NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D1.4.4 Reasoning over Distributed Networked Ontologies and Data Sources

Deliverable Co-ordinator: **Georgios Trimponias, Peter Haase**

Deliverable Co-ordinating Institution: **Universität Karlsruhe (TH) (UKARL)**

Main Author: **Georgios Trimponias (UKARL)**

Other Authors: **Chan Le Duc, Antoine Zimmermann (INRIA), Simon Schenk (UKOB)**

Document Identifier:	NEON/2009/D1.4.4/v1.0	Date due:	February 28, 2009
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 28, 2009
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
- ONTO
- UKOB
- INRIA

Change Log

Version	Date	Amended by	Changes
0.1	24.11.2008	Georgios Trimponias	Initial creation of document
0.3	01.02.2008	Georgios Trimponias	Chapters 1, 2, 3, 4, 8
0.4	07.02.2008	Chan Le Duc	Chapter 6
0.5	23.02.2008	Simon Schenk	Chapter 7
0.6	24.02.2008	Georgios Trimponias	Chapter 5
0.9	26.03.2008	All authors	Modifications after review
1.0	27.03.2008	Georgios Trimponias	Final Version

Executive Summary

The concept of the semantic web is based on the ability to reason over explicitly declared or defined knowledge, so as to infer new, implicit knowledge in a sound way. The ontology, as a *shared specification of a conceptualization*, provides the basic tool for explicitly authoring this specified knowledge. Towards this direction, the World Wide Web Consortium has already endorsed the Web Ontology Language (OWL). OWL consists of a family of languages, diversified along their expressiveness and decidability. It is based on different fragments of Description Logics, a powerful fragment of First Order Predicate Calculus, which allows for knowledge description in a formal and well-understood way. On the other hand, the reasoning component has been primarily served by the theoretical results on description logics and as a result efficient reasoners have been proposed and implemented.

Theoretical research and practical implementations so far have usually focused on the centralized reasoning paradigm. Under this paradigm, one can assume the existence of one global ontology, defining a number of concepts and their relationships as well, and of a reasoner that executes a reasoning algorithm on the ontology and produces the inferred knowledge. Such a model, however, does not mirror the aspirations of the Semantic Web. Indeed, we argue that more has to be done, since large-scales are an indispensable part of the semantic web vision.

In the current Deliverable we aim at investigating the distributed aspects of knowledge representation formalisms and the reasoning procedures they are equipped with. Moreover, we propose three novel approaches that deal with the problems of 1) efficient reasoning in distributed knowledge bases, 2) reasoning with integrated distributed description logics, and 3) reasoning with (distributed) temporarily unavailable data sources, respectively.

In this direction, the Deliverable is organized as follows:

- Chapter 1 introduces the general motivation behind distributed knowledge representation formalisms and distributed inference processes and presents a number of interesting dimensions/features that are later used for a comparative analysis of various approaches.

- Chapter 2 attempts to demonstrate how the current Deliverable is highly related to the ongoing work in the main NeOn use case studies, namely the fisheries ontology of Work Package 7 and the invoice ontology of Work Package 8.

- Chapter 3 is a very central Chapter of the current work, as it investigates the state of the art in distributed knowledge representation formalisms and the corresponding inference procedures and makes a comparative analysis along the dimensions proposed in Chapter 1.

- Chapter 4 is devoted to the presentation of 1) the extended OWL metamodel for $\mathcal{E}$-Connections, which was initially introduced in the Deliverable 1.1.5, and 2) repeats the metamodel for mapping support in OWL, which has already been presented in Deliverable 1.1.2. The choice to repeat it in this deliverable is not accidental and reflects its high relevance to the subject of distributed knowledge representation formalisms.

- Chapter 5 presents a new approach for efficient reasoning with large data volumes (ABoxes). More concretely, it considers a number of ontologies connected with $\mathcal{E}$-Connections, then transforms them

into an equivalent $\mathcal{EL}^{++}$knowledge base, which is then transformed into an equivalent Datalog program. We then use this program for conjunctive query answering with PTIME Data Complexity.

- Chapter 6 proposes the integrated distributed description logic formalism (IDDL) and, moreover, describes a reasoning process in the context of this formalism. IDDL has already been presented in Deliverable 1.3.3 for formalizing modular ontologies. Since Deliverable 1.3.3 deal with formalization of modular ontologies, it did not involve reasoning procedure for IDDL.
- Chapter 7 introduces an approach based on multi-valued logics for dealing with temporarily unavailable data sources. This work has also been described in Deliverable 1.2.4 as they share the technical foundations. More precisely, Chapter 7 is a specialized application of the more general idea described in Deliverable 1.2.4.
- Chapter 8 contains the conclusion of the deliverable, along with some open challenges for future research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

The concept of the semantic web is based on the ability to reason over explicitly declared or defined knowledge, so as to infer new, implicit knowledge in a sound way. The ontology, as a *shared specification of a conceptualization*[1], provides the basic tool for explicitly authoring this specified knowledge. Towards this direction, the World Wide Web Consortium has already endorsed the Web Ontology Language (OWL). OWL consists of a family of languages, diversified along their expressiveness and decidability. It is based on different fragments of Description Logics, a powerful fragment of First Order Predicate Calculus, which allows for knowledge description in a formal and well-understood way. On the other hand, the reasoning component has been primarily served by the theoretical results on description logics and as a result efficient reasoners, such as FACT$++$[2], Pellet[3], or KAON2[4] have been proposed and implemented.

These two dimensions of explicitly declared specifications on the one hand and implicitly inferred knowledge on the other, along with the notion of the semantic web as defined and envisioned by Tim Berners Lee and other researchers, have been thoroughly studied and have found their way to practical implementation. Despite the impressive results in the field, we argue that more has to be done, since large-scales are an indispensable part of this vision.

Theoretical research and practical implementations so far have usually focused on the centralized reasoning paradigm. Under this paradigm, one can assume the existence of one global ontology, defining a number of concepts and their relationships as well, and of a reasoner that executes a reasoning algorithm on the ontology and produces the inferred knowledge. Such a model, however, does not mirror the aspirations of the Semantic Web.

In the emerging semantic web universe it would be naive to assume the existence of such global ontologies for every conceptualization. First, one would expect that different communities will provide different ontologies, or in other words different specifications of the same conceptualization. For example, the task and purpose of the ontology heavily influences the ontology design, while cultural, organizational or administrative aspects may well lead to ontologies with very different characteristics. (It is very interesting to notice that though the formal definition of an ontology assumes a shared specification, in practice different perspectives of the same concepts are to be expected). Furthermore, in practical applications where scalability, maintenance, reuse and understandability parameters are of high importance, it is common practice to consider a number of different ontologies, each of them covering a different subdomain of the application. For example,

[1] Some researchers tend to disagree with this definition, as in its heart lies the notion of *conceptualization*, for which no definition is available. So, a number of problems arise with respect to what one means by a conceptualization. Does it, for example, just encompass things of our real world, only well-understood objective terms or any mental construct? For their interest, these questions remain beyond the scope of this report, so we have decided to use the definition without further thoughts.

[2] `http://owl.man.ac.uk/factplusplus/`.

[3] `http://clarkparsia.com/pellet/`.

[4] `http://kaon2.semanticweb.org/`.

in the context of the NeOn project[5], the FOA[6] ontology consists of a number of smaller ontologies covering various aspects of interest, like fish species, landscapes or vessels.

Indeed, if we consider a target ontology describing a broader domain, then the description of this domain might be distributed over a number of ontologies, that describe special subdomains of the larger domain. In case they describe the same subdomain from a different perspective, then we can simply speak of ontologies with different points of view. On the other side of the spectrum, when the subdomains are completely disjoint, we are closer to the case of modularization, where each ontology can act as an (independent) module. It is expected that in practical applications more complicated scenarios could appear more often, e.g. ontologies with overlapping subdomains.

In general, an all-encompassing global ontology is very hard to achieve in practical scenarios. On the other hand, a centralized inference process exhibits a number of significant shortfalls, making it in practice highly inappropriate:

- When dealing with very large scales, e.g. the Internet scale, we could end up with a vast number of different ontologies, whose physical integration would be infeasible. What would make sense in such a setting is the establishment of suitable relationships (like for example mappings, connections, links) that show how the different schemata are related one to another and then use the available information during a (distributed) reasoning process. As an example of this approach, we mention the Open Linked Data initiative[7], which tries to extend the Web with a data commons by publishing various open datasets as RDF on the Web and by setting RDF links between data items from different data sources. This situation is actually reminiscent of how the Internet works in its present form: different resources, written by people with different perspectives and needs, targeted to communities with very different characteristics. There is every reason to believe that the semantic web should also have such a form. In fact, being able to cope with this issue seems to be at the very heart of the semantic web vision.

- For data whose size is typically much larger than the size of the schema, it is even more unrealistic to create one global database, for the same reasons that this is impossible for relational databases. Instead, decentralized infrastructures and networking are desired. Very similar arguments pushed considerably the database research into the direction of distributed databases. In general, the bigger the systems get, the harder the complexity issues.

- Even if we could overcome the obstacles of the integration process, it would be non-scalable and consequently unrealistic to suppose that a centralized reasoning process can cope with ontologies of very large sizes. There are of course techniques, like module extraction, whose goal is to extract only this part of the ontology that is relevant to query answering, disregarding the rest of the ontology, but they would be highly inefficient for very large ontology sizes, as the ones expected in the semantic web. Instead, it would greatly help to keep the schemata separate (along with their relationships) and to come up with a distributed reasoning procedure aiming at precipitating the reasoning task.

On the other hand, in comparison to the previous discussion, the benefits that the distribution of 1) the inference process and 2) the ontologies bring about are mainly focused around the notion of *complexity* and especially of *scalability*. As knowledge systems grow larger, a highly desirable property is actually their ability to efficiently handle very large data volumes. Indeed, a prominent goal and research direction of database systems, for instance, has always been scalability *for querying data*, as their size grows significantly. When dealing with semantics, this goal must be extended to include scalability *for reasoning on ontologies*, since apart from the explicitly authored knowledge, implicit knowledge can be additionally inferred (let alone how the reasoning task is already exacerbated by the very costly algorithms).

[5] http://www.neon-project.org.
[6] http://www.fao.org/aims/neon.jsp.
[7] http://en.wikipedia.org/wiki/Linked_Data.

Moreover, as is the case with all large knowledge and software systems, the system development is usually realized in a collaborative way, where a number of different teams contribute different parts of the system. Distribution can in this case play a catalytic role, as it facilitates development, evolution and maintenance of the ontologies, while in the special case of modularization it allows ontology reuse. By contrast, a large centralized ontology could make even simple tasks inflexible, time-taking and laborious.

The purpose of the current Chapter is to present existing approaches in the subject of distributed reasoning with ontologies, by attempting at the same time a comparative analysis of them along different dimensions, as explained later in more detail.

1.2 Distributed Ontologies versus Distributed Reasoning

At this stage, we should distinguish between the notions of distributed ontologies and distributed reasoning.

On the one hand, the notion of *distributed ontologies* refers to a collection of ontologies, *physically* or *logically* dispersed over multiple nodes. It differs from the classical case of just one single ontology, in the sense that in this case there are multiple ontologies residing on different nodes. A first example, well-studied in the literature, is when the set of the ontologies under consideration describe the same domain and each of them has its own point of view of the concepts that it defines. In that case, an additional set of relationships among these ontologies can reveal how they are related. A second example, lying on a completely different direction, refers to a network of ontology modules covering different subdomains/aspects of one global ontology and that should still be considered as a collection of distributed ontologies. The latter case is, for instance, the case when partitioning a large ontology into smaller equivalent modules, and it can be considered as a specific case of the more general notion of distributed ontologies as stated earlier. Nevertheless, throughout this report emphasis will almost exclusively be laid on the first kind of ontology distribution, as the second one is a subject of modularization and is beyond the scope of this work. From the previous subsection it immediately follows that a network of distributed ontologies under the former approach is in the very nature of the semantic web vision, where different communities provide various specifications of their conceptualizations, and these conceptualizations are dispersed over the web.

On the other hand, the notion of *distributed reasoning* refers to the case where the reasoning procedure is not centralized any more. For example, given a network of distributed ontologies, distributed reasoning decentralizes the reasoning algorithm, so that the overall process can be split upon the various nodes and thus be precipitated. So, the main focus in this case is to improve reasoning performance through distribution of the reasoning task. Distributed reasoning tries to face the scalability challenge, by distributing the reasoning task among different (sub)reasoners in a decentralized manner, thus making sure that the reasoning procedure can handle even very large collections of ontologies, what would otherwise be a practically insurmountable problem.

To be more concrete, even in this case these is often in literature an additional distinction: parallelized vs. distributed processes. Parallelized processes usually refer to centralized applications where the objective is to increase system performance. Parallelization is, under this view, mainly concerned with partitioning a problem for performance. This is especially the case if there is a single ontology for which there is central control of how to partition it. Moreover, parallelization can also deal with the problem of algorithmic parallelization of the inference process. For example, when building a tableau for checking TBox consistency, we can deal with the non-determinism introduced by disjunction by assigning the construction of the two branches of the tableau to different processors and by then combining their results. Distribution, on the other hand, usually refers to the case where the application is distributed across multiple physical locations. In this case, this control is not given, thus such parallelization cannot be achieved in the same manner. Instead, what we are interested in is decentralized reasoning procedures across the nodes of the application. So, while parallel systems pay more attention to the cost/performance arguments, distributed systems often address issues like distributed, heterogeneous components for which no global control is available. In the context of the current survey, we will mainly focus on distributed rather than on parallelized reasoning.

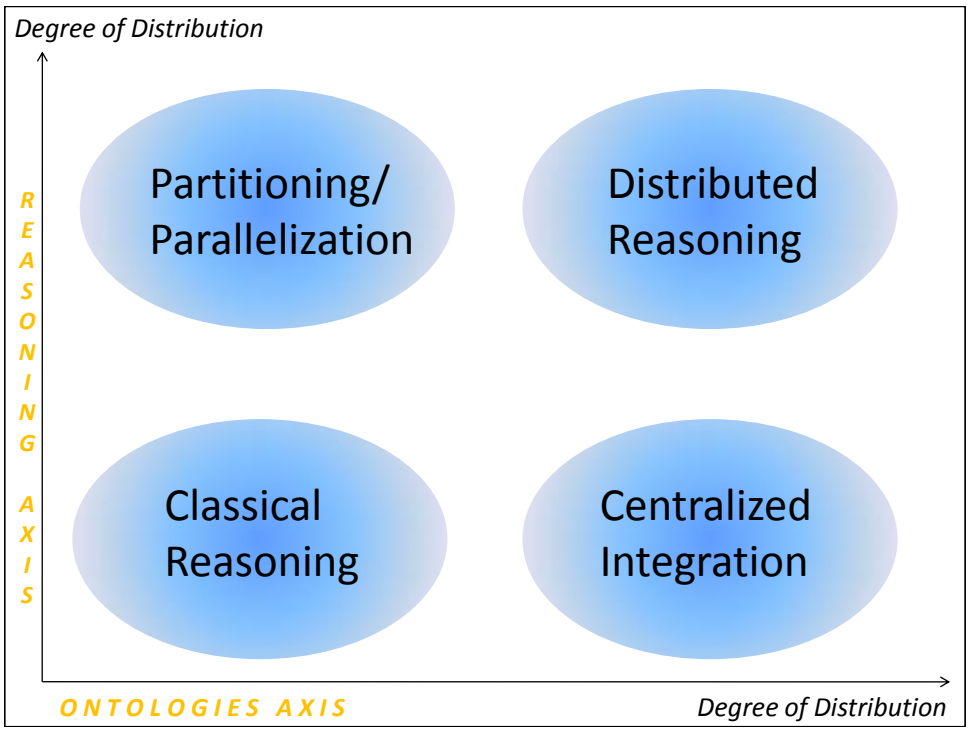

Figure 1.1: Distributed Ontologies versus Distributed Reasoning.

We could furthermore say that the two different notions of distributed ontologies and distributed reasoning provide us with two very interesting and useful dimensions, along which we can categorize the problem of (distributed) reasoning with (distributed) ontologies. More concretely, by combining these two dimensions we can derive four basic cases. These cases are graphically illustrated in Figure 1.1, using two axes, the horizontal *ontology* axis, and the vertical *reasoning* axis. The two axes are orthogonal one to another and they should rather be conceived as representing the distribution size in a continuous way.

The first case refers to the classic case, where we have one global ontology upon which we apply centralized reasoning. The second one includes all problems where we have a single ontology, where the reasoning procedure is distributed. For example, we could think of an ALC ontology, on which we apply distributed ordered resolution, as is suggested in [SS08b]. The second case is closer to what we called parallel reasoning in the previous section and has also a great connection to the problem of partitioning a single ontology for gaining performance. The third case refers to the case where we have a network of distributed ontologies, but we only apply one reasoning procedure/reasoner. For instance, one could attempt to logically integrate the networked ontologies through $IDDLs$ [Zim07] and then apply a centralized reasoning procedure. The last and most interesting one is the case where a network on distributed ontologies is available, upon which we reason using distributed reasoning procedures. The last case will be thoroughly studied in the next sections through a variety of existing methods. We also mention that in the bibliography this last paradigm is usually simply called distributed reasoning.

1.3 Dimensions of Interest

Ultimate goal of the current presentation is the comparative analysis of the different approaches, so that their different characteristics can be better understood and used in practise. For this purpose, we first need to come up with dimensions of interest, along which the comparative study will take place[8]. The following

[8]Perhaps the use of the word *dimension* is not very proper. In other scientific contexts dimensions tend to refer to independent features, whereas in the current context the dimensions are very interdependent one with another, as we will see later.

dimensions are the ones that we will mainly deal with:

- The *language* dimension refers to the language that the explicit specifications are expressed in. As mentioned earlier, ontologies are usually expressed using suitable fragments (depending on issues as expressivity and tractability) of the Description Logics formalism.
- The *relationships* are very relevant to the subject of reasoning with distributed ontologies. As already mentioned, distributed ontologies can contain different perspectives of conceptualizations, which are then related in one way or another in the networked environment. For example, we could have ontology modules distributed over the internet, which then import one another, as is the case with OWL. Or we could organize our ontologies in package hierarchies and then create links between these different packages. Networking relationships are of utmost importance in the context of this report and will be later thoroughly analyzed.
- The *reasoning task* describes the main task of the reasoning procedure, that is what we expect the reasoner to do. It can range from traditional reasoning tasks, such as concept subsumption and classification to more complex tasks, such as query answering.
- The *reasoning paradigm* depends on the underlying reasoning model of the reasoner. For example, there is the extensively studied paradigm of tableaux-based methods in description logics.
- The *algorithmic features*, as the name suggests, are the dimension that includes all these characteristics that we are usually interested in when studying an algorithm. These for example include the complexity of the reasoning procedure or its scalability capabilities. Depending on the case, we could also look into even more theoretical features, as for example decidability.

1.4 Overview of the Deliverable

The current report is briefly organized as follows:

- Chapter 2 provides use cases of reasoning over distributed networked ontologies.
- Chapter 3 provides the state of the art in reasoning with distributed data - distributed reasoning.
- Chapter 4 describes the metamodel for mapping support in OWL and the extended OWL metamodel for dealing with $\mathcal{E}$-Connections.
- Chapter 5 looks into distributed reasoning with $\mathcal{E}$-Connections in the context of the FOA ontology.
- Chapter 6 examines the approach proposed at the [Zim07] regarding reasoning with integrated Description Logics.
- Chapter 7 deals with reasoning with temporarily unavailable data sources.
- Chapter 8 summarizes the results and contains open problems for future research.

In order to structure the report in a uniform and comprehensible manner, every section is organized in a very similar way. A *motivation* part discusses the problems that are addressed by the approach under consideration, then a *formalism* part presents (briefly) the syntax and semantics and finally a *reasoning* part describes procedures that are used for reasoning under this formalism.

The current Chapter will, of course, not even attempt to give an overview of all techniques that have been proposed for distributed reasoning in description logics, as that would be well beyond its scope. Instead, we have chosen to concentrate on a number of approaches that are well studied in the literature and/or come with a system implementation. The main motivation behind this has to do with the objective of the

survey, which is not so much about presenting all formalisms, but rather about attempting to show how the different formalisms approach the problem of distributed reasoning and in what ways the are the same or they differ. In other words, ultimate goal is the demonstration of the challenges that can arise when building distributed systems and how each of these approaches tackles some of these challenges. Moreover, for the approaches that are discussed in the context of the current report we have chosen not to go into details about their formalisms, reasoning processes, applications etc., but rather to focus on a more restricted part that is enough to demonstrate the main ideas behind the method. This is in accordance with what we consider the general objectives of any survey, namely summarization and comparison.

Chapter 2

Use Cases of Reasoning over Distributed Networked Ontologies

2.1 Introduction

As already stated in Chapter 1, targeting in practice an all-encompassing ontology is a very unrealistic goal. Regardless of the scenario under consideration, a centralized inference process on an all-encompassing global ontology exhibits a number of significant shortfalls, making it in practice highly unrealistic. Let us briefly summarize them:

- When dealing with very large scales, e.g. the Internet scale, we could end up with a vast number of different ontologies, whose physical integration would be infeasible. What would make sense in such a setting is the establishment of suitable relationships (like for example mappings, connections, links) that show how the different schemata are related one to another and then use the available information during a (distributed) reasoning process.

- For data, whose size is typically much larger than the size of the schema size, it is even more unrealistic to create one global database, for the same reasons that this is impossible for relational databases. Instead, decentralized infrastructures and networking are desired.

- Merging different schemata is an immensely difficult process, with many obstacles that have to be surpassed, and for which automatic algorithms are very hard to design.

- Even if we could overcome the obstacles of the integration process, it could be quite non-scalable to assume that a centralized reasoning process can cope with ontologies of very large sizes.

On the other hand, in comparison to the previous discussion, the benefits that the distribution of 1) the inference process and 2) the ontologies bring about are mainly focused around the notion of *complexity* and especially of *scalability*. As knowledge systems grow larger, a highly desirable property is actually their ability to efficiently handle very large data volumes. Indeed, a prominent goal and research direction of database systems, for instance, has always been scalability *for querying data*, as their size grows significantly. When dealing with semantics, this goal must be extended to include scalability *for reasoning on ontologies*, since apart from the explicitly authored knowledge, implicit knowledge can be additionally inferred (let alone how the reasoning task is already exacerbated by the very costly algorithms).

Moreover, as is the case with all large knowledge and software systems, the system development is usually realized in a collaborative way, where a number of different teams contribute different parts of the system. Distribution can in this case play a catalytic role, as it facilitates development, evolution and maintenance of the ontologies, while in the special case of modularization it allows ontology reuse. By contrast, a large centralized ontology could make even simple tasks inflexible, time-taking and laborious.

To make the previous discussion/motivation a bit more specific and to better illustrate the reasons that make (distributed) reasoning with distributed ontologies so necessary, in this chapter we will attempt to present use case scenarios in the context of the NeOn Project, where 1) distribution in ontologies/data sources, and/or 2) decentralized inference processes can indeed lead to a number of specific benefits.

2.2 The FOA Case Study

One of the two major case studies in the NeOn Project (with the other one being the WP8 pharmaceutical ontology case study) is the WP7 case study, which deals with the creation of an ontology-driven Fisheries Stock Depletion Assessment System (FSDAS). This system will use FAO and non-FOA resources on Fisheries to assist in the assessment of fish stock and will be empowered by means of a network of ontologies. The user requirements for this case study are included in the Deliverable 7.1.2 "Revised Specifications of User Requirements for the Fisheries Case Study". In the following discussion we will restate some of the objectives/requirements of the WP7 case study as described in this deliverable and then we will attempt to show how distributed (reasoning) with distributed data sources is indeed related to them.

2.2.1 Objectives and User Requirements

The main objectives of this case study can be summarized as follows:

1. Creation and maintenance of networked ontologies in the fisheries knowledge community.
2. Exploitation of ontologies within web applications and development of a Fish Stock Depletion Assessment System.
3. The success of a number of case study indicators.

Objective 1 means in practice that users should be able to implement the fisheries ontologies network and map them to exploit and use the Fisheries electronic resources, taking care of the continuous growth of the information and knowledge made available made in the domain. For the realization of Objective 2, it is beyond others expected that mechanisms are provided for static and especially run-time modularization of networked ontologies. This is very relevant considering the large size of the case study ontologies and the high level performance expected from applications using the ontologies.

On the other hand, a major requirement for ontology engineers for the WP7 case study has to do with *ontology reuse, reengineering* and *integration*. New ontologies may be created on the basis of existing ones, either by transforming the conceptual model of an existing and implemented ontology into a new one (reengineering) or by including the existing ontology into the new one (integration).

Perhaps the most important goal for ontology engineers is, however, *modularization*. Especially in case of large ontologies, it is important that facilities be provided to support users in defining and selecting "fragments" of ontologies, which we call modules. Common reasons for these facilities are: to improve efficiency (by selecting only the part of the ontology most frequently used), sharing editorial duties, visualization purposes and user rights. In this direction, mechanisms should be in place to allow ontology engineers to create (at least) modules *by language, by code, by topic* and for *editorial duties*.

Indeed, in Deliverable 7.2.2 "Revised/Enhanced Fisheries Ontologies" it is stated that one of the lessons learned was that *if efficiency is an issue, modularization is required.* That means that the user should be in the position to select and load only the portion of ontologies that need to accessed, visualized etc. In this direction, the generated ontologies were produced by the following hierarchies of reference data:

- Land Areas
- Fishing Areas

- Biological Entities
- Fisheries Commodities
- Vessel Types and Size
- Gear Types

This particular choice is inextricably related to the idea of modularization, as it results in the creation of a number of ontologies describing disjoint domains. Indeed, the whole idea of modularization is based on the ability to define a number of local modules, which can then be composed under some special operators to generate the target ontology. In case the domain are disjoints, we can gain some concrete benefits that will be described in the next subsection. Needless to say, as also stated at the beginning of the chapter, this is not the only possibility. For instance, in case the sub-ontologies offer different perspectives of the same concepts, then they refer to the same domain and in that case the mappings need to show how concepts and instances described in one local ontology are related to other concepts and instances described in a different local ontology. In Chapter 3, this distinction is better illustrated through an overview of the state of the art, where different kinds of relationships between the local ontologies are defined, depending by and large on how the various subdomains are related one to another.

2.2.2 Reasoning over Distributed Networked Ontologies in the Context of WP7 Use Case

In the previous subsection we showed how one of the most basic requirements of the fisheries ontology use case has to do with *modularization* and how it can increase *efficiency*, while at the same time we briefly summarized the local reference ontologies out of which the revised/enhanced fisheries ontology comprises. We will next argue that a distributed reasoning process over distributed data sources can indeed achieve these goals. Chapter 5 will then discuss an approach that is well-fitted in such a setting, providing all the necessary technical details.

An environment of distributed ontologies (schemata, TBoxes) and data sources (instances, TBoxes) can indeed be proved very helpful in the case of modularization, especially in the fisheries ontology case study, where the various local ontologies cover disjoint domains. Each of these modules can reside on a different node (physically or logically) and then a set of relationships between the nodes can show how these local ontologies are connected/related to each other. In contrast to a centralized setting, where all ontologies would reside on the same node, extracting thus a very heavy toll on system scalability and reducing performance, in the case of decentralized architectures, the generated systems are far more scalable, flexible and the complexity can be better handled.

More specifically, modularization can lead to the following benefits:

- Scalability for querying data and reasoning on ontologies, as stated in the Introduction.
- Scalability for development, evolution and maintenance of the ontology modules.
- As far as the design phase is concerned, the previous point means better complexity management, while the usage phase is drastically facilitated, as understandability is significantly increased.
- Re-use. This case is evidently the same as with modular software and it has been one of the strongest motivations behind the development of modular systems (with the other one being scalability).

Moreover, the domain-disjointness is, in fact, a very convenient feature of our system, when it comes to physical (networking) distribution. Indeed, the fact that the different local ontologies cover non-overlapping aspects practically means that placing each module on a different node won't produce any sort of conflicts, which usually arise in settings of overlapping domains. For instance, having two different ontologies describing the same concepts can potentially lead to inconveniences if we place the modules on different nodes,

since answering a query would require the different nodes to cooperate in some way to produce the desired answer and that could possibly be facilitated by placing the ontologies together. Of course, one can argue that this is not entirely correct even in our case. For example, if two modules are expected to appear together in a very high percentage of the posed queries, then it makes sense if they both appear on the same node, instead of being separate. Nevertheless, in principle disjoint domains can avoid many of these problems, when it comes to networking.

The second dimension of distribution, orthogonal to the ontologies axis, is the reasoning axis, as was graphically depicted in the Introduction, Figure 1.1. In the WP7 case study, distributed reasoning can drastically contribute to achieve the desired goals, and especially efficiency/performance. A centralized inference process would require the integration of the different components needed for query answering, canceling thus partially the benefits of a distributed architecture. In order to fully exploit the benefits acquired from a decentralized architecture, we must also be in the position to apply some distributed reasoning process, that will take into consideration the ontology/data distribution and will avoid -to the biggest possible degree- to integrate the content residing on distinct nodes.

So far, we only described in practical terms how distributed reasoning with distributed ontologies and data can be very helpful in the use case scenario of WP7. In Chapter 5, a suitable approach will be discussed along with all formalisms and technical details.

2.3 The Pharma Case Study

As mentioned in Deliverable 8.3.1 in the context of the NeOn Pharmaceutical case studies, the use of electronic invoices for commercial transactions has grown exponentially. This results in large heterogeneity of the represented invoice information, which is further aggravated by the lack of invoice standards among the main players of the sector. To overcome this situation, finding mappings between networked ontologies and reasoning over distributed networked ontologies provide a real solution allowing one i) to automate invoice exchange between business peers, and ii) to ensure consistency of exchanged invoice data.

More concretely, consider two Pharmaceutical Reference ontologies *Digitalis* and *BOTPlus* which gather the knowledge represented in the schema of respective databases. Assume that these ontologies are located in different nodes. A system supporting networked ontologies with the features described above should provide the following functions:

- Adapting the ontologies *Digitalis* and *BOTPlus* in accordance with the networked ontology framework;
- Detecting automatically or semi-automatically mappings between ontologies;
- Checking consistency of ontologies with mappings;
- Identifying correspondences which are responsible for an inconsistency if any.

In such a scenario, it would be desired that checking global consistency is distributed, i.e. each node is associated with a local reasoner that is able to answer a query about the consistency of the ontologies located in this node with respect to the mappings.

Such a solution can be provided by the Integrated Distributed Description Logics (IDDLs) formalism, which will be presented in more detail in Chapter 6.

Chapter 3

State of the Art in Reasoning with Distributed Data - Distributed Reasoning

3.1 Introduction

Distribution both in reasoning and ontologies is a necessary pillar for the realization of the semantic web, as explained in the motivation section 1.1 of Chapter 1. Distributed Reasoning, in particular, is a relatively new area of research in the field of Description Logics and has not reached yet a point of maturity. Nevertheless, a number of approaches stand out as important results and in this chapter we will attempt an overview of the state of the art, providing at the same time a brief comparison along the dimensions that were introduced in the section 1.3 of Chapter 1.
The current chapter is briefly organized as follows:

- Section 2. presents the Distributed Description Logics (DDLs). DDLs suppose the existence of a number of local ontologies, each of them adopting its unique perspective, and then define bridge rules that show how the various ontologies are related one to another. A distributed extension of the classic tableau algorithm is available for these logics.
- Section 3. deals with the $\mathcal{E}$-Connection approach. $\mathcal{E}$-Connections also consider a number of local ontologies, but instead of just defining correspondences between them, it allows for link constructors, that act as super-roles, between the different ontologies. A combined tableau can be used for reasoning.
- Section 4. describes the Distributed First Order Logics. This formalism provides a common framework for both DDLs and $\mathcal{E}$-Connections and though no general distributed process is available for this formalism, we consider that it fits greatly to this report.
- Section 5. looks into Package-based Description Logics (P-DLs). This formalism deals with some problems of the DDLs and $\mathcal{E}$-Connections and provides a collaborative environment for the construction, sharing and usage of ontologies, through an importing mechanism. A distributed message-based inference procedure can be used for reasoning.
- Section 6. examines the approach proposed at the SomeWhere system. This approach considers a peer-to-peer setting, where every local ontology is simple, in the sense that it can be described in the full propositional fragment of the Description Logics. It then presents a decentralized consequence finding algorithm for query answering.
- Section 7. firstly describes the extension of OWL with DL-safe mappings and then investigates how it can be applied to a peer-to-peer setting. Though the reasoning process integrates the relevant data and is thus not distributed, it is relevant in the sense that it integrates the part of the distributed ontologies that is relevant to query answering.

- Section 8. summarizes the results.

In order to structure the report in a uniform and comprehensible manner, every section is organized in a very similar way. A *motivation* part discusses the problems that are addressed by the approach under consideration, then a *formalism* part presents (briefly) the syntax and semantics and finally a *reasoning* part describes procedures that are used for reasoning under this formalism.

3.1.1 Related Work and Scope of the Chapter

Borgida and Serafini [BS03] propose an extension of classic Description Logics with additional directional binary relations describing the correspondences between the ontology domains, in what they name Distributed Description Logics (DDLs). Serafini and Tamilin propose and implement at [ST05] a distributed tableau algorithm for this formalism for distributed TBoxes under the restriction that no cyclical bridge rule dependencies exist, while [SBT05] takes care of local inconsistencies by introducing a special interpretation, called a hole. Ghidini et al. describe at [GST07] an extension of the bridge rules between roles and define a formalism allowing for heterogeneous mappings. Extending the syntax and semantics of OWL to allow for the representation of contextual ontologies has resulted in the development of the language C-OWL [BGvH$^+$03]. An implementation of modular ontologies with the DDLs formalism has been suggested by Stuckenschmidt at [Stu06].

Another direction in a similar spirit has been followed using $\mathcal{E}$-Connections. The general formalism has been introduced at [KLWZ04] and provides fundamental theoretical results on whether decidability is preserved when connecting decidable abstract description systems. Based on this theory, Grau et al. propose an extension of OWL that integrates a simplified form of the $\mathcal{E}$-Connections formalism and they provide a combined tableau for the SHIF(D) ontologies without ABoxes [GPS04b]. Practical tableau algorithms for very expressive description logics, namely, SHIQ, SHIO, SHOQ, are extensively discussed at [GPS04a].

Integrated Distributed Description Logics (IDDLs) [Zim07] is another distributed approach, where the use of an equalizing function maps local domains to a global domain and gives the semantics an integrating character. Reasoning in a network of ontologies connected through IDDLs based on the notion of local configurations is presented at [ZD08].

Contrary to the linking/mapping approach followed by the approaches mentioned above, Bao et al. suggest an importing approach called Package-Based Description Logics (PDLs) [BCH06b] which allows an ontology to make direct references to terms defined in other ontologies and relaxes the domain disjointness condition which exists explicitly or implicitly in DDLs and $\mathcal{E}$-Connections. A distributed tableau algorithm for ALC that allows importing of concepts between packages using an asynchronous message passing protocol is described at [BCH06d].

Apart from the above approaches that constitute the main pillar of distributed reasoning in DLs, other formalisms have been proposed in the bibliography. The work of Haase and Wang at [HW07] describes a decentralized peer-to-peer infrastructure for query answering over distributed ontologies, called KAONp2p, which is based on the notion of DL-safe mappings [HM05]. This approach does not present a really distributed paradigm, but takes into consideration only the relevant information in order to integrate only this part of the ontologies along with the mappings that are relevant to query answering. For a peer-to-peer setting of simple ontologies expressed in the propositional fragment of description logics, a decentralized message-based consequence finding algorithm has been suggested for query answering by Adjiman et al. [ACG$^+$06]. This algorithm works by using propositional encodings of both the local ontologies and the query and by producing its rewritings. Finally, Schlicht et al. [SS08b] addresses the problem of scalable reasoning in ALC by proposing a novel parallel algorithm that decides satisfiability of terminologies and which is based on applying resolution to DLs.

It goes without saying, the current chapter will not even attempt to give an overview of all techniques that have been proposed for distributed reasoning in description logics, as that would be well beyond its scope. Instead, we have chosen to concentrate on a number of approaches that are well studied in the literature and/or come with a system implementation, leaving aside other potentially very promising methods. The main

motivation behind this is not so much about presenting all formalisms, but rather about attempting to show how the different formalisms approach the problem of distributed reasoning and in what ways the are the same or they differ. In other words, ultimate goal is the demonstration of the challenges that can arise when building distributed systems and how each of these approaches tackles some of these challenges. Moreover, for the approaches that are discussed in the context of the current chapter we have chosen not to go into details about their formalisms, reasoning processes, applications etc., but rather to focus on a more restricted part that is enough to demonstrate the main ideas behind the method. This is in accordance with what we consider the general objectives of any state of the art survey, namely summarization and comparison.

3.2 Distributed Description Logics

3.2.1 Motivation

The semantic web vision involves, as already mentioned, a number of distributed ontologies, each providing its unique view of the world. In order to be able to exploit the distributed knowledge, one must first establish some kind of relationships between the ontologies in the distinct nodes, so that the reasoning algorithm can then be able to combine different pieces of information residing on different nodes. For example, one information source alone might not contain all the relevant information, but the established semantic relationships could help us use knowledge from other sources, in order to get the complete answer.

The relationships/mappings between the different distributed sources are very interesting to investigate. In the general case, one cannot make many assumptions about their form. For example, when it comes to relationships between individuals, these relationships do not need to have an one-to-one character, as it is often the case that an individual belonging to the domain of one ontology can be mapped to a number of individuals belonging to the domain of a remote ontology. For a similar reason, injections or surjections cannot be adequate to capture the possible correspondences between individuals in more general cases. Also, one usually has to express further constraints about these relationships, as for example that an instance of domain A can be corresponded to exactly two instances from domain B. Another very important feature is that the established correspondences between the two distributed ontologies, should not necessarily be the same in the two directions. Instead, one needs in practice a pair of correspondences, one in each direction of the relation.

Getting inspiration from the characteristics of relationships between distributed ontologies, Borgida and Serafini [BS03] propose an extension of classic Description Logics with additional directional binary relations describing the correspondences between the ontology domains, in what they name Distributed Description Logics ($DDLs$). The novelty of their approach resides in the fact that using bridge rules they can implicitly constrain the relationships between the different domains. Additionally, the directional bridge rules respect the potentially different point of views that each ontology can exhibit.

3.2.2 Formalism

Consider a collection of description logics DL_i over a non-empty set of indices I. By T_i we denote the TBox of the i ontology expressed in DL_i, while the distributed TBox T is defined as $\bigcup_{i \in I} T_i$, that is the union of the local TBoxes. Every description of the i-th ontology is labeled with the prefix i : to make it clear which ontology it belongs to. For example, the concept E described in the ontology i will be referred to as $i : E$, whereas the axiom $D \sqsubseteq E$ in ontology j will be written as $j : D \sqsubseteq E$.

The definition of bridge rules is then as follows:

Definition 1 *A bridge rule B_{ij} from i to j is any expression of the following two forms:*

- $i : A \xrightarrow{\sqsupseteq} j : B$ *(onto - bridge rule)*
- $i : C \xrightarrow{\sqsubseteq} j : D$ *(into - bridge rule)*

, where A, C and B, D are concepts of description logics DL_i and DL_j respectively.

What is very interesting is that bridge rules do not try to capture the meaning of the ontology mappings from an *objective* point of view. That would be the case for example if we tried to map all local ontologies into a global ontology. Instead, the bridge rule represents a directional relation which is considered from the *subjective* point of view of each ontology and thus the bridge rule B_{ji} is not necessarily the inverse of the bridge rule B_{ij}, in fact it does not even have to exist. This central idea reflects according to [BS03] the structure of the semantic web, where it is not realistic to consider global point of views for all participating ontologies, but rather subjective perceptions of the semantic relations.

So, the onto - bridge rule $i : A \xrightarrow{\sqsupseteq} j : B$ states that from the point of view of the $j-th$ ontology the concept A in ontology i is less general than the concept B in ontology j. Similarly, the bridge rule $i : C \xrightarrow{\sqsupseteq} j : C$ states that from the point of view of the $j - th$ ontology the concept C in ontology i is more general than the concept D in ontology j. The "more general" and "less general" concepts imply some kind of relation between the two concepts that maps instances of the domain of ontology i to instances of the domain of ontology j, as seen from $j's$ perspective. This purpose is served by the domain relations r_{ij}:

Definition 2 *A domain relation r_{ij} from the domain Δ_i of ontology i to the domain Δ_j of ontology j is a subset of the cartesian product Δ_i x Δ_j.*

Obviously, its purpose is to map instances of the domain Δ_i to instances of the domain Δ_j, as seen from the perspective of j. That means that when defining the interpretation of a distributed TBox the classical interpretation structure $< \Delta^I, \cdot^I >$ must be extended, so that it also takes into account the domain relations between the distributed ontologies. More concretely:

Definition 3 *An interpretation J of a distributed TBox consists of local interpretations for the local TBoxes T_i on the local domains Δ_i and a collection of domain relations r_{ij} between the different local domains. J satisfies the elements of a distributed T-Box $T = < \{T_i\}_{i \in I}, B >$ according to the following clauses, where i,j $\in$ I:*

- $J \models i : A \xrightarrow{\sqsubseteq} j : B$, *if* $r_{ij}(A^{I_i}) \subseteq B^{I_j}$ *(Satisfaction of onto - bridge rule)*
- $J \models i : A \xrightarrow{\sqsupseteq} j : B$, *if* $r_{ij}(A^{I_i}) \supseteq B^{I_j}$ *(Satisfaction of into - bridge rule)*
- $J \models i : A \sqsubseteq B$, *if* $I_i \models A \sqsubseteq B$ *(Satisfaction of local subsumptions)*
- $J \models T_i$, *if* $I_i \models T_i$ *(Satisfaction of local T-Boxes)*
- $J \models T$, *if for every i* $\in$ *I,* $J \models T_i$, *and J satisfies every bridge rule in the union of the sets in B.*

Furthermore, $T \models i : C \sqsubseteq D$ if, for every distributed interpretation J, $J \models T \Longrightarrow J \models i : C \sqsubseteq D$.

At this point we mention that the $DDLs$ formalism is additionally equipped with mappings between individuals belonging to the different ontologies. These correspondences can have the form of a mapping from one instance of ontology A to either exactly one instance of ontology B or multiple instances of ontology B. The reason we chose not to go into details with regard to these mappings is that distributed reasoning under this formalism has been developed only for knowledge bases with only TBoxes, as we shall discuss later.

The definitions of the $DDLs$ have some very interesting implications. Among them, we mainly focus on the simple and generalized subsumption propagation properties, since they provide the theoretical foundations for the distributed reasoning process. More concretely:

- The simple subsumption propagation allows for the propagation of subsumption axioms across ontologies using a combination of onto- and into-bridge rules. For example, if B_{ij} contains $i : A \xrightarrow{\sqsupseteq} j : G$ and $i : B \xrightarrow{\sqsubseteq} j : H$, then: $J \models i : A \sqsubseteq B \Longrightarrow T \models j : G \sqsubseteq H$.

- The generalized subsumption propagation is, as its name suggests, a generalization of the simple propagation rule and takes into account the case where multiple subsumption rules are available. So, if B_{ij} contains $i : A \xrightarrow{\sqsupseteq} j : G$ and $i : B_k \xrightarrow{\sqsubseteq} j : H_k$ for $1 \leq k \leq n$ and $n \geq 0$, then: $J \models i : A \sqsubseteq \bigsqcup_{k=1}^{n} B_k \Longrightarrow T \models j : G \sqsubseteq \bigsqcup_{k=1}^{n} H_k$.

In the next section we will see how this two properties constitute the main pattern in the distributed reasoning procedure.

3.2.3 Reasoning

Traditionally, the basic reasoning mechanism provided by classic DL reasoners checked the subsumption of concepts, since it can be shown that all other TBox inferences can be reduced to subsumption verification. The same can be said for distributed TBoxes, with the difference that besides the local reasoning tasks, which can be implemented by a reasoner residing on a local ontology, one must also consider the various interactions between the distributed ontologies because of the semantic mappings between them. So, apart from the local reasoning task, a distributed reasoner has to take advantage of the semantic relations between the different ontologies, so as to infer "hidden" subsumptions.

Using as basic reasoning pattern the property of generalized subsumption propagation, Serafini and Tamilin propose at [ST05] a distributed tableau algorithm for determining whether $J \models i : A \sqsubseteq B$. Their approach is based on the contextual reasoning paradigm. Contrary to other approaches that try to reduce the distributed reasoning problem into a reasoning in a global ontology that encodes both the local TBoxes and the mappings, this approach has the advantage of checking concept satisfiability by combining local tableau procedures that are executed inside the local ontologies.

The following definition provides us with the same tool for the distributed reasoning in $DDLs$:

Definition 4 *Given the ontologies i and j and the set of bridge rules B_{ij}, the bridge rule operator $B_{ij}(\cdot)$ takes as input the T-Box T_i of ontology i and produces a TBox of ontology j as follows:*

$$B_{ij}(T_i) = \left\{ G \sqsubseteq \bigsqcup_{k=1}^{n} H_k \;\middle|\; \begin{array}{l} T_1 \models A \sqsubseteq \bigsqcup_{k=1}^{n} B_k, \\ i : A \xrightarrow{\sqsupseteq} j : G \in B_{ij}, \\ i : B_k \xrightarrow{\sqsubseteq} j : H_k \in B_{ij}, \\ \textit{for } 1 \leq k \leq n \textit{ and } n \geq 0 \end{array} \right\}$$

n=0 denotes the bottom concept $\bot$. The intuition behind the introduction of the bridge operator is the isolation the non-local part of the distributed inference process. Indeed, the theorem that follows next shows that for a certain family of distributed TBoxes all non-local inferences can be provided by the computation of the bridge operator:

Theorem 1 *Suppose T_{ij} is the distributed T-Box $< T_i, T_j, B_{ij} >$, then:*
$T_{ij} \models j : X \sqsubseteq Y \Longleftrightarrow T_j \cup B_{ij}(T_i) \models X \sqsubseteq Y$.

What the previous theorem actually states is that the decision whether the subsumption $X \sqsubseteq Y$ holds in ontology j can be *correctly* and *completely* reduced to checking whether the subsumption axiom holds int the local TBox T_j augmented by the T-Box of the bridge operator. So, what is needed is an effective procedure to compute the outcome of the bridge operator.

Indeed, the reasoning procedure proposed at [ST05] answers this problem for a particular category of distributed TBoxes, called *acyclic* TBoxes. Namely, a distributed TBox is acyclic if the set of indexes is a partial order $(I, <)$, such that $i < j$, if and only if $B_{ij} \neq \emptyset$. The algorithm restricts to this case and not to the general one, as in the former case the interactions between the ontology domains can be represented as a chain, along which subsumption axioms are propagated. That is not the case for general-form distributed TBoxes, where the semantic links can be much more complicated.

The main task of the algorithm is to check whether $T \models i : A \sqsubseteq B$, by reducing the subsumption problem to the satisfiability problem of the complex concept $A \sqcap \not B$. To that purpose every local ontology belonging

to the network of the distributed ontologies has a local distributed tableau procedure that tries to build a representation of the model for the concept and consists of two procedures:

- An exclusively local procedure that is in charge of the local reasoning process and makes use of the standard Description Logics expansion rules.
- A distributed tableau procedure that uses the local reasoning procedure plus a set of expansion bridge rules, so as to use additional information from the set of bridge rules and infer new subsumption relations holding in the local ontology.

The proposed algorithm certainly has a number of limitations, as for example the existence of acyclic distributed TBoxes with no individuals, but on the other hand is quite simple to implement. Indeed, an implementation of the previous procedures dealing with the problem of distributed reasoning has been proposed in what is known as Distributed Reasoning Architecture for a Galaxy of Ontologies (DRAGO). This system combines different reasoning procedures for mapped ontologies and directly uses ontologies and links between them published on the web, without requiring a tight interaction with central repositories or between the local ontologies and a global one. Despite its simplicity, its design philosophy constitutes a true paradigm of a distributed tableau procedure.

3.3 $\mathcal{E}$-Connections

3.3.1 Motivation

Despite their ability to combine different loosely coupled federated data sources, $DDLs$ are rather restricted, since they allow only one type of domain relations. Indeed, bridge rules can be thought of as binary relations between the two domains; under that approach relations of arbitrary arity n are not possible. Moreover, instead of one domain relation, as in the $DDLs$, we can now have as many links between the data sources as we need. These links can be conveniently be thought of as "super-roles", that allow to define relationships between disjoint domains, as opposed to the classic case in Description Logics, where roles are binary relations over the one and unique domain. In this section, we define $\mathcal{E}$-Connections in the context of abstract description systems and we present a number of results with regard to them, based on the work at [KLWZ04].

The main idea is that given n systems defined in terms of abstract description systems (ADSs) with disjoint interpretation domains, we want to establish n-ary relations between them, in such a way that if the n components are decidable, then the resulting connected system also retains the decidability.

We begin by introducing ADSs and $\mathcal{E}$-Connections among them.

3.3.2 Formalism

Abstract Description Systems

Abstract description systems came as a result of the effort to combine different logical formalisms into a single logical formalism. Examples of the different formalisms include description logics, logics of topological spaces (e.g. the modal logic S4 extended with the universal modality), logics of metric spaces that can offer not only qualitative but also quantitative information (e.g. the family MS), and propositional temporal logics. That actually suggests that the abstract description system formalism is a very expressive formalism, able to handle quite different kinds of logics, and not restricted just to the DLs field.

The syntax is provided by the abstract description language, which determines the set of terms and assertions. Here we only give the trimmed-down version of ADSs that doesn't take into consideration A-Boxes. We begin by defining abstract description languages, which provide the building stones for writing terms, and abstract description models, that provide the terms with semantics.

Definition 5 *An abstract description language (ADL) $\mathcal{L}$ is described by a countably infinite set $\mathcal{V}$ of set variables and a countable set $\mathcal{F}$ of function symbols f of any arity m_f, such that $\neg, \wedge \notin \mathcal{F}$. The terms of t_j of $\mathcal{L}$ are built in the following way:*
$t_j ::= x \mid \neg t_1 \mid t_1 \wedge t_2 \mid f(t_1, ..., t_{m_f})$, *where $x \in \mathcal{V}$ and $f \in \mathcal{F}$. The term assertions of $\mathcal{L}$ are of the form $t_1 \sqsubseteq t_2$.*
An abstract description model (ADM) for an ADL $\mathcal{L} = (\mathcal{V}, \mathcal{F})$ is a structure of the form $\mathfrak{M} = \langle W, \mathcal{V}^{\mathfrak{M}} = (x^{\mathfrak{M}})_{x \in \mathcal{V}}, \mathcal{F}^{\mathfrak{M}} = (f^M)_{f \in \mathcal{F}} \rangle$,where W is a non-empty set, $x^{\mathfrak{M}} \subseteq W$ and each $f^{\mathfrak{M}}$ is a function mapping m_f-tuples $\langle X_1, ..., X_{n_f} \rangle$ of subsets of W to a subset of W. The value $t^M \subseteq W$ of an $\mathcal{L}$-term t in $\mathfrak{M}$ is defined inductively by taking: $(\neg t)^{\mathfrak{M}} = W - t^{\mathfrak{M}}$, $(t_1 \wedge t_2)^{\mathfrak{M}} = t_1^{\mathfrak{M}} \cap t_2^{\mathfrak{M}}$, and $(f(t_1, ..., t_{m_f}))^{\mathfrak{M}} = f^{\mathfrak{M}}(t_1^{\mathfrak{M}}, ..., t_{m_f}^{\mathfrak{M}})$.

Also, $\mathfrak{M} \models t_1 \sqsubseteq t_2$ iff $t_1^{\mathfrak{M}} \subseteq t_2^{\mathfrak{M}}$ (truth-relation for a term-assertion) and in this case we say that the assertion $t_1 \sqsubseteq t_2$ is satisfied in $\mathfrak{M}$. For sets Γ of assertions, we write $\mathfrak{M} \models \Gamma$ if $\mathfrak{M} \models \varphi$ for all $\varphi \in \Gamma$.
We now define an ADS as a pair of an ADL and a class of ADMs.

Definition 6 *An abstract description system is a pair $(\mathcal{L}, \mathcal{M})$, where $\mathcal{L}$ is an ADL and $\mathcal{M}$ is a class of ADMs for $\mathcal{L}$ that is closed under the following operation: if $\mathfrak{M} = \langle W, \mathcal{V}^{\mathfrak{M}}, \mathcal{F}^{\mathfrak{M}} \rangle$ is in $\mathcal{M}$ and $\mathcal{V}^{\mathfrak{M}'} = (x^{\mathfrak{M}'})_{x \in \mathcal{V}}$ is a new assignment of set variables in W then $\mathfrak{M}' = \langle W, \mathcal{V}^{\mathfrak{M}'}, \mathcal{F}^{\mathfrak{M}} \rangle \in \mathcal{M}$.*

What the closure condition suggests is that the set variables can be interpreted as arbitrary sets of the interpretation domain, so they are treated as variables in any ADS, while the interpretation of the function symbols remains the same. That is a property that all DLs comply with.
The main reasoning task for an ADS is the satisfiability problem for finite sets of term assertions.

Definition 7 *Let $S = (\mathcal{L}, \mathcal{M})$ be an ADS. A finite set Γ of term assertions is called satisfiable in S if there exists an ADM $\mathfrak{M} \in \mathcal{M}$ such that $\mathfrak{M} \models \Gamma$.*

So far, we have presented an overview of abstract description systems. It is not hard to see how many description logics formalisms can be reduced to their corresponding ADS. Let's think for example of the description language ALC which is comprised of concept names A_1, A_2, ..., role names R_1, R_2, ..., the Boolean operators $\neg$ and $\wedge$, and the existential and the universal restrictions $\exists$ and $\forall$ respectively. Then set variables would correspond to concept names, function symbols to concept constructors and term assertions to general concept inclusion axioms. The exact transformation is not presented here for brevity reasons, but can be found in [KLWZ04]. Analogous transformations are possible for more expressive DLs, like for example the $SHIQ$ language, and even for other family of formalisms, as for example propositional dynamic logics.

$\mathcal{E}$-Connections in Abstract Description Systems

So far, we have provided the framework of ADSs which enables us to translate very different formalisms into the fore mentioned abstract formalisms. On the other hand, our main interest resides in the ability to establish general form link relations between the components of the system. This is accomplished by $\mathcal{E}$-Connections, which can be viewed as the combination of the system components.
The $\mathcal{E}$-Connection $C^{\mathcal{E}}(S_1, ..., S_n)$ of n ADSs $S_1, ..., S_n$, where $S_i = (\mathcal{L}_i, \mathcal{M}_i)$ with $1 \leq i \leq n$, contains a set of terms and a set of assertions both partitioned into n sets, and is interpreted by a class of models.
We define the i-terms inductively: 1) every set variable of $\mathcal{L}_i$ is an i-term, 2) the set of i-terms is closed under $\neg$, $\wedge$ and the function symbols of $\mathcal{L}_i$, and 3) if $(t_1, ..., t_{i-1}, t_{i+1}, ..., t_n)$ is a sequence of k-terms t_k for $k \neq i$, then
$\langle E_j \rangle^i (t_1, ..., t_{i-1}, t_{i+1}, ..., t_n)$ is an i-term, for every $j \in J$.
The set of terms of $C^{\mathcal{E}}(S_1, ..., S_n)$ is the union of the set of the i-terms, for $1 \leq i \leq n$. The term assertions of $C^{\mathcal{E}}(S_1, ..., S_n)$ are of the form $t_1 \sqsubseteq t_2$, where both t_1 and t_2 are i-terms, for $1 \leq i \leq n$.

The semantics of $C^{\mathcal{E}}(S_1, ..., S_n)$ is given through the structure$\mathfrak{M} = \langle(\mathfrak{M}_i)_{i\leq n}, E^{\mathfrak{M}} = (E_j^{\mathfrak{M}})_{j\in J}\rangle$,where $\mathfrak{M}_i \in \mathcal{M}_i$ for $1 \leq i \leq n$ and $E_j^{\mathfrak{M}} \subseteq W_1 \times \cdots \times W_n$ for each $j \in J$. This structure is called a model for $C^{\mathcal{E}}(S_1, ..., S_n)$. The extension $t^{\mathfrak{M}} \subseteq W_i$ of an i-term t is defined by induction. For set variables X of $\mathcal{L}_i$ we put $X^{\mathfrak{M}} = X^{\mathfrak{M}_i}$, while the inductive steps for the Booleans and function symbols are the same as in Definition 5. Moreover, if $\overline{t_i} = (t_1, ..., t_{i-1}, t_{i+1}, ..., t_n)$ is a sequence of j-terms t_j, with $1 \leq j \leq n \wedge j \neq i$, then: $(\langle E_j\rangle^i(\overline{t_i}))^{\mathfrak{M}} = x \in W_i \mid \exists_{l\neq i} x_l \in t_l^{\mathfrak{M}}(x_1, ..., x_{i-1}, x, x_{i+1}, ..., x_n) \in E_j^{\mathfrak{M}}$.
The satisfiability of a set of assertions is defined as in previous section. The following theorem is the fundamental transfer result proved by Kutz, Lutz, Wolter and Zakharyaschev:

Theorem 2 *Let $C^{\mathcal{E}}(S_1, ..., S_n)$ be an $\mathcal{E}$-Connection of ADSs $S_1, ..., S_n$. If the satisfiability problem for each of $S_1, ..., S_n$ is decidable, then it is decidable for $C^{\mathcal{E}}(S_1, ..., S_n)$ as well.*

So, the general framework of $\mathcal{E}$-Connections defined in terms of abstract description systems preserves the satisfiability of the system components and moreover, it can be proved that an upper complexity bound for the satisfiability problem for $C^{\mathcal{E}}(S_1, ..., S_n)$ is one exponential higher that the time complexity of the original decision procedures for $S_1, ..., S_n$, while the combined decision procedure is non-deterministic. Nevertheless, it is not yet known whether this complexity result is optimal.
The $\mathcal{E}$-Connections mentioned so far are basic in the sense that they do not allow for more complex operations on link relations. For this purpose, a variety of extensions of basic $\mathcal{E}$-Connections has been defined and investigated in [KLWZ04]. These results are beyond the scope of the current work and will not be mentioned here.

3.3.3 Reasoning

Grau, Parsia and Sirin explore at [GPS04b] a variant form of $\mathcal{E}$-Connections among distributed ontologies expressed in $SHIF(D)$ and propose an extension of the ordinary tableau calculus for reasoning upon the connected system.
Let $D_1, D_2, ..., D_n$ n disjoint distributed domains and L_i, $1 \leq i \leq n$, propositionally closed Description Logics (extensions of ALC) that allow us to talk about the corresponding domain D_i, that can be expressed as abstract description systems and are decidable. As was the case before, we want to establish connections between the system components S_i, so as to express possible relations between the different domains. However, instead of using n-ary links to describe these relations, we now use binary links, each describing a relation between two different domains. So, we define countable and disjoint sets ϵ_{ij}, with $i, j = 1, ..., n$, of link names and in that case a link property $E \in \epsilon_{ij}$ can be seen as a relation associating elements from the domain D_i to the domain D_j, as opposed to relations in Description Logics, which associate elements of the same domain.
The semantics of $\mathcal{E}$-Connections defined as above, is given by a combined interpretation $I = (\{I_i\}_{i=1}^n, \{\epsilon_{ij}^I\}_{i,j=1,i\neq j}^n)$ of the local domains on the one hand and of the binary relations on the other. Additionally, we can define the two concept constructors $\exists E.C$ and $\forall E.C$, defined on the link property $E \in \epsilon_{ij}$, with the following semantics:

- $(\exists E.C)^I = \{x \in W_i \mid \exists y \in W_j, (x, y) \in E^I, y \in C^I\}$
- $(\forall E.C)^I = \{x \in W_i \mid \forall y \in W_j,$ if $(x, y) \in E^I$ then $y \in C^I\}$

The combined tableau method that was mentioned in the beginning of this section is defined for the combined logic $C^{\mathcal{E}}(SHIF(D))$, which is the $SHIF(D)$ family of DLs (on which the ontology language OWL-Lite is based), extended with the link properties (defined earlier) and containing only T-Box assertions. Essentially, the combined logic consists of n T-Boxes K_i, $i = 1, ..., n$, where each T-Box contains inclusion axioms of the form $C \sqsubseteq D$, where C and D are i-concepts. The reasoning algorithm works on a finite combined completion, which is a forest of $SHIF(D)$ completion trees, and it tries to construct a model for an input concept in the combined T-Box. If it succeeds it returns satisfiable, otherwise unsatisfiable.

The algorithm generates n kinds of nodes and n different kinds of trees, namely i-nodes and i-trees respectively. An i-tree is expanded with the expansion rules, plus the additional rules, and can contain i-nodes and j-nodes. Its root must always be an i-node, whereas j-nodes can only be leafs of the tree. If the algorithm is building an i-tree and at some point during the expansion a j-node is created, then the algorithm goes on with the expansion of the i-tree and only after the i-tree is completed does the algorithm proceed with the expansion of the j-node. A j-node g in an i-tree contains a clash iff the corresponding j-tree contains a clash when expanded. Otherwise, it is satisfiable and g is marked with the label "visited".
The three expansion rules added by the algorithm are:

- $\exists_{link}\ Rule$: If $\exists E.C \in L_i(x)$ ($E \in \epsilon_{ij}$), x is not blocked and has no E-successor y with $L(x,y)$ = $\{E\}$, and $\{C\} \in L_j(y)$, then create a new E-successor (j-node) y of x with $L_j(y) = \{C\}$. A new j-tree is created with root y labeled with $L_j(y)$. The j-tree won't be expanded until no more rules apply in the i-tree.
- $\forall_{link}\ Rule$: If $\forall E.C \in L_i(x)$ ($E \in \epsilon_{ij}$), x is not blocked and there exists an E-successor y of x, such that $\{C\} \notin L_j(y)$, then $L_j(y) = L_j(y) \cup \{C\}$.
- $CE\ Rule$: If $C_{K_i} \notin L_i(x)$, then $L_i(x) \leftarrow L_i(x) \cup \{C_{K_i}\}$, where $C_{K_i} = \sqcap_{C_i^j \sqsubseteq D_i^j \in K_i}(\neg C_i^j \sqcup D_i^j)$.

The first rules are the analogous to the expansion rules for qualified existential and universal qualification, where now the binary relation in the quantification associates different domains, so we have to distinguish between i-nodes and j-nodes. The third rule ensures that every i-node contains the concept C_{K_i}. Indeed, we cannot internalize the combined T-Box into a single concept and thus cannot reason with respect to an empty T-Box. In that case, the third rule makes sure that the i-node will satisfy the i component K_i of the combined T-Box.
Given as an input an i-concept X and a combined T-Box, the algorithm initially creates an i-tree with a single i-node x_i, labeled $L_i(x_i) = X$, and a j-tree for each j = 1, ..., n, j $\neq$ i, each with a single j-node labeled with the empty set.
In order to ensure termination, an additional blocking condition has to be added. More specifically, before starting to expand a j-tree T with root node g, created as a successor of some i-node, the algorithm has to check whether another i-node x exists in a not yet completed j-tree such that $L_j(g) \subseteq L_j(x)$. Should that be the case, we say that the node g is blocked by the node x, and the algorithm returns that the j-tree T is satisfiable. An i-tree tree is *complete* if either it contains a clash or no more can be applied to its i-nodes and all its produced j-nodes are "visited". In case, all trees can be expanded through the expansion rules in a way that each initial tree yields to a complete, non-clash tree, the algorithm returns "satisfiable" for the input i-concept X. otherwise it returns "unsatisfiable". The work at [GPS04b] proves that this tableau algorithm has the desirable properties of tableau algorithms in DLs, namely that 1) it *terminates*, and 2) there is a clash-free and complete combined completion iff the concept is satisfiable with respect to the knowledge base.

3.4 Distributed Terminological Knowledge under the Distributed First Order Logic Framework

Clearly, in both the $DDLs$ and the $\mathcal{E}$-Connections approach, the resulting system consists of two components: a family of terminological knowledge subcomponents described in DLs and a set of rules/connections that represent mappings between the subcomponents. A powerful logical framework that allows for the encoding of these approaches (as well as of others) is Distributed First Order Logic (DFOL). This logical formalism generally enables the representation of first order theories along with a set of axioms describing the relations between these theories. Based on this formalism, Serafini et. al. investigate at [SSW05] how DFOL can be used to encode distributed terminological knowledge. In this section, we provide a rather extended summary of their results.

3.4.1 Formalism

We first consider a set of first order languages with equality $\{L_i\}$ defined over a non-empty set of indexes. Obviously, each local language $L_i$by the i-th knowledge base. The signature of L_i is extended with a new set of symbols that are used to denote objects that are related to other objects in different ontologies. So, for each variable and each index $j \in J$ with $j \neq i$ two new symbols $x_{\rightarrow j}$ and $x_{j\rightarrow}$ are added. called arrow variables. Terms and formulas of L_i are defined as usual, while quantification on arrow variables is not permitted. The notation $\phi(x)$ is used to denote the formula ϕ as well as the fact that the free variables of ϕ are $x = x_1, ..., x_n$.

Definition 8 *A set of local models of L_i are a set of first order interpretations of L_i, on a domain dom_i which agree on the interpretation of L_i^c, the complete fragment of L_i.*

Two or more models can describe the same part of the world, in which case we say that they semantically overlap. DFOL explicitly represents semantic overlapping via a domain relation, where a domain relation from dom_i to dom_j is a binary relation $r_{ij} \subseteq dom_i \times dom_j$. A pair $< d, d' >$ in r_{ij} means that from the point of view of j, d in dom_i corresponds to d' in dom_j. The two-way correspondences need of course not to be the same, since each of them describes the correspondences from a subjective point of view, so in general we have $r_{ij} \neq r_{ji}$. Using the notion of domain relations one can define the notion of a model for a set of local models.

Definition 9 *A DFOL model M is a pair $< \{M_i\}, \{r_{ij}\} >$ where, for each $i \neq j \in I : M_i$ is a set of local models for L_i, and r_{ij} is a domain relation from dom_i to dom_j.*

We now extend the classical notion of assignment, so that arrow variables are also taken into consideration. We note that assigning arrow variables is not always possible, since the domain relation might be such that there is no consistent way of assigning arrow variables.

Definition 10 *Let M = $< \{M_i\}, r_{ij} >$ be a model for $\{L_i\}$. An assignment a is a family $\{a_i\}$ of partial functions from the set of variables and arrow variables to dom_i, such that:*

- $a_i(x) \in dom_i$,
- $a_i(x^{j\rightarrow}) \in r_{ji}(a_j(x))$,
- $a_j(x) \in r_{ij}(a)i(x^{\rightarrow j}))$.

An assignment a is admissible for a formula $i : \phi$ if a_i assigns all the arrow variables occurring in f. WE can now define the notion of satisfiability in first order logic:

Definition 11 *Let M = $< \{M_i\}, \{r_{ij}\} >$ be model for $\{L_i\}$, $m \in M_i$, and a an assignment. An i-formula ϕ is satisfied by m w.r.t. a, or $m \models_D \phi[a]$ if*

1. *a is admissible for $i : \phi$ and*
2. *$m \models \phi[a_i]$, according to the definition of satisfiability of first order logic.*

M $\models \Gamma[a]$ if for all $i : \phi \in \Gamma$ and $m \in M_i$, $m \models_D \phi[a_i]$.

From the point of view of DFOL, mappings can in reality be considered as additional constraints that involve more than one knowledge base and so they restrict the set of DFOL models that can interpret the combined knowledge base. In the DFOL framework they are defined by introducing a new kind of formula and are called interpretation constraints: Concretely:

Definition 12 *An interpretation constraint from $i_1, ..., i_n$ to i with $i_k \neq i$ for $1 \leq k \leq n$ is an expression of the form:*

$i_1 : \phi_1, ..., i_n : \phi_n \longrightarrow i : \phi$.

For the satisfiability of interpretation constraints we have:

Definition 13 *A model M satisfies the interpretation constraint, in symbols $M \models i_1 : \phi_1, ..., i_n : \phi_n \longrightarrow i : \phi$ if for any assignment a strictly admissible for $\{i_1 : \phi_1, ..., i_n : \phi_n\}$, if $M \models i_k : \phi_k[a]$ for $1 \leq k \leq n$, then a can be extended to an assignment a' admissible for $i : \phi$ and such that $M \models i : \phi[a']$.*

Depending on where the arrow variable appears (on the left or the right side of the interpretation constraint), the variable has a different meaning (universal or existential respectively). So:

1. $M \models i : P(x^{\rightarrow j}) \longrightarrow j : Q(x)$ iff
   For all $d \in \|P\|_i$ and for all $d' \in r_{ij}(d), d' \in \|Q\|_j$.
2. $M \models i : P(x) \longrightarrow j : Q(x^{i\rightarrow})$ iff
   For all $d \in \|P\|_i$ there is a $d' \in r_{ij}(d)$, s.t. $d' \in \|Q\|_j$.
3. $M \models j : Q(x^{i\rightarrow}) \longrightarrow i : P(x)$ iff
   For all $d \in \|Q\|_j$ and for all d' with $d \in r_{ij}(d'), d' \in \|P\|_i$.
4. $M \models j : Q(x) \longrightarrow i : P(x^{\rightarrow j})$ iff
   For all $d \in \|Q\|_j$ there is a d' with $d \in r_{ij}(d')$, s.t. $d' \in \|P\|_i$.

3.4.2 Associating Mapping Languages with the DFOL Formalism

Without entering in more details, we just notice that formalisms for mapping languages are based on four main parameters: local languages, local semantics used to specify the local knowledge, and mapping languages and semantics for mappings, used to specify the semantic relations between the local knowledge. From the previous sections it has been made clear that the local knowledge is encoded in a suitable fragment of DLs, which means that the local language is a suitable fragment of first order languages. Local semantics is expressed through the TBox models, which means that local knowledge is associated with a t most one FOL interpretation. To make this compatible with the DFOL framework, we just need to declare each L_i to be a complete language. This implies that all $m \in M_i$ have to agree on the interpretation of L_i symbols. Let us now see how the two mapping formalisms, namely $DDLs$ and $\mathcal{E}$-Connections, can be translated in the DFOL framework.

$DDLs$

We are interested in the following bridge rules:

1. $i : \phi \xrightarrow{\sqsubseteq} j : \psi$.
   The corresponding DFOL interpretation constraint is $i : \phi(x^{\rightarrow j}) \longrightarrow j : \psi(x)$.
2. $i : \phi \xrightarrow{\sqsupseteq} j : \psi$.
   The corresponding DFOL interpretation constraint is $j : \psi(x) \longrightarrow i : \phi(x^{\rightarrow j}))$.
3. $i : \phi \xrightarrow{\not\sqsubseteq} j : \psi$.
   For this bridge rule there is no corresponding DFOL interpretation constraint.

$\mathcal{E}$-Connections

In $\mathcal{E}$-Connections, we can have multiple links associating concepts from two subcomponents. To represent this, we label each arrow variable with the proper link name, so for example instead of just writing $x^{\rightarrow j}$ we also specify the link relation by writing $x^{\overset{E}{\rightarrow} j}$, where E the name of the link. With this syntactic extension of DFOL, $\mathcal{E}$-Connections can be translated in DFOL interpretation constraints as follows:

1. $\phi \sqsubseteq \exists E.\psi$

   The corresponding DFOL interpretation constraint is $i : \phi(x) \longrightarrow j : \psi(x^{i \overset{E}{\rightarrow}})$.

2. $\phi \sqsubseteq \forall E.\psi$

   The corresponding DFOL interpretation constraint is $i : \phi(x^{\overset{E}{\rightarrow} j}) \longrightarrow j : \psi(x)$.

3. $\phi \sqsubseteq \geq nE.\psi$

   The corresponding DFOL interpretation constraint is $i : \bigwedge_{k=1}^{n} \phi(x_k) \longrightarrow j : \bigwedge_{k \neq h=1}^{n} \psi(x_k^{i \overset{E}{\rightarrow}}) \wedge x_k \neq x_h$.

4. $\phi \sqsubseteq \leq nE.\psi$

   The corresponding DFOL interpretation constraint is $i : \phi(x) \wedge \bigwedge_{k=1}^{n+1} x = x_k^{\overset{E}{\rightarrow} j} \longrightarrow j : \bigvee_{k=1}^{n+1}(\psi(x_k) \supset \vee_{h \neq k} x_k = x_h)$.

We notice that it is not possible to model Boolean combinations of links, while it is possible to represent inverse links and link inclusion axioms.

3.5 Package-Based Description Logics

3.5.1 Motivation

The semantic web vision has as an important pillar the existence of collaborative environments for the construction, sharing and usage of ontologies. Indeed, collaboration between several developer groups with expertise in specific areas, with each developer contributing only a part of the ontology, is highly expected. Several problems that need to be addressed can arise in such a setting:

1. Local vs. Global Semantics: When integrating the independently developed ontology modules, we should keep in mind that each module represents a local point of view and is locally consistent. However, one cannot expect that the different modules also need to be globally consistent, since semantic conflicts between the different modules might occur. In such cases, additional care should be taken to manage these inconsistencies, without the need to discard any ontology.

2. Partial Reuse vs. Total Reuse: When building a new ontology, it is often necessary to import just some part of another ontology and not the whole ontology. It would thus be helpful, if the ontology has a modular structure and every time only some relevant part of it is being imported, leaving the irrelevant part outside the importing scope.

3. Organizational Structure vs. Semantic Structure: It is useful to distinguish between two different types of ontology structures: organizational structure and semantic structure. Organizational structure refers to the arrangement of the ontology in modules in such a way that they are easy to use. On the other hand, semantic structure, deals with the relationship between meanings (semantics) of terms in an ontology.

4. Knowledge Hiding vs. Knowledge Sharing: In most cases, the provider of the ontology is expected not to make the entireness of the ontology visible to outside user or developer communities, but it is rather likely that only some parts of the ontology will be exposed to specified communities. Moreover, the ontology component usually offers only a limited query interface, while the details of the implementation are kept hidden from the use. In both cases, knowledge hiding is a required feature of the ontology modules. Knowledge hiding allows thus for a type of semantic encapsulation, where detailed information is hidden from view, while a simpler query interface is provided.

Package-based Description Logics provide an interesting framework for achieving the requirements for an ontology collaborative environment and overcoming the problems described above, as will be shown in the following sections.

3.5.2 Formalism

Syntax

In $PDLs$ the whole ontology is composed of a set of packages (modules). We have the next definition:

Definition 14 *Let O = (S, A) be an ontology, where S the set of terms and A the set of axioms of the ontology. A package P = (Δ_S, Δ_A) of the ontology O is a fragment of O, where $\Delta_S \subseteq S$ and $\Delta_A \subseteq A$. The set of all possible packages is denoted as Δ_P. A term $t \in \Delta_S$ or an axiom $t \in \Delta_A$ is called a member of P ($t \in P$) and P is called the home package of t (HP(t) = P).*

A package P can also use terms defined in another package Q, called foreign terms in P. In that case we say that P imports Q and write $P \longmapsto Q$. The importing closure $I_{\longmapsto}(P)$ of a package P contains all packages that are either directly or indirectly imported into P. A package-based Description Logic ontology, or a PDL ontology, consists of multiple packages, each of them expressed in DL.

As mentioned in the previous section, it is usually useful to impose a hierarchical organizational structure over the PDL ontology, which coexists with the semantic structure and doesn't be the same as it. More specifically, we have:

Definition 15 *A package P_1 can be nested in one and only one other package P_2. In that case we write $P_1 \in_N P_2$ and say that P_1 is the subpackage of P_2 and P_2 is the superpackage of P_1. The collection of all package nesting relations in an ontology constitutes the organizational hierarchy of the ontology. By $P_1 \in_N^* P_2$ we denote the transitive closure of the nesting relationship.*

Semantics

For each package in a PDL, we can define the local interpretation of the package in a way similar to the interpretation for a description logic ontology. Since a local interpretation interprets everything in the local domain, there semantics of the foreign terms are not taken into account in the local domain. For the same reason, the same term can be interpreted differently in two packages and they do not need to be in accordance one with another.

Contrary to the local interpretation, a global interpretation is the interpretation of all packages of which the ontology consists. It interprets the ontology from the global point of view of all packages and not from the local point of view of a single package.Given that the global interpretation tries to interpret the ontology from the global point of view of all its packages, it is evident that a PDL ontology is globally consistent, iff a global interpretation exists. While it perfectly makes sense to demand that a local interpretation for each package exists, the same doesn't hold for the stronger requirement of global interpretation of all packages. Indeed, if local consistency cannot be guaranteed, integrity of any information that is based on that package cannot be guaranteed. On the other hand, taking into consideration that the different modules may be independently

developed from developer groups with different perspectives and interests, it could be that global consistency does not exist.

Finally, a distributed interpretation witnessed by a package P represents the semantics of axioms in P and its importing closure. It is different from a local interpretation, in that a local interpretation provides an interpretation only for the local package, handling foreign terms as any local terms. It is also different from a global interpretation, since the latter is a global model for the whole ontology, whereas the former is a model for only some packages in the ontology.

Definition 16 *For a package-based ontology* $< \{P_i\}, \{P_i \xrightarrow{t} P_j\}_{i \neq j} >$*, a distributed interpretation is a pair M =* $< \{I_i\}, \{r^t_{ij}\}_{i \neq j} >$*, where* $I_i = < \Delta_i, (\cdot)_i >$ *is the local interpretation of package* P_i*,* $r^t_{ij} \subseteq \Delta_i \times \Delta_j$ *is the interpretation for the importing relation* $P_i \xrightarrow{t} P_j$*. For any such relation r from* Δ_i *to* Δ_j *an any individual* $d \in \Delta_i$*, r(d) denotes the set* $\{d' \in \Delta_j | < d, d' > \in r\}$*. For any subset* $D \subseteq \Delta_i$*, r(D) denotes* $\cup_{d \in D} r(d)$ *and we call it the image set of D.*

The importing relation must satisfy the following three conditions: 1) every importing relation is one-to-one in that it maps an object of t^{I_i} to a single unique object in t^{I_j}, 2) importing relations are consistent for different terms, i.e. for any $i : t_1 \neq i : t_2$ and any $x, x_1, x_2 \in \Delta_i$, $r^{t_1}_{ij}(x) = r^{t_2}_{ij}(x)$ and $r^{t_1}_{ij}(x_1) = r^{t_2}_{ij}(x_2) \neq \emptyset \rightarrow x_1 = x_2$, and 3) Compositional Consistency: if $r^{i:t_1}_{ik}(x) = y_1$, $r^{i:t_2}_{ij}(x) = y_2$, $r^{i:t_3}_{jk}(x) = y_3$, and y_1, y_2, y_3 are not empty set, then $y_1 = y_3$.

The *image domain relation* between I_i and I_j is $r_{ij} = \cup_t r^t_{ij}$ and is strictly one-to-one. We can imagine the image domain relation as an isomorphism that "copies" the relevant part of the I_i domain to the I_j domain and establishes unambiguous communication between the two packages. A very strong contradiction to PDL importing with regard to $DDLs$ and $\mathcal{E}$-Connections lies in the fact that local domain disjointness doesn't need to hold anymore. In fact, there are cases where a local model partially overlaps with the model of an imported module.

A concept $i : C$ is satisfiable with respect to a PDL O = $< \{P_i\}, \{P_i \xrightarrow{t} P_j\}_{i \neq j} >$ if there exists a distributed interpretation of O such that $C^{I_i} \neq \emptyset$. A package P_k witnesses subsumption $i : C \sqsubseteq j : D$ iff $r^C_{ik}(C^{I_i}) \subseteq r^D_{jk}(D^{I_j})$ holds for every model of P_k and all packages in its importing closure.

3.5.3 Reasoning

The main motivation behind a tableau algorithm for $ALCP_C$ (the extension of the closed propositional Description Logic ALC with packages) is distribution. Instead of a centralized inference procedure constructing a single global tableau of the ontologies, we would prefer multiple federated local tableau, which would result in the distribution of the inference load and precipitation of the inference. That is possible by the image relations, i.e. a local tableau can create "image" nodes of its nodes in another local tableau. Specifically:

Definition 17 *A distributed tableau for an* ALC_P *ontology* $\{P_i\}$ *is a tuple* $< \{T_i\}, \{r_{ij}\}_{i \neq j} >$*, where* T_i *is a local* ALC *tableau for the package* P_i*,* r_{ij} *is the image relation between* T_i *and* T_j*, such that it creates one-to-one mappings from a subset of nodes in* T_i *to a subset of nodes in* T_j*. For any i-node* x *and any j-node* y*,* $(x, y) \in r_{ij}$*,* x *is called the pre-image of* y *and* y *the image of* x*. The local label* $L_i(x)$ *of* x *in a local tableau* T_i *only contains i-concepts.*

It is important to notice that unlike the combined tableau for $\mathcal{E}$-Connections, in the distributed tableau environment all local tableau will be autonomously created and maintained by local reasoners, while the communication between them is more relaxed, resembling peer-to-peer networks. The communication between a local tableau T_j and another local tableau T_i is accomplished by the following set of primitive messages:

1) *Membership m(y,C)*: given an individual y and an i-concept C, querying whether there is a pre-image or image y' of y in T_i, such that $C \in L_i(y')$;

2) *Reporting r(y,C)*: given an individual y and an i-concept C, if there is a pre-image or image y' of y in T_i,

$C \notin L_i(y')$, then $L_i(y') = L_i(y') \cup \{C\}$; if there is no existing pre-image or image y' of y, then create a y' with $L(y') = \{C\}$ and add (y', y) to the image relation r_{ij};
3) *Clash*$\bot(y)$: an individual y in T_j contains a clash; and
4) *Model*$\top(y)$: no expansion rules can be applied on y or any of its descendants in T_j.

In the trivial case where $i = j$, the messages above are reduced to local operations: $m(y, C)$ is reduced to querying whether $Cb \in L_j(y)$, $r(y, C)$ is reduced to adding C to $L_j(y)$. $\bot(y)$ reports local inconsistency, and $\top(y)$ reports completion of the local tableau.

For these primitive messages it is important that a local tableau knows each time which other local tableau to address to, that is the destination package. Formally:

Definition 18 *An atomic concept C or its negation $\neg C$ destination is the home package of C, i.e. HP(C). A complex concept C destination is the package in which it is generated. Destination of C is denoted as $\delta(C)$.*

$ALCP_C$ Tableau Expansion Rules

The expansion rules for ALC are well studied a concise presentation of them can be found at [BCH06d]. $ALCP_P$ expansion rules modify the ALC expansion rules, so that each module is only locally internalized, and not globally internalized w.r.t. to a combined TBox; moreover, a local tableau can create copies of its local nodes in other local tableau when needed during an expansion. The expansion rules are:

- $\sqcap$-rule: if $C_1 \sqcap C_2 \in L_i(x)$ and x is not blocked, then (1) if $m(x, C_1)$=false, then do $r(x, C_1)$ (2) if $m(x, C_2)$=false, then do $r(x, C_2)$.
- $\sqcup$-rule: if $C_1 \sqcup C_2 \in L_i(x)$, x is not blocked, but $m(x, C_1)$ or $m(x, C_2)$ is false, then do $r(x, C_1)$ or $r(x, C_2)$.
- $\exists$-rule: if $\exists R.C \in L_i(x)$, x is not blocked, and x has no R-successor y with $m(y, C)$ = true, then create a new node y with $L_i(< x, y >)$=$\{R\}$ and do $r(y, C)$.
- $\forall$-rule: if $\forall R.C \in L_i(x)$, x is not blocked, and x has an R-successor y with $m(y, C)$ = false, then do $r(y, C)$.
- CE-rule: if $C_{T_i} \notin L_i(x)$, then $L_i(x)$=$L_i(x) \cup C_{T_i}$, where C_{T_i} is the internalized concept for the TBox of P_i.

We say that a distributed tableau is distributively complete if no $ALCP_C$ expansion rules can be applied on any of its local tableau, and it is distributively consistent if all of its local tableau are consistent. We assume that a satisfiability query regarding some concept C is initially sent to a package P_j, called the witness package. We first create the local tableau T_j with a node x_0 labeled by $\{C_{T_j\}\sqcap C}$ ($\{C_{T_j\}\sqcap C} \in L_j x_0$), and apply the $ALCP_C$ expansion rules, until a distributively complete and consistent tableau is found, or all expansion processes eventually fail.

In order to ensure that no infinite expansion of the tableau will take place, one has to impose a suitable blocking strategy on the tableau. To make things simpler, we assume that there are no cycles in the importing relations, that is if some package P_i is being imported directly or indirectly by a package P_j, then the package P_j cannot import directly or indirectly the package P_i. In that case the required *blocking strategy* to guarantee termination is that a node x is blocked by a node y, iff both nodes are in the same local tableau T_i, y is a local ancestor of x, and $L_i(x) \subseteq L_i(y)$.

The reason this blocking mechanism works has to do with the acyclical importing relations and the way the tableau expansion works. More concretely, if a tableau T_i contains an image node in another tableau T_j, then due to the acyclical nature of importing the tableau T_j as well as its descendant tableau will never report a message to the tableau T_j. So, if there is a path from a node x to a node y in tableau T_i, then this path will only contain nodes from the local tableau without traversing through image nodes in other tableau, and thus it is enough to restrict blocking to the local tableau.

The coordination between different local tableau with acyclic importing makes use of the following operations:
1) *Waiting*: If a node x has sent a reporting message to another local tableau, x is temporarily blocked until it receives a clash message from any of its image nodes, or model messages from all of its image nodes;
2) *Clash Message*: If a node x contains a local clash, or it receives a clash report, a) if there is no other search choice that can be applied to x, send clash reports $\perp(x)$ to x's local direct ancestor and all preimage nodes, and destroy all image relations from x, or b) if there are other search choices, restore the state (including image relations) of x to the state before the last choice, and try the next choice; and
3)*Model Message*: If no $ALCP_C$ expansion rules can be applied to x, x has no clash nor received any clash message, there is no image node of x, or all image nodes return model messages, then send model messages $\top(x)$ to x's local direct ancestor and all preimage nodes.

The algorithm terminates when the root node in the local tableau of the witness package receives a clash message or a model message. In the former case the input concept is unsatisfiable, while in the latter case it is satisfiable. Moreover, it is *sound* and *complete*, and its worst case run time complexity is no greater than that of the classical tableau for ALC w.r.t. the size of the input concept and the total size of the combined terminology set of all packages.

3.6 A Decentralized Consequence Finding Algorithm in a Peer-To-Peer Setting

3.6.1 Motivation

Most approaches dealing with distribution in the semantic web (and the approaches examined so far are no exception to this) adopt a "the more complex, the better" point of view. More concretely, a very active aim of these approaches is to stretch the expressivity of the constituent ontologies of the system as far as possible. The main incentive behind this is that the richer the expressivity the better the application domain can be modeled through the ontology and, consequently, the more functional and useful the whole system will be. However, this high expression usually comes along with high reasoning costs and the resulting system might be inefficient. Moreover, there are already a number of applications on the web, like social networks, where the systems consist of a very big number of peers, each of whom describes a relatively simple taxonomy, and the primary objective is to achieve scalability and efficient dynamic behavior.

A "small is beautiful" approach is definitely more appropriate in these cases. We can imagine such systems consisting of a big number of peers, each of them providing a simple personalized ontology (for example a taxonomy), and the peers can be connected with other peers through logical mappings. Such mappings contribute to a collaborative exchange of data between adjacent peers, so that one peer can have access to relevant data stored by other peers, to which it is connected through mappings. In such a setting there is no peer trying to impose itself on the other peers its own ontology and there is nothing like a super-peer or a central mediator, which controls the system and has a global overview of the peers and their connections, as in that case the scalability of the system would be severely suppressed. On the contrary, we consider distributed architectures, where every peer can dynamically enter and leave the network, and in addition to its own ontology it can also mediate between some other peers of the system to ask and answer queries. Such an approach promotes a novel nature of the semantic web, where the semantic web is actually conceived as a peer-to-peer data management system of very large dimensions.

This approach is extensively examined at [RAC$^+$06] by Rousset et al. The "small is beautiful" character is realized in this case by considering the full propositional fragment of OWL-DL to express the local ontologies instead of other more expressive languages, which come nevertheless at a greater cost. Queries are asked to a peer in terms of its local ontology and are answered using available mappings of the given peer with its acquainted nodes. They are then translated into a correspondent distributed propositional theory and the produced query rewritings are evaluated by a message passing algorithm named DeCA (Decentralized Consequence Finding Algorithm). These results are the main contributions of the work at [ACG$^+$06]. In the sections to follow, we provide a summary of these results.

3.6.2 Formalisms

Data Model

Each peer ontology contains a local ontology expressed in the full propositional fragment of the OWL-DL language. Atomic concepts, the top concept ($\top$) and the bottom concept ($\bot$) form the inductive basis of the concept descriptions, which are inductively defined as the union ($\cup$), the intersection ($\cap$) and the negation ($\neg$) of concept descriptions. Axioms of class definitions are either equivalence (complete definition) or inclusion (partial definition) axioms, where the left side is always an atomic concept. Terminological assertions on the available class descriptions is possible through axioms on class descriptions, and can can be equivalence, inclusion and disjointness. It is clear that the underlying model of the peer ontology provides the same level of expressivity as the full propositional calculus. It mainly allows for taxonomies of class descriptions along with disjointness axioms.

The local ontology of the peer provides a specification of the modeling level. On the low level we have the data storage and for this purpose we need another specification that captures the classes that are responsible for the data storage. This is accomplished through 1) the declaration of atomic extensional classes in terms of atomic classes of the peer ontologies, and 2) assertional statements over the extensional classes. More concretely, declaration of atomic extensional classes is possible with inclusion assertions between the extensional class and a class description of the ontology.

When a peer enters a peer-to-peer network, it establishes a number of mappings with its acquainted peers. A mapping in this context can be a disjointness, equivalence or inclusion axiom involving atomic classes of different peers. The mappings express the correspondences between different peers and are taken into account during the distributed reasoning procedure.

In this setting no super-peer with a global, unified schema exists. Contrary to that, there are only peers with knowledge of their local ontology and the nodes, which they are acquainted through mappings with. For each such peer P_i let O_i, V_i and M_i be the assertions defining the local ontology, the extensional classes and the mappings, respectively. The schema of the peer-to-peer network P, written as S(P), is defined as the union $\cup_{i=1..n} O_i \cup V_i \cup M_i$.

Semantics is defined in terms of interpretations, as is the case with DLs (and first order logics in general), so we choose not to enter into details. We only notice that a model is an interpretation that satisfies all the axioms of the distributed schema.

Query Rewriting through Propositional Encoding

A query is posed to a peer using its vocabulary and it is a logical combination of class descriptions of the given peer ontology. Besides the local peer it is possible to infer possible answers through remote nodes that it is acquainted with, so interactions between the local nodes and the acquainted nodes must additionally be taken into consideration. To achieve this, the query is being rewritten into a set of rewritings that are expressed in terms of only extensional classes, both of the local peer an of its adjacent nodes. These rewritings provide in intensional form the answers to the initial query and can then be further processes in order to take the (extensional) answers. We have the following definition:

Definition 19 *Given a peer-to-peer network P = $\{P_i\}_{i=1..n}$, a propositional combination Q_e of extensional classes is a rewriting of a query Q iff P $\models Q_e \sqsubseteq Q$. Q_e is a proper rewriting if there exists some model I of S(P) such that $Q_e^I \neq \emptyset$. A conjunctive rewriting is defined as a rewriting which is a conjunction of extensional classes. A rewriting Q_e is called a maximal (conjunctive) rewriting if there does not exist another (conjunctive) rewriting of Q strictly subsuming Q_e.*

The reason we are so interested in query rewriting is a result proved at [GR04] stating that when a query has a finite number of maximal conjunctive rewritings, then the query answers can be obtained as the the union of the answers of the the query rewritings. Obviously, in such a context the main problem is to find an efficient

procedure for query rewriting. In the full propositional calculus every set of propositions can be translated into its equivalent clausal form. Inspired by this observation and by the restriction of the language to the propositional DL fragment, Rousset et al. first encode the distributed schema of the peer-to-peer network and the query to their propositional encodings. Then they apply a novel message-based algorithm for consequence finding in distributed propositional theories and this algorithm returns a set of logical combinations of extensional classes. Since the (intentional) answer set can be transformed to its equivalent (finite) normal clausal form, the condition of a finite number of maximal conjunctive rewritings holds, and we can then get the query answers as described above. Let us now see that in a little more detail.

For propositional encoding, we translate the query and every schema axiom to an equivalent propositional formula, where atomic classes play the role of propositional variables. The propositional encoding of a class description is:

- Prop($\top$) = true, Prop($\bot$) = false
- Prop(A) = A, if A is an atomic class
- Prop($D_1 \sqcap D_2$) = Prop(D_1) $\wedge$ Prop(D_2)
- Prop($D_1 \sqcup D_2$) = Prop(D_1) $\vee$ Prop(D_2)
- Prop($\neg D$) = $\neg$Prop(D)

The propositional encoding of the axioms is on the other hand as follows:

- Prop($C \sqsubseteq D$) = Prop(C) $\Rightarrow$ Prop(D)
- Prop($C \equiv D$) = Prop(C) $\Leftrightarrow$ Prop(D)
- Prop($C \sqcap D \equiv \bot$) = $\neg$Prop(C) $\vee$ $\neg$Prop(D)

In order to establish the main propositional transfer result, we need one more definition from propositional calculus. More specifically, if T a clausal theory and q a clause, then a clause m is called a prime implicate of q w.r.t. T iff $T \cup \{q\} \models m$ and for any other clause m', if m' is an implicate of q w.r.t. T and $m' \models m$ then $m' \equiv m$. Moreover, m is called proper if we have additionally that $T \not\models m$. The following theorem is fundamental:

Theorem 3 *Let P be a peer-to-peer network and let Prop(S(P)) be the propositional encoding of the distributed schema. Then:*

- *S(P) is satisfiable iff Prop(S(P)) is satisfiable.*
- *q_e is a maximal conjunctive rewriting of a query q iff $\neg$Prop(q_e) is a proper prime implicate of $\neg$Prop(q) w.r.t. Prop(S(P)) such that all its variables are extensional classes.*

From the above theorem, it immediately follows that the maximal conjunctive rewritings can be obtained by the negation of the proper prime implicates of $\neg$q w.r.t. the propositional encoding of the schema of S(P). We notice that since the number of proper prime implicates of a clause w.r.t. to a clausal theory is always finite, every query has a finite number of query rewritings and, thus, the answers to the query can be computed as the union of the answers to the given rewritings. Moreover the data complexity is in PTIME [GR04].

The remaining problem is consequence finding in distributed propositional theories, which computes the proper prime implicates of a query w.r.t. a distributed propositional theory. Rousset et al. propose at [ACG+06] a message-based algorithm that takes as input a literal and produces all the proper prime implicates for a given set of target variables. The fact that the algorithm takes as input a literal and not a more complex logical combination poses no problem, since we can decompose the query to its literals, find the corresponding maximal rewritings and combine them in a way that we get the resulting maximal conjunctive queries. The next section gives a short overview of this algorithm.

3.6.3 Reasoning

The message-based algorithm for consequence finding in distributed propositional theories is based on these observations:

- The initial query posed to a certain local node can after its propositional encoding decomposed into its corresponding literals. Finding the query rewritings for each of them is enough, since taking their logical combinations can produce the rewritings for the initial query.
- For each such literal the peer which we pose the query to checks resolves this literal upon the clauses of its local propositional theory and keeps only the clauses where the non-shared variables belong to the target variables. If a contradiction is reached, then no rewritings can be obviously produced. Of course before the resolution takes place, the local peer must ensure that the literal has not already computed the rewritings and also that no rewritings for the complement of the literal have been computed in the reasoning branch, otherwise we infer a contradiction.
- For the clauses obtained by the previous post the peer queries its acquaintances so as to find other possible query rewritings by passing them query messages.
- After getting back the rewritings from its acquainted peers (in an asynchronous way) through answer messages the peer combines the different rewritings and produces thus the full rewritings of the initial query, where only target variables are used. It takes care so that every rewriting is produced only once.
- During this procedure every peer needs to know if all the rewritings have been sent back to him, otherwise termination of the algorithm would not be guaranteed. This condition is satisfied by demanding that the peers send explicit final messages to the peers above them in the reasoning branch so as to let them know that all rewritings asked have been produced.

This algorithm has the properties of soundness, completeness (depending on the acquaintance graph) and termination. In the context of the present work, we have omitted proofs and its interesting technical details. The interested reader is referred to [ACG+06].

3.7 A Mapping System for Query Answering in a Peer-To-Peer Setting

3.7.1 Motivation

When building large-scale applications in the Semantic Web, three important challenges need to be addressed. The first is the coordination of the different nodes. In completely centralized environments, we assume the existence of a super-node having control over all other nodes of the system and helping to their coordination, whereas in completely decentralized infrastructures every local node can act autonomously and no global coordination exists. The second challenge has to do with the heterogeneity of the available data sources. In a large-scale scenario with multiple developer communities, each contributing a fraction of the total application, the adoption of a global schema is highly unrealistic. On the contrary, it makes more sense to think of heterogeneous data sources, in which case the need for specifying the relationship of one to another arises. The third difficulty that we have to overcome is the development of efficient techniques for reasoning over multiple nodes, by also taking into account the semantics of mappings.

The work of Haase and Wang at [HW07] describes a decentralized infrastructure for query answering over distributed ontologies, called KAONp2p. Each of the aforementioned challenges is answered at their work by a suitable approach: the decentralized architecture with no global coordination is provided by a peer-to-peer network (whose motivation we described in the previous session), heterogeneity in ontologies can be overcome by establishing mappings among the local ontologies residing on different nodes that have the form of conjunctive queries, whereas efficient reasoning techniques are available by transforming the relevant

local ontologies along with all corresponding mappings into their equivalent disjunctive datalog program, for which efficient evaluation and optimization techniques, like magic sets, are available.

Their work is mainly based on the previous work by Haase and Motik [HM05], which defines a mapping system for the integration of OWL-DL ontologies and provides a query answering mechanism for the integrated system.

In this section, we will look initially into the formalisms and the query answering process suggested at [HM05] and we will then see how the KAONp2p system allows for a distributed reasoning process.

3.7.2 Formalisms

Preliminaries

Reducing $SHIQ$ to Disjunctive Datalog. A common characteristic among the approaches investigated so far is that at the center of the reasoning procedure lies the tableau calculus. This is of course not surprising, given the very robust results obtained in practice by most tableau reasoners. Indeed, the experience shows that these systems perform in practice much better that what their ExpTime worst-time complexity suggests.

Despite their very good performance for TBox assertions, these systems exhibit a relatively poor performance when queried over large ABoxes, which can be summarized to two main reasons: 1) tableau-based algorithms treat all individuals the same, that is they don't try to group them according to common properties, and 2) only a rather small part of the ABox is every time relevant to answer the query, nevertheless the tableau calculus does not really take the query information into account [MSS05].

A novel approach proposed by Motik at [Mot06] overcomes this problem by reducing $SHIQ$ Description Logics to disjunctive datalog programs, while preserving the semantics of the knowledge base. Under the disjunctive datalog formalism one can make use of different optimization techniques such as magic sets or join-order optimization. Magic sets permit that only a set of relevant facts is derived during the query evaluation. while join order optimization manipulates individuals in sets and then applies inference rules to all individuals in a set rather than to each individual separately. The reduction from DLs to disjunctive datalog is quite extended and will not be described in this report.

Data Integration Foundations. Mappings server primarily as a way to integrate different ontologies. Lenzerini [Len02] has investigated the theoretical foundations of data integration through mappings.

Definition 20 *A data integration system I consists of a triple $< G, S, M >$, where:*

- *G is the global schema, expressed in a language L_G over an alphabet A_G. The alphabet comprises a symbol for each element of G.*
- *S is the source schema, expressed in a language L_S over an alphabet A_S. The alphabet A_S includes a symbol for each element of the sources.*
- *M is the mapping between G and S, constituted by a set of assertions of the forms 1) $q_S \rightsquigarrow q_G$ and 2) $q_G \rightsquigarrow q_S$, where q_S and q_G are two queries of the same arity, respectively over the source schema S and over the global schema G. Queries q_S are expressed in a query language $L_{M,S}$ over the alphabet A_S, and queries q_G are expressed in a query language $L_{M,G}$ over the alphabet A_G.*

The source schema describes the structure of the sources, where the real data lie, whereas the global schema provides a reconciled, integrated and virtual view of the underlying sources. Queries to the data integration system are posed in terms of the global schema G, and are expressed in a query language L_Q over the alphabet A_G.

In order to assign semantics to a data integration system $I = < G, S, M >$, we first consider a database D that conforms to the source schema S and satisfies all constraints in S. We can then specify the information

content of the global schema G. We call global database for I any databases for G. A global database B for I is said to be legal with respect to D. if: 1) B is legal with respect to G, i.e. B satisfies all the constraints of G, and 2) B satisfies the mapping M with respect to D.

It is very interesting to notice that the semantics of a data integration system depends on how we interpret the mapping component of the integrated system. The two main approaches are the local as view approach (LAV) and the global as view approach (GAV). In the former approach the content of each source element s is characterized in terms of a view q_G over the global schema ($s \rightsquigarrow q_G$), while in the latter the content of each element of the global schema g is characterized in terms of a view q_S over the sources ($g \rightsquigarrow q_S$).

Finally, given a source database D for I, the answer $q_{I,D}$ to a query q in I with respect to D, is the set of tuples t of objects in a fixed domain Γ such that $t \in q_B$ for every global database B that is legal for I with respect to D. The set $q_{I,D}$ is called the set of certain answers to q in I with respect to D.

Rules and Queries

Conjunctive queries over a $SHOIN(D)$ knowledge base are defined as follows:

Definition 21 *Let KB be a $SHOIN(D)$ knowledge base, and let N_p be a set of predicate symbols, such that all $SHOIN(D)$ concepts and all abstract and concrete roles are in N_P. An atom has the form $P(\mathbf{s_1}, ..., \mathbf{s_n})$, denoted also as $P(\mathbf{s})$, where $P \in N_P$, and $\mathbf{s_i}$ are either variables or individuals from KB. An atom is called ground atom, if it is variable-free. An atom is called a DL-atom if P is a $SHOIN(D)$ concept, or an abstract or a concrete role. Let $\mathbf{x_1}, ..., \mathbf{x_n}$ and $\mathbf{y_1}, ..., \mathbf{y_m}$ be sets of distinguished and non-distinguished variables, denoted also as $\mathbf{x}$ and $\mathbf{y}$, respectively. A conjunctive query over KB, written as $Q(\mathbf{x},\mathbf{y})$, is a conjunction of atoms $\wedge P_i(\mathbf{s_i})$, where all $\mathbf{s_i}$ together exactly contain $\mathbf{x}$ and $\mathbf{y}$. A conjunctive query $Q(\mathbf{x}, \mathbf{y})$ is DL-safe if each variable occurring in a DL-atom also occurs in a non-DL atom in $Q(\mathbf{x}, \mathbf{y})$. The translation of such a query into a first order formula is:*

$$\pi(Q(\mathbf{x}, \mathbf{y})) = \exists y : \bigwedge \pi(P_i(\mathbf{s_i}))$$

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:

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

The main inferences for conjunctive queries are:

- *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, such that $\pi(KB) \models \pi(Q(\mathbf{x}\theta), \mathbf{y})$.*
- *Checking query containment. A query $Q_2(\mathbf{x}, \mathbf{y_2})$ is contained in a query $Q_1(\mathbf{x}, \mathbf{y_1})$ w.r.t. KB, if $\pi(KB) \models \pi(Q_2(\mathbf{x}, \mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x}, \mathbf{y_1}))$.*

We now give the definition of rules:

Definition 22 *A rule over a $SHOIN(D)$ knowledge base KB has the form $H \leftarrow Q(\mathbf{x}, \mathbf{y})$, where H is an atom and $Q(\mathbf{x}, \mathbf{y})$ a query over KB. We assume rules to be safe, i.e. each variable from H occurs in $\mathbf{x}$ as well. A rule is DL-safe if and only if $Q(\mathbf{x}, \mathbf{y})$ is DL-safe. We extend the operator π to translate rules into first-order formulas as follows:*

$$\pi(H \leftarrow Q(\mathbf{x}, \mathbf{y})) = \forall x : \pi(H) \leftarrow \pi(Q(\mathbf{x}, \mathbf{y}))$$

A program P is a finite set of rules and we say that P is DL-safe if all rules are DL-safe. A combined knowledge base is a pair (KB,P) and we define $\pi((KB, P)) = \pi(KB) \cup \pi(P)$. The main inference in (KB,P) is query answering, i.e. deciding whether $\pi((KB, P)) \models A$ for a ground atom A.

An integration system for OWL-DL

Based on the section 3.7.2, Haase and Motik proceed at [HM05] with the introduction of a mapping system for OWL-DL ontologies. Such a mapping system is still a triple $< S, T, M >$, where now the global schema is just the target OWL-DL ontology T, the source schema is the source OWL-DL ontology S, and for the mapping assertions $q_S \rightsquigarrow q_T$ we have that q_S and q_T are conjunctive queries over S and T, respectively, with the same set of distinguished variables **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. $S \cup T$, and is equivalent to an axiom $\forall \mathbf{x} : q_T(\mathbf{x}, \mathbf{y_T}) \leftarrow q_S(\mathbf{x}, \mathbf{y_S})$; an assertion $q_S \sqsupseteq q_T$ is called a complete mapping, requiring that q_T is contained by q_S w.r.t. $S \cup T$, and is equivalent to an axiom $\forall \mathbf{x} : q_S(\mathbf{x}, \mathbf{y_S}) \leftarrow q_T(\mathbf{x}, \mathbf{y_T})$; and an assertion $q_S \equiv q_T$ is an exact mapping, requiring it to be sound and complete.

The semantics of the mapping system is defined through translation into first order logic.

Definition 23 *For a mapping system MS = (S, T, M), let*

$$\pi(MS) = \pi(S) \cup \pi(T) \cup \pi(M).$$

The main inference for MS is computing answers of $Q(\mathbf{x}, \mathbf{y})$ with respect to MS, for $Q(\mathbf{x}, \mathbf{y})$ a conjunctive query.

This semantics is equivalent to the usual model theoretic semantics based on local and global models, where a query answer must be an answer in every global model. Unfortunately, query answering in such a general setting is undecidable, so two special types of mappings have been introduced that lead to decidable query answering.

The first class of mappings, called full implication mappings, captures the mappings that can be directly expressed in OWL-DL. In this case, q_S and q_T are DL-atoms of the form $P_s(\mathbf{x})$ and $P_t(\mathbf{x})$, where P_s and P_t are either DL concepts (concept mappings) or abstract or concrete roles (role mappings).

The second class of mappings is inspired by the fact that the undecidability of query answering for general implication mappings is due to the unrestricted use of non-distinguished variables in either q_S or q_T. To overcome this, we disallow the use of non-distinguished variables in the query that is located in the head of the assertion. These mappings are called safe mappings. Query answering with such mappings is still undecidable in the general case. Therefore, we require the query in the body of the assertion to be DL-safe, thus limiting the applicability of the rules to known individuals. Thus mappings correspond to (one or more) DL-safe rules, for which efficient algorithms for query answering are known.

Extending the second class of mappings, we can relax the restrictions introduced by DL-safety, by eliminating non-distinguished variables through a reduction of tree-like parts of a query to a concept. In that case, we get mappings with tree-like query parts.

3.7.3 Reasoning

The original work at at [HM05] bases on the notion of DL-safe mappings and allows query answering by integrating the local ontologies along with the mappings established between them. As this is highly non-scalable, the work at [HW07] avoids doing the integration of the whole system and instead it takes into account only these ontologies that are considered relevant to query answering according to some criteria. In this section, we first discuss how query answering is realized through integration of the whole system, and then we describe how we can only restrict reasoning to ontologies with relevant information.

Query Answering

For a set of local sours ontologies $S_1, ..., S_n$, a global ontology T and corresponding mapping systems $MS_1, ..., MS_n$ with MS_i = (S_i, T, M_i), an ontology integration system IS is a mapping system (S, T, M)

with $S = \cup_{i\in\{1...n\}} S_i$ and $M = \cup_{i\in\{1...n\}} M_i$. The main inference task for IS is still to compute answers of a conjunctive query $Q(\mathbf{x}, \mathbf{y})$ over T w.r.t. $S \cup T \cup M$.

The algorithm proposed is based on the reduction from $SHIQ(D)$ to disjunctive datalog (Section 3.7.2), from which it inherits two limitations: 1) IS is required to be based on $SHIQ(D)$ knowledge bases, and 2) the conjunctive query $Q(\mathbf{x}, \mathbf{y})$ and the queries in mappings are not allowed to contain transitive roles. The algorithm starts by eliminating non-distinguished variables from $Q(\mathbf{x}, \mathbf{y})$ and the mappings using the roll-up technique. After roll-up, the obtained mappings and queries are required to be DL-safe, which is needed for decidable query answering. In this case, the source ontology, target ontology and the mappings are converted into a disjunctive datalog program, and the original query is answered in the obtained program. This algorithm exactly computes the answer of the query in the integrated system.

Reasoning in KAONp2p

As mentioned at the beginning of the section, the KAONp2p infrastructure consists in reality of a peer-to-peer network. Each node of the network accommodates the local OWL-DL ontology, whereas between the different nodes we can define mappings of the form discussed in section 3.7.2. Moreover, each node of the distributed system can be asked a conjunctive query. Obviously, the main objective of the system is to answer the query posed to the local node by taking into account only the relevant information residing on remote nodes, without having to integrate the whole system, as the latter option would be highly unscalable. Below we briefly discuss how this is made visible under the KAONp2p system by singling out the main components of the architecture.

Firstly, a suitable API/User Interface in every node is used to pose the query on the node. The Query Manager, the component responsible for answering queries selects on a local level (without additional external coordination as would be the case in centralized infrastructures) all network resources that are relevant to answer that particular query. For that purpose, every local node keeps a Metadata Repository containing metadata information about the available local or distributed resources. These metadata can be descriptive, provenance, dependency, and statistical. The selecting algorithm matches the subject of the query against the subject found in the descriptive metadata. For describing the system nodes themselves as long as the established mappings among them is used. The outcome of the resource selection phase is a virtual ontology that logically integrates the relevant heterogeneous ontologies, which are connected through their mappings.

Finally, a local Reasoning Engine in every node takes care of the query evaluation against the virtual integrated ontology, more specifically the KAON2 reasoner. It is important to note that the distributed ontologies making the virtual integrated ontology do not need to reside locally, but can be materialized in a distributed way. In order to compute the equivalent disjunctive datalog program, we only need to physically integrate the T-Box parts of the different ontologies along with their mappings, which are usually much smaller than the ABoxes and pose therefore negligible computational burden. As far as the ABox parts are concerned, only the extensions from relevant predicates appearing in the datalog program have to be accessed.

Without entering into more technical and experimental details, we only mention that in such a setting is can be experimentally shown that the time cost of query answering is by and large due to the data size, whereas the extent of distribution and heterogeneity only slightly affects the computational cost. This rather proves the success of the underlying query answering technique, which allows for a performance similar to one homogenous node infrastructures.

3.8 Conclusion

During the last decade the semantic web has been constantly evolving coming to some extent closer to the semantic web vision. A number of W3C recommendations are now available, while the OWL recommendation has very recently been proceeded by the extended OWL 2 recommendation. On the other hand, more and more applications using semantic web standards (mainly RDF/RDFS) are emerging, especially around the areas of semantic search engines, travel planning applications, electronic commerce and knowledge

Approach \ Dimension	*Language*	*Relationships*	*Reasoning Task*	*Reasoning Paradigm*	*Algorithmic Features*
DDLs	$SHIQ$	Bridge Rules	Subsumption	Distributed Tableau Calculus	$SHIQ$ Complexity
$\mathcal{E}$-Connections	$SHIF(D)$	Link Constructors	Subsumption	Combined Tableau Calculus	no less than $SHIF(D)$ Complexity
PDLs	$ALCP_C$	Import of Foreign Terms	Subsumption	Message-Based Distributed Tableau	ALC Complexity
SomeWhere	Propositional DLs	Inclusion Axioms over Atomic Classes	Propositional Query Answering	Message-Based Consequence Finding Algorithm	PTIME Data Complexity
DL-Safe Mappings	$SHIQ(D)$	DL-Safe Mappings	DL-Safe Conjunctive Query answering	Disjunctive Datalog	NP-Complete Data Complexity

Table 3.1: Comparison of the described approaches.

management, while the semantic web community is actively pushing towards this direction through a number of initiatives, like the Billion Triple Challenge.

All these steps point to a relative maturity of the semantic web field and a big change on the impact of the semantic web technologies on the application-level. However, as long as the issue of scalability to very large schema and (mainly) data volumes is unresolved, the practical feasibility of a complete or even partial fulfilment of the semantic web vision cannot be guaranteed. It is thus sine qua non that the research community focuses on the problem of data distribution and distributed reasoning and comes up with novel ideas in the direction of system scalability. This is straightly analogous to the problems faced and to a large extent resolved by the database community in the previous decades.

In this chapter an overview of the state of the art in distributed reasoning was presented and compared along a number of different dimensions. We discussed a number of different approaches for distributed reasoning by giving a relative emphasis on the main technical details of the underlying formalisms for the interested reader. Most of this has already been discussed at the introduction and will not be repeated here. We summarize some basic dimensions in Table 3.1. It is important to mention that independent of the specific technique, two features are always present:

- *Relationships*: Since we consider distributed, heterogeneous data sources (data or schemata) a number of relationships between the different sources has to be established that shows how the different sources are related one to another. These relationships are in essence formalisms that consist of syntax and semantics and their nature can be radically different. They are of exceptional importance in a distributed environment, since their definition has a very special impact on the semantics of the whole system and the inference procedures. Defining these formalisms is a central part of distributed systems and a lot of careful consideration has to take place prior to their definition.

- *Tradeoff between Expressiveness and Complexity*: Obviously, high expressiveness comes at the cost of high complexity. On the other hand, a basic motive behind the distributed systems has to do with performance precipitation. That implies that a balance between the two features has to be accomplished if practical feasibility is to be achieved. So far, this is no different from what is already the case with description logics formalisms. For distributed systems, it is additionally very important that the resulting system scales well, which ideally means that the complexity of the resulting system can be reduced to the complexity of the local parts. Unfortunately, this is usually the case for systems with simple structure, like acyclic structures, where we have no recursion. The general problem of reasoning distributedness is of especially hard nature and has been posing grave challenges to the AI community for many decades.

Chapter 4

Metamodels for $\mathcal{E}$-Connections and Mapping Formalisms

4.1 The Extended OWL Metamodel for $\mathcal{E}$-Connections

Deliverable 1.1.2 [D1.] introduced the core of the networked ontology model, which is a MOF-based metamodel. For this purpose, the core modeling features provided by MOF were used, while for stating the metamodel even more precisely, we augmented it with OCL-constraints, which specify invariants that have to be fulfilled by all models that instantiate the metamodel.

In general, a metamodel for an ontology language can be derived from the modeling primitives offered by the language. The metamodel for OWL ontologies on Deliverable 1.1.2 had a one-to-one mapping to the functional-style syntax of OWL 1.1 and thereby to its formal semantics.Along with the explanation of the various OWL constructs, it introduced and discussed the corresponding metaclasses, their properties and their constraints. In order to simplify the understanding of the metamodel, we added accompanying UML diagrams to our discussion.

On the other hand, in the context of the WP7, we have developed a formalism for safe conjunctive query answering in $\mathcal{E}$-Connections of $\mathcal{EL}^{++}$Knowledge Bases. Since $\mathcal{E}$-Connections play a prominent role in that case and given that Deliverable 1.1.2 does not look into the networked ontology model of $\mathcal{E}$-Connected ontologies, we consider it appropriate to devote this Chapter into presenting the core metamodel along with the accompanying UML diagrams for the $\mathcal{E}$-Connections metamodel.

This chapter extends the MOF-based metamodel for OWL 1.1 with $\mathcal{E}$-Connections, based on the functional-style syntax discussed in [GPS06]. Instead of presenting the whole metamodel, we prefer to focus exclusively on the E-Connections extension. The Chapter is structured in six sections: Section 4.1.1 provides the general motivation of the $\mathcal{E}$-Connections, Section 4.1.2 starts a general discussion on the subject of link properties and $\mathcal{E}$-Connections, after which Section 4.1.3 presents the link property entity. Next, Section 4.1.4 demonstrates class descriptions based on link properties and Section 4.1.5 presents link property axioms. Finally, Section 4.1.6 presents assertions.

4.1.1 Motivation of $\mathcal{E}$-Connections

Effective knowledge reuse and sharing has been a major goal of the Web Ontology Working Group. Indeed, as ontologies get larger, they become harder for reasoners to process and for humans to understand. Ontology reuse can in this case prove very useful, as it allows one to take into account existing knowledge and link it or integrate it into their knowledge base, without having to construct a large ontology from scratch. The intuition behind this is that if we think of ontologies as terminological and assertional descriptions of a certain domain, then we could split the broader ontology into different smaller ontologies that are independent and self-contained modules and that describe some more specific domain of our application.

Of course, ontology reuse requires that suitable tools should be provided for integrating and connecting

the modular ontologies. The Web Ontology Language (OWL) provides some support for integrating web ontologies by defining the $owl : imports$ construct, which allows one to include by reference in a knowledge base the axioms contained in another ontology, published somewhere on the Web and identified by a global name (a URI). However, this construct is not adequate enough. The most fundamental problem lies in the fact that it brings into the original ontology all the axioms of the imported one. This prohibits partial reuse and it results in exclusively syntactical modularity, as opposed to the logical modularity.

One approach that has been thoroughly studied in the last years is that of $\mathcal{E}$-Connections [KLWZ04]. The $\mathcal{E}$-Connections formalism provides a general framework for combining logical languages that are expressible as Abstract Description Systems (ADS). The main motivation and contribution is that it combines decidable logics in such a way that the resulting combined formalism remains decidable, although that can potentially come at the cost of higher worst-case complexity.

Based on this theoretical framework, the authors in [GPS06] present a syntactical and semantic extension of OWL that covers $\mathcal{E}$-Connections of OWL-DL ontologies, showing how such an extension can be used to achieve modular ontology development on the Semantic Web and how $\mathcal{E}$-Connections provide a suitable framework for integration of Web Ontologies.

4.1.2 Preliminaries

In this section, we extend the definitions of the networked ontology model from Chapter 4 to take into account the $\mathcal{E}$-Connections formalism.

We start by taking a closer look at the $\mathcal{E}$-Connections. We first define the *Link Property*, which is the building stone of the $\mathcal{E}$-Connection.

Definition 24 *A set of* Link Properties ϵ_{AB} *between two ontologies A and B with disjoint domains is a set of special relations between them, which relate the domain of ontology A to the domain of ontology B. A Link be conveniently conceived as a superrole relating elements of the two domains, as opposed to object properties, which relate elements of the same interpretation domain.*

Based on the notion of the *Link Properties* we can build the *combined knowledge base* or $\mathcal{E}$-Connection of the two ontologies A and B, which talks about the domains of A and B and in addition to that about the relationships between them. [GPS04b] offers a closer look at the syntax of the combined language (e.g. new available link constructors) and at the semantics by defining suitable interpretations.

4.1.3 The Link Property Entity

Entities are the fundamental building blocks of OWL 2 ontologies. OWL 2 has five entity types: data types, OWL classes[1], individuals, object properties and data properties. In the extended model, we also consider an additional entity, the *link property*, in line with what was discussed in the previous section.

OWL traditionally distinguishes two types of property expressions: object property expressions and data property expressions, represented by the respective metaclasses *ObjectProperty* and *DataProperty*. In the extended metamodel, we also introduce the *LinkProperty*.

A link property can possibly be defined as the inverse of an existing link property (see Example 1). The metamodel has an abstract superclass *LinkPropertyExpression* for both the usual link property and the link property defined as an inverse of another. Figure 4.1 gives its subclasses *LinkProperty* and *InverseLinkProperty*, respectively representing normal and inverse link properties. The same figure also represents the *LinkProperty* as a subclass of the more general *Entity*, as discussed previously. An inverse link property can be defined based on any, normal or inverse, link property, hence the association *inverseLinkProperty* from the metaclass *InverseLinkProperty* is connected to the superclass *LinkPropertyExpression*.

[1] OWL provides two classes with predefined URI and semantics: owl:Thing defines the set of all objects (top concept), whereas owl: Nothing defines the empty set of objects (bottom concept).

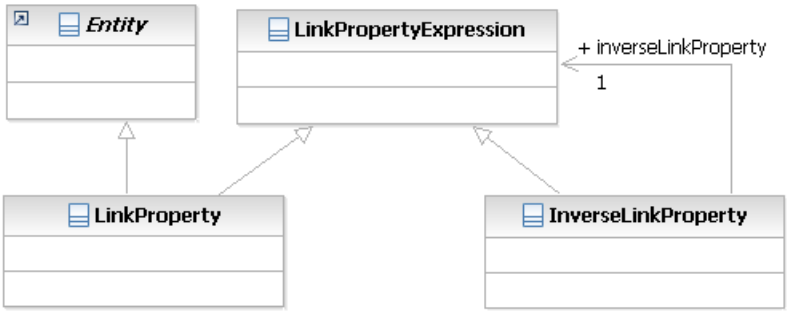

Figure 4.1: Extended metamodel: link property expressions

Example 1 *The following example illustrates the definition of an inverse object property: InverseLinkProperty(MadeFromIngredient)*

4.1.4 Class Expressions

Classes group similar resources together and are the basic building blocks of class axioms. The class extension is the set of individuals belonging to the class. To define classes, OWL 2 provides next to a simple class definition (metaclass *OWLClass*) several very expressive means for defining classes. The metamodel defines a metaclass *ClassExpression* as abstract superclass for all class definition constructs.

Link properties provide suitable constructors for building complex concept expressions, by placing different types of restrictions restrictions on link properties. This group of restrictions is actually reminiscent of the constraints placed on object properties.

Figure 4.2 gives the first group of restrictions on the property value of object properties for the context of the class.

The metaclass *LinkAllValuesFrom* represents the class construct that defines a class as the set of all objects which have only objects from a certain other class expression as value for a specific link property (see Example 2).

Example 2 *The following example illustrates the definition of a class as a subclass of another class, which is defined through a class description by restricting an object property: SubClassOf(Medication ObjectAllValuesFrom(Treats Disease))*

The OWL construct that defines a class as all objects which have at least one object from a certain class expression as value for a specific link property, is represented by the metaclass *LinkSomeValuesFrom*. Both *ObjectAllValuesFrom* and *ObjectSomeValuesFrom* have an association called *foreignClassExpression*, specifying the class description of the construct, whereas their association *linkPropertyExpression* specifies the link property on which the restriction is defined. The label *foreignClassExpression* describes the role of the association, namely the class expression has to come from the foreign ontology, with which the local ontology is related via the link property under consideration.[2] To define a class as all objects which have a certain individual as value for a specific link property, the extended model provides a construct that is represented in the metamodel by the metaclass *ObjectHasValue*, its association *linkPropertyExpression* specifying the link property, and its association *value* specifying the link property value.

The second group of link property restrictions is depicted in Figure 4.3 and it deals with restrictions on the cardinalities of link properties.

[2] This important constraint is not totally captured by the UML diagram. Although it might be somehow possible to enforce such

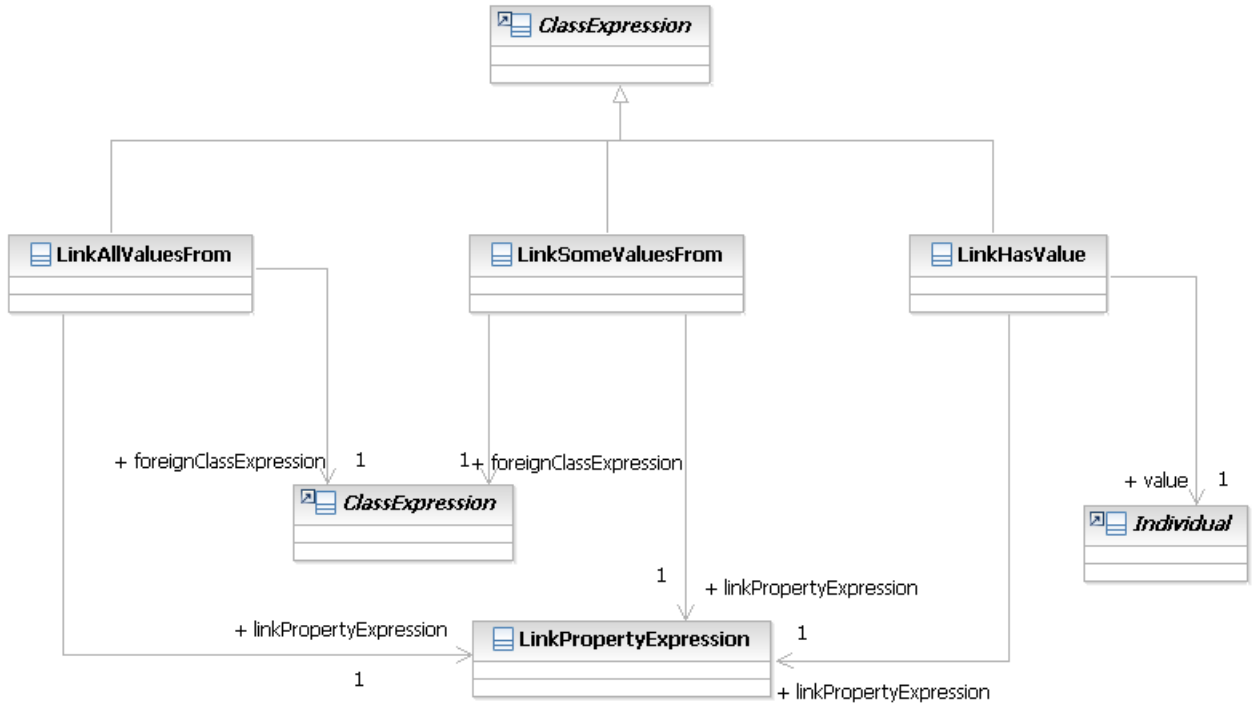

Figure 4.2: Extended metamodel: link property restrictions

A cardinality of a link property for the context of a class can be defined as a minimum, maximum or exact cardinality. The first one of this group is represented by the metaclass *LinkMinCardinality*, which defines a class of which all individuals have at least N different individuals of a certain class as values for the specified link property (N is the value of the cardinality constraint) (see Example 3).

Example 3 *The following example illustrates the definition of a class as a subclass of another class, which is defined through a class description by restricting the cardinality of an object property:*
SubClassOf(Medication LinkMinCardinality(1 madeFromIngredient Ingredient))

Secondly, the construct represented by the metaclass *LinkMaxCardinality* defines a class of which all individuals have at most N different individuals of a certain class as values for the specified link property. Finally, the construct represented by the metaclass *LinkExactCardinality* defines a class of which all individuals have exactly N different individuals of a certain class as values for the specified link property. To specify the cardinality (N) of these constructs, which is a simple integer, all three metaclasses have an attribute *cardinality*. OCL constraints define that this cardinality must be a nonnegative integer[3]:

1. The cardinality must be nonnegative:
   context LinkExactCardinality inv:
   self.cardinality >= 0

2. The cardinality must be nonnegative:
   context LinkMaxCardinality inv:
   self.cardinality >= 0

restrictions by defining suitable OCL constraints, it would be cumbersome and thus it is preferable that tools that implement the metamodel support these constraints.

[3]Note that it would make sense to define a constraint which specifies that when a minimum and a maximum cardinality on a link property are combined to define a class, the minimum cardinality should be less than or equal to the maximum cardinality. However, as the extended do not define this restriction and rely on applications to handle this, we also do not define an OCL constraint in the metamodel.

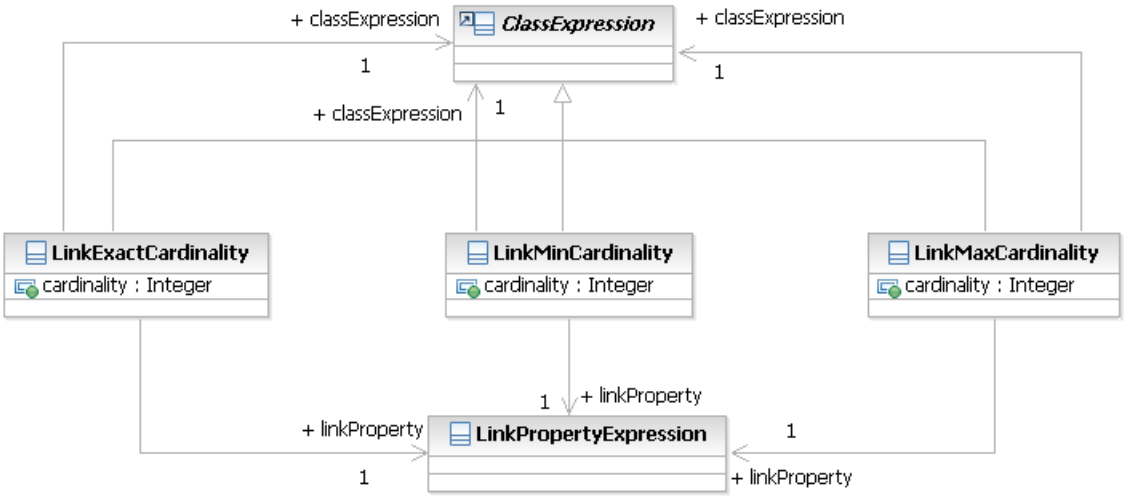

Figure 4.3: Extended metamodel: link property cardinality restrictions

3. The cardinality must be nonnegative:
   context LinkMinCardinality inv:
   self.cardinality >= 0

Additionally, they all have an association *class* and an association *linkProperty* representing the class respectively the object property involved in the statement. Note that the multiplicity of the associations called *class* have multiplicity 'zero or one' as the $\mathcal{E}$-Connections syntax does not define the explicit specification of the restricting class description as mandatory[4].

4.1.5 Link Property Axioms

OWL defines six different kinds of axioms: entity annotations, declarations, class axioms, object property axioms, data property axioms and assertions. In the extended $\mathcal{E}$-Connection model we consider in addition link property axioms.

The group of link property axioms contains constructs defining relations between different link properties, definitions of domain and range and specifications of property characteristics. We first introduce the various relations between link properties in Figure 4.4.

The first one, represented by the metaclass *SubLinkPropertyOf*, defines that the property extension of one link property, specified through the association *subLinkPropertyExpression*, is a subset of the extension of another link property, specified through the association *superLinkPropertyExpression*. The metaclass *EquivalentLinkProperty* represents a construct that defines that the property extensions of two or more link properties are the same.

To define the class to which the subjects of a link property belong, the extended model provides the domain concept (see Example 4), whereas a range specifies the class to which the objects of the property, the property values, belong. Figure 4.5 presents the metamodel elements for the constructs defining a link property domain and range.

[4]In the case where it is not explicitly defined, called an unqualified cardinality restriction, the description is owl:Thing. That is similar as with the

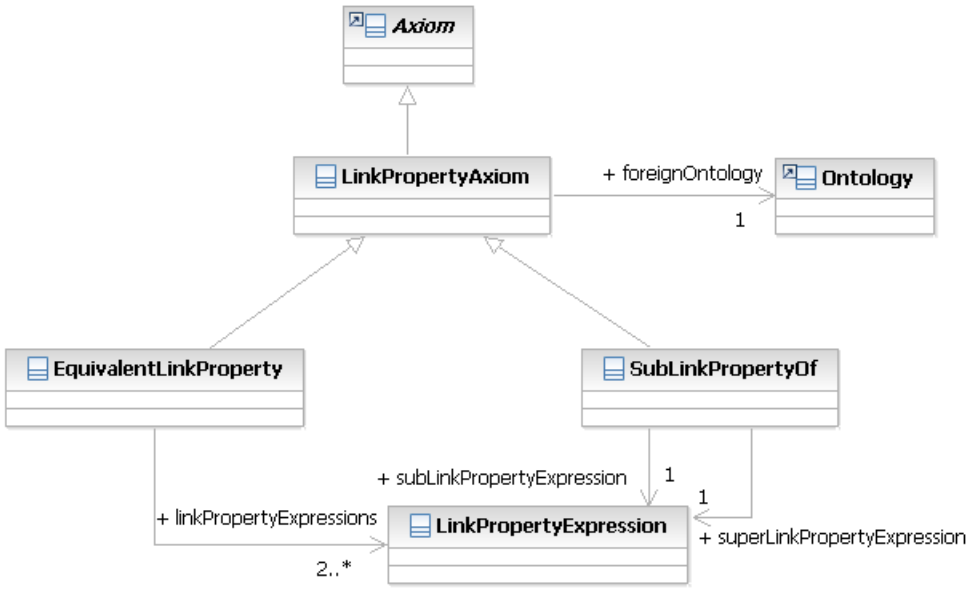

Figure 4.4: Extended metamodel: link property axioms - part 1

Example 4 *The following example illustrates the definition of the domain of an object property: LinkPropertyDomain(madeFromIngredient Medication)*

The metaclass *LinkPropertyDomain* specifies a link property and its domain via the associations *property* respectively *domain*. Similarly, the metaclass *LinkPropertyRange* represents the construct to define the range of a link property.

The remaining OWL link property axioms take a link property and assert a characteristic to it. In doing so, link properties can be defined to be functional or inverse functional.

A functional link property is a property for which each subject from the domain, can have only one value in the range, whereas an inverse functional link property can have only one subject in the domain for each value in the range.

Figure 4.6 demonstrates that each of these two axioms has an own metaclass with an association *linkProperty* to the class *LinkPropertyExpression*, specifying the property on which the characteristic is defined.

4.1.6 Assertions

In addition to the existing OWL assertions, the extended model defines a further assertion axiom to state assertions about link properties.

To specify the value of a specified individual under a certain link property, the extended model provides the link property assertion construct (see Example 5).

Example 5 *The following example illustrates how the value of an individual under a link property, is defined: LinkPropertyAssertion(madeFromIngredient aspirin acetylsalicylicacid)*

Figure 4.7 depicts the construct in the metamodel as the metaclass *LinkPropertyAssertion*. It has an association to *property* representing the involved link property, and two associations to *Individual* representing the subject and the object of the assertion.

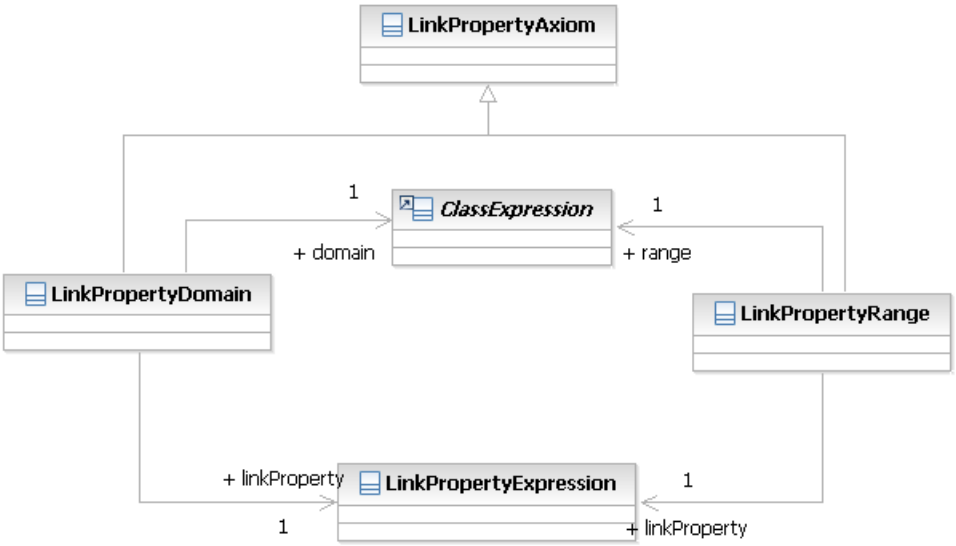

Figure 4.5: Extended metamodel: link property axioms - part 3

4.2 Mapping Support (OWL)

When people are modeling the same domain, they mostly produce different results, even when then use the same language. Mappings have to be defined between these ontologies to achieve an interoperation between applications or data relying on these ontologies. This chapter provides an extension for the metamodel for OWL and SWRL to give additional support for mappings between heterogeneous ontologies.

Section 4.2.1 introduces the metamodel extension in the same way as we did in the introduction of the metamodel for OWL and SWRL, and for F-Logic. While introducing the various mapping aspects[5], we discuss their representation in the metamodel. Accompanying UML diagrams document the understanding of the metamodel.[6]

The metamodel is a common metamodel for the different OWL mapping languages. On top of this common metamodel, we define two sets of constraints to concretize it to two specific OWL ontology mapping languages, DL-safe mappings and C-OWL. Section 4.2.2 starting on page 56 presents the extension for C-OWL mappings, consisting of a set of constraints. Similarly, Section 4.2.3 starting on page 57 presents the extension for DL-Safe Mappings.

4.2.1 A Common MOF-based Metamodel Extension for OWL Ontology Mappings

This section presents the common metamodel extension for OWL ontology mappings in two subsections: Section 4.2.1 presents mappings, after which Section 4.2.1 presents queries.

[5] Remember, however, that the OWL ontology mapping languages and their general aspects, are not part of our contribution.
[6] In doing so, meta-classes that are colored or carry a little icon again denote elements from the metamodel for OWL or SWRL.

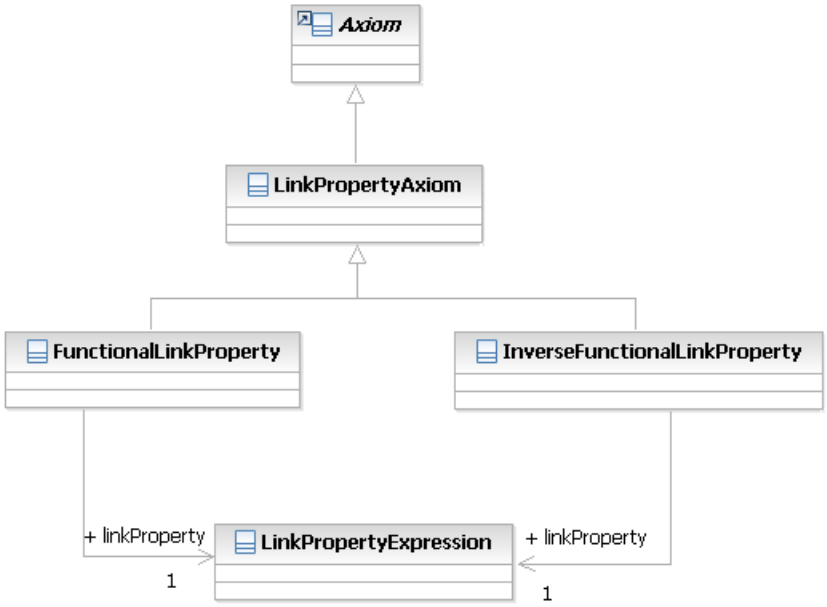

Figure 4.6: Extended metamodel: link property axioms - part 2

Mappings

We use a mapping architecture that has the greatest level of generality in the sense that other architectures can be simulated. In particular, we make the following choices:

- A mapping is a set of mapping assertions that consist of a semantic relation between mappable elements in different ontologies. Figure 4.8 demonstrates how this structure is represented in the metamodel by the five metaclasses *Mapping*, *MappingAssertion*, *Ontology*, *SemanticRelation* and *MappableElement* and their associations.
- Mappings are first-class objects that exist independent of the ontologies. Mappings are directed, and there can be more than one mapping between two ontologies. The direction of a mapping is defined through the associations *sourceOntology* and *targetOntology* of the metaclass *Mapping*, as the mapping is defined from the source to the target ontology. The cardinalities on both associations denote that to each *Mapping* instantiation, there is exactly one *Ontology* connected as source and one as target.

These choices leave us with a lot of freedom for defining and using mappings. For each pair of ontologies, several mappings can be defined or, in case of approaches that see mappings as parts of an ontology, only one single mapping can be defined. Bi-directional mappings can be described in terms of two directed mappings.

The central class in the mapping metamodel, the class *Mapping*, is given four attributes. For the assumptions about the domain, the metamodel defines an attribute *DomainAssumption*. This attribute may take specific values that describe the relationship between the connected domains: *overlap*, *containment* (in either direction) or *equivalence*.

The question of what is preserved by a mapping is tightly connected to the hidden assumptions made by different mapping formalisms. A number of important assumptions that influence this aspect have been identified and formalized in [SSW05]. A first basic distinction concerns the relationship between the sets of objects (domains) described by the mapped ontologies. Generally, we can distinguish between a global domain and local domain assumption:

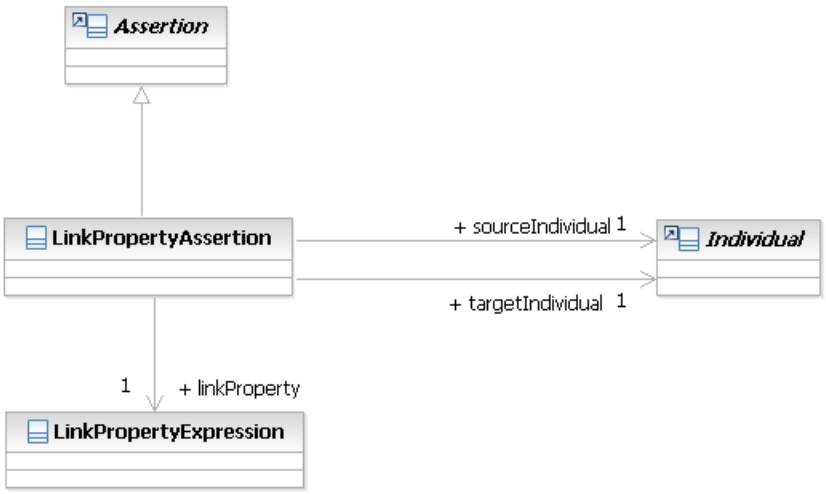

Figure 4.7: Extended metamodel: assertions

Global Domain assumes that both ontologies describe exactly the same set of objects. As a result, semantic relations are interpreted in the same way as axioms in the ontologies. This domain assumption is referred to as *equivalence*, whereas there are special cases of this assumption, where one ontology is regarded as a global schema and describes the set of all objects, other ontologies are assumed to describe subsets of these objects. Such domain assumption is called *containment*.

Local Domains do not assume that ontologies describe the same set of objects. This means that mappings and ontology axioms normally have different semantics. There are variations of this assumption in the sense that sometimes it is assumed that the sets of objects are completely disjoint and sometimes they are assumed to overlap each other, represented by the domain assumption called *Overlap*.

These assumptions about the relationship between the domains are especially important for extensional mapping definitions, because in cases where two ontologies do not talk about the same set of instances, the extensional interpretation of a mapping is problematic as classes that are meant to represent the same aspect of the world can have disjoint extensions.

The second attribute of the metaclass *Mapping* is called *inconsistencyPropagation*, and specifies whether the mapping propagates inconsistencies across mapped ontologies. *uniqueNameAssumption*, the third attribute of the metaclass *Mapping*, specifies whether the mappings are assumed to use unique names for objects, an assumption which is often made in the area of database integration. The fourth attribute, *URI*, is an optional URI which allows to uniquely identify a mapping and refer to it as a first-class object.

The set of mapping assertions of a mapping is denoted by the relationship between the two classes *Mapping* and *MappingAssertion*. The elements that are mapped in a *MappingAssertion* are defined by the class *MappableElement*. A *MappingAssertion* is defined through exactly one *SemanticRelation*, one source *MappableElement* and one target *MappableElement*. This is defined through the three associations starting from *MappingAssertion* and their cardinalities.

A number of different kinds of semantic relations have been proposed for mapping assertions and are represented as subclasses of the abstract superclass *SemanticRelation*:

Equivalence (≡) Equivalence, represented by the metaclass *Equivalence*, states that the connected elements represent the same aspect of the real world according to some equivalence criteria. A very

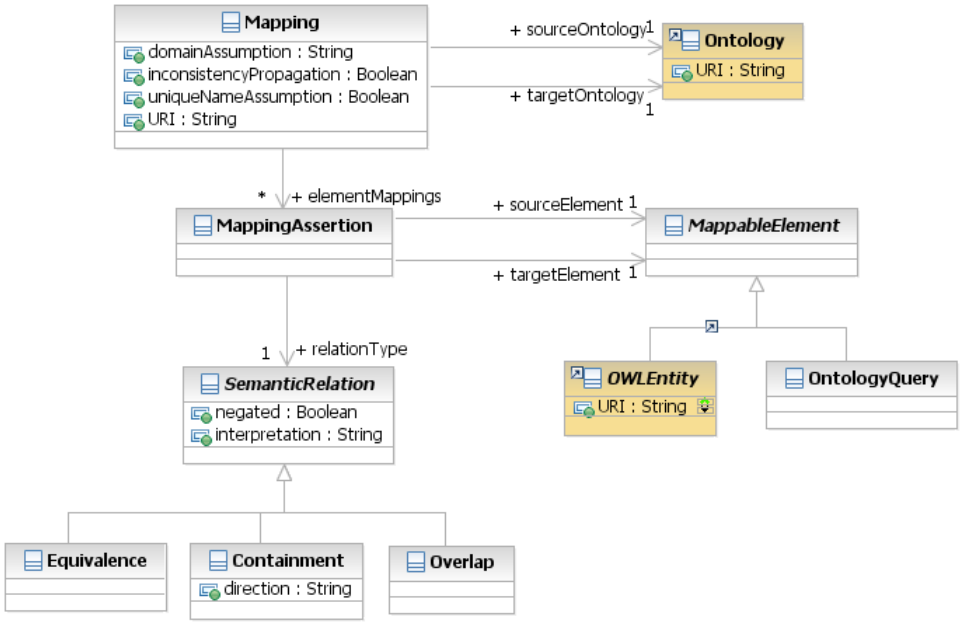

Figure 4.8: OWL mapping metamodel: mappings

strong form of equivalence is equality, if the connected elements represent exactly the same real world object. Specific forms of the equivalence relation are to be defined as subclasses of *Equivalence* in the specific metamodels of the concrete mapping formalisms.

Containment ($\sqsubseteq, \sqsupseteq$) Containment, represented by the metaclass *Containment*, states that the element in one ontology represents a more specific aspect of the world than the element in the other ontology. Depending on which of the elements is more specific, the containment relation is defined in the one or in the other direction. This direction is specified in the metamodel by the attribute *direction*, which can be *sound* ($\sqsubseteq$) or *complete* ($\sqsupseteq$). If this attribute value is *sound*, the source element is more specific element than the target element. In case of the attribute value *complete*, it is the other way around, thus the target element is more specific than the source element.

Overlap (o) Overlap, represented by the metaclass *Overlap*, states that the connected elements represent different aspects of the world, but have an overlap in some respect. In particular, it states that some objects described by the element in the one ontology may also be described by the connected element in the other ontology.

In some approaches, these basic relations are supplemented by their negative counterparts, for which the metamodel provides an attribute *negated* for the abstract superclass *SemanticRelation*. For example, a negated *Overlap* relation specifies the disjointness of two elements. The corresponding relations can be used to describe that two elements are *not* equivalent ($\not\equiv$), *not* contained in each other ($\not\sqsubseteq$) or *not* overlapping or disjoint respectively ($\emptyset$). Adding these negative versions of the relations leaves us with eight semantic relations that cover all existing proposals for mapping languages.

In addition to the type of semantic relation, an important distinction is whether the mappings are to be interpreted as extensional or as intensional relationships, specified through the attribute *interpretation* of the metaclass *SemanticRelation*.

Extensional The extension of a concept consists of the things which fall under the concept. In extensional mapping definitions, the semantic relations are interpreted as set-relations between the sets of objects represented by elements in the ontologies. Intuitively, elements that are extensionally the same have to represent the same set of objects.

Intensional The intension of a concept consists of the qualities or properties which go to make up the concept. In the case of intensional mappings, the semantic relations relate the concepts directly, i.e. considering the properties of the concept itself. In particular, if two concepts are intensionally the same, they refer to exactly the same real world concept.

As mappable elements, the metamodel contains the class *OWLEntity* that represents an arbitrary part of an ontology specification. While this already covers many of the existing mapping approaches, there are a number of proposals for mapping languages that rely on the idea of view-based mappings and use semantic relations between (conjunctive) queries to connect models, which leads to a considerably increased expressiveness. These queries are represented by the metaclass *OntologyQuery*. Note that the metamodel in principle supports all semantic relations for all mappable elements. OCL constraints for specific mapping formalisms can restrict the combinations of semantic relations and mappable elements.

Queries

A mapping assertion can take a query as mappable element. Figure 4.9 demonstrates the class *Query* that reuses constructs from the SWRL metamodel.

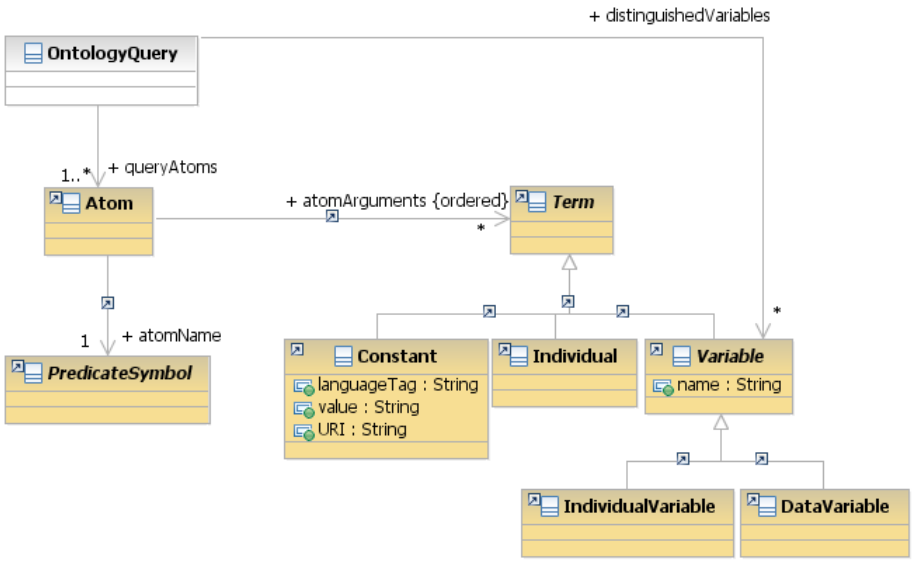

Figure 4.9: OWL mapping metamodel: queries

We reuse large parts of the rule metamodel as conceptual rules and queries are of very similar nature [TF05]: A rule consists of a rule body (antecedent) and rule head (consequent), both of which are conjunctions of logical atoms. A query can be considered as a special kind of rule with an empty head. The distinguished variables specify the variables that are returned by the query. Informally, the answer to a query consists of all variable bindings for which the grounded rule body is logically implied by the ontology. A *Query* atom also contains a *PredicateSymbol* and some, possibly just one, *Term*s. In the SWRL metamodel, we defined the permitted predicate symbols through the subclasses *Description*, *DataRange*, *DataProperty*, *ObjectProperty* and *BuiltIn*. Similarly, the different types of terms, *Individual*, *Constant*, *IndividualVariable* and *DataVariable* are specified as subclasses of *Term*. Distinguished variables of a query are differentiated through an association between *Query* and *Variable*. An OCL constraint a restriction on the use of distinguished variables:

1. A variable can only be a distinguished variable of a query if it is a term of one of the atoms of the query:
   self.distinguishedVariables→forAll(v: Variable |
   self.queryAtoms→exists(a: Atom | a.atomArguments→exists(v | true)))

4.2.2 OCL Constraints for C-OWL

We define OCL constraints on the common mapping metamodel extension to concretize it according to the specific formalism C-OWL [BGvH+03]. We list the specific characteristics of C-OWL and introduce the necessary constraints for them. For each constraint, firstly the context of the constraint, so the class in the metamodel on which it is to be defined, is defined using the OCL syntax "context *classname* inv:"[7]. Some existing reasoners support only a subset of C-OWL. Additional constraints could be defined to support this.

1. C-OWL does not have unique name assumption. To reflect this in the metamodel, a constraint defines that the value of the attribute *uniqueNameAssumption* of the class *Mapping* is always 'false':
   context Mapping inv:
   self.uniqueNameAssumption = false

2. C-OWL does not have inconsistency propagation. Similary, a constraint is defined to set the value of the attribute *inconsistencyPropagation* of the class *Mapping* to 'false':
   context Mapping inv:
   self.inconsistencyPropagation = false

3. The relationship between the connected domains of a mapping in C-OWL is always assumed to be *overlap*. A constraint sets the value of the attribute *domainAssumption* of the class *Mapping* to 'overlap':
   context Mapping inv:
   self.domainAssumption = 'overlap'

4. C-OWL does not allow to define mappings between queries. Moreover, only object properties, classes and individuals are allowed as mappable elements. A constraint defines that any mappable element in a mapping must be an *ObjectProperty*, an *OWLClass* or an *Individual*:
   context MappableElement inv:
   self.oclIsTypeOf(ObjectProperty) or
   self.oclIsTypeOf(OWLClass) or
   self.oclIsTypeOf(Individual)

5. A mapping assertion can only be defined between elements of the same kind. As the previous constraint defines that mappings can only be defined between three specific types of elements, an additional constraint can easily define that when the source element is one of these specific types, then the target elements must be of that type as well:
   context MappingAssertion inv:
   (self.sourceElement.oclIsTypeOf(OWLClass) implies
   self.targetElement.oclIsTypeOf(OWLClass)) and
   (self.sourceElement.oclIsTypeOf(ObjectProperty) implies
   self.targetElement.oclIsTypeOf(ObjectProperty)) and
   (self.sourceElement.oclIsTypeOf(Individual) implies
   self.targetElement.oclIsTypeOf(Individual))

6. As semantic relations in mappings, C-OWL supports *equivalence*, *containment* (sound as well as complete), *overlap* and *negated overlap* (called *disjoint*). A constraint defines this by specifying which different subclasses of *SemanticRelation* are allowed. When only the non-negated version of the semantic relation is allowed, the constraint defines that the attribute *negated* of the class *SemanticRelation* is set to 'false':
   context SemanticRelation inv:
   (self.oclIsTypeOf (Equivalence) and self.negated = false) or

[7] where 'inv' stands for the constraint type *invariant*.

(self.oclIsTypeOf(Containment) and self.negated = false) or
self.oclIsTypeOf(Overlap)

4.2.3 OCL Constraints for DL-Safe Mappings

This section provides OCL constraints on the common metamodel for OWL ontology mappings to concretize it to the formalism DL-Safe Mappings [HM05]. Again, we highlight the specific characteristics of the language and provide the appropriate constraints.

1. DL-Safe Mappings do not have unique name assumption. To reflect this in the metamodel, a constraint defines that the value of the attribute *uniqueNameAssumption* of the class *Mapping* is always 'false':
   context Mapping inv:
   self.uniqueNameAssumption = false

2. DL-Safe Mappings always have inconsistency propagation. A constraint is defined to set the value of the attribute *inconsistencyPropagation* of the class *Mapping* to 'true':
   context Mapping inv:
   self.inconsistencyPropagation = true

3. The relationship between the connected domains of a DL-Safe Mapping is always assumed to be *equivalence*. A constraint sets the value of the attribute *domainAssumption* of the class *Mapping* to 'equivalence':
   context Mapping inv:
   self.domainAssumption = 'equivalence'

4. DL-Safe Mappings support mappings between queries, properties, classes, individuals and datatypes. A constraint defines that the type of a *MappableElement* must be one of these subclasses:
   context MappableElement inv:
   self.oclIsTypeOf(OntologyQuery) or
   self.oclIsTypeOf(ObjectProperty) or
   self.oclIsTypeOf(DataProperty) or
   self.oclIsTypeOf(OWLClass) or
   self.oclIsTypeOf(Individual) or
   self.oclIsTypeOf(Datatype)

5. In DL-Safe Mappings, elements being mapped to each other must be of the same kind. Thus, when one wants to map for instance a concept to a query, the concept should be modelled as a query. A constraint defines that when the source element is of a specific type, the target element must be of the same type:
   context MappingAssertion inv:
   (self.sourceElement.oclIsTypeOf(OntologyQuery) implies
   self.targetElement.oclIsTypeOf(OntologyQuery)) and
   (self.sourceElement.oclIsTypeOf(ObjectProperty) implies
   self.targetElement.oclIsTypeOf(ObjectProperty)) and
   (self.sourceElement.oclIsTypeOf(DataProperty) implies
   self.targetElement.oclIsTypeOf(DataProperty)) and
   (self.sourceElement.oclIsTypeOf(OWLClass) implies
   self.targetElement.oclIsTypeOf(OWLClass)) and
   (self.sourceElement.oclIsTypeOf(Individual) implies
   self.targetElement.oclIsTypeOf(Individual)) and
   (self.sourceElement.oclIsTypeOf(Datatype) implies
   self.targetElement.oclIsTypeOf(Datatype))

6. DL-Safe Mappings specify that queries that are mapped to each other, must contain the same distinguished variables. A constraint defines that when both elements are a query, each variable that exists as distinguished variable in the source element, must exist as distinguished variable in the target element:
   context MappingAssertion inv:
   self.sourceElement.oclIsTypeOf(Query) and
   self.targetElement.oclIsTypeOf(Query) implies
   self.sourceElement.oclAsType(Query).distinguishedVariables→
   forAll(v: Variable | self.targetElement.oclAsType(Query).distinguishedVariables→
   exists(v | true))

7. The interpretation of semantic relations in DL-Safe Mappings is always assumed to be extensional. A constraint defines that the value of the attribute *interpretation* of the class *SemanticRelation* must be set to 'extensional':
   context SemanticRelation inv:
   self.interpretation = 'extensional'

8. DL-Safe Mappings support the semantic relations *equivalence* and *containment* (sound as well as complete). A constraint specifies this by defining which types *SemanticRelation* can have and what its value for the attribute *negated* should be:
   context SemanticRelation inv:
   (self.oclIsTypeOf(Equivalence) and self.negated = false) or
   (self.oclIsTypeOf(Containment) and self.negated = false)

Chapter 5

An integration of $\mathcal{EL}^{++}$ with $\mathcal{E}$-Connections for Safe Conjunctive Query Answering

Effective knowledge reuse and sharing has been a major goal of the Web Ontology Working Group. Indeed, as ontologies get larger, they become harder for reasoners to process and for humans to understand. Ontology reuse can in this case prove very useful, as it allows one to take into account existing knowledge and link it or integrate it into their knowledge base, without having to construct a large ontology from scratch. The intuition behind this is that if we think of ontologies as terminological and assertional descriptions of a certain domain, then we could split the broader ontology into different smaller ontologies that are independent and self-contained modules and that describe some more specific domain of our application.

Of course, ontology reuse requires that suitable tools should be provided for integrating and connecting the modular ontologies. The Web Ontology Language (OWL) provides some support for integrating web ontologies by defining the $owl : imports$ construct, which allows one to include by reference in a knowledge base the axioms contained in another ontology, published somewhere on the Web and identified by a global name (a URI). However, this construct is not adequate enough. The most fundamental problem lies in the fact that it brings into the original ontology all the axioms of the imported one. This prohibits partial reuse and it results in exclusively syntactical modularity, as opposed to the logical modularity.

One approach that has been thoroughly studied in the last years is that of $\mathcal{E}$-Connections [KLWZ04]. The $\mathcal{E}$-Connections formalism provides a general framework for combining logical languages that are expressible as Abstract Description Systems (ADS). The main motivation and contribution is that it combines decidable logics in such a way that the resulting combined formalism remains decidable, although that can potentially come at the cost of higher worst-case complexity.

$\mathcal{E}$-Connections in expressive fragments of Description Logics have already been investigated and special extensions of the tableau calculus have been proposed for the reasoning task of concept satisfiability [GPS04a]. Despite their correctness, their high computational complexity (at least exponential) places an undesired restriction to their use in practice. In addition to that, while the subsumption task can be proved very useful for classifying large ontologies, in the case of ontologies with large ABoxes one is usually much more interested in query answering, as is also the case with database systems.

To overcome these obstacles, we propose in this Chapter a suitable restriction of the existing theories, which consists in two directions. First, we do not examine the general problem of $\mathcal{E}$-Connections between expressive Description Logics, but we instead focus on a very useful tractable fragment of Description Logics, namely the $\mathcal{EL}^{++}$ family. The selection of $\mathcal{EL}^{++}$ as a candidate tractable knowledge representation formalism is not accidental. Indeed, as demonstrated in [BBL05b], despite its relatively weak expressive power, it is however sufficient for a number of different applications. We consider that in the context of the Work Package 7 use case, namely the fisheries ontology, where the terminological knowledge is not complex, choosing a tractable Description Logic with strong expressivity in practice can reconcile in an effective our need for expressivity, on the one hand, and for low complexity, on the other.

Second, despite the usefulness of subsumption and satisfiability in traditional reasoning tasks, such as clas-

sification, we choose to concentrate on the task of conjunctive query answering. This is especially useful for knowledge bases with large data volumes, where we are interested to ask information about the data. Conjunctive queries are a well-studied family of queries that can capture the user needs in many practical applications and this is the primary motivation behind our interest in them and are related to Horn-clause logics and rules. Unfortunately, because of the open-world assumption made in knowledge representation formalisms, query answering is not even obvious to define, as for instance is the case with database systems, where the close-world assumption is adopted. Moreover, combining OWL-DL with rules leads to undecidability (a very strong intuition why that happens is provided by Motik in [MSS05]). To surmount this difficulty, we do not consider general conjunctive queries, but only queries with DL-safe variables, as discussed in [HM05].
The current Chapter is structured as follows: we first discuss the tractable family of $\mathcal{EL}^{++}$; we then define $\mathcal{E}$-Connections between $\mathcal{EL}^{++}$ components and we prove that the resulting system is decidable and, moreover, that it can be translated into an equivalent $\mathcal{EL}^{++}$ knowledge base, thus showing that it retains its tractability; in the end, we make use of a recent work that translates ELP knowledge bases into equisatisfiable Datalog programs, which we use for DL-Safe conjunctive query answering.

5.1 The Description Logic $\mathcal{EL}^{++}$

In Description Logics, a set of atomic concepts N_C, a set of atomic roles N_R and a set of individual names N_I serve as the building blocks to construct more complex concepts. N_C, N_R and N_I are always considered (infinitely)countable and pairwise disjoint. The concept constructors that are available in $\mathcal{EL}^{++}$ are the top concept, the bottom concept, concept conjunction, existential restriction and nominals (concrete domains are also possible, but we don't deal with them at this paper, as they can be treated in a similar way). More formally:

Definition 25 *Let N_C, N_R, N_I be countable and pairwise disjoint sets of concept names, role names and individuals, respectively. The set of concepts is the smallest set such that:*

1. *It contains the special concepts $\top$ (Top) and $\bot$ (Bottom).*
2. *Every concept name $A \in N_C$ is a concept.*
3. *If C, D are concepts, $R \in N_R$ and $o \in N_I$, the following are also concepts:*
   - *$C \sqcap D$ (Conjunction).*
   - *{o} (Nominals).*
   - *$\exists R.C$ (Existential Restriction).*

Let C, D be concepts, then the assertion $C \sqsubseteq D$ is called a general concept inclusion axiom (GCI). Moreover, if $r_1, ..., r_n, R \in N_R$, then the assertion $r_1 \circ ... \circ r_n \sqsubseteq R$ is called a role inclusion axiom (RI). An $\mathcal{EL}^{++}$ constraint box (CBox) is a finite set of GCIs and RIs.

The semantics is defined through an interpretation as follows:

Definition 26 *An interpretation I is a pair $I = (\Delta^I, \cdot^I)$, where Δ^I is a non-empty set, called the interpretation domain, and $\cdot^I$ is the interpretation function, which maps:*

- *Each concept name $A \in N_C$ to a subset A^I of Δ^I.*
- *Each role name $R \in N_R$ to a subset R_I of $\Delta^I \times \Delta^I$.*

The interpretation function can be inductively extended to the aforementioned concept constructors as shown in Table 5.1.
An interpretation I satisfies the RI $r_1 \circ ... \circ r_n \sqsubseteq R$ iff $r_1^I \circ ... \circ r_n^I \subseteq R^I$ and it satisfies the GCI $C \sqsubseteq D$ iff $C^I \subseteq D^I$. The interpretation I is a model of the knowledge base iff it satisfies all the axioms in the CBox.

Name	Syntax	Semantics
Top	$\top$	Δ^I
Bottom	$\bot$	$\emptyset$
Conjunction	$C \sqcap D$	$C^I \cap D^I$
Existential Restriction	$\exists R.C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I, (x,y) \in R^I, y \in C^I\}$
Nominals	$\{o\}$	$\{o\}^I \in \Delta^I$

Table 5.1: Concept Constructors in $\mathcal{EL}^{++}$: Syntax and Semantics

The main reasoning task in $\mathcal{EL}^{++}$ is concept subsumption: a concept D is subsumed by concept E w.r.t. the CBox C and we write $D \sqsubseteq_C E$ iff $D^I \subseteq E^I$ for all models I of C. It can be proved that subsumption is expressive enough to express many interesting reasoning tasks (concept satisfiability, ABox consistency, instance problem) and vice versa [BBL05a].

A very useful property of any CBox C is that it can be transformed in linear time to its *normal form*, which is an equivalent form of the original CBox, in which:

- All GCIs are of the form: $C \sqsubseteq D$, $C_1 \sqsubseteq \exists R.C_2$, $C_1 \sqcap C_2 \sqsubseteq D$, or $\exists R.C \sqsubseteq D$, where $C,\ C_1,\ C_2 \in BC_C$[1] and $D \in BC_C \cup \{\bot\}$,
- All RIs are of the form: $r \sqsubseteq s$, $r_1 \circ r_2 \sqsubseteq s$, where $r_1,\ r_2,\ r,\ s$ are roles.

The following result reveals the tractable nature of subsumption in $\mathcal{EL}^{++}$ [BBL05b]:

Theorem 4 *Subsumption in $\mathcal{EL}^{++}$ w.r.t. CBoxes can be decided in polynomial time.*

5.2 $\mathcal{E}$-Connections of $\mathcal{EL}^{++}$ Components

5.2.1 Abstract Description Systems

Abstract description systems [KLWZ04] came as a result of the effort to combine different logical formalisms into a single logical formalism. Examples of the different formalisms include description logics, logics of topological spaces (e.g. the modal logic S4 extended with the universal modality), logics of metric spaces that can offer not only qualitative but also quantitative information (e.g. the family MS), and propositional temporal logics. That actually suggests that the abstract description system formalism is a very expressive formalism, able to handle quite different kinds of logics, and not restricted just to the DLs field.

The syntax is provided by the abstract description language, which determines the set of terms and assertions. We give a trimmed-down and slightly modified version of it [KLWZ04], which best fits our purpose, without any affect on the main results.

Definition 27 *An abstract description language (ADL) $\mathcal{L}$ is described by a countably infinite set $\mathcal{V}$ of set variables, a countably infinite set $\mathcal{X}$ of object variables, (possibly infinite) countable set $\mathcal{R}$ of relation symbols R of arity 2 and a countable set F of function symbols f of arity n_f, such that $\neg, \wedge \notin \mathcal{F}$. These sets are pairwise-disjoint.*
The terms of t_j of $\mathcal{L}$ are built in the following way:
$t_j ::= \top \mid \bot \mid x \mid t_1 \wedge t_2 \mid f(t_1, ..., t_{m_f})$,
where $x \in \mathcal{V}$ and $f \in \mathcal{F}$.
The term assertions of $\mathcal{L}$ are of the form $t_1 \sqsubseteq t_2$, for all terms t_1 and t_2, and the object assertions are either of the form $R(a,b)$, for $a, b \in X$ and $R \in \mathcal{R}$, or of the form $(a : t)$, for $a \in \mathcal{X}$ and t a term.

[1] The set BC_C is defined as the smallest set that contains the set N_C of concept names, the top concept $\top$ and all concept descriptions of the form $\{o\}$ appearing in C.

As assertions of the ADL we define the sets of term and object assertions.
An abstract description model (ADM) for an ADL $\mathcal{L} = (\mathcal{V}, \mathcal{X}, \mathcal{R}, \mathcal{F})$ *is a structure of the form* $\mathfrak{M} = \langle W, \mathcal{V}^{\mathfrak{M}} = (x^{\mathfrak{M}})_{x\in\mathcal{V}}, \mathcal{X}^{\mathfrak{M}} = (a^{\mathfrak{M}})_{a\in\mathcal{X}}, \mathcal{F}^{\mathfrak{M}} = (f^{\mathfrak{M}})_{f\in\mathcal{F}}, \mathcal{R}^{\mathfrak{M}} = (R^{\mathfrak{M}})_{R\in\mathcal{R}}\rangle$,
where W *is a non-empty set,* $x^{\mathfrak{M}} \subseteq W$, $a^{\mathfrak{M}} \in W$, *each* $f^{\mathfrak{M}}$ *is a function mapping* n_f*-tuples* $\langle X_1, ..., X_{n_f}\rangle$ *of subsets of* W *to a subset of* W, *and the* $R^{\mathfrak{M}}$ *are* m_R*-ary relations on* W. *The value* $t^{\mathfrak{M}} \subseteq W$ *of an* $\mathcal{L}$*-term* t *in* $\mathfrak{M}$ *is defined inductively by taking*

- $\top^{\mathfrak{M}} = W$ *and* $\bot^{\mathfrak{M}} = \emptyset$,
- $(t_1 \wedge t_2)^{\mathfrak{M}} = t_1^{\mathfrak{M}} \cap t_2^{\mathfrak{M}}$, *and*
- $(f(t_1, ..., t_{m_f}))^{\mathfrak{M}} = f^{\mathfrak{M}}(t_1^{\mathfrak{M}}, ..., t_{m_f}^{\mathfrak{M}})$.

The truth-relation $\mathfrak{M} \models \varphi$ *for an assertion* φ *is defined as follows:*

- $\mathfrak{M} \models R(a_1, ..., a_{m_R})$ *iff* $R^{\mathfrak{M}}(a_1^{\mathfrak{M}}, ..., a_{m_R}^{\mathfrak{M}})$,
- $\mathfrak{M} \models a : t$ *iff* $a^{\mathfrak{M}} \in t^{\mathfrak{M}}$,
- $\mathfrak{M} \models t_1 \sqsubseteq t_2$ *iff* $t_1^{\mathfrak{M}} \subseteq t_2^{\mathfrak{M}}$.

In this case we say that the assertion φ *is satisfied in* $\mathfrak{M}$. *For sets* Γ *of assertions, we write* $\mathfrak{M} \models \Gamma$ *if* $\mathfrak{M} \models \varphi$ *for all* $\varphi \in \Gamma$.

We now define an ADS as a pair of an ADL and a class of ADMs.

Definition 28 *An abstract description system is a pair* $(\mathcal{L}, \mathcal{M})$, *where* $\mathcal{L}$ *is an ADL and* $\mathcal{M}$ *is a class of ADMs for* $\mathcal{L}$ *that is closed under the following operations:*

- *if* $\mathfrak{M} = \langle W, \mathcal{V}^{\mathfrak{M}}, \mathcal{X}^{\mathfrak{M}}, \mathcal{F}^{\mathfrak{M}}, \mathcal{R}^{\mathfrak{M}}\rangle$ *is in* $\mathcal{M}$ *and* $\mathcal{V}^{\mathfrak{M}'} = (x^{\mathfrak{M}'})_{x\in\mathcal{V}}$ *is a new assignment of set variables in* W *then* $\mathfrak{M}' = \langle W, \mathcal{V}^{\mathfrak{M}'}, \mathcal{X}^{\mathfrak{M}}, \mathcal{F}^{\mathfrak{M}}, \mathcal{R}^{\mathfrak{M}}\rangle \in \mathcal{M}$.
- *for every finite* $\mathcal{G} \subseteq \mathcal{F}$, *there exists a finite set* $\mathcal{X}_\mathcal{G} \subseteq \mathcal{X}$ *such that, for every* $\mathfrak{M} = \langle W, \mathcal{V}^{\mathfrak{M}}, \mathcal{X}^{\mathfrak{M}}, \mathcal{F}^{\mathfrak{M}}, \mathcal{R}^{\mathfrak{M}}\rangle$ *from* $\mathcal{M}$ *and every assignment* $\mathcal{X}^{\mathfrak{M}'} = (a^{\mathfrak{M}'})_{a\in\mathcal{X}}$ *of object variables in W such that* $a^{\mathfrak{M}} = a^{\mathfrak{M}'}$ *for all* $a \in \mathcal{X}_\mathcal{G}$, *there is an interpretation* $\mathcal{F}^{\mathfrak{M}'} = (f^{\mathfrak{M}'})_{f\in\mathcal{F}}$ *of the function symbols such that* $f^{\mathfrak{M}'} = f^{\mathfrak{M}}$ *for all* $f \in \mathcal{G}$ *and* $\mathfrak{M}' = \langle W, \mathcal{V}^{\mathfrak{M}}, \mathcal{X}^{\mathfrak{M}'}, \mathcal{F}^{\mathfrak{M}'}, \mathcal{R}^{\mathfrak{M}}\rangle \in \mathcal{M}$.

What the closure condition suggests is that the set variables can be interpreted as arbitrary sets of the interpretation domain, so they are treated as variables in any ADS, while the interpretation of the function symbols remains the same. That is a property that all DLs comply with.
The main reasoning task for an ADS is the satisfiability problem for finite sets of term assertions.

Definition 29 *Let* $S = (\mathcal{L}, \mathcal{M})$ *be an ADS. A finite set* Γ *of term assertions is called satisfiable in S if there exists an ADM* $\mathfrak{M} \in \mathcal{M}$ *such that* $\mathfrak{M} \models \Gamma$.

The next proposition states the correspondence between the DL $\mathcal{EL}^{++}$ and the notion of ADSs. The detailed proof for $\mathcal{EL}^{++}$ and even more expressive description logics can be found in [KLWZ04, BLSW02].

Proposition 1 *The Description Logic* $\mathcal{EL}^{++}$ *corresponds to an ADS.*

Proof 1 *A formal proof for more expressive DLs can be found at [KLWZ04, BLSW02]. Here we only provide a simplification of the original proof suited to* $\mathcal{EL}^{++}$.
First, we translate the syntax of the $\mathcal{EL}^{++}$ *to the corresponding ADL* $\mathcal{L}$.

- *Every atomic concept $A \in N_C$ can be associated with a set variable x_A in $\mathcal{L}$.*
- *The top concept $\top$ is translated to the fixed set variable $x_\top$, while the bottom concept $\bot$ to the fixed set variable $x_\bot$.*
- *Every individual name i in $\mathcal{EL}^{++}$ can be treated as an object variable a_i in $\mathcal{L}$.*
- *The set of relation symbols of $\mathcal{L}$ is exactly the set of role name $R \in N_R$ in $\mathcal{EL}^{++}$.*
- *The set of function symbols of $\mathcal{L}$ is the smallest set, that:*
  1. *for every role $R \in \mathcal{R}$, it contains the unary function symbol $f_{\exists R}$, and*
  2. *for every o that belongs to the set of nominals, it contains a function symbol f_o of arity 0.*

Conjunction between $\mathcal{EL}^{++}$ concepts can be straightforwardly translated to conjunction between the corresponding terms in $\mathcal{L}$. Indeed, $C \sqcap D$ in $\mathcal{EL}^{++}$ is regarded as $t_C \wedge t_D$ in $\mathcal{L}$. Moreover, $\mathcal{EL}^{++}$ GCIs correspond to $\mathcal{L}$-term assertions, while object assertions correspond to ABox assertions in the $\mathcal{EL}^{++}$.
Second, we proceed with the definition of the class $\mathcal{M}$ of ADMs for the language $\mathcal{L}$ that we just described. For this purpose, for every $\mathcal{EL}^{++}$-model $I = \langle \Delta, A_1^I, ..., R_1^I, ..., a_1^I, ... \rangle$, the class $\mathcal{M}$ contains the model

$$\mathfrak{M} = \langle W, \mathcal{V}^{\mathfrak{M}}, \mathcal{X}^{\mathfrak{M}}, \mathcal{F}^{\mathfrak{M}}, \mathcal{R}^{\mathfrak{M}} \rangle,$$

where for every concept name $A \in N_A$, role name $R \in N_R$, and every object name o:

- $x_A^{\mathfrak{M}} = A^I$, $x_\top = W$, *and* $x_\bot = \emptyset$
- $o^{\mathfrak{M}} = a^I$,
- $R^{\mathfrak{M}} = R^I$,
- $f_o^{\mathfrak{M}} = \{o^{\mathfrak{M}}\}$, *and*
- $f_{\exists R}^{\mathfrak{M}}(X) = \{w \in \Delta \mid \exists u((w,u) \in R^I \wedge u \in X)\}$.

Concept interpretations can be changed arbitrarily, so closure condition (1) of Definition 28 is satisfied. Closure condition (2) is also satisfied. Indeed, consider a finite subset $\mathcal{G}$ of $\mathcal{F}$ and let $\mathcal{X}_\mathcal{G}$ denote the finite set of object variables o for which the nullary function $f_o \in \mathcal{G}$. Then, for any new assignment of the object variables in $\mathcal{X} - \mathcal{X}_\mathcal{G}$, the new interpretation of the function symbols not occurring in G consists in interpreting every nominal f_o, $a \in \mathcal{X} - \mathcal{X}_\mathcal{G}$, as the singleton set containing the object newly assigned to a, while the interpretation of the rest of the function symbols remains the same as before.
Thus the pair $(\mathcal{L}, \mathcal{M})$ indeed corresponds to an ADS.

5.2.2 $\mathcal{E}$-Connections in Abstract Description Systems

So far, we have provided the framework of ADSs which enables us to translate very different formalisms into the fore mentioned abstract formalisms. On the other hand, our main interest resides in the ability to establish general form link relations between the components of the system. This is accomplished by $\mathcal{E}$-Connections, which can be viewed as the combination of the system components.
The $\mathcal{E}$-Connection $C^{\mathcal{E}}(S_1, ..., S_n)$ of n ADSs $S_1, ..., S_n$, where $S_i = (L_i, M_i)$ with $1 \leq i \leq n$, contains a set of terms and a set of assertions both partitioned into n sets, and is interpreted by a class of models.
We define the i-terms inductively:

- every set variable of L_i is an i-term,
- the set of i-terms is closed under $\neg, \wedge$ and the function symbols of L_i,

- if $(t_1, ..., t_{i-1}, t_{i+1}, ..., t_n)$ is a sequence of k-terms t_k for $k \neq i$, then $\langle E_j \rangle^i (t_1, ..., t_{i-1}, t_{i+1}, , t_n)$ is an i-term, for every $j \in J$.

Apart from the term and object assertions that we mentioned before, a new kind of assertion, the link assertion, is now defined, offering the ability to establish connections between the various components. All of them together form the set of assertions of the $\mathcal{E}$-Connection, and a finite set of assertions is called a knowledge base of the $\mathcal{E}$-Connection. Formally, for $1 \leq i \leq n$:

- the i-term assertions are of the form $t_1 \sqsubseteq t_2$, where both t_1 and t_2 are i-terms,
- the i-object assertions are of the form $a : t$ or $R(a_1, ..., a_{m_R})$, where a and $a_1, ..., a_{m_R}$ are object variables of L_i, t is an i-term, and R is a relation symbol of L_i,
- the link assertions are of the form $(a_1, ..., a_n) : E_j$, where the a_i are object variables of L_i, $1 \leq i \leq n$, and $j \in J$.

The semantics of $C^E(S_1, ..., S_n)$ is given through the structure $\mathfrak{M} = \langle (\mathfrak{M}_i)_{i \leq n}, E^{\mathfrak{M}} = (E_j^{\mathfrak{M}})_{j \in J} \rangle$, where $\mathfrak{M}_i \in \mathcal{M}_i$ for $1 \leq i \leq n$ and $E_j^{\mathfrak{M}} \subseteq W_1 \times \cdots \times W_n$ for each $j \in J$. This structure is called a model for $C^{\mathcal{E}}(S_1, ..., S_n)$. The extension $t^{\mathfrak{M}} \subseteq W_i$ of an i-term t is defined by induction. For set variables X of L_i we put $X^{\mathfrak{M}} = X^{\mathfrak{M}_i}$, while the inductive steps for the Booleans and function symbols are the same as in Definition 27. Moreover, if $\overline{t_i} = (t_1, ..., t_{i-1}, t_{i+1}, ..., t_n)$ is a sequence of j-terms t_j, with $1 \leq j \leq n \wedge j \neq i$, then: $(\langle E_j \rangle^i (\overline{t_i}))^{\mathfrak{M}} = x \in W_i \mid \exists_{l \neq i} x_l \in t_l^{\mathfrak{M}} (x_1, ..., x_{i-1}, x, x_{i+1}, ..., x_n) \in E_j^{\mathfrak{M}}$.
The truth-relation and the satisfiability of a set of assertions is defined as in previous section. The following theorem is the fundamental transfer result proved by Kutz, Lutz, Wolter and Zakharyaschev:

Theorem 5 *Let $C^{\mathcal{E}}(S_1, ..., S_n)$ be an $\mathcal{E}$-Connection of ADSs $S_1, ..., S_n$. If the satisfiability problem for each of $S_1, ..., S_n$ is decidable, then it is decidable for $C^{\mathcal{E}}(S_1, ..., S_n)$ as well.*

From the theorem above and the proposition 1, the following corollary immediately follows:

Corollary 1 *The satisfiability problem for the $\mathcal{E}$-Connection $C^{\mathcal{E}}(S_1, ..., S_n)$ of the $\mathcal{EL}^{++}$ DLs $S_1, ..., S_n$ is decidable.*

Practical $\mathcal{E}$-Connections Instead of using n-ary $\mathcal{E}$-Connections which involve all system components, it is a common practice to consider a trimmed-down version of them, namely only binary $\mathcal{E}$-Connections [BCH06b]. Apparently, the restricted version is a subcase of the more general n-ary version, where the arguments of the n-ary $\mathcal{E}$-Connection that do not participate in the binary $\mathcal{E}$-Connection are just replaced by their respective top concepts. The simplified version is nevertheless somehow more natural, since it actually constitutes an extension of the traditional binary role to a binary "super"-role, which, contrary to the common role, connects individuals of components with disjoint domains. Throughout the paper we will only consider this simplifies version of $\mathcal{E}$-Connections.

5.3 Translating the $\mathcal{E}$-Connection of $\mathcal{EL}^{++}$ Components into an Equivalent $\mathcal{EL}^{++}$ Knowledge Base

We will follow the approach suggested in [BS03], modifying it whenever needed. This approach builds an equivalent global DL, which encodes the information that is available in the different $\mathcal{EL}^{++}$ knowledge bases and in their corresponding $\mathcal{E}$-Connections.
In this direction, let us first think of n $\mathcal{EL}^{++}$ Description Logics, namely $EL_1, ..., EL_n$. As stated before, every $\mathcal{EL}^{++}$ CBox can be written to its equivalent normal form, which contains conjunction, nominals and existential quantification as concept constructors and role composition as the sole role constructor. No

interaction exists between the various CBoxes, so the transformation process is expected to proceed as usual for each CBox, yielding to n equivalent *normal forms*. Contrary to that, let's now consider the $\mathcal{E}$-Connection $C^{\mathcal{E}}(EL_1, ..., EL_n)$, which additionally provides linking mechanisms, which have the form of an existential restriction, with the difference that the domain and range of it are now disjoint. This additional feature introduces interaction between the components, so one cannot just proceed with transforming each component into its equivalent form, since it is expected that each component is potentially dependent on other components, too.

In order to overcome this obstacle, we will try to build a new $\mathcal{EL}^{++}$ CBox corresponding to an equivalent global knowledge base KB_G, in the sense that a concept is subsumed by another concept in the $\mathcal{E}$-Connection $C^{\mathcal{E}}(EL_1, ..., EL_n)$ if and only if the subsumption holds in the global constructed CBox. Such a global knowledge base mirrors in reality an integration of the different system components, by encoding the information that is available in the $\mathcal{E}$-Connected System.

Building the knowledge base KB_G requires first specifying a language L_G, which will provide us with the ability to explicitly state which roles and concepts belong to which component (the link does not belong to one specific component, since it connects different ones). So, if t an atomic concept (or an atomic role) of component i, L_G will contain the atomic concept (or atomic role) $i : t$. The language of the global system is equipped with the same set of concept, role and link constructors as each $\mathcal{EL}^{++}$ component, therefore it can express (at least) the same complex descriptions that are allows in the individual parts. L_G can additionally express the concepts $ANYTHING$ and $NOTHING$, which refer to the global domain of the integrated system. In order to be able to express the local domain and range restrictions for concepts and roles belonging to the i-th component, we introduce besides the global $ANYTHING$ and $NOTHING$ the concepts Top_i and Bot_i that refer to the local domain of the i-th component. Out of technical reasons that will be made clear later L_G also contains a special role symbol $\hat{P}$.

After having defined the global language L_D, we proceed with (1) the translation of the concept and role descriptions in each component, and (2) the translation of the whole CBox of each component, that contains general concept inclusions and role inclusions. (1) is quite straightforward and is based on the recursive definition of the concept and role descriptions, while (2) needs the introduction of extra conditions that guarantee that the integrated system respects a number of constraints. We mention that instead of writing the translation for every single constructor, we represent every constructor as a function with k arguments and we provide a uniform translation for all of them in the spirit of [BS03].

The translation of the concept and role descriptions is as follows:

1. $\#(i, M) = i : M$ for atomic concepts and roles M, including the top and bottom concepts ANYTHING and NOTHING respectively.
2. $\#(i, \{o\}) = \{i : o\}$.
3. $\#(i, C \sqcap D) = \#(i, C) \sqcap \#(i, D)$.
4. $\#(i, \exists R.C) = \exists \#(i, R).\#(i, C)$ for existential restrictions.
5. $\#(i, \exists E.C) = \exists \#(i, E).\#(j, C)$ for link restrictions $E \in E_{ij}$.
6. $\#(i, r_1 \circ \cdots \circ r_n) = \#(i, r_1) \circ \cdots \circ \#(i, r_n)$.

The global CBox $\#(T)$ of the $\mathcal{E}$-Connection $C^{\mathcal{E}}(DL_1, ..., DL_n)$ will then consist of the following axioms:

1. *General Concept Inclusion Axioms.*

   $\#(i, A) \sqsubseteq \#(i, B)$ for all i-term assertions $A \sqsubseteq B$ in DL_i.

2. *Force the bottom concept of every component to be interpreted as the empty set.*

   $Bot_i \sqsubseteq NOTHING$

3. *Force every atomic concept (basis) to be subsumed by the top concept of the corresponding component.*

   $i : A \sqsubseteq Top_i$, for every atomic concept A of DL_i.

4. *Force every nominal to be subsumed by the top concept of the corresponding component.*

   $\{i : o\} \sqsubseteq Top_i$, for every nominal o of DL_i.

5. *Declare that the top concept of each component must be interpreted by a non-empty set. $\hat{P}$ is an auxiliary role, whose interpretation does not matter after all.*

   $ANYTHING \sqsubseteq \exists \hat{P}.Top_i$

6. (1) *Force the range of any role to be subsumed by the top concept of the corresponding component.*

   Normally, in order to express that restriction we would write $Top_i \sqsubseteq \forall(i : p).(Top_i)$ for every role p of DL_i. The universal constructor is unfortunately not in $\mathcal{EL}^{++}$. The only way to express this restriction is to introduce the constructor $range(p)$ that denotes the range of a role and then restrict it through proper inclusion axioms. While this seems quite "innocent", the interaction of general form role inclusions and range restrictions can lead to undecidability, let alone tractability [BBL08]. In our case, the role inclusions have all the very nice property that all roles that participate in a role inclusion axiom have as range the corresponding local top concept and according to [BBL08] this syntactical restriction of the role ranges guarantees decidability and tractability of $\mathcal{EL}^{++}$. Without entering into any details, we just mention that since every concept is restricted to range over the local top concept and the same holds for the roles, it is possible to omit this axiom.

   (2) *Force the domain of any role to be subsumed by the top concept of the corresponding component.*

   $\exists(i : p).ANYTHING \sqsubseteq Top_i$

7. (1) *Force the range of any link to be subsumed by the top concept of the component of the second argument.*

   As before, we omit including the range axiom (again, for details [BBL08]).

   (2) *Force the domain of any link to be subsumed by the top concept of the component of the first argument.*

   $\exists E.ANYTHING \sqsubseteq Top_i$

The main result is provided by the following theorem:

Theorem 6 *Let us consider an $\mathcal{E}$-Connection $C^{\mathcal{E}}(DL_1, ..., DL_n)$ of the $\mathcal{EL}^{++}$ DLs $DL_1, ..., DL_n$ with only term assertions and the corresponding translated global Description Logic $\#(T)$. Then*
$\#(T) \models \#(i, X) \sqsubseteq \#(i, Y)$ *iff*
$C^{\mathcal{E}}(DL_1, ..., DL_n) \models i : X \sqsubseteq Y$.

Proof 2 *Main idea*

It is quite easy to see that the class of models of the $\mathcal{E}$-Connected $\mathcal{EL}^{++}$ ontologies and the class of models of the translated combined knowledge base are actually isomorphic. This is enough to prove the claim. [CK07, BS03]

5.4 DL-Safe Query Answering in the $\mathcal{EL}^{++}$ $\mathcal{E}$-Connection

5.4.1 Translating the $\mathcal{E}$-Connection into Datalog

The work in [KRH08] presents a reasoning algorithm for ELP based on a polynomial translation from ELP to a specific kind of Datalog programs that can be evaluated in polynomial time. ELP is in a fact a decidable

fragment of the Semantic Web Rule Language that is based on the tractable description logic $\mathcal{EL}^{++}$, and encompasses an extended notion of the DL Rules for that logic. Thus ELP extends $\mathcal{EL}^{++}$ with a number of features introduced by the forthcoming OWL (such as disjoint roles, local reflexivity, certain range restrictions, and the universal role).

In the context of the current Chapter, we will not pay attention to the interesting details of this work, but we will rather make use of the proposed transformation/reduction to Datalog, in order to use it as an indispensable step in our algorithm for conjunctive query answering that will be presented in the subsequent Section 5.4.3. Indeed, the translation described in the previous section results in an equisatisfiable $\mathcal{EL}^{++}$ global knowledge base. On the other hand, every knowledge base can be transformed into an equisatisfiable datalog program by the procedure described in [KRH08]. In the end, satisfiability of the $\mathcal{E}$-Connection is reduced to satisfiability in the global knowledge base, which in turn is reduced to satisfiability in the produced datalog program.

5.4.2 DL-Safe Query Answering

Conjunctive queries over an $\mathcal{EL}^{++}$ knowledge base are defined as follows [HM05]:

Definition 30 *Let KB be an $\mathcal{EL}^{++}$ knowledge base, and let N_p be a set of predicate symbols, such that all $\mathcal{EL}^{++}$ concepts and all abstract and concrete roles are in N_P. An atom has the form $P(s_1, ..., s_n)$, denoted also as $P(s)$, where $P \in N_P$, and s_i are either variables or individuals from KB. An atom is called ground atom, if it is variable-free. An atom is called a DL-atom if P is a $\mathcal{EL}^{++}$ concept, or an abstract or a concrete role. Let $x_1, ..., x_n$ and $y_1, ..., y_m$ be sets of distinguished and non-distinguished variables denoted also as $\mathbf{x}$ and $\mathbf{y}$, respectively. A conjunctive query over KB, written as Q($\mathbf{x}, \mathbf{y}$), is a conjunction of atoms $\wedge P_i(s_i)$, where all s_i together exactly contain $\mathbf{x}$ and $\mathbf{y}$. A conjunctive query Q($\mathbf{x}, \mathbf{y}$) is DL-safe if each variable occurring in a DL-atom also occurs in a non-DL atom in Q($\mathbf{x}, \mathbf{y}$). The translation of such a query into a first order formula is:*

$$\pi(Q(\mathbf{x}, \mathbf{y})) = \exists y : \bigwedge \pi(P_i(s_i))$$

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:

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

The main inferences for conjunctive queries are:

- *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, such that $\pi(KB) \models \pi(Q(x\theta), y)$.*
- *Checking query containment. A query $Q_2(\mathbf{x}, \mathbf{y_2})$ is contained in a query $Q_1(\mathbf{x}, \mathbf{y_1})$ w.r.t. KB, if $\pi(KB) \models \pi(Q_2(\mathbf{x}, \mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x}, \mathbf{y_1}))$.*

In practice, the DL-safety of the variables that appear inside DL-atoms can be guaranteed either by introducing a new predicate that enforces all variables to belong to the known individuals or we can relax the restrictions introduced by DL-safety, by eliminating non-distinguished variables through a reduction of tree-like parts of a query to a concept [HT02].

First, we define the so-called tree-like parts of a query:

Definition 31 *For a set of unary and binary literals S, the* coincidence graph *of S is a directed graph with the following structure:*

- *Each variable from S is associated with a unique node.*
- *Each occurrence of a constant n S is associated with a unique node, i.e.occurrences of the same constant are associated with distinct nodes.*

- *For each literal $C(s) \in S$, the node s is labeled with C.*
- *For each literal $R(s,t) \in S$, the nodes s and t are connected with a directed arc labeled R.*

The subset Γ of DL-atoms of a conjunctive query $Q(\mathbf{x},\mathbf{y})$ is called a tree-like *part of $Q(\mathbf{x},\mathbf{y})$ with a root s if*

- *no variable from Γ occurs in $Q(\mathbf{x},\mathbf{y})$ Γ,*
- *the coincidence graph of Γ is a connected tree with a root s,*
- *all nodes apart from s are non-distinguished variables of $Q(\mathbf{x},\mathbf{y})$.*

The reason why we need these tree-like parts is that using the query roll-up technique from [HT02], we can eliminate non-distinguished variables by reducing a tree-like part of a query to a concept, without losing semantic consequences.

5.4.3 Algorithm for Conjunctive Query Answering

In this section we will present an outline of the algorithm for conjunctive query answering in the combined $\mathcal{EL}^{++}$ knowledge base, which borrows its main ideas form the work described in [HM05]. The algorithm is depicted in Figure 5.1 takes as input: 1) a set of $\mathcal{EL}^{++}$ knowledge bases, which are connected with $\mathcal{E}$-Connections, and 2) a conjunctive query $Q(\mathbf{x},\mathbf{y})$. The algorithms then produces an equivalent knowledge base, according to the theory described in Section 5.3. The new combined knowledge base is then transformed and reduced into an equisatisfiable Datalog program, as we explained in Section 5.4.1, and this reduction has polynomial time complexity.

After this initial phase, the algorithm eliminates non-distinguished variables from $Q(\mathbf{x},\mathbf{y})$ using the query roll-up technique that we mentioned in Section 5.4.2. After roll-up, the obtained mappings and queries are required to be DL-Safe to ensure decidable query answering. If this precondition is fulfilled, then the original query in answered in the obtained above Datalog program. By our previous discussion, it is straightforward to see that the algorithm exactly computes the answer of $Q(\mathbf{x},\mathbf{y})$ in the original $\mathcal{E}$-Connected ontologies.

Input
$EL_1, ..., EL_n$: n knowledge bases in $\mathcal{EL}^{++}$ combined with $\mathcal{E}$-Connections
$Q(\mathbf{x},\mathbf{y})$: the conjunctive query

Algorithm
1: Transform the $EL_1, ..., EL_n$ knowledge system into the equivalent EL_{equiv} knowledge base
2: Transform EL_{equiv} into the equivalent datalog program DD
3: Roll-up tree-like parts of query $Q(\mathbf{x},\mathbf{y})$
4: Stop if $Q(\mathbf{x},\mathbf{y})$ is not DL-Safe
5: Compute the answer of $Q(\mathbf{x},\mathbf{y})$ in DD

Figure 5.1: Algorithm for Conjunctive Query Answering.

Chapter 6

Reasoning with Integrated Distributed Description Logics

In this chapter, we focus on the IDDL formalism and an algorithm for reasoning on modular ontologies with the IDDL semantics. In addition, we show how it can be efficiently implemented in an IDDL reasoner thanks to some optimization techniques.

6.1 Motivation

Reasoning on a network of multiple ontologies can be achieved by integration of several knowledge bases or by using non standard distributed logic formalisms. With the first, knowledge must be translated into a common logic, and reasoning is fully centralized. The second option, which has been chosen for Distributed Description Logics (DDL) [BS03], $\mathcal{E}$-connections [KLWZ04], Package-based Description Logics (P-DL) [BCH06b] or [Len02] consists in defining new formalisms which allow reasoning with multiple domains in a distributed way. The non-standard semantics of these formalisms reduces conflicts between ontologies, but they do not adequately formalize the quite common case of ontologies related with ontology alignments produced by third party ontology matchers. Indeed, these formalisms assert cross-ontology correspondences (bridge rules, links or imports) from one ontology's point of view, while often, such correspondences are expressed from a point of view that encompasses both aligned ontologies. Consequently, correspondences, being tied to one "context", are not transitive, and therefore, alignments cannot be composed in these languages.

IDDL addresses this situation and offer sound alignment composition. The principle behind it was presented in [ZE06] under the name *integrated distributed semantics*, and particularized for Description Logics in [Zim07]. This chapter aims at providing a distributed reasoning procedure for IDDL, which has the following interesting characteristics:

- the distributed process takes advantage of existing DL reasoners (e.g. Pellet, Racer, FacT++ etc.);
- local ontologies are encapsulated in the local reasoning system, so it is not necessary to access the content of the ontologies in order to determine the consistency of the overall system i.e. it is sufficient that local reasoners provide results about the consistency of local ontologies;
- the expressiveness of local ontologies is not limited as long as it is a decidable description logic.

Regarding two notions of distributed ontologies and distributed reasoning which are discussed in Section 1.2, one can consider that IDDL supports the approach, namely *centralized integration*. The reason is that an IDDL system considers not only local ontologies but also mappings between them as independent knowledge pieces. From this underlying conception, a reasoning procedure for IDDL would rely on one global reasoner and several local reasoners. The global reasoner deals with knowledge from ontology mappings and gets consistency results from each local reasoner without knowing details about formalisms used in local

ontologies. This feature of IDDL-based systems is very different from those based on DDL, P-DL, etc. in which a reasoner decides local consistency by taking into account knowledge propagated from other ontologies through mappings. This difference is originated from the absence of global consistency notion from the mentioned formalisms which do not seem to focus on mappings.

In term of computation complexity, global consistency (and inconsistency) is basically harder than local consistency (inconsistency) since the former requires to check each model of mappings against all local ontologies.

6.2 Formalism

This section reminds some essential points of IDDL which was presented in Deliverable 1.1.3. for formalizing ontology modules.

6.2.1 Semantics of the local content of modules

Interpreting the local content of a module is equivalent to interpreting axioms of a non-modular ontologies. Since the formalism used to write axioms in our module framework is based on OWL, this local semantics corresponds to a description logic semantics.

Definition 32 (Interpretation) *Given a set of ontology elements $Elem$ (individuals, classes and properties), an* INTERPRETATION *of $Elem$ is a pair $\langle \Delta, [\![.]\!] \rangle$, where*

Δ is a non-empty set, called the DOMAIN*,*

$[\![.]\!]$ is a function from $Elem$ to $\Delta \cup \mathcal{P}(\Delta) \cup \mathcal{P}(\Delta \times \Delta)$, where $\mathcal{P}(x)$ is the part set of x.

In our specific case (considering OWL), the function $[\![.]\!]$ maps

- an individual $\mathtt{a}$ to an element of Δ: $[\![\mathtt{A}]\!] \in \Delta$
- a class $\mathtt{C}$ to a subset of Δ: $[\![C]\!] \subseteq \Delta$
- a property $\mathtt{p}$ to a binary relation between elements of Δ: $[\![\mathtt{p}]\!] \subseteq \Delta \times \Delta$

In fact, an interpretation of the named elements Nam, uniquely defines an interpretation of the ontology elements by applying inductive interpretation rules:

- $[\![C \sqcap D]\!] = [\![C]\!] \cap [\![D]\!]$,
- $[\![C \sqcup D]\!] = [\![C]\!] \cup [\![D]\!]$,
- $[\![\exists R.C]\!] = \{x | \exists y.y \in [\![C]\!] \wedge \langle x, y \rangle \in [\![R]\!]\}$,
- etc.

Interpretations are related to axioms thanks to the satisfaction relation $\models$.

Definition 33 (Satisfaction) *An interpretation $\langle \Delta, [\![.]\!] \rangle$ satisfies:*

- *an axiom $\mathtt{C} \sqsubseteq \mathtt{D}$ if $[\![\mathtt{C}]\!] \sqsubseteq [\![\mathtt{D}]\!]$*
- *an axiom $\mathtt{C}(\mathtt{a})$ if $[\![\mathtt{a}]\!] \in [\![\mathtt{C}]\!]$*
- *an axiom $\mathtt{p}(\mathtt{a}, \mathtt{b})$ if $([\![\mathtt{a}]\!], [\![\mathtt{a}]\!]) \in [\![\mathtt{p}]\!]$*

If an interpretation I satisfies an axiom α, it is denoted by $I \models \alpha$.

The local content of a module is characterized by a set of axioms.

Definition 34 *An interpretation is a model of a set of axioms O if it satisfies all the axioms in O. The set of models of a set of axioms O is denoted $Mod(O)$.*

6.2.2 Satisfied mapping

A mapping connects entities from 2 different ontologies or modules. Interpreting them implies interrelating both ontology (or module) interpretations.
Entities appearing in a correspondence can be interpreted according to the ontology language semantics. Since each ontology may be interpreted in different domains, we define a notion of equalizing function which helps making these domains commensurate.

Definition 35 (Equalizing function) *Let Ω be a set of ontologies and for all $O \in \Omega$, $I_O = \langle \Delta_O, [\![.]\!]_O \rangle$ be an interpretation of O. An equalizing function ϵ for $(I_O)_{O\in\Omega}$ assigns to each o a function $\epsilon_O : \Delta_O \to \Delta$ to a common global domain of interpretation Δ.*

Besides, the mapping language defines a set of relation symbols that are used to express relations between ontology entities. The interpretation of such relation is defined by the mapping language semantics, according to the global domain of interpretation. More precisely, each relation symbol $r \in \mathfrak{R}$ and each global domain Δ is associated to a binary relation $r^\Delta \subseteq \Delta \times \Delta$.
In this deliverable, the mapping language is characterized by the relation symbols $R = \{\sqsubseteq, \sqsupseteq, \equiv, \bot, \not\sqsubseteq, \not\sqsupseteq, \not\equiv, /\!\!\bot\}$. The binary relations associated to them are: set inclusion $r^\Delta = \{(X,Y) \in \Delta \times \Delta \mid X \subseteq Y\}$, set containment $r^\Delta = \{(X,Y) \in \Delta \times \Delta \mid X \supseteq Y\}$, set equality $r^\Delta = \{(X,Y) \in \Delta \times \Delta \mid X = Y\}$, set disjunction $r^\Delta = \{(X,Y) \in \Delta \times \Delta \mid X \cap Y = \emptyset\}$, and their complements.
Using these notions, we can determine whether a correspondence is satisfied by the interpretations of the mapped ontologies.

Definition 36 (Satisfied correspondence) *Let $c = \langle e_1, e_2, r \rangle$ be a correspondence in a mapping between O_1 and O_2. A correspondence is satisfied by two interpretations $\mathcal{I}_1 = \langle \Delta_1, [\![.]\!]_1 \rangle$ and $\mathcal{I}_2 = \langle \Delta_2, [\![.]\!]_2 \rangle$ of O_1 and O_2 respectively, if there exists an equalizing function ϵ for $(\mathcal{I}_1, \mathcal{I}_2)$ over global domain Δ such that $(\epsilon_1([\![e_1]\!]_1), \epsilon_2([\![e_2]\!]_2)) \in r^\Delta$. This is written $\mathcal{I}_1, \mathcal{I}_2 \models_\epsilon c$.*

For instance, consider the correspondence $c = \langle \mathtt{Cottage}_1, \mathtt{Building}_2, \sqsubseteq \rangle$, then $\mathcal{I}_1, \mathcal{I}_2 \models c$ iff $\epsilon_1([\![\mathtt{Cottage}_1]\!]) \subseteq \epsilon_2([\![\mathtt{Building}_1]\!])$.

Definition 37 (Satisfied mapping) *A mapping A of ontologies O_1 and O_2 is satisfied by a pair of interpretations $\langle \mathcal{I}_1, \mathcal{I}_2 \rangle$ if there exists an equalizing function ϵ of $\langle \mathcal{I}_1, \mathcal{I}_2 \rangle$ such that for each $c \in A$, $\mathcal{I}_1, \mathcal{I}_2 \models_\epsilon c$.*

Note that a mapping can be satisfied by interpretations that are not themselves models of the local ontologies. This is useful when one needs to determine consistency of a mapping, but do not have access to the ontologies. Moreover, this also ensures encapsulation at the mapping level, since it prevents mapping satisfiability to be dependent on a particular ontology implementation.

6.2.3 Global interpretation of modules

The interpretation of a module is recursively defined in function of the interpretations of its imported modules.
This recursive definition assumes that there is no cycle in the import chain, so each chain eventually leads to a base module with no import. Detection of cycles should be syntactically checked, since this definition is not

well founded otherwise. If one thinks in term of software engineering, this is not a major limitation. Indeed, when a new module is designed, it has to import existing modules. This way, it is not possible to have cyclic references.

Definition 38 (Base module interpretation) *Let $\mathfrak{M} = \langle \emptyset, \emptyset, \emptyset, O, E \rangle$ be a base module. An interpretation of $\mathfrak{M}$ is a local interpretation $\mathcal{I}$ of the content O of $\mathfrak{M}$, with domain of interpretation $\mathcal{D}$.*

A module interpretation is defined recursively according to the import chain.

Definition 39 (Module interpretation) *Let $\mathfrak{M} = \langle M, I, A, O, E \rangle$ be a module. An interpretation of $\mathfrak{M}$ is a triple $\mathfrak{I} = \langle \mathcal{I}, (\mathcal{I}_m)_{m \in M}, \epsilon \rangle$ such that:*

- *For each imported module $m \in M$, $\mathcal{I}_m$ is a* module interpretation *of m over domain of interpretation $\mathcal{D}_m$;*
- *ϵ is an equalizing function for $(\mathcal{I}_m)_{m \in M}$, over a global domain of interpretation Δ;*
- *$\mathcal{I} = \langle \Delta, [\![.]\!] \rangle$ is a (local) interpretation of the content O of $\mathfrak{M}$, with domain of interpretation Δ. Δ is also called the* domain of interpretation *of module $\mathfrak{M}$;*
- *the interpretation of the imported terms $t_m \in I_m$ of module $m \in M$ is defined by $[\![t_m]\!] = \epsilon_m([\![t_m]\!]_m)$.*

In order for an interpretation to satisfy a module, there are three conditions:

1. the local interpretation must be a model of the content of the module, i.e., all local axioms must be satisfied;
2. the imported modules must be satisfied by their respective interpretations;
3. the mappings between the imports must be satisfied by the respective pairs of interpretations;

Definition 40 (Model of a module) *Let $\mathfrak{M} = \langle M, I, A, O, E \rangle$ be a module and $\mathfrak{I} = \langle \mathcal{I}, (\mathcal{I}_m)_{m \in M}, \epsilon \rangle$ a module interpretation of $\mathfrak{M}$. $\mathfrak{I}$ is a* model *of $\mathfrak{M}$ (written $\mathfrak{I} \models \mathfrak{M}$) iff:*

- *for each imported module $m \in M$, $\mathcal{I}_m \models m$ (i.e., each imported module is locally satisfied);*
- *$\mathcal{I}$ is a model of O (i.e., the local content of $\mathfrak{M}$ is satisfied);*
- *for each pair of modules $m, m' \in M$, $\mathcal{I}_m, \mathcal{I}_{m'} \models A_{m,m'}$ (i.e., all mappings are satisfied).*

The set of all the models of a module $\mathfrak{M}$ is written $\mathrm{Mod}(\mathfrak{M})$ too.

The notion of models is essential for automatic deduction in modular ontologies. It serves to define which formulas are semantic consequences of a module (i.e., entailment).

6.2.4 Consequences of a module

In order to reason with modular ontologies, we have to define what are the semantic consequences of a module. They are defined as follows:

Definition 41 (Consequences of a module) *Let $\mathfrak{M} = \langle M, I, A, O, E \rangle$ be a module. Let δ be an axiom built upon the signature of the content of $\mathfrak{M}$ (which includes the import interfaces of I). δ is a consequence of $\mathfrak{M}$, written $\mathfrak{M} \models \delta$ iff for all $\langle \mathcal{I}, (\mathcal{I}_m)_{m \in M}, \epsilon \rangle \in \mathrm{Mod}(\mathfrak{M})$, $\mathcal{I} \models \delta$.*

Obviously, if a formula is a consequence of the content ontology of a module, then it is a consequence of the module itself.[1] Additionally, it is desirable to derive knowledge about the imported terms according to the imported modules knowledge. However, if something is true about a concept C in a module, it is not necessarily true in another module that imports C. For instance, in a description logic knowledge base, if a module m is such that $m \models \neg C \sqsubseteq D$, it does not follow that, considering a module $\mathfrak{M} = \langle\{m\}, \{C, D\}, \emptyset, \emptyset, \emptyset\rangle$, $\mathfrak{M} \models \neg C \sqsubseteq D$, as the domains of interpretation of $\mathfrak{M}$ and m may not be the same.

In order to characterize formulas that can be propagated from a module to its importers, we define a general notion of locality, inspired by [GHKS07]. A formula is semantically local when its satisfiability in a module implies its satisfiability in a module that imports its terms.

Definition 42 (Semantic locality) *Let m be a module. Let α be a formula written in terms of the export interface. α is* semantically local *iff for all modules $\mathfrak{M}$ that uses m and import the terms of α:*

$$m \models \alpha \longrightarrow \mathfrak{M} \models \alpha$$

A module m is semantically local iff all its axioms are local.

As seen in [GHKS07], locality can be computationally checked in the description logic $\mathcal{SHOIQ}$. Semantic locality is clearly a desirable property to design "safe" modules. Indeed, in a semantically local module, what is true of a term in the module, is also true in a module that imports it.

6.3 The IDDL Reasoner for Ontology Modules

The IDDL reasoner checks the consistency of a distributed modular ontology. This section briefly presents the principle of the algorithm and its implementation for a reduced IDDL system which does not allow disjointness correspondences to occur in mappings (or alignments). This restriction does not lead to a serious drawback of the expressiveness since alignments generated by the majority of matching algorithms do not often include disjointness correspondences.

6.3.1 Algorithm and Optimization

The algorithm is based on [ZD08] which provides much more details about it. However, the vocabulary used in [ZD08] differs from the one used here. According to the metamodel, wherever *ontology* is used in [ZD08], we can replace it by *module* without loss of correctness. Where *alignment* is used in [ZD08], we use *mapping* here. Finally, a *correspondence* in [ZD08] is the equivalent of a *mapping assertion* here.

Preliminary assumption

The IDDL reasoner works by having a module reasoner communicate with imported modules' reasoners. The imported modules reasoners are supposed to be encapsulated so that their implementation is unknown but they can be used via an interface. Consequently, for each imported module m_i, we assume that there exists an oracle F_i which takes a set of DL axioms A_i as arguments and returns a boolean equal to $\mathrm{Consistency}(m_i \cup A_i)$.

Definition 43 (Reasoning oracle) *Let O be an ontology defined in a logic L. A* reasoning oracle *is a boolean function $F : \mathcal{P}L \rightarrow \mathrm{Bool}$ which returns $F(A) = \mathrm{Consistency}_L(O \cup A)$, for all sets of axioms $A \in \mathcal{P}L$.*

[1] Note that only the consequences related to the exported terms are useful to an external module that imports them.

The term ontology in Definition 43 is to be taken in a general sense. It can be a module or even a distributed system, as long as the associated reasoner can interpret DL axioms and offers correct and complete reasoning capabilities. In practice, such oracles will be implemented as an interface which encapsulates a reasoner like Pellet, Fact++, or a module reasoner.

A module reasoner must call the oracles associated with the imported modules with well chosen axioms in order to determine consistency. The choice of axioms will be explained below. In addition, the module reasoner have access to the mappings that may exist between imported modules. From the importing module's point of view, these mappings are treated like local axioms. Therefore, we can consider that the mappings are equivalent to an ontology (called *the alignment ontology* in [ZD08]).

Definition 44 (Alignment ontology) *Let $\mathbf{A}$ be a set of mappings. The alignment ontology is an ontology $\widehat{\mathbf{A}}$ such that:*

- *for each mapping assertion $i:C \xleftrightarrow{\sqsubseteq} j:D$ with C and D local concepts,*
  - *$\widehat{i:C}$ and $\widehat{i:D}$;*
  - *$\widehat{i:C} \sqsubseteq \widehat{i:D} \in \widehat{\mathbf{A}}$;*
- *for each mapping assertion $i:P \xleftrightarrow{\sqsubseteq} j:Q$ with P and Q local roles,*
  - *$\widehat{i:P}$ and $\widehat{i:Q}$;*
  - *$\widehat{i:P} \sqsubseteq \widehat{i:Q} \in \widehat{\mathbf{A}}$.*

In order to check the global consistency of the module, we also assume that there is a reasoning oracle F_A associated to $\widehat{\mathbf{A}}$. The algorithm consists in questioning all the reasoning oracles with well chosen axioms that are detailed just below.

Algorithm

In [ZD08], it is formally proven that consistency checking of an IDDL system with only subsumption mapping assertions can be reduced to determining the emptiness and non emptiness of specific concepts. More precisely, we define the notion of configuration, which serves to explicitly separate concepts between empty concepts and not empty concepts, among a given set of concepts. It can be represented by a subset of the given set of concepts, which contains the asserted non-empty concepts.

Definition 45 (Configuration) *Let $\mathscr{C}$ be a set of concepts. A* configuration *Ω over $\mathscr{C}$ is a subset of $\mathscr{C}$.*

In principle, a configuration Ω implicitly assert that for all $C \in \Omega$, $C \sqsubseteq \bot$ and for all $C \notin \Omega$, $C(a)$ for some individual a. A similar notion of *role configuration* is also defined in [ZD08], but for the sake of simplicity we will only present it for concepts.

The algorithm then consists in choosing a configuration over the set of all concepts of the alignment ontology. The axioms associated with the configuration are then sent to the oracles to verify the consistency of the resulting ontologies. If they all return *true* (i.e., they are all consistency with these additional axioms) then the modular ontology is consistent. Otherwise, another configuration must be chosen. If all configurations have been tested negatively, the modular ontology is inconsistent, according to the proof in [ZD08]. Since there is a finite number of configurations, this algorithm is correct and complete.

The sets of axioms that must be used to query the oracles are defined according to a configuration Ω as follows.

Let A (resp. $A_1, \ldots, A_n$) be the sets of axioms associated with the oracles of the alignment ontology (resp. with the oracle of modules $m_1, \ldots, m_n$).

For all imported modules m_i,

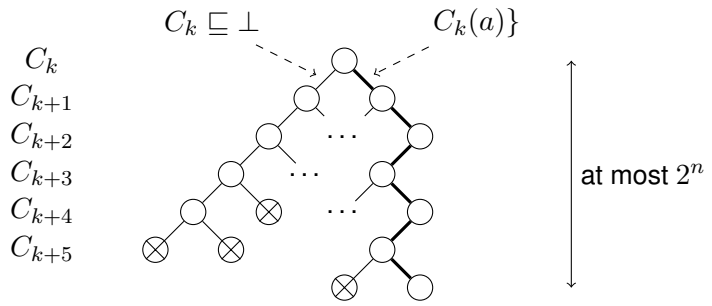

Figure 6.1: At each node, the left branch indicates that the concept C_k is asserted as an empty concept ($C_k \sqsubseteq \bot$), while the right branch indicates a non empty concept ($C_k(a)\}$). The thick path indicates a possible configuration for the distributed system.

- for all concepts $i\!:\!C \in \Omega$, $\widehat{i\!:\!C}(a) \in A$ and $C(a_C) \in A_i$ where a is a fixed individual and a_C is a new individual in m_i.
- for all concepts $i\!:\!C \notin \Omega$, $\widehat{i\!:\!C} \sqsubseteq \bot \in A$ and $C \sqsubseteq \bot \in A_i$.

Limitations

The algorithm was originally designed to reason with a network of aligned ontologies, not with modules. This leads to notable limitations:

- the content of the module must be empty or only contains subsumption axioms $A \sqsubseteq B$ with A and B primitive;
- this module reasoning procedure cannot be used as an oracle for an imported module, which means that there cannot be a chain of import longer than one.

Optimization

The algorithm, as it is described here, will answer negatively if and only if it has tested all possible configurations. So, every time a module is inconsistent, the reasoner must call all the oracles 2^n times, where n is the number of concepts in the alignment ontology. This situation is hardly acceptable in a practical reasoner. However, optimizations can be carried out to improve this situation. In particular, backtrack algorithms can be applied to this procedure. Indeed, for each concept C appearing in a mapping, it must be decided whether it is empty or not. There are cases when it can be deduced that C is empty (resp. not empty). In this case, it not necessary to test configurations where C is not empty (resp. empty). This can be visualized in Figure 6.1.

6.3.2 IDDL Reasoner API

The IDDL reasoner provides a basic interface to check consistency of an ontology module which consists of a set of imported ontologies in OWL and mappings between them. The current IDDL reasoner uses Pellet reasoner as local reasoner for checking consistency of an imported ontology from an ontology module. The interface between the IDDL reasoner and local reasoners is designed so that any other reasoner, e.g. FaCT++, Racer, Drago, etc., can easily replace Pellet reasoner. Additionally, the IDDL reasoner uses Alignment API [Euz04] to manipulate correspondences during reasoning process.

In addition, the reasoner offers a possibility to get an explanation for an inconsistency of an IDDL system. The method

```
ReasonerManager.isConsistent(Vector<URI> ontoUris, Vector<URI> alignUris, Semantics sem)
```

allows users to call the reasoner with the following three parameters :

1. **ontoUris :** URIs of OWL ontologies
2. **alignUris :** URIs of alignments
3. **sem :** indicates the semantics of the system. The type **Semantics** takes two possible values :
   (a) **IDDL :** represents the IDDL semantics in the context where alignments are obtained from matching ontologies which are not necessary imported ontologies of an ontology given.
   (b) **DL :** represents the semantics of Description Logics which interprets all ontologies and alignments of the system into a unique interpretation domain. Consequently, consistency of the system is equivalent to that of the union of OWL ontologies and alignments.

The method

```
IDDLReasoner.isConsistent ()
```

will be invoked from

```
ReasonerManager.isConsistent(Vector<URI> ontoUris, Vector<URI> alignUris, Semantics sem)
```

to check consistency of an IDDL system which has been loaded by the method

```
IDDLReasoner.loadIDDLSystem()
```

or

```
IDDLReasoner.loadMIDDLSystem()
```

To make an IDDL reasoner available to reason for a new IDDL system, the method

```
IDDLReasoner.unloadIDDLSystem()
```

has to be called.

6.3.3 Further work

The current version of the IDDL reasoner provides only explanations for inconsistencies which are caused by correspondences propagated from mappings to imported ontologies. A future version of the IDDL reasoner should take advantages of explanations from local reasoners to give more details about how propagated correspondences impact on an imported ontology.

As mentioned at the beginning of the present section, the current version of the IDDL reasoner does not allow disjointness correspondences to occur in alignments. This limitation prevents us from supporting axiom entailment since it is equivalent to inconsistency of an IDDL system including disjointness correspondences. For instance, the current IDDL reasoner does not know whether $\langle \mathbf{O}, \mathbf{A} \rangle \models i{:}C \sqsubseteq j{:}D$ where $\mathbf{O}$, $\mathbf{A}$ are the sets of imported ontologies and mappings from an ontology module.

In a future version, we plan to extend the reasoner such that it takes into account only disjointness correspondences translated from entailment but not those initially included in alignments. Allowing disjointness correspondences in this controlled way may not lead to a complexity blow-up.

6.4 Integration with the NeOn Toolkit

In this section we describe the principle of the integration of the IDDL reasoner within the NeOn Toolkit. This integration is performed by developing a plug-in, namely IDDL reasoner plug-in, which plays an interface role between the IDDL reasoner and the NeOnToolkit plug-ins, e.g. the module API, Ontology Navigator, Alignment Plugin, etc.

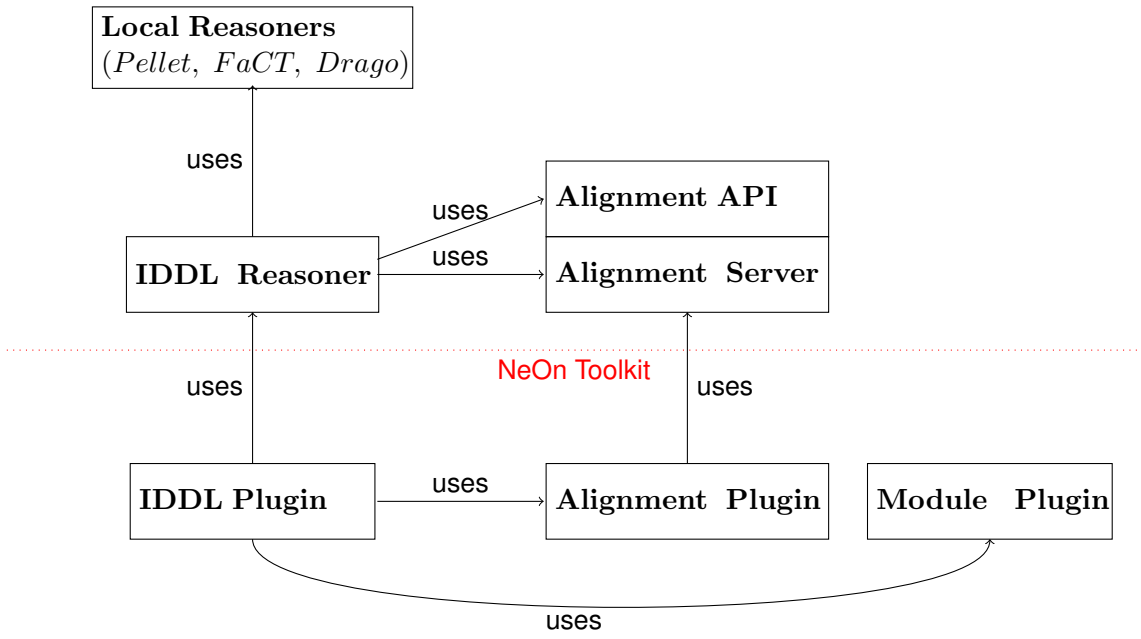

Figure 6.2: IDDL reasoner plug-in and related components.

6.4.1 Principle of the integration

The IDDL reasoner API for ontology modules uses the module API to access to **mappings** and **imported ontologies** of an ontology module. In other terms, the IDDL reasoner plug-in is developed such that it can get access to the mappings and imported ontologies from an ontology module and pass them to the IDDL reasoner with help of the IDDL reasoner API as described in Section 6.3.2.

More precisely, from the NeOn toolkit environment the IDDL reasoner plug-in gets URIs of the imported ontologies and the mappings from an ontology module. In the most of cases where mappings are not available from the ontology module, the plug-in can fetch available alignments or mappings from an alignment server. This feature allows users to use alignments permanently stored on servers and select the most suitable alignments for an intended purpose.

6.4.2 IDDL reasoner plug-in

Figure 6.2 shows relationships between the IDDL reasoner plug-in and the other components from the NeOn Toolkit and the IDDL reasoner. The IDDL reasoner plug-in relies on the IDDL reasoner, and the core module API which provides basic operations to manipulate ontology modules and mappings. By using the core module API, the IDDL reasoner plug-in can get necessary inputs from an ontology module. In the case where the mappings obtained from the ontology module in question are not appropriate, the plug-in can connect to the Alignment Server [LBD+08] to fetch alignments available. For this purpose, the plug-in offers to users an interface allowing to visualize and select alignments.

From a determined input, the IDDL reasoner plug-in can obtain an answer for consistency from the IDDL reasoner. In the case where the answer is negative the plug-in can obtain an explanation indicating configurations and/or correspondences which are responsible for that inconsistency.

6.5 Use Example

The answer time of the IDDL reasoner depends on the following elements which are taken into account in the optimized algorithm design.

1. If there are more unsatisfiable or non-empty concepts occuring in correspondences, the answer time is shorter;
2. If there are more equivalent concepts or properties occuring in correspondences, the reasoner answers faster;
3. If ontology module is inconsistent, the reasoner has to check likely all configurations. Therefore, the answer time would be long.

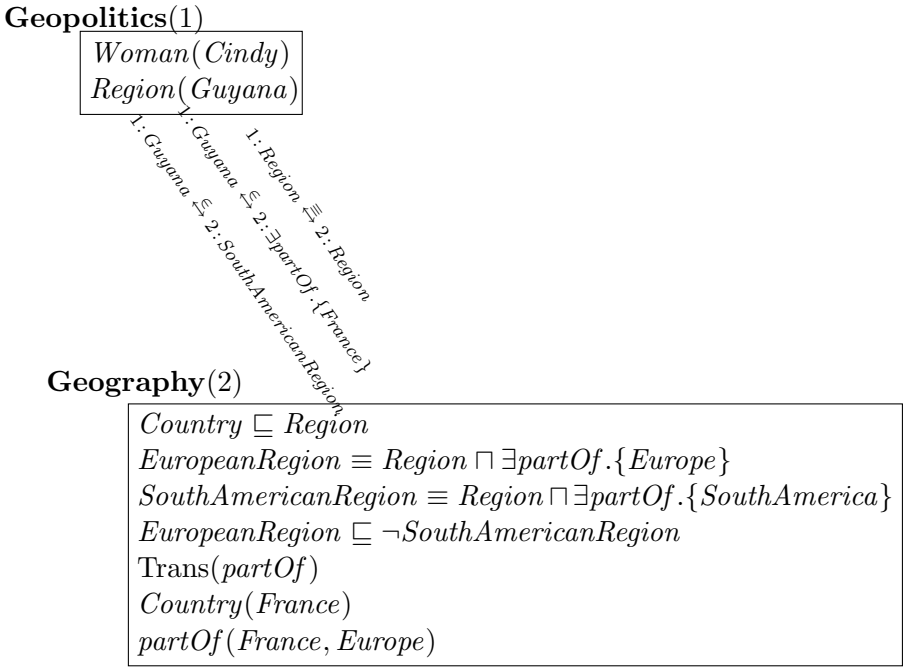

Figure 6.3: An example of an ontology module with mappings.

In this Example 6.3, we have two imported ontologies, $\mathbf{Geopolitics}$ and $\mathbf{Geography}$, with a mapping between them. The axioms of the ontologies and mapping are expressed in a description logic and they can be directly coded in OWL-DL. We consider the following cases:

1. If these imported ontologies are merged with the correspondences of the mapping, we obtain an OWL ontology which is not consistent. The reason is that the mapping allows one to deduce that two classes "EuropeanRegion" and "SouthAmericanRegion" are not disjoint, which contradicts the disjointness axiom in $\mathbf{Geography}$.
2. However, in the context of ontology module the IDDL reasoner can check consistency of the module and answer that the module is consistent (Figure 6.4).
3. If we now add to the following mapping $1 : Guyana \overset{\in}{\leftrightarrow} 2 : SouthAmericanRegion \sqcap EuropeanRegion$, the IDDL reasoner answers that the module is no longer consistent. An explanation of the inconsistency is shown in Figure 6.5.

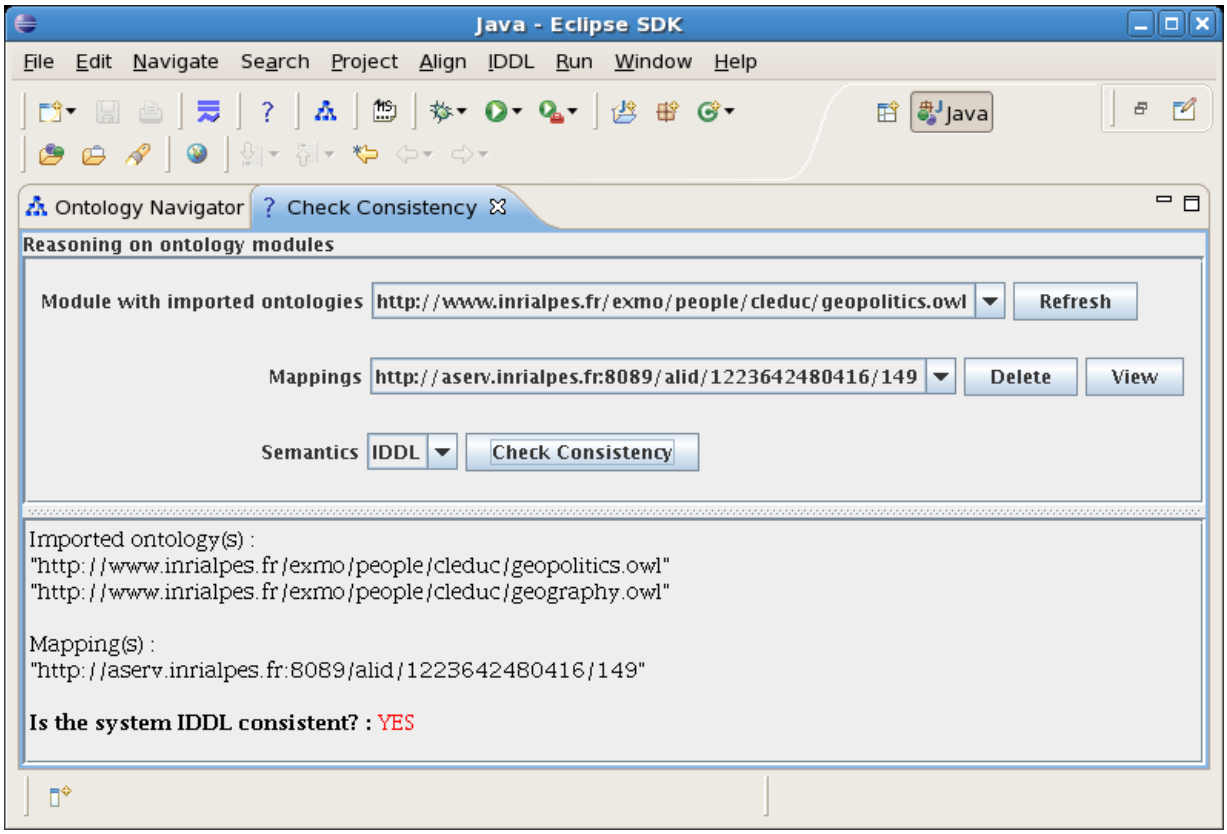

Figure 6.4: A consistent IDDL system.

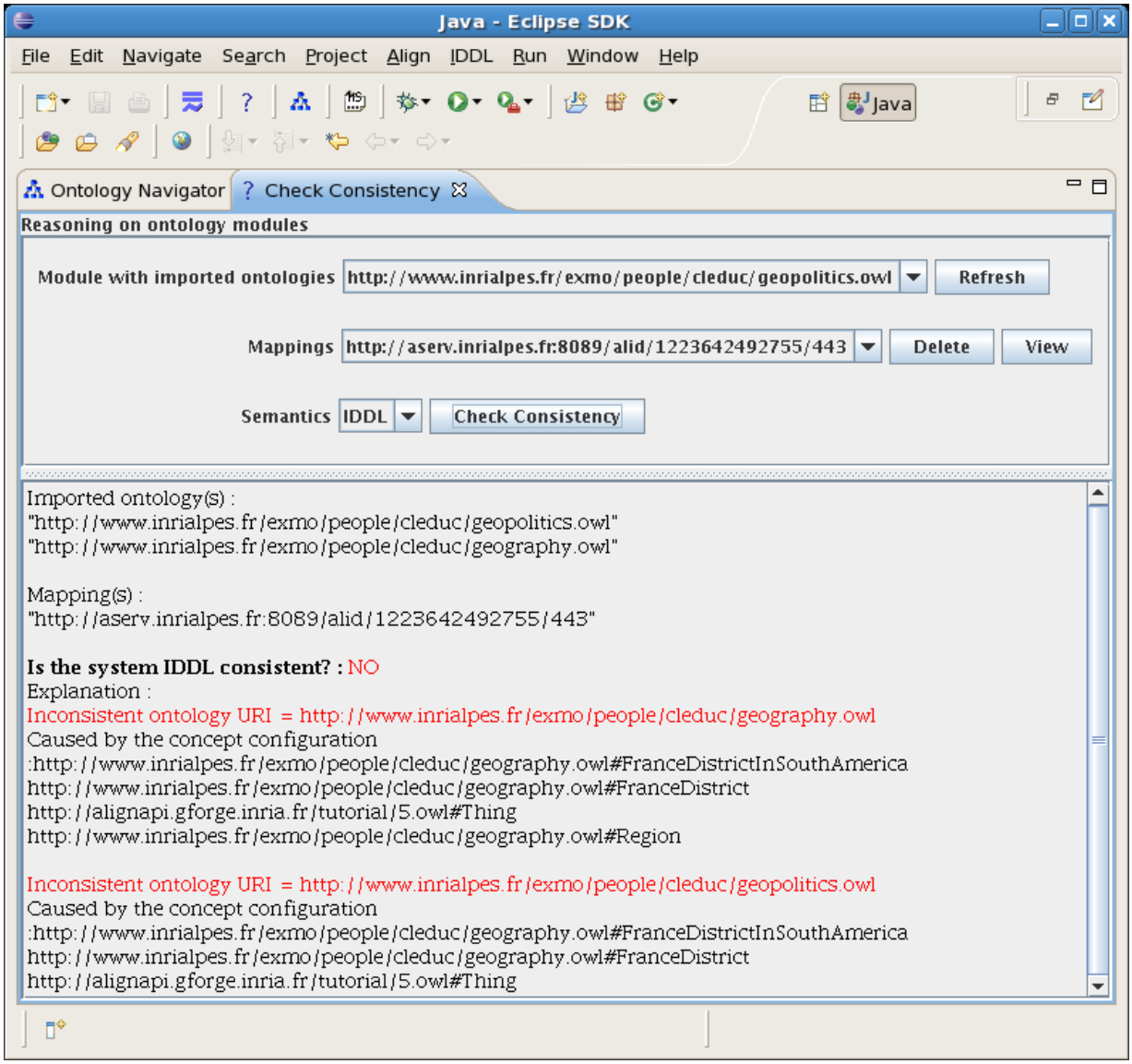

Figure 6.5: An inconsistent IDDL system.

Chapter 7

Reasoning with Temporarily Unavailable Data Sources

The Semantic Web is envisioned to be a *Web of Data* [BL98]. As such, it integrates information from various sources, may it be through rules, data replication or similar mechanisms.

When doing reasoning over such distributed data, availability of the data sources is usually assumed and data is directly requested from these sources. However, in order to provide an increased level of resistance against failures or in order to improve performance, caching of remote data may be employed during reasoning. In such cases, however the cache needs to be kept up to date. In cases, when this is not possible, information must be available, that the inferences drawn are based on stale information. Alternatively, some default truth value could be assumed for unavailable information.

In this chapter we propose a framework for reasoning with such cached knowledge, which allows to give additional information on the reliability of results to the user. In particular, we are able to tell whether a statement's truth value is inferred based on really accessible information, or whether it might change in the future, when cached or default values are updated. The work presented here shares foundations with trust based reasoning as presented in [QHS+09] and is a specialization of general trust based reasoning.

Dealing with cached information in relevant to most distributed reasoning mechanisms and as such orthogonal to the approaches presented in the previous chapters. particularly, KAONp2p locally integrated the T-Boxes of the ontologies involved in a reasoning task. There, the formalism proposed in this chapter is directly applicable to the T-Box reasoning.

The problem of reasoning in a distributed environment without reliability guarantees is highly relevant, because fault tolerance and reliable data integration is a main prerequisite for a distributed system like the semantic web. The level of reliability of a piece of information can strongly influence further usability of derived information. For this reason, our approach can be seen as a bridge between the rules, proof and trust layers of the semantic web layercake.

The availability of multiple integration mechanisms for distributed resources makes formulating a generic framework a non-trivial task. Moreover, classical two-valued logic fails to capture the 'unknown' truth value of unavailable information. In fact, many applications today rely on simple replication of all necessary data, instead of more flexible mechanisms.

Our approach extends a very flexible basis of most logical frameworks, namely bilattices, which allow to formalize many logics in a coherent way [HW05]. Hence, it is applicable to a broad range of logical languages. We propose an extension $\mathcal{FOUR}-\mathcal{C}$ to the $\mathcal{FOUR}$ bilattice.

We investigate support for connected and interlinked autonomous and distributed semantic repositories (for RDF or OWL) — a basic idea behind the semantic web effort. These repositories exchange RDF and OWL data statically (e.g. by copying whole RDF graphs, as in the caching scenario) or dynamically using views or rules.

Our approach is based on assigning different trust levels to cached and local information. Information about

trust levels is aggregated and propagated to inferred axioms during the reasoning process.

7.1 $\mathcal{FOUR}$

Most logic programming paradigms, including classical logic programming, stable model and well founded semantics, and fuzzy logics can be formalized based on bilattices of truth values and fixpoints of a direct consequence operator on such a bilattice. Therefore, if we build our extension into this foundational layer, it will directly be available in many different formalisms.

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 **??**. 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.

In traditional, two valued logic programming without negation, only t and f would be allowed as truth values. In contrast, e.g. the stable model semantics, allows to use $\top$ and $\bot$. In this case, multiple stable models are possible. For example, we might have a program with three clauses:

$man(bob) \leftarrow person(bob), \neg woman(bob).$

$woman(bob) \leftarrow person(bob), \neg man(bob).$

$person(bob).$

Using default f, we might infer both $man(bob) \wedge \neg woman(bob)$ and $woman(bob) \wedge \neg man(bob)$. While in two valued logics we would not be able find a model, in four values, we could assign truth values $t \oplus f = \top$ and $t \otimes f = \bot$. In fact, both would be allowed under the stable model semantics, resulting in multiple models for a single program.

The well founded semantics distinguishes one of these models — the minimal one, which is guaranteed to always exist and only uses t, f, and $\bot$. In a similar way, other formalisms can be expressed in this framework as well. Particularly, we can also formalize open world based reasoning, using $\bot$ instead of f as default value. For a detailed introduction of logical bilattices, we refer the reader to the very good overview in [Fit02].

7.2 $\mathcal{FOUR}-\mathcal{C}$

To apply our work to a variety of different logical formalisms, we directly extend $\mathcal{FOUR}$ as the theoretical basis.

To distinguish between certain information, which is local or currently available online, and cached information (or information derived from cached information), we extend the set of possible truth values: For information, of which we know the actual truth value we use the truth values $\{t_k, f_k, \top_k, \bot_k\}$. For cached information, we use a different set: $\{t_c, f_c, \top_c, \bot_c\}$. The basic idea of the extension is that cached information is always potentially outdated. For example a cached *false* value might actually be *true*. Therefore, we assume cached information to be always a bit less false or true than certain information — as the truth value might have changed.

In our scenario, let us assume Project1's web site is currently inaccessible. In a normal closed world setting, we would assume $published(report1, \_)$ to be f, hence also $timelyDeliverable(report1)$, by rule (4). Changing our default to $\bot$ – unknown – would also not help us determine, whether Project1's web site is just

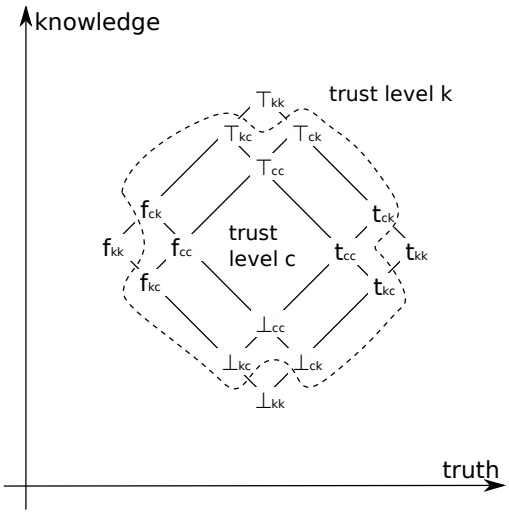

Figure 7.1: $\mathcal{FOUR}-\mathcal{C}$

updated slowly, or whether the available information might be inaccurate. In $\mathcal{FOUR}-\mathcal{C}$, we assign f_c (or $\bot_c$ in an open world setting) to $published(report1, \_)$. We can then conclude from (1-4) and the unavailability of (5) that $timelyDeliverable(report1)$ is $t_k \wedge f_c = f_c$, and hence possibly outdated. Therefore, Oscar will simply update his report later, when all relevant data sources are available again, instead of sending a reminder by mistake. Analogously, if we run into an inconsistency, we want to be aware, if this inconsistency could potentially be resolved by updating the cache. Summarizing, our operators should act as in $\mathcal{FOUR}$, if we only compare truth values on the same trust level. If we compare values from multiple trust levels, we would like to come up with analogous truth values as in the four valued case, but on the trust level, which is the lowest of the compared values.

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 lattice based operators from section 7.1 — then $\mathcal{L}$ will be as well.

We use $\mathcal{L}_1 = \mathcal{L}_2 = t_k > t_c > f_c > f_k$ as input lattices, resulting from our basic idea that cached values are a bit less true and false. As $\mathcal{L}_1$ and $\mathcal{L}_2$ are totally ordered sets, they are complete lattices and hence infinitely distributive. The resulting $\mathcal{FOUR}-\mathcal{C}$ bilattice shown in fig. 7.1. In fig. 7.1 we label nodes of the form $\langle f_x, t_y\rangle$ with $\top_{xy}$, $\langle t_x, f_y\rangle$ with $\bot_{xy}$, $\langle f_x, f_y\rangle$ with f_{xy} and $\langle t_x, t_y\rangle$ with t_{xy}.

The artificial truth values $f_{kc}, f_{ck}, t_{kc}, t_{ck}$, $\top_{kc}, \top_{ck}, \bot_{kc}$ and $\bot_{ck}$ are only used for reasoning purposes. Users will only be interested in trust levels, which are equivalence classes of truth values: Given an order of $k > c$, the trust level of a truth value is the minimal element in its subscript. For example the trust level of $t_c, \bot_{kc}$ and $\top_{ck}$ is c. In fig. 7.1, these equivalence classes are separated by dotted lines. As we only have two trust levels here, there are exactly two equivalence classes - one for currently accessible (truth values $t_{kk}, f_{kk}, \top_{kk}, \bot_{kk}$) and one for cached information (truth values $t_{cc}, f_{cc}, \top_{cc}, \bot_{cc}$) and information derived from cached information (truth values $t_{kc}, f_{kc}, \top_{kc}, \bot_{kc}, t_{ck}, f_{ck}, \top_{ck}, \bot_{ck}$).

Obviously, $\mathcal{FOUR}-\mathcal{C}$ meets our requirements from the beginning of this section: We have two subbilattices isomorphic to $\mathcal{FOUR}$, one on each trust level. Additionally, we always come up with truth values on the right trust level, e.g. $f_{kk} \oplus t_{cc} = \top_{kc}$, which is on trust level c and $\top_{kk} \wedge \top_{cc} = \top_{ck}$, which is on trust

level c and correctly reflects the fact that the result may be inaccurate in case $\top_{cc}$ needs to be corrected to some f_{xx}.

In the caching scenario, we can assume a default truth value of $\bot$ or f (depending on whether we do open or closed world reasoning) to all statements, where the actual truth value can not be determined at the moment. However, some more information may be available for example due to caching, statistics or similar. Using $\mathcal{FOUR}-\mathcal{C}$, we can still do inferencing in the presence of such unreliable sources. Moreover, a user or application can determine, whether a piece of information is completely reliable, or if more accurate information may become available.

In [Sch08] we describe, how the stable and well founded semantics for logic programs, and for Semantic Web rules based on Networked Graphs [SS08a] can be extended towards reasoning with unavailable data sources based of $\mathcal{FOUR}-\mathcal{C}$. Here, we briefly sketch the extension of OWL 2. As reasoning with unavailable data sources is a specialization of trust based reasoning described in [QHS$^+$09], we refer the reader to [SS08a] and [QHS$^+$09] for a detailed specification.

7.3 Extension towards OWL

In this section we extend $\mathcal{SROIQ}$, 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]. $\mathcal{SROIQ}-\mathcal{T}$ is even more general than needed here, as it allows to use any logical bilattice as discussed in [QHS$^+$09]. For reasoning with cached knowledge, we use $\mathcal{FOUR}-\mathcal{C}$ as the underlying bilattice.

Definition 46 *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 47 (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 $\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}}$.*

This generalization allows to assign one of multiple truth values from a logical bilattice, instead of only *true*.
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. 7.1. The satisfaction of axioms is defined analogously and listed in [QHS$^+$09] in detail.

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

Table 7.1: Extended Class Interpretation Function

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. As an illustration, remember that in description logics, membership of an instance in a class is crisp, i.e. it is a member or not. Hence, classes are usually modeled as sets and set membership corresponds to a truth value of *true* and the default is false. Here, in contrast, we can assign any truth value from the bilattice. Hence, such a simple modeling is no longer possible and we need a function instead of a set to model classes.

Definition 48 (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.

u represents the reliability of an infered result: If an axiom is k satisfiable, it can be inferred from available axioms. On the other hand, a c satisfiable axiom is inferred from axioms, of which some are temporarily unavailable. Hence, this inferred knowledge may be erroneous.
Finally, we define entailment:

Definition 49 (Entailment) *O entails a $\mathcal{SROIQ} - \mathcal{C}$ 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.*

The following theorem shows that we have indeed defined a strict extension of $\mathcal{SROIQ}$:

Theorem 7 *If $\mathcal{FOUR}$ is used as logical bilattice, $\mathcal{SROIQ}-\mathcal{T}$ is isomorphic to $\mathcal{SROIQ}$.*

7.4 Related Work

Relevant related work comes from the fields of semantic caching, multi-valued logics — particularly based on logical bi-lattices — from belief revision and trust. The following works are closely related:

The term Semantic Caching refers to the caching of semantic data. Examples are such diverse topics as caching results of semantic web service discovery [SHH07], caching of ontologies [B. 06] and caching to improve the performance of query engines [KKM06]. These approaches have in common, that they discuss how to best do caching of semantic data. In this chapter, we describe which additional information about the reliability of knowledge we can infer, given a heterogeneous infrastructure containing semantic caches.

Much work has been done about basing logical formalisms on bilattices (cf. [Fit02], [HW05]). Most of these works, however propose a certain logic by manually designing a suitable bilattice, or discuss how a particular logic can be formalized using a bilattice. In contrast to these works, we do not propose a fixed bilattice or logic. Instead, we automatically derive logical bilattices for trust based reasoning. Hence, we propose a whole family of logics, which can automatically be tailored to the problem at hand. We instantiate this framework for reasoning with cached knowledge.

7.5 Conclusion

We have proposed an extension to the logical bilattice $\mathcal{FOUR}$, called $\mathcal{FOUR}-\mathcal{C}$, which allows to reason with temporarily unavailable data. As bilattices are a basis for various logical formalisms, this allows to extend many languages with trust based reasoning.

Our extension is applicable in both, open and closed world reasoning, in particular for rules and the description logic $\mathcal{SROIQ}$. We will investigate the complexity of trust based reasoning with description logics and plan an implementation, extending existing reasoning engines.

Chapter 8

Discussion

8.1 Summary

During the last decade the semantic web has been constantly evolving coming to some extent closer to the semantic web vision. A number of W3C recommendations are now available, while the OWL recommendation has very recently been proceeded by the extended OWL 2 recommendation. On the other hand, more and more applications using semantic web standards (mainly RDF/RDFS) are emerging, especially around the areas of semantic search engines, travel planning applications, electronic commerce and knowledge management, while the semantic web community is actively pushing towards this direction through a number of initiatives, like the Billion Triple Challenge.

All these steps point to a relative maturity of the semantic web field and a big change on the impact of the semantic web technologies on the application-level. However, as long as the issue of scalability to very large schema and (mainly) data volumes is unresolved, the practical feasibility of a complete or even partial fulfilment of the semantic web vision cannot be guaranteed. It is thus sine qua non that the research community focuses on the problem of data distribution and distributed reasoning and comes up with novel ideas in the direction of system scalability. This is straightly analogous to the problems faced and to a large extent resolved by the database community in the previous decades.

In this deliverable, we have attempted to attack various problems that arise when dealing with distributed, networked ontologies. More concretely:

- Chapter 1 discussed the general background and motivation of reasoning and ontology distribution and presented the main dimensions of interest in distributed reasoning.
- Chapter 2 provided use cases of reasoning over distributed networked ontologies.
- Chapter 3 provided the state of the art in reasoning with distributed data - distributed reasoning.
- Chapter 4 described the metamodel for mapping support in OWL.
- Chapter 5 looked into distributed reasoning with $\mathcal{E}$-Connections in the context of the FOA ontology.
- Chapter 6 examined the approach proposed at the [Zim07] regarding reasoning with integrated Description Logics.
- Chapter 7 dealt with reasoning with temporarily unavailable data sources.

8.2 Challenges

The discussion in this deliverable reveals a certain level of realization in the semantic web community of the importance that distribution in reasoning procedures will play in coming years and provide different solutions

to the distribution problem. Nevertheless, the field is still not very mature and a number of points for future work have to be considered:

- *Both schema and data distribution*. There are, for example, techniques like DDLs that define formalisms for both TBox and ABox distribution (by providing the ability to define relationships between individuals residing on different nodes), but present practical algorithms only for the case of TBoxes. While it is true that if one focuses only on inference tasks as concept subsumption then one does not need to take into account the ABoxes, in the general case and in most practical applications it would be necessary to consider distributed data all over the system which could then be used for query answering. We also note that an exception to this is the approach of OWL-DL mappings that allows for ABoxes, since it is based on the paradigm of disjunctive Datalog.

- *Enrich existing techniques*. All approaches present a framework that allows for distributed reasoning and/or reasoning with distributed data, but there is still room for further extensions/enrichments, so that they can be used in even more general settings, thus increase their applicability. For example, DDLs currently provide an algorithm for bridge rules only between concepts, that could in future work be extended to the case of bridge rules between roles. General directions for enhancement could include allowing for more expressive representation languages and more expressive relationships between different nodes or, in some cases, restricting the current framework, so as to obtain more convenient complexity results.

- *Look into the pragmatics of distributed reasoning, especially for very large data volumes*. Even the requirements from distributed reasoning when dealing with very big ABoxes is not something very clear in the semantic web community. For very large scales one can argue that the ability for sound and complete reasoning should give place to the ability to receive back only the fraction of the full answer set that is most relevant according to a list of criteria. This is still a very controversial point, since it somehow distorts the ideal picture of reasoning that several people support. Nevertheless, approximate reasoning mechanisms could provide another useful direction for very large ABox volumes.

- *Clear out which approaches are better suited to different applications*. In the semantic web universe it is pointless to seek for only one technique that encompasses every desirable feature and provides a solution to all applications. Contrary to that no single method can currently be considered as a panacea. It would instead make sense to look into the pragmatics of the different approaches and look for use cases for each of them. Better understanding the pragmatics could then provide a framework for their comparative evaluation, the discovery of their relative strengths and weaknesses and the selection of the most suitable approach in the context of a given application.

Bibliography

[ACG+06] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *J. Artif. Intell. Res. (JAIR)*, 25:269–314, 2006.

[B. 06] B. Liang et al. Semantic Similarity Based Ontology Cache. In *APWeb*, 2006.

[BBL05a] F. Baader, S. Brandt, and C. Lutz. Pushing the el envelope. LTCS-Report LTCS-05-01, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005. See http://lat.inf.tu-dresden.de/research/reports.html.

[BBL05b] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the el envelope. In *IJCAI*, pages 364–369, 2005.

[BBL08] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the el envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.

[BCH] Jie Bao, Doina Caragea, and Vasant G Honavar. Towards collaborative environments for ontology construction and sharing.

[BCH06a] Jie Bao, Doina Caragea, and Vasant Honavar. Modular ontologies - a formal investigation of semantics and expressivity. In *ASWC*, pages 616–631, 2006.

[BCH06b] Jie Bao, Doina Caragea, and Vasant Honavar. On the semantics of linking and importing in modular ontologies. In *International Semantic Web Conference*, pages 72–86, 2006.

[BCH06c] Jie Bao, Doina Caragea, and Vasant Honavar. Package-based description logics - preliminary results. In *International Semantic Web Conference*, pages 967–969, 2006.

[BCH06d] Jie Bao, Doina Caragea, and Vasant Honavar. A tableau-based federated reasoning algorithm for modular ontologies. In *Web Intelligence*, pages 404–410, 2006.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BGvH+03] Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. C-owl: Contextualizing ontologies. In *International Semantic Web Conference*, pages 164–179, 2003.

[BL98] Tim Berners-Lee. Semantic Web Road Map. http://www.w3.org/DesignIssues/Semantic.html [2008-05-12], 1998.

[BLN86] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[BLSW00] Franz Baader, Carsten Lutz, Holger Sturm, and Frank Wolter. Fusions of description logics. In *Description Logics*, pages 21–30, 2000.

[BLSW02] Franz Baader, Carsten Lutz, Holger Sturm, and Frank Wolter. Fusions of description logics and abstract description systems. *J. Artif. Intell. Res. (JAIR)*, 16:1–58, 2002.

[BS03] Alexander Borgida and Luciano Serafini. Distributed description logics: Assimilating information from peer sources. *J. Data Semantics*, 1:153–184, 2003.

[CK07] Bernardo Cuenca Grau and Oliver Kutz. Modular ontology languages revisited. In *SWeCKa 2007: Proc. of the IJCAI-2007 Workshop on Semantic Web for Collaborative Knowledge Acquisition , Hyderabad, India, January 7, 2007*, 2007.

[D1.] D1.1.2 networked ontology model. `http://www.neon-project.org/web-content/index.php?option=com_weblinks&view=category&id=17&Itemid=73`.

[D7.a] D7.1.2 revised specifications of user requirements for the fisheries case study. `http://www.neon-project.org/web-content/images/Publications/neon_2008_d7.1.2.pdf`.

[D7.b] D7.2.2 revised and enhanced fisheries ontologies. `http://www.neon-project.org/web-content/images/Publications/neon_2007_d7.2.2.pdf`.

[Euz04] Jérôme Euzenat. An API for ontology alignment. In *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture notes in computer science*, pages 698–712, Hiroshima (JP), 2004.

[Fit02] Melvin Fitting. Fixpoint Semantics for Logic Programming - A Survey. *Theoretical Computer Science*, 278(1-2), 2002.

[G. 07] G. Deschrijver et al. A Bilattice-based Framework for Handling Graded Truth and Imprecision. *Uncertainty, Fuzziness and Knowledge-Based Systems*, 15(1), 2007.

[GGSS98] Chiara Ghidini, Chiara Ghidini, Luciano Serafini, and Luciano Serafini. Distributed first order logics. In *Frontiers Of Combining Systems 2, Studies in Logic and Computation*, pages 121–140. Research Studies Press, 1998.

[GHKS07] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. A logical framework for modularity of ontologies. In *IJCAI*, pages 298–303, 2007.

[GHW08] Jennifer Golbeck and Christian Halaschek-Wiener. Trust-Based Revision for Expressive Web Syndication. *Logic and Computation*, to appear, 2008.

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

[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.

[Gol06] J. Golbeck. Trust on the World Wide Web: A Survey. *Web Science*, 1(2):131–197, 2006.

[GPS04a] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Tableau algorithms for econnections of description logics. Technical report, 2004.

[GPS04b] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Working with multiple ontologies on the semantic web. In *International Semantic Web Conference*, pages 620–634, 2004.

[GPS06] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Combining owl ontologies using epsilon-connections. *J. Web Sem.*, 4(1):40–59, 2006.

[GR04] François Goasdoué and Marie-Christine Rousset. Answering queries using views: A krdb perspective for the semantic web. *ACM Trans. Internet Techn.*, 4(3):255–288, 2004.

[GST07] Chiara Ghidini, Luciano Serafini, and Sergio Tessaris. On relating heterogeneous elements from different ontologies. In *CONTEXT*, pages 234–247, 2007.

[HHR+06] Peter Haase, Pascal Hitzler, Sebastian Rudolph, Guilin Qi, Marko Grobelnik, Igor Mozetič, Damjan Bojadžiev, Jerome Euzenat, Mathieu d'Aquin, Aldo Gangemi, and Carola Catenacci. Context languages – state of the art. Deliverable D3.1.1, NeOn, 2006.

[HM05] Peter Haase and Boris Motik. A mapping system for the integration of owl-dl ontologies. In *IHIS*, pages 9–16, 2005.

[HT02] Ian Horrocks and Sergio Tessaris. Querying the semantic web: A formal approach. In *International Semantic Web Conference*, pages 177–191, 2002.

[HW05] Pascal Hitzler and Matthias Wendt. A uniform approach to logic programming semantics. *TPLP*, 5(1-2), 2005.

[HW07] Peter Haase and Yimin Wang. A decentralized infrastructure for query answering over distributed ontologies. In *SAC*, pages 1351–1356, 2007.

[KG06] Yarden Katz and Jennifer Golbeck. Social Network-based Trust in Prioritized Default Logic. In *Proc. of AAAI*, 2006.

[KKM06] Alissa Kaplunova, Atila Kaya, and Ralf Möller. Experiences with load balancing and caching for semantic web applications. In *Proc. of DL Workshop*, 2006.

[KLWZ04] Oliver Kutz, Carsten Lutz, Frank Wolter, and Michael Zakharyaschev. E-connections of abstract description systems. *Artif. Intell.*, 156(1):1–73, 2004.

[Kos00] Donald Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32(4):422–469, 2000.

[KRH08] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Elp: Tractable rules for owl 2. In Sheth et al. [SSD+08], pages 649–664.

[LBD+08] Chan Leduc, Jesus Barrasa, Jérôme David, Jérôme Euzenat, Raul Palma, Rosario Plaza, Marta Sabou, and Boris Villazón-Terrazas. Matching ontologies for context. Deliverable D3.2.2, NeOn, 2008.

[Len02] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[Mot06] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany, January 2006.

[MSS05] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for owl-dl with rules. *J. Web Sem.*, 3(1):41–60, 2005.

[P. 05] P. Haase et al. A Framework for Handling Inconsistency in Changing Ontologies. In *Proc. of ISWC*, 2005.

[Pol07] Axel Polleres. From SPARQL to rules (and back). In *Proc. of WWW*, 2007.

[Prz90] Teodor C. Przymusinski. The Well-Founded Semantics Coincides with the Three-Valued Stable Semantics. *Fundamenta Informaticae*, 13(4), 1990.

[QHS+09] Guilin Qi, Peter Haase, Simon Schenk, Steffen Stadtmüller, and Pascal Hitzler. D1.1.2 d1.2.4 inconsistency-tolerant reasoning with networked ontologies, 2009.

[R. 04] R. Gavriloaie et al. No Registration Needed: How to use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web. In *Proc. of ESWS*, 2004.

[RAC+06] Marie-Christine Rousset, Philippe Adjiman, Philippe Chatalic, François Goasdoué, and Laurent Simon. Somewhere in the semantic web. In *SOFSEM*, pages 84–99, 2006.

[SBT05] Luciano Serafini, Alexander Borgida, and Andrei Tamilin. Aspects of distributed and modular ontology reasoning. In *IJCAI*, pages 570–575, 2005.

[Sch08] Simon Schenk. On the semantics of trust and caching in the semantic web. In Sheth et al. [SSD+08], pages 533–549.

[SHH07] M. Stollberg, M. Hepp, and J Hoffmann. A Caching Mechanism for Semantic Web Service Discovery. In *Proc. of ISWC*, 2007.

[SS08a] Simon Schenk and Steffen Staab. Networked Graphs: A Declarative Mechanism for SPARQL Rules, SPARQL Views and RDF Data Integration on the Web. In *Proc. of WWW*, 2008.

[SS08b] Anne Schlicht and Heiner Stuckenschmidt. Distributed resolution for alc. In *Description Logics*, 2008.

[SSD+08] Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad Thirunarayan, editors. *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, volume 5318 of *Lecture Notes in Computer Science*. Springer, 2008.

[SSST08] Bernhard Schueler, Sergej Sizov, Steffen Staab, and Duc Thanh Tran. Querying for meta knowledge. In *Proceedings of WWW*, Bejing, China, 4 2008.

[SSW05] Luciano Serafini, Heiner Stuckenschmidt, and Holger Wache. A formal investigation of mapping languages for terminological knowledge. In *BNAIC*, pages 379–380, 2005.

[ST05] Luciano Serafini and Andrei Tamilin. Drago: Distributed reasoning architecture for the semantic web. In *ESWC*, pages 361–376, 2005.

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

[Stu06] Heiner Stuckenschmidt. Implementing modular ontologies with distributed description logics. In *WoMO*, 2006.

[TF05] S. Tessaris and E. Franconi. Rules and Queries with Ontologies: a Unifying Logical Framework. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, volume 147 of *CEUR Workshop Proceedings*, Edinburgh, Scotland, UK, July 2005. CEUR-WS.org.

[vRS91] Allen van Gelder, Kenneth Ross, and John S. Schlipf. The Well-Founded Semantics for General Logic Programs. *J. of the ACM*, 38(3), 1991.

[ZD08] Antoine Zimmermann and Chan Le Duc. Reasoning with a network of aligned ontologies. In *RR*, pages 43–57, 2008.

[ZE06] Antoine Zimmermann and Jérôme Euzenat. Three semantics for distributed systems and their relations with alignment composition. In *International Semantic Web Conference*, pages 16–29, 2006.

[Zim07] Antoine Zimmermann. Integrated distributed description logics. In *Description Logics*, 2007.

[ZL08] Antoine Zimmermann and Chan Leduc. Reasoning on a Network of Aligned Ontologies. In Thomas Eiter, Diego Calvanese, and Georg Lausen, editors, *The Second International Conference on Web Reasoning and Rules Systems, RR 2008, Karlsruhe, Germany, October 31st-November 1st, 2008, Proceedings*, October 2008. to appear.