meta(B) $<_D$ meta(C)$\}$ ;
    degree ← $\{$meta(B) | B $\in$ add $\}$ ;
    repeat
        module ← module $\cup$ add syn ← $\{B \in O | B$ is syntactically relevant for module$\}$ ;
        add ← $\{B \in$ syn | $\exists d \in$ degree: meta(B) $>=_D$ d$\}$ ;
    until add $\subseteq$ module ;
    syn ← $\{B \in O | B$ is syntactically relevant for module$\}$ ;
until module $\models A$ ;
return module, degree ;
```

Theorem 1 *Let O be an ontology, A an axiom such that $O \models A$ and D a meta-knowledge dimension with a total order.* `MetaA(O, A, D)` *computes the meta-knowledge degree of $O \models A$ in dimension D.*

Proof: First we show that MetaA terminates. The inner loop terminates when *add* is empty. *add* is assigned the subset of O, which is syntactically relevant to *module* and has a meta-knowledge degree $>=_D$ *degree*. *add* is then added to module. The loop terminates when *add* $\subseteq$ *module*. As O is finite, this must eventually be the case. Analogously in the outer loop, *module* is grown as a subset of O. Again, as O is finite and $O \models A$, the outer loop must eventually terminate. In this case *module* indeed $\models A$ and *degree* $=$ *meta(module)*.

It remains to show that if MetaA terminates, it returns the correct meta-knowledge degree of $O \models A$. As the corresponding meta-knowledge formula is a supremum of infima, we look for the pinpoint with the greater meta knowledge degree. *degree* is initialized with $\top_D$ and gradually reduced in the outer loop, until *module* $\models A$. *module* only contains axioms with a meta-knowledge degree greater or equal *degree*. When the outer loop terminates, *module* contains a (possibly superset of a) greatest pinpoint of A wrt. O. Assume there is a pinpoint, which has not been found and has a greater meta-knowledge degree. Then it must already be part of *module*, as *module* contains all directly or indirectly syntactically relevant axioms with meta-knowledge degrees $>=_D$ *degree*.

For partial orders the resulting *degree* may contain multiple elements, which need to be merged. Moreover, the computed *module* may be too large, as we grow it with all relatively maximal axioms, even though some of their meta-knowledge values may be irrelevant for the final result. MetaPartial computes accurate *degree* and *module* values for partial orders. It uses MetaA to compute an approximation first and then computes all pinpoints within the approximated module to come up with precise results.

Theorem 2 *Let O be an ontology, A an axiom such that $O \models A$ and D a meta-knowledge dimension.* `MetaPartial(O, A, D)` *computes the meta-knowledge degree of $O \models A$ in dimension D.*

The proof is an extension of the one for MetaA. In chapter 4 we evaluate the performance of the optimized algorithm using both randomly generated meta-knowledge and change tracks from a real world ontology editing system.

Algorithm 3: MetaPartial(O, A, D)

```
module, degree $\leftarrow$ MetaA(O, A, D);
pinpoints $\leftarrow$ GetAllPinpoints(A, module);
module $\leftarrow \{B|P \in \text{pinpoints}, B \in P\}$;
singledegree $\leftarrow \bigoplus_{B \in \text{module}} meta(B)$;
**return** *module,* $\{$*singledegree*$\}$ ;
```

Chapter 3

A Segmentation-based Approach for Approximate Query over Distributed and Inconsistent Ontologies

3.1 Preliminaries

We first introduce some prerequisite knowledge about conjunctive query answering [HT00] problem over description logics $\mathcal{SHIQ}$ *KB* that is proved to be decidable in [GHLS07]. We also introduce an ontology mapping system by following the definition of conjunctive query.

3.1.1 Description logic $\mathcal{SHIQ}$

Let $\mathsf{N_R}$ be a set of *role names* with both transitive and normal role names $\mathsf{N_{tR}} \cup \mathsf{N_{nR}} = \mathsf{N_R}$, where $\mathsf{N_{tR}} \cap \mathsf{N_{nR}} = \varnothing$. A $\mathcal{SHIQ}$*-role* is either some $R \in \mathsf{N_R}$ or an *inverse role* R^- for $R \in \mathsf{N_R}$. $\mathsf{Trans}(R)$ and $R \sqsubseteq S$ represent the transitive and inclusion role axioms, respectively, where R and S are roles. $\mathcal{R}$ as a finite set of transitive and inclusion role axioms. A simple role is a $\mathcal{SHIQ}$*-role* that neither its sub-roles nor itself is transitive. Given a set of *concept names* $\mathsf{N_C}$, the set of $\mathcal{SHIQ}$-concepts is the smallest set such that: (1) Every concept name is a concept; (2) if C and D are concepts, R is a role, S is a simple role, and n is a positive integer, then the following expressions are also concepts: ($\top$), ($\bot$), ($\neg C$), ($C \sqcap D$), ($C \sqcup D$), ($\exists R.C$), ($\forall R.C$), ($\leqslant nSC$) and ($\geqslant nSC$). A TBox $\mathcal{T}$ is a finite set of axioms with the form $C \sqsubseteq D$ where C and D are $\mathcal{SHIQ}$-concepts, and an ABox $\mathcal{A}$ is a finite set of axioms with the form $C(x)$, $R(x, y)$, and $x \approx y$ ($x \not\approx y$). A $\mathcal{SHIQ}$ knowledge base (*KB*) is a triple $(\mathcal{R}, \mathcal{T}, \mathcal{A})$ which is also considered as ontology in the semantic information systems here. The semantics of $\mathcal{SHIQ}$ *KB* is given by the interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of $\mathcal{I}$) and the function $\cdot^{\mathcal{I}}$ as usual (e.g., see [HS04]). The reasoning and decidability of $\mathcal{SHIQ}$ is also introduced in [HS04]. The interpretation $\mathcal{I}$ is the model of $\mathcal{R}$ and $\mathcal{T}$ if for each $R \sqsubseteq S \in \mathcal{R}$, $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ and for each $C \sqsubseteq D \in \mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

3.1.2 Conjunctive query answering over $\mathcal{SHIQ}$ *KB*

Let *KB* be a $\mathcal{SHIQ}$ knowledge base, $\mathsf{N_P}$ be a set of names such that all concepts and roles are in $\mathsf{N_P}$. An *atom* $P(s_1, ..., s_n)$ has the form $P(s_1, \ldots, s_n)$, denoted as $P(\mathbf{s})$, where $P \in \mathsf{N_P}$, and s_i are either variables or individuals from *KB*. An atom is called a *DL-atom* if P is a $\mathcal{SHIQ}$-concept or role; it is called *non-DL-atom* otherwise.

Definition 14 (Conjunctive Queries) *Let* $x_1, \ldots, x_n$ *and* $y_1, \ldots, y_m$ *be sets of* distinguished *and* non-distinguished *variables, denoted as* $\mathbf{x}$ *and* $\mathbf{y}$*, respectively. A* conjunctive query $Q(\mathbf{x}, \mathbf{y})$ *over a KB is a conjunction of atoms* $\bigwedge P_i(\mathbf{s_i})$*, where the variables in* $\mathbf{s_i}$ *are contained in either* $\mathbf{x}$ *or* $\mathbf{y}$*. We denote operator*

π [MSS05] to translate $Q(\mathbf{x},\mathbf{y})$ into a first-order formula with free variables $\mathbf{x}$: $\pi(Q(\mathbf{x},\mathbf{y}))$=$\exists\mathbf{y}$: $\bigwedge(P_i(\mathbf{s_i}))$. $\Diamond$

For $Q_1(\mathbf{x},\mathbf{y_1})$ and $Q_2(\mathbf{x},\mathbf{y_2})$ conjunctive queries, a *query containment* axiom $Q_2(\mathbf{x},\mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x},\mathbf{y_1})$ has the following semantics:

$$\pi(Q_2(\mathbf{x},\mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x},\mathbf{y_1})) = \forall\mathbf{x} : \pi(Q_1(\mathbf{x},\mathbf{y_1})) \leftarrow \pi(Q_2(\mathbf{x},\mathbf{y_2}))$$

Definition 15 (Conjunctive Query Answering) *An* answer *of a conjunctive query $Q(\mathbf{x},\mathbf{y})$ w.r.t. KB is an assignment θ of individuals to distinguished variables, using* Ans(Q, *KB*) *as a function, such that π(KB) $\models$ $\pi(Q(\mathbf{x}\theta,\mathbf{y}))$.* $\Diamond$

We refer readers to [GHLS07, MSS05, HT00] for further issues in conjunctive query answering for ontologies. We follow the general framework of [Len02] to formalize the notion of a mapping system for DL ontologies, where mappings are expressed as correspondences between conjunctive queries[1] over ontologies.

Definition 16 (Ontology Mapping System) *An* ontology mapping system *$\mathcal{MS}$ is a triple $(\mathcal{O}_1,\mathcal{O}_2,\mathcal{M})$, where*

- *$\mathcal{O}_1$ is the* source ontology*, $\mathcal{O}_2$ is the* target ontology*,*
- *$\mathcal{M}$ is the mapping between $\mathcal{O}_1$ and $\mathcal{O}_2$, i.e. a set of assertions $q_S \rightsquigarrow q_T$, where q_S and q_T are conjunctive queries over $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively, with the same set of distinguished variables $\mathbf{x}$, and $\rightsquigarrow \in \{\sqsubseteq,\sqsupseteq,\equiv\}$.*

An assertion $q_S \sqsubseteq q_T$ is called a sound *mapping, requiring that q_S is contained by q_T w.r.t. $\mathcal{O}_1 \cup \mathcal{O}_2$; an assertion $q_S \sqsupseteq q_T$ is called a* complete *mapping, requiring that q_T is contained by q_S w.r.t. $\mathcal{O}_1 \cup \mathcal{O}_2$; and an assertion $q_S \equiv q_T$ is called an* exact *mapping, requiring it to be sound and complete.* $\Diamond$

To have the same segmentation result for ontology integration system introduced later, we do not consider mapping transitivity here, i.e., if $\mathcal{O}_1$ and $\mathcal{O}_2$ have mapping $\mathcal{M}_{12}$; $\mathcal{O}_2$ and $\mathcal{O}_3$ have mapping $\mathcal{M}_{23}$, it does not imply the existence of mapping $\mathcal{M}_{13}$ between $\mathcal{O}_1$ and $\mathcal{O}_3$. Furthermore, several mappings between two ontologies are considered as one single mapping. This form of mapping is decidable in inferencing while it is restricted to DL-safe mappings [HM05]. Mappings discussed in this paper are referred as DL-safe mapping by default. Further details about semantics and restrictions of ontology mapping system can be found in [HM05].

3.2 Segment-based Conjunctive Query Answering over Distributed Ontologies

To discover the possible optimizations using approximate conjunctive query answering in the distributed environment, we need to analyze the distributed ontologies by considering their integration via mappings. We here define ontology integration system using mappings for distributed networking scenario. In the following, we denote $I = \{1,\ldots,n\}, n \in \mathbb{N}$ and $i \neq j; i,j \in I$.

Definition 17 (Distributed Ontology Integration System (DOIS)) *A distributed ontology integration system (DOIS) is a triple $(\{\mathcal{MS}_i\},\mathcal{N},\mathsf{Loc})$, where*

1. *$\{\mathcal{MS}_i\}$ is a set of mapping systems. We denote $\mathbf{O}$ and $\mathbf{M}$ as the ontologies and mappings included in $\{\mathcal{MS}_i\}$, respectively.*

[1]We denote a conjunctive query as $q(\mathbf{x},\mathbf{y})$, with $\mathbf{x}$ and $\mathbf{y}$ sets of *distinguished* and *non-distinguished* variables, respectively.

2. *$\mathcal{N}$ is a set of distributed nodes where the ontologies and mappings reside;*

3. *$\mathsf{Loc} : \mathbf{O} \cup \mathbf{M} \to \mathcal{N}$ is a location function such that $N_i = \mathsf{Loc}(\mathcal{O}_i)$ and $N_{ij} = \mathsf{Loc}(\mathcal{M}_{ij})$, where $N_i, N_{ij} \in \mathcal{N}$ and $\mathcal{O}_i \in \mathbf{O}, \mathcal{M}_{ij} \in \mathbf{M}$. This function aims to relate an ontology or mapping to a specific distributed node.* $\diamondsuit$

Given DOIS $(\{\mathcal{MS}_i\}, \mathcal{N}, \mathsf{Loc})$ over $\mathcal{O}$, we use $\mathsf{Ans}_e(Q, \{\mathcal{MS}_i\}, \mathcal{O})$ to denote the complete set of answers for conjunctive query $Q(\mathbf{x}, \mathbf{y})$ over $\mathbf{O} \cup \mathbf{M}$.

To simplify the presentation, in the following we introduce the notion of distributed system, which is inspired from [ZE06] but not exactly the same.

Definition 18 (Distributed System) *Distributed system $\mathfrak{D}$ is a set of mapping systems $\{\mathcal{MS}_i\}$. Or equivalently, $\mathfrak{D} = (\mathbf{O}, \mathbf{M})$ where $\mathbf{O} = \{\mathcal{O}_i\}$ is a set of ontologies and $\mathbf{M} = \{\mathcal{M}_{ij}\}$ is a set of mappings between $\mathcal{O}_i$ and $\mathcal{O}_j$ in $\mathbf{O}$.* $\diamondsuit$

We introduce the notion of distributed system in addition to Definition 17 for better understanding our graph-based segmentation approach and algorithms, because by looking at Example 1, it is very easy to see $\mathfrak{D}$ is a graph with ontologies as vertex and mappings as edges.

Example 1 *The DOIS depicted in Figure 3.1 contains five ontologies distributed over five nodes, where $\mathcal{O}_t$ is the target ontology that has non-empty mappings with all other source ontologies. $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ are connected by mappings, presented as dotted line.*

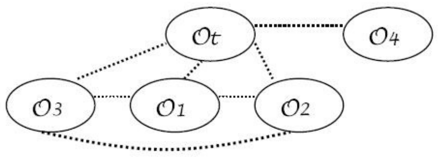


Figure 3.1: Figure for Example 1. Given $\mathfrak{D} = (\mathbf{O}, \mathbf{M})$, source ontology $\mathcal{O}_i \in \mathbf{O}$, target ontology $\mathcal{O}_t \in \mathbf{O}$ and mappings (light-dot lines) between them. Circles are ontologies distributed on different nodes. $\triangle$

We define the notion of an inconsistent distributed system.

Definition 19 *[MST07] Given a distributed system $\mathfrak{D} = (\mathbf{O}, \mathbf{M})$ where $\mathbf{O} = \{\mathcal{O}_i\}$ is a set of ontologies and $\mathbf{M} = \{\mathcal{M}_{ij}\}$ is a set of mappings between $\mathcal{O}_i$ and $\mathcal{O}_j$ in $\mathbf{O}$. The* union *$Union(\mathfrak{D})$ of $\mathbf{O}$ and $\mathbf{M}$ is defined as $Union(\mathfrak{D}) = \cup_{\mathcal{O}_i \in \mathbf{O}} \mathcal{O}_i \cup_{\mathcal{M}_{ij} \in \mathbf{M}} \{t(m) : m \in \mathcal{M}_{ij}\}$ with t being a translation function that converts a correspondence into an axiom in the following way: $t(\langle C, C', r, \alpha \rangle) = CrC'$.*

That is, we first translate all the correspondences in $\mathbf{M}$ into DL axioms, then the union of the ontologies connected by the mappings is the set-union of the ontologies and the translated axioms.

Definition 20 (Inconsistent Distributed System) *Given a distributed system $\mathfrak{D} = (\mathbf{O}, \mathbf{M})$ where $\mathbf{O} = \{\mathcal{O}_i\}$ is a set of ontologies and $\mathbf{M} = \{\mathcal{M}_{ij}\}$ is a set of mappings between $\mathcal{O}_i$ and $\mathcal{O}_j$ in $\mathbf{O}$. $\mathfrak{D}$ is inconsistent if $Union(\mathfrak{D})$ is inconsistent.* $\diamondsuit$

Given a DOIS, we know the following information: (1) A mapping $\mathcal{M}$ and its source and target ontologies; (2) the locations of those mappings or ontologies in the distributed network. Now the we are able to query over distributed ontologies that are integrated as a global ontology in mapping systems. Because a DOIS consists of different ontologies, it is possible to improve the overall performance by dividing the distributed system of a DOIS into several segments (Example 2). Let's consider a simple example: The Law School and Faculty of Physics normally do not share projects or professors, or the shared information is not usually recognized by people. Therefore, ontologies describing these two departments are going to be grouped into different segments.

Definition 21 (Distributed Subsystem) *Given a distributed system $\mathfrak{D}$, distributed subsystem of $\mathfrak{D}$ is a pair $(\mathbf{O}', \mathbf{M}')$, denoted as $\mathfrak{D}' \sqsubseteq \mathfrak{D}$, iff $\mathbf{O}' \subseteq \mathbf{O}$ and $\mathbf{M}' = \{\mathcal{M}_{ij} \in \mathbf{M} : \mathcal{O}_i, \mathcal{O}_j \in \mathbf{O}'\}$. A distributed subsystem is also a distributed system.* $\Diamond$

Definition 22 (Segment) *Given a distributed system $\mathfrak{D} = (\mathbf{O}, \mathbf{M})$, a* segment *of $\mathfrak{D}$ is distributed subsystem $\mathfrak{S} = (\mathbf{O}', \mathbf{M}')$ such that*

1. *for all $\mathcal{M}_{ij} \in \mathbf{M}'$, we have $\mathcal{M}_{ij} \neq \varnothing$;*
2. *for any distributed subsystem $\mathfrak{S}'$ of $\mathfrak{D}$, if $\mathfrak{S} \sqsubset \mathfrak{S}'$, then $\mathfrak{S}'$ does not satisfy 1.* $\Diamond$

Different from [SR06], which aims to generate segments from a large domain ontology to facilitate ontology engineering, we can see a *segment* here is distributed subsystem $\mathfrak{S}$ of $\mathfrak{D}$ which satisfies the condition that all the ontologies in it are connected by non-empty mappings and any other distributed subsystem of $\mathfrak{D}$ which strictly includes $\mathfrak{S}$ does not satisfy this condition. This has two benefits:

1. If the ontologies of a distributed subsystem are not all connected by non-empty mappings, then this distributed subsystem can be divided into smaller distributed subsystems to improve performance.
2. We achieve completeness of answers as much as possible after segmentation by requiring the segment to be the inclusion maximal distributed subsystem which satisfies Condition 1 in Definition 22.

Therefore, our definition of segment *perfectly captures* the idea of balancing the trade-off between performance and completeness in querying distributed ontologies – the Example 2 presents how the segmentation is processed.

We are able to develop an algorithm for segmenting a distributed system $\mathfrak{D}$ directly based on existing algorithms (e.g., union-find algorithm [GI91]) to process graphs. Note that not all the complete subgraphs of G can be interpreted to segments, e.g., assuming a segment $(\{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3\}, \{\mathcal{M}_{12}, \mathcal{M}_{13}, \mathcal{M}_{23}\})$, apparently, $(\{\mathcal{O}_1, \mathcal{O}_2\}, \{\mathcal{M}_{12}\})$ forms a complete graph but it doesn't satisfy Definition 22. Let's see an example about how mappings affect segmenting result to form DOISs with different distributed nodes.

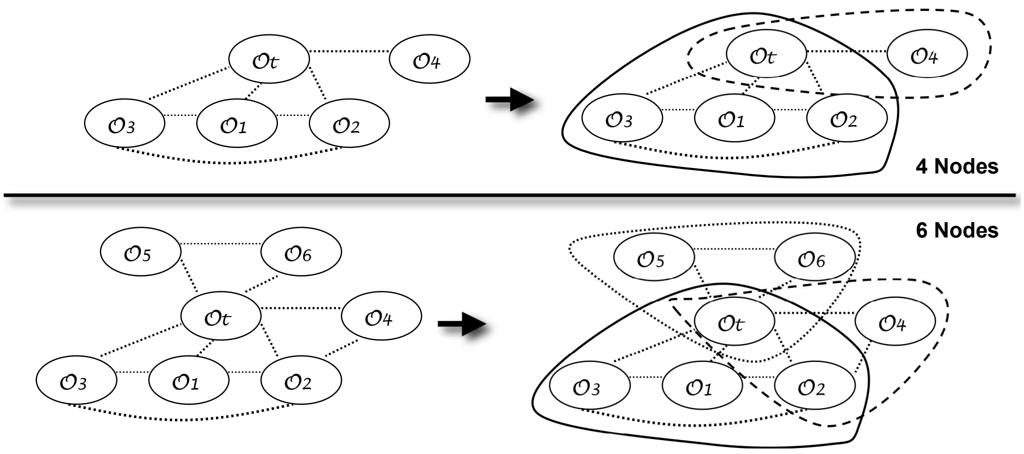


Figure 3.2: Figure for Example 2. Given $\mathfrak{D} = (\mathbf{O}, \mathbf{M})$, source ontology $\mathcal{O}_i \in \mathbf{O}$, target ontology $\mathcal{O}_t \in \mathbf{O}$ and mappings (light-dot lines) between them.

Example 2 *In this example, we use 5 and 7 distributed ontologies for each DOIS, respectively. Next we will segment the DIOS to show how our segmenting approach works based on definitions above.*

1. *Let's first look at the case of four distributed nodes with one ontology on each node. Before segmenting the DOIS, we have mappings $\mathcal{M}_{12}$, $\mathcal{M}_{13}$ and $\mathcal{M}_{23}$ between ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$, $\mathcal{O}_1$ and $\mathcal{O}_3$, $\mathcal{O}_2$ and $\mathcal{O}_3$, $\mathcal{O}_2$ and $\mathcal{O}_4$, respectively. We also have mappings from target ontology to each source ontology*

$\mathcal{M}_{t1}$, $\mathcal{M}_{t2}$, $\mathcal{M}_{t3}$ and $\mathcal{M}_{t4}$. According to the definition of segments, we have the segmentation result depicted by following the arrow:

(a) $(\{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \mathcal{O}_t\}, \{\mathcal{M}_{t1}, \mathcal{M}_{t2}, \mathcal{M}_{t3}, \mathcal{M}_{13}, \mathcal{M}_{12}, \mathcal{M}_{23}\})$

(b) $(\{\mathcal{O}_4, \mathcal{O}_t\}, \{\mathcal{M}_{t4}\})$.

So there are two segments in this case.

2. *It is very often the case that the distributed nodes and extra mappings are added into the current system. Our seven nodes example indicates the segmentation status if we add $\mathcal{O}_5$ and $\mathcal{O}_6$ with mappings $\mathcal{M}_{24}$ and $\mathcal{M}_{56}$ to the five nodes distributed system. Mappings between target ontology and $\mathcal{O}_5$ and $\mathcal{O}_6$ are $\mathcal{M}_{t5}$ and $\mathcal{M}_{t6}$, respectively. Then we have the following segmentation result:*

   (c) $(\{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \mathcal{O}_t\}, \{\mathcal{M}_{t1}, \mathcal{M}_{t2}, \mathcal{M}_{t3}, \mathcal{M}_{13}, \mathcal{M}_{12}, \mathcal{M}_{23}\})$

   (d) $(\{\mathcal{O}_2, \mathcal{O}_4, \mathcal{O}_t\}, \{\mathcal{M}_{t2}, \mathcal{M}_{t4}, \mathcal{M}_{24}\})$

   (e) $(\{\mathcal{O}_5, \mathcal{O}_6, \mathcal{O}_t\}, \{\mathcal{M}_{t5}, \mathcal{M}_{t6}, \mathcal{M}_{56}\})$

   So there are three segments in this case. $\triangle$

As mentioned in the introduction, our aim is to find a balance between the completeness and performance for query answering over distributed ontologies by using segments of distributed systems. Thus, the union of the individual query answering result of each segment obviously *may not* be equal to the exact answers. We therefore introduce our approach to find out *segment-based answers* in querying distributed ontologies. In the mean time, for uses who want to achieve complete answers, we also provide an alternative approach to compute them (Algorithm 5).

Definition 23 (Segment-based Query Answering) *Given a DOIS with $\mathfrak{D} = (\mathbf{O}, \mathbf{M})$ and a conjunctive query Q, let $\mathbf{S} = \{\mathfrak{S}_i\}_{i=1,\ldots,n;\ n\in I}$ and* $\mathsf{Ans}_e(Q, \mathfrak{D}, \mathcal{O})$ *be* complete answers to query *over $\mathcal{O}$, where $\mathcal{O}$ is the target ontology and $\mathcal{O} \in \mathbf{O}$.* Segment-based Query Answering *in DOIS is defined as:*

$$\mathsf{Ans}_a(Q, \mathfrak{D}, \mathcal{O}) = \bigcup_{i\in\{1,\ldots,n\}} \mathsf{Ans}_e(Q, \mathfrak{S}_i, \mathcal{O})$$

where Ans_a *stands for segment-based query answering.* $\diamondsuit$

Obviously, the set of segment-based answers are subset of the set of complete answers because the number of answers monotonically increases. Let's illustrate this by an example.

Example 3 *Assume ontologies from different departments $\mathcal{O}_1$, $\mathcal{O}_2$ and mapping $\mathcal{M}$ bridging them:*

- $\mathcal{O}_1$=$\{\mathsf{Professor} \sqsubseteq \mathsf{Faculty} \sqcap \exists \mathsf{teach.Course}\}$
- $\mathcal{O}_2$=$\{\mathsf{Professor} \sqsubseteq \mathsf{Staff} \sqcap \exists \mathsf{teach.Lecture}\}$
- $\mathcal{M}$=$\{1{:}\mathsf{Professor} \sqsubseteq 2{:}\mathsf{Professor}, 1{:}\mathsf{Faculty} \sqsubseteq 2{:}\mathsf{Staff}, 2{:}\mathsf{Lecture} \sqsubseteq 1{:}\mathsf{Course}\}$

Here, 1:Professor *means concept* Professor *in $\mathcal{O}_1$. If $\mathcal{M}$ holds and we ask for professors who teaches a certain course, we get complete answers because $\mathcal{O}_1$ and $\mathcal{O}_2$ are integrated with $\mathcal{M}$ as a global ontology. However, if $\mathcal{M}$ does not exist, then we can ask for segment-based answers (i.e., $\mathcal{O}_1$ and $\mathcal{O}_2$ are divided into different segments). In this case the professors who give lectures with other departments are not included in the answers but can be computed in the end of our algorithm.* $\triangle$

3.3 Algorithms for Segment-based Query Answering

There are three algorithms in this segment-based conjunctive query answering approach: (1) Segmentation, (2) query distribution and answering, and (3) termination and results collection. Different query distribution approaches are adopted for segments with different number of mapping elements for better allocating computing resources. The *cardinality* of $\mathfrak{S}$ is $\text{Card}(\mathfrak{S}) = |\mathbf{M}|$ that indicates the number of elements contained in $\mathbf{M}$. Then,

- a segment is called *single-element segment*, denoted as $\mathfrak{S}_s$, iff $\text{Card}(\mathfrak{S}_s) = 1$;
- a segment is called *multiple-element segment*, denoted as $\mathfrak{S}_m$, iff $\text{Card}(\mathfrak{S}_m) > 1$.

The *single-element segment* is a segment which contains only two ontologies (i.e., a target ontology $\mathcal{O}_t$ and a source ontology $\mathcal{O}_i$ with mapping $\mathcal{M}_{ti}$ between them, e.g., segment (b) in Example 1 is a single-element segment). On the other hand, the *multiple-element segment* consists more than one mapping ontologies connecting more than two ontologies (e.g., all segments in Example 1 except (b) are multiple-element segments). The exact conjunctive query answering are denoted as $\text{Ans}_e(Q, \mathfrak{S}_s, \mathcal{O})$ and $\text{Ans}_e(Q, \mathfrak{S}_m, \mathcal{O})$ for single/multiple-element segments, respectively, where Q is a conjunctive query over a common target ontology shared by all the segments of $\mathfrak{D}$.

Algorithm 4: Segmentation

```
Data: A DOIS $(\mathfrak{D}, \mathcal{N}, \text{Loc})$, a target ontology $\mathcal{O}_t$, empty ontology set $\mathbf{O}$, empty mapping set $\mathbf{M}$ and empty
      graph set $\mathcal{G}$
begin
    get all ontologies that have non-empty mappings with $\mathcal{O}_t$ and put them to $\mathbf{O}$;
    get all mappings related $\mathcal{O}_t$ to $\mathbf{O}$ and put them to $\mathbf{M}_t$;
    get all mappings among ontologies in $\mathbf{O}$ and put them to $\mathbf{M}$, add $\mathcal{O}_t$ to $\mathbf{O}$, add $\mathbf{M}_t$ to $\mathbf{M}$;
    establish graph $G$ with $n$ vertices $\mathbf{O}$ and edges $\mathbf{M}$, where $n$ is the number of ontologies in $\mathbf{O}$;
    $k = n$;
    while $k \geqslant 2$ do
        get all complete subgraphs of $G$ with number of vertices $k$ and put them to $\mathcal{G}_k$;
        remove all subgraphs that are already in $\mathcal{G}$ from $\mathcal{G}_k$;
        add $\mathcal{G}_k$ to $\mathcal{G}$;
        $k - -$;
    establish a set of segments $\mathbf{S}$ of $\mathfrak{D}$ based on $\mathcal{G}$;
    remove segments that are inconsistent from $\mathbf{S}$;
    for $\mathfrak{S}_l \in \mathbf{S}$ do
        put $\mathfrak{S}_l$ into $\mathbf{S}_s$ if $\mathfrak{S}_l$ is a single-element segment, else, put $\mathfrak{S}_l$ into $\mathbf{S}_m$;
    output $\mathbf{S}_s$ and $\mathbf{S}_m$;
end
```

Algorithm 4 starts with input of a DOIS with distributed system $\mathfrak{D}$ and a target ontology $\mathcal{O}_t$ over which the query is about to be executed. The system gets all ontologies connected to $\mathcal{O}_t$ and the corresponding mappings to establish a graph G (Step 1–4), and then it extracts the segments with k number of ontologies ($2 \leqslant k \leqslant n$) iteratively by computing the complete subgraphs of G with k vertices using classic *union-find algorithm* [Gl91] (Step 5–12), this is exactly the segmentation procedure presented in Example 1). In the meanwhile, we need to eliminate those subgraphs of complete subgraphs of G with higher number vertices to make sure all the generated subgraphs are maximal complete subgraphs (Step 8). Then, we interpret the generated set of maximal complete subgraphs $\mathcal{G}$ back to a set of segments, written as $\mathbf{S}$ (Step 10). We remove those segments that are inconsistent from $\mathbf{S}$ (Step 11). To facilitate query distribution, we classify the segments into single and multiple-element segments as input of the consequent Algorithm 5 (Step 12–15). We show all segments should contain $\mathcal{O}_t$.

Proposition 4 *Given a DOIS* $(\mathfrak{D}, \mathcal{N}, \mathsf{Loc})$*, a target ontology* $\mathcal{O}_t$ *and a set of segments* $\mathbf{S}$ *generated in the Step 14 of Algorithm 4, the target ontology* $\mathcal{O}_t$ *is included in all the segments in* $\mathbf{S}$*.*

Proof sketch. If G is complete graph without vertex $\mathcal{O}_t$, then G and $\mathcal{O}_t$ forms a complete graph. According to Definition 22, G can not be interpreted as a segment. □

Algorithm 5: Query distribution and consistent query answering

```
Data: A DOIS $(\mathfrak{D}, \mathcal{N}, \mathsf{Loc})$, a conjunctive query $Q$, a target ontology $\mathcal{O}_t$, segments $\mathbf{S}_s$ and $\mathbf{S}_m$
begin
    create an empty list $L_N$ for nodes that are idle (i.e., nodes that are not involved in executing query
    answering tasks);
    for $\mathfrak{S}_{s_i} \in \mathbf{S}_s$ do
        get ontology source ontologies $\mathcal{O}_i$ of $\mathcal{O}_t$ in $\mathfrak{S}_{s_i}$; get remote node $N_i = \mathsf{Loc}(\mathcal{O}_i)$, put $N_i$ into $L_N$;
        compute $\mathsf{Ans}_a(Q, \mathfrak{S}_{s_i}, \mathcal{O})$ on node $N_i$ in parallel (i.e., parallel processing means this task is
        executed in parallel in different nodes over distributed network);
    for $\mathfrak{S}_{m_j} \in \mathbf{S}_m$ do
        get source ontology set $\mathbf{O}_j$ of $\mathcal{O}_t$ in $\mathfrak{S}_{m_j}$;
        find an arbitrary idle node $N_j = \mathsf{Loc}(\mathcal{O}_j)$, where $\mathcal{O}_j \in \mathbf{O}_j$;
        put $N_j$ into $L_N$;
        compute $\mathsf{Ans}_a(Q, \mathfrak{S}_{m_j}, \mathcal{O})$ on node $N_j$ in parallel;
    get a random idle node $N_i$, compute $\mathsf{Ans}_e(Q, \mathfrak{D}, \mathcal{O})$ on $N_i$ in parallel ;
end
```

In Algorithm 5, for a single-element segment, the system sends the query to the distributed node where the source ontology in $\mathfrak{S}_{s_i}$ resides (Step 2–5, please note there is only one source ontology in each $\mathfrak{S}_{s_i}$ so the nodes are not occupied at this stage); for multiple-element segment, the system sends the query to an arbitrary unused node where an arbitrary source ontology resides (Step 6–11). To achieve exact query answering (Step 12) as supplement, the algorithm simply integrates all ontologies and mapping in $\mathfrak{D}$ on an unused remote node without segmentation and query over $\mathfrak{D}$. In Step 6–12, if all nodes are occupied, the algorithm waits until an arbitrary node finishes its querying task and put the next query answering task to this node – this procedure managed independently by Algorithm 6 for termination and result collection.

Algorithm 6: Termination and results collection

```
Data: list of nodes $L_N$ that is in-use (not idle), termination boolean signal $U$, timeout preset $T$
begin
    while $L_N \neq \emptyset$ and $U$ =FALSE and $T \neq$ TIMEOUT do
        if $N_i$ then
            returns anwser;
        send out the answer from $N_i$, remove $N_i$ from $L_N$;
    final results collection;
end
```

Algorithm 6 returns the real time segment-based answers, manages the distributed nodes and monitors the terminating signal. Once all distributed nodes are not in use, or querying process are terminated. The system also terminates in a preset, maximum allowed execution time, considering one or several distributed tasks doesn't respond due to possible system or network failures.

In Algorithm 4, the optimized clique discovery problem to find complete subgraphs in graphs with size n, which is the number of ontologies, has complexity LOG-TIME [GI91]. The complexity of conjunctive query answering over $\mathcal{SHIQ}$ *KB* with size m, which is the number of concepts, is CO-NP-COMPLETE [GHLS07], which is the major computation in Algorithm 5. Our algorithms do not reduce the computational complexity of query answering over distributed ontologies. Because the cope of our approach is to balance the trade-off between completeness and performance in distributed ontology query answering for actual web information systems, but not to optimize the query processing algorithms in local query processing.

Chapter 4

Evaluation

In this chapter, we provide some preliminary evaluation results on our approach for reasoning with meta knowledge and our approach for approximate query over distributed ontologies separately. Although the distributed ontologies considered in our experiments are consistent, we expect that positive results can be achieved even if inconsistent distributed ontologies are considered.

4.1 Complexity and Evaluation of Reasoning with Meta Knowledge

The complexity of both the naive and the optimized approach for computing meta knowledge is equivalent to the computation of pinpoints in the underlying logic as the meta knowledge formula can be evaluated in polynomial time. If it is expressed in normal form, however, the size of the formula can blow up exponentially. Approaches for computing pinpoints like [BP07] which, rather than representing pinpoints formula in a normal form, derive a compact representation of the pinpoints formula benefit the computation of meta knowledge since they avoid exponential blowup.

While in the average case the complexity of computing meta knowledge is the same as the complexity of finding all pinpoints and hence quite high, in the average case we can do much better using the optimized algorithm.

Evaluation Goals We want to evaluate whether meta knowledge based on pinpointing is fast enough to support a users assessment of the reliability of inferences in real time.

Evaluation Criteria In order to demonstrate the scaleability of our very general approach for determining meta knowledge, we have performed several type of experiments. In the first type of experiments, we have enriched existing ontologies with artificial meta knowledge dimensions and values. For this purpose, we selected ontologies that had been under investigation in related work demonstrating the efficiency of traditional computation of pinpoints (ontologies 1-6). In the second type of experiments, we have used an actual log of changes to a large and expressive ontology (ontology 7) to check whether our findings based on artificial data carry over to the real world.

Let us now consider the experiments. We have evaluated the average absolute time needed to compute the meta knowledge for an inconsistency, considering all inconsistent classes per ontology. For the ontology 7, which contains the real world change tracks, we had to add some inconsistencies to run the experiment.

The evaluation is performed with four different kinds of data. (Naive) As a baseline we use the pinpointing algorithm in the OWLAPI to compute all pinpoints. (Optimized Random) We augmented ontology 1-6 with random meta knowledge in a single dimension and compute the meta knowledge of the inconsistencies. (Optimized Cluster 1 Dimension and Optimized Cluster 2 Dimensions) We augmented ontology 1-6 with semi random meta knowledge, such that clusters of axioms, which are syntactically relevant to each other are assigned similar meta knowledge degrees. This shall reflect the fact that a user usually does not do random modifications, but changes a part of the ontology focused around a certain class or property. We performed this experiment with a single and with two dimensions to investigate the influence of combining

ID	Ontology	Expressivity	Axioms
1	People+Pets	$\mathcal{ALCHOIN}$	108
2	MiniTambis	$\mathcal{ALCN}$	173
3	University	$\mathcal{SOIN}$	52
4	Economy	$\mathcal{ALCH}(\mathcal{S})$	1625
5	Chemical	$\mathcal{ALCHF}$	114
6	Transport	$\mathcal{ALCH}$	1157
7	BIO	$\mathcal{SHOIN}$(D)	8839

Table 4.1: Ontologies used in experiment

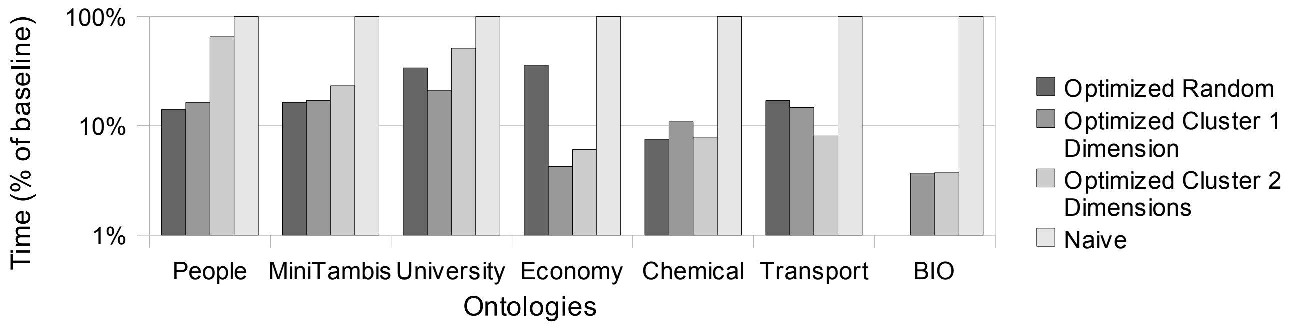


Figure 4.1: Average Time to Compute Meta Knowledge for an Inconsistency

dimensions. (Real life change tracks) For ontology 7 we performed the same tests as for ontologies 1-6, but the meta knowledge is taken from real world change tracks instead of being generated.

Evaluation Setting The prototype has been implemented in Java using Pellet and OWLAPI. For the naive implementation the black box algorithm in the OWLAPI is used for computing all pinpoints. An overview of the ontologies used for evaluation is shown in table 4.1. This dataset has already been used for testing the computing time of laconic justifications in [HPS08], except for ontology 7. The evaluation was run on a dual core Atom 1,6Ghz Processor running Windows XP.

Evaluation Results The results of the evaluation are shown in Figure 4.1. We have normalized the results to the processing time for the naive approach and used a logarithmic scale, as the absolute numbers ranged over three orders of magnitude. Our optimized algorithm performs significantly better for all cases and scales very well. Moreover, in every case, meta knowledge can be computed in well under a second, which is fast enough for interactive applications. The absolute times needed on our rather slow machine ranged from 71ms for ontology 6 to 812ms for ontology 7 for the "Optimized Cluster 2 Dimensions" experiment.

Furthermore, we observe that reasoning with many independent dimensions can have a negative performance impact due to lots of incomparable values in the combined dimension. In such cases, meta knowledge in the independent dimensions should be computed separately. Results for the real world data from the BIO ontology show, that in reality this is less a problem, as modification dates and creator are not independent there.

4.2 Evaluation of Distributed Reasoning

We have implemented our approach for distributed reasoning using KAON2 as the query answering engine for this evaluation[1]. Since our approach aims to improve the overall performance in querying distributed ontologies approximately, we need to evaluate the performance increases (presented by time saved) against the loss of completeness (presented by rates of the cardinality of the set of segment-based answers to that of

[1]The experimental implementation is available upon request

the set of complete answers). The hypothesis for this evaluation is that the time saved outweighs the rates of approximation to complete answers. We show our evaluation by first introducing the setting for experiments, presenting and discussing the results afterwards.

4.2.1 Experiment settings

We used 17 virtual distributed nodes to simulate a distributed network. The nodes are "virtual" because virtual machines were deployed on four actual computers, therefore the CPU power was shared. Each node held an ontology instance data set with fixed size. The instance data have different schemata that are either heterogeneous or homogenous. In our experiment, we are using four ontology schemata: (1) The Lehigh University Benchmark (LUBM) ontology[2]; (2) Proton ontology[3]; (3) SWRC ontology [SBH+05]; (4) FOAF ontology.[4] The instance data includes the following:

1. LUBM automatically generated ontologies that includes instance data of information about university life
2. Digital library data in Proton schema
3. Documents and publication data in Institute AIFB, University of Karlsruhe
4. Project and personal contacting information in FOAF schema

The ontology mappings were created manually for actual use in different projects (e.g., Proton–SWRC mapping was created for EU IST SEKT project[5], FOAF–LUBM mapping was created for Traditional Chinese Medicine project [CWW+06].)

Each node held a data set with either SWRC, Proton, LUBM or FOAF schema with corresponding ontology instance data with size approximately 1MB. Let's look at a 5 segments example of our experiment setting to see how the data and schemata are allocated. The 17 ontologies and the mappings between them constitute a distributed system $\mathfrak{D}$, whereas the segmentation process only applies to those ontologies that hold mappings with target ontology.

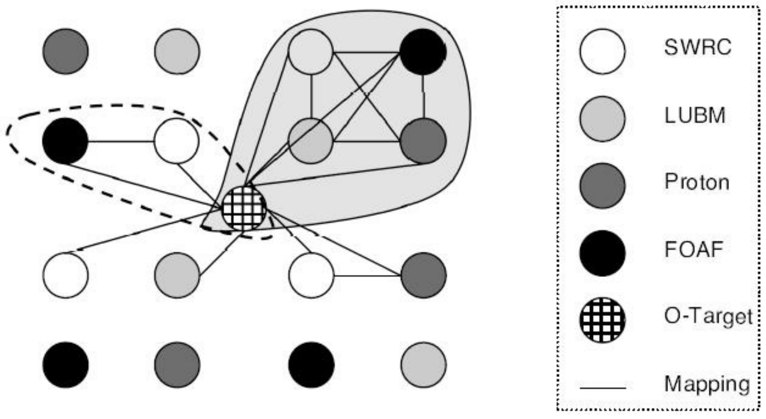


Figure 4.2: Example of data allocation in 5 segments case. The circles stand for distributed ontologies with different schemata; O-Target means the target ontology.

We used two conjunctive queries: One was to search the documents with their corresponding authors who were professors; the other was to find out all the abstracts of the documents written by Yimin Wang in 2006 with the associated projects. The data had many schemata created for different purpose, for example, data in SWRC schema were created for the local research group website portal, data in FOAF schema were created

[2]http://swat.cse.lehigh.edu/projects/lubm/
[3]http://proton.semanticweb.org/
[4]http://xmlns.com/foaf/0.1/
[5]http://www.sekt-project.com

for the Chinese Traditional Medicine project management, etc. Therefore, it was possible that Yimin Wang was working for a project information in FAOF schema and has publication information in SWRC schema – the mapping was used to infer complete information in this case.

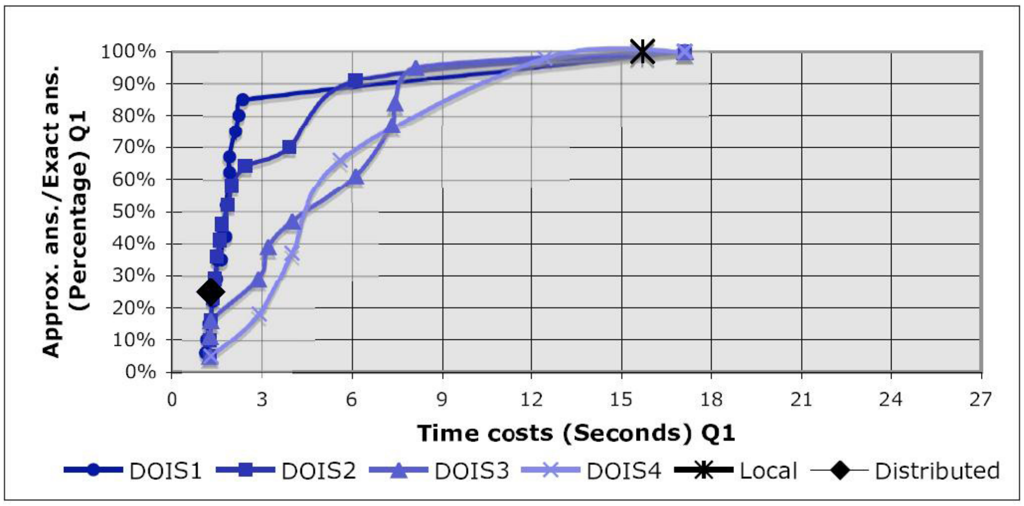
```
1. SELECT ?x ?y
   WHERE { ?x rdf:type Professor . ?y rdf:type Document . ?y publicationAuthor ?x }

2. SELECT ?x ?y ?k
   WHERE { ?x year "2006". ?x author ?z . ?x abstract ?y .
           ?z name "Yimin Wang" . ?z worksFor ?k . ?k rdf:type Project}
```

An experiment unit was one execution of one query over a certain setting with different mappings covered. Based on Definition 22, it's easy to find when the number of mappings changes, the segmentation results are different. In this experiment, we set the DOISs to have four different settings with 16, 12, 9, 5 segments (Figure 4.2 depicts the 5 segments case), respectively. The two queries above were executed for 20 times for each experiment unit and the average execution time was computed, recorded and compared with the execution time held by exact query answering over same size of data. We compared times saved by using segment-based approach against the rates of approximation in different stages. We present both the time costs using global integration with and without segmentation approach applied.

4.2.2 Results and discussions

In the experiments, the two queries above were executed in the distributed network and the results were collected and presented in Figure 4.3.



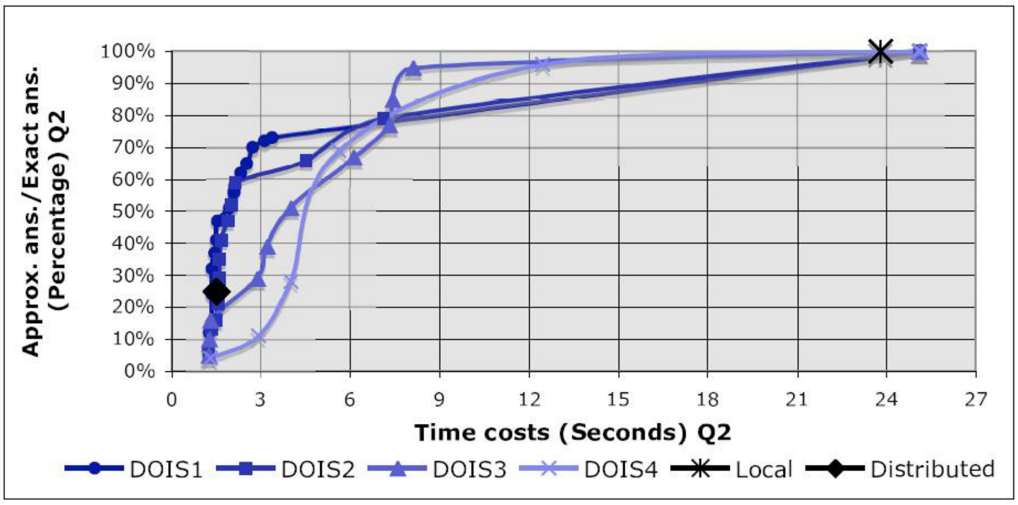


Figure 4.3: Experiment results for the two queries. respectively.

The X axis presents the time costs and the Y axis shows the corresponding rates of approximation (i.e. rate of approximation equals to $\frac{N_a}{N_e}$, where N_a are the numbers of segment-based and N_e are all answers, respectively).

The system computes the answers of segments on each distributed node and returns the answers one by one incrementally. Thus, in Figure 4.3, there are 16 points indicating the time saved against the rates of approximation. Similarly, DOIS$_2$, DOIS$_3$ and DOIS$_4$ have 12, 9 and 5 points in the figures, respectively. The black star and diamond symbols in two figures are time costs of querying global integration of all ontologies in $\mathfrak{D}$ in local space, and querying complete distributed without considering mappings and integration. We can obviously see these two extreme situations are significantly short of either performance or completeness.

By aligning to the motivated scenario in Section 3.2 that is considered to be common on today's Web, there are three major messages delivered by this evaluation: First, in the motivated scenario, data in individual departments of a big organizations are usually reasonably (not heavily) interconnected. Moreover, the mapping among the departments are usually under control to avoid unintended information sharing. In this case, our approach have good rates of approximation with remarkable time saved. Taking DOIS$_1$ in Figure 4.3 for example, when the results have more than 80% rate of approximation to the set of complete answers by using fully global integration in local space, up to 85%($\frac{15.7-2.3}{15.7}$) and 68% ($\frac{23.8-7.5}{23.8}$) execution time has been saved for the first and second queries respectively. Compare to the case without considering mappings and integration, we see the completeness is only about 25%. This is quite obvious: We have ontologies with four different schemata, so we can approximately get one of four answers if we do not integrate the distributed ontologies using mappings.

Second, however, we have also realized the rate of approximation relies on the queries. For instance, in Figure 4.3, the first query has better rate of approximation than the second one because the first query is less possible to access remote ontologies: The information about professors and their publications usually reside in individual departments, whereas in the second query the projects are very likely to be shared across departments or universities, resulting frequent usage of mappings. For certain queries, it is possible that the approximate process returns the complete answers, or doesn't give any result at all – that's why we also provide the exact query answering functionality.

Last but not least, personalization of query answering task is a key issue in our approach. It's configurable to make certain mappings integrated or not. People just need to tune the parameters to terminate the query answering procedure if they are satisfied with the segment-based answers, or wait for all the answers. Figure 4.3 indicates the performance is very promising if anticipated answers are achieved and process is terminated in the middle of runtime.

In a nutshell, our approach, which addresses segment-based conjunctive query answering over distributed ontologies, well meets our target, resulting in sound but *may* incomplete answers in a very efficient manner, especially when the answers are acceptable in practical scenarios.

Chapter 5

Conclusion

In this deliverable, we first reduced bilattice based reasoning to meta knowledge reasoning by adjusting the technique developed for $\mathcal{ALC}\triangle$ in D1.2.4. An efficient algorithm was then provided for meta knowledge reasoning. After that, we described a novel approach to address the problem of balancing the trade-off between completeness and performance in conjunctive query answering over inconsistent and distributed ontologies. We introduced the notions of segment-based conjunctive query answering to achieve better performance with acceptable completeness by introducing distributed system segmentation for query distribution. We designed and implemented three algorithms to support our approach. We provided evaluation results for both the approach for meta knowledge reasoning and the approach for distributed reasoning. The evaluation results indicated that our approaches are very promising in the motivated scenario.

There is a possibility to combine the segmentation-based approach with the billatice-based semantics. That is, if we do not require that segments returned by Algorithm 4 be consistent, then we can apply the billatice-based semantics to handle those inconsistent segments. This work will be left as future work.

Bibliography

[AdA07] B. Thomas Adler and Luca de Alfaro. A content-driven reputation system for the wikipedia. In *WWW2007*. ACM, 2007.

[AGPR99] Swarup Acharya, Phillip B. Gibbons, Viswanath Poosala, and Sridhar Ramaswamy. Join synopses for approximate query answering. In *Proc. of SIGMOD Conference*, pages 275–286, 1999.

[BP07] Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. In *TABLEAUX '07: Proceedings of the 16th international conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 11–27, Berlin, Heidelberg, 2007. Springer-Verlag.

[CGL01] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In *Proceedings of the First Semantic Web Working Symposium*, pages 303–316, 2001.

[CWW$^+$06] Huajun Chen, Yimin Wang, Heng Wang, Yuxin Mao, Jinmin Tang, Cunyin Zhou, Ainin Yin, and Zhaohui Wu. Towards a semantic web of relational databases: A practical semantic toolkit and an in-use case from traditional chinese medicine. In *Proc. of the 5th International Semantic Web Conference*, pages 750–763, 2006.

[GHLS07] Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In *Proc. of IJCAI-07*. AAAI Press, 2007.

[GI91] Zvi Galil and Giuseppe F. Italiano. Data structures and algorithms for disjoint set union problems. *ACM Comput. Surv.*, 23(3):319–344, 1991.

[Gin92] Matthew L. Ginsberg. Multivalued Logics: A Uniform Approach to Inference in Artificial Intelligence. *Computational Intelligence*, 4(3), 1992.

[GKT07] Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance Semirings. In *PODS*, pages 31–40, 2007.

[GM08] Bernardo Cuenca Grau and Boris Motik. OWL 2 Web Ontology Language: Model-Theoretic Semantics. http://www.w3.org/TR/owl2-semantics/ [2008-05], 2008.

[GR03] Francois Goasdoue and Marie-Christine Rousset. Querying distributed data through distributed ontologies: A simple but scalable approach. *IEEE Intelligent Systems*, 18(5):60–65, 2003.

[HM05] Peter Haase and Boris Motik. A mapping system for the integration of OWL-DL ontologies. In *IHIS'05*, pages 9–16. ACM Press, NOV 2005.

[HPS08] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in owl. In *ISWC '08: Proceedings of the 7th International Conference on The Semantic Web*, pages 323–338, Berlin, Heidelberg, 2008. Springer-Verlag.

[HS04] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. *Artificial Intelligence*, 160(1–2):79–104, December 2004.

[HT00] I. Horrocks and S. Tessaris. A conjunctive query language for description logic Aboxes. In *Proc. of AAAI/IAAI-06*, pages 399–404. AAAI Press / The MIT Press, 2000.

[JQH08] Q. Ji, G. Qi, , and P. Haase. A relevance-based algorithm for finding justifications of DL entailments. Technical report, University of Karlsruhe, 2008.

[KPCGS06] A. Kalyanpur, B. Parsia, B. Cuenca-Grau, and E. Sirin. Axiom pinpointing: Finding (precise) justifications for arbitrary entailments in OWL-DL. Technical report, 2006.

[Len02] M. Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM Press, 2002.

[MH09] Yue Ma and Pascal Hitzler. Paraconsistent reasoning for owl 2. In Axel Polleres and Terrance Swift, editors, *RR*, volume 5837 of *Lecture Notes in Computer Science*, pages 197–211. Springer, 2009.

[MHL08] Yue Ma, Pascal Hitzler, and Zuoquan Lin. Paraconsistent reasoning for expressive and tractable description logics. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Description Logics*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[MSS05] Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with rules. *J. of Web Semantics*, 3(1):41–60, 2005.

[MST07] Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin. Repairing ontology mappings. In *Proc. of AAAI'07*, pages 1408–1413, 2007.

[QS09] Guilin Qi and Simon Schenk. D1.2.4 inconsistency-tolerant reasoning with networked ontologies. Technical Report D1.2.4, Universität Karlsruhe, 2009.

[SBH$^+$05] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle. The SWRC ontology - semantic web for research communities. In C. Bento, A. Cardoso, and G. Dias, editors, *Proceedings of the 12th Portuguese Conference on Artificial Intelligence*, volume 3803 of *LNCS*, pages 218 – 231, Covilha, Portugal, DEC 2005. Springer.

[SDS09] Simon Schenk, Renata Dividino, and Steffen Staab. Reasoning with Provenance, Trust and all that other Meta Knowlege in OWL2. In *SWPM2009*. CEUR, 2009.

[SR06] Julian Seidenberg and Alan Rector. Web ontology segmentation: Analysis, classification and use. In *Proc. of the 15th International World Wide Web Conference*, Edinburgh, June 2006.

[SS09] Renata Dividino Simon Schenk and Steffen Staab. Reasoning with provenance, trust and all that other meta knowlege in owl. In Paolo Missier Juliana Freire and Satya S. Sahoo, editors, *Semantic Web and Provenance Management (SWPM)âĂŹ09 Workshop (co-located with ISWC 2009)*, 2009.

[SST08] B. Schueler, S. Sizov, and D. T. Tran. Querying for Meta Knowledge . In *WWW2008*, pages 625–634. ACM, 2008.

[Str06] Umberto Straccia. A Fuzzy Description Logic for the Semantic Web. In *Fuzzy Logic and the Semantic Web*. Elsevier, 2006.

[THD$^+$09] Georgios Trimponias, Peter Haase, Chan Le Duc, Antoine Zimmermann, and Simon Schenk. D1.4.4 reasoning over distributed networked ontologies and data sources. Technical Report D1.4.4, Universität Karlsruhe, 2009.

[ZE06] Antoine Zimmermann and Jérôme Euzenat. Three semantics for distributed systems and their relations with alignment composition. In *Proc. of ISWC'06*, pages 16–29, 2006.