Linköping Studies in Science and Technology. Thesis No. 1644

Licentiate Thesis

Integration of Ontology Alignment and Ontology Debugging for Taxonomy Networks

by

Valentina Ivanova

Department of Computer and Information Science
Linköping University
SE-581 83 Linköping, Sweden

Linköping 2014

This is a Swedish Licentiate's Thesis

Swedish postgraduate education leads to a Doctor's degree and/or a Licentiate's degree.

A Doctor's degree comprises 240 ECTS credits (4 year of full-time studies).

A Licentiate's degree comprises 120 ECTS credits.

Copyright © 2014 Valentina Ivanova

ISBN 978-91-7519-417-2

ISSN 0280-7971

Printed by LiU Tryck 2014

URL: <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-102953>

Abstract

Semantically-enabled applications, such as ontology-based search and data integration, take into account the semantics of the input data in their algorithms. Such applications often use ontologies, which model the application domains in question, as well as alignments, which provide information about the relationships between the terms in the different ontologies.

The quality and reliability of the results of such applications depend directly on the correctness and completeness of the ontologies and alignments they utilize. Traditionally, ontology debugging discovers defects in ontologies and alignments and provides means for improving their correctness and completeness, while ontology alignment establishes the relationships between the terms in the different ontologies, thus addressing completeness of alignments.

This thesis focuses on the integration of ontology alignment and debugging for taxonomy networks which are formed by taxonomies, the most widely used kind of ontologies, connected through alignments.

The contributions of this thesis include the following. To the best of our knowledge, we have developed the first approach and framework that integrate ontology alignment and debugging, and allow debugging of modelling defects both in the structure of the taxonomies as well as in their alignments. As debugging modelling defects requires domain knowledge, we have developed algorithms that employ the domain knowledge intrinsic to the network to detect and repair modelling defects.

Further, a system has been implemented and several experiments with real-world ontologies have been performed in order to demonstrate the advantages of our integrated ontology alignment and debugging approach. For instance, in one of the experiments with the well-known ontologies and alignment from the Anatomy track in Ontology Alignment Evaluation Initiative 2010, 203 modelling defects (concerning incomplete and incorrect information) were discovered and repaired.

This work has been supported by the Swedish National Graduate School in Computer Science (CUGS), the Swedish e-Science Research Center (SeRC) and Vetenskapsrådet (VR).

Acknowledgements

When life brought me to Sweden I had never imagined the wonderful possibilities I would discover. They did not come for granted, though. The path through the research world is thorny, going up and down, turning at the most unpredictable moments. I believe I have managed to put those to my advantage and now I welcome the next challenge.

I am sincerely thankful to my supervisor Professor Patrick Lambrix who has introduced me to the challenging area of ontologies. While working under his supervision I have improved my calm judgement of circumstances and, in general, my analytical skills. He provided encouraging and relaxed work environment and guided me during all stages of this work. Thank you, Patrick!

I am especially grateful to Professor Nahid Shahmehri, my second supervisor, who is the main reason for me being at this university. She is the one who first believed in my research talent and kindly advised me. I am also thankful to Associate Professor Lena Strömbäck and David Byers who made me believe I possess the strength to take this adventure. They have introduced me to the wonderful world of research.

The time here would not have been that enjoyable without my colleagues who make the work environment so friendly. I also thank the people at the IDA administrative department, and especially Anne, for their timely and always kind assistance in various administrative issues. I say thank you to Brittany Shahmehri for proof reading this thesis and providing valuable remarks.

I am greatly thankful to my family and friends for their unquestioning support and encouragement. Their belief in the successful end of this adventure has always been driving me forward.

This work would not have been possible without my life partner Pavel. He shares the sunny and stormy weather with me. Thank you, Pavel, for your love and for being here!

*Valentina Ivanova
January 2014
Linköping, Sweden*

Contents

1	Introduction	1
1.1	Semantic Web	1
1.2	Ontologies	3
1.2.1	Ontology alignment	4
1.2.2	Ontology debugging	4
1.2.3	Ontology networks	5
1.2.4	Benefits from the integration of ontology alignment and ontology debugging	5
1.3	Problem formulation	6
1.4	Contributions	7
1.5	Thesis structure	8
1.6	List of publications	9
1.6.1	Thesis based on	9
1.6.2	Related publications	9
1.6.3	Other publications	10
2	Background	11
2.1	Ontologies	11
2.1.1	Components	12
2.1.2	Classification	15
2.1.3	Applications	17
2.2	Ontology alignment	17
2.3	Ontology debugging	20
2.3.1	Classification of defects	21
2.4	Definitions	23
2.4.1	Ontologies and ontology networks	23
2.4.2	Knowledge bases	23
3	Framework and Algorithms	25
3.1	Framework and workflow	26
3.2	Methods in the framework	28
3.2.1	Detect missing and wrong is-a relations and mappings	28
3.2.2	Repair missing and wrong is-a relations and mappings	31
3.3	Algorithms in the debugging component	35

3.3.1	Detect and validate candidate missing is-a relations and mappings	35
3.3.2	Repair missing and wrong is-a relations and mappings	38
3.4	Algorithms in the alignment component	43
3.4.1	Detect and validate candidate missing mappings	43
3.4.2	Repair missing and wrong mappings	44
3.5	Interactions between the alignment component and the debugging component	45
4	Implemented System	47
4.1	Detect and validate candidate missing is-a relations and mappings	48
4.1.1	Detect and validate candidate missing is-a relations	48
4.1.2	Detect and validate candidate missing mappings	49
4.2	Repair missing and wrong is-a relations and mappings	51
4.2.1	Repair wrong is-a relations and mappings	51
4.2.2	Repair missing is-a relations and mappings	52
5	Experiments and Discussions	55
5.1	Ontology debugging	55
5.1.1	OAEI Anatomy 2010	55
5.2	Integration of ontology debugging and ontology alignment	60
5.2.1	OAEI Anatomy 2011	60
5.2.2	OAEI Benchmark 2010	64
5.2.3	ToxOntology-MeSH use case	70
5.3	Discussion	76
6	Related work	79
6.1	Ontology debugging	79
6.1.1	Debugging modelling defects	79
6.1.2	Debugging semantic defects	82
6.2	Ontology alignment	86
6.3	Integration of ontology alignment and ontology debugging	88
7	Conclusions and Future Work	91
7.1	Conclusions	91
7.1.1	Debugging of ontologies and alignments	92
7.1.2	Benefits from the integration of ontology alignment and ontology debugging	92
7.1.3	Implemented system	93
7.2	Future work	93
7.2.1	Extending the system	94
7.2.2	Long-term future work	95

List of Figures

2.1	(Part of an) Ontology network.	13
2.2	Part of the is-a hierarchy in the Wine ontology.	14
2.3	Part of the Wine ontology.	15
2.4	A general alignment framework.	18
2.5	An unsatisfiable concept in the Pizza ontology.	22
3.1	Workflow.	27
3.2	Initialization for detection.	35
3.3	Initialization for repairing.	38
3.4	Algorithm for generating repairing actions for wrong is-a relations and mappings.	39
3.5	Algorithm for generating repairing actions for missing is-a relations and mappings.	41
4.1	Generating and validating CMIs.	49
4.2	Aligning.	50
4.3	Repairing wrong is-a relations.	51
4.4	Repairing missing is-a relations.	53

List of Tables

- 5.1 Ontology debugging: OAEI Anatomy 2010—ontologies and alignment. 56
- 5.2 Ontology debugging: OAEI Anatomy 2010—final result. . . . 56
- 5.3 Ontology debugging: OAEI Anatomy 2010—recommendations. 57
- 5.4 Ontology debugging: OAEI Anatomy 2010—first iteration results. 58
- 5.5 Ontology alignment and debugging: OAEI Anatomy 2011—Run I results—debugging of the alignment. 61
- 5.6 Ontology alignment and debugging: OAEI Anatomy 2011—Run I results—debugging of the ontologies. 62
- 5.7 Ontology alignment and debugging: OAEI Benchmark 2010—ontologies and alignments. 64
- 5.8 Ontology alignment and debugging: OAEI Benchmark 2010—Run I—final result. 65
- 5.9 Ontology alignment and debugging: OAEI Benchmark 2010—Run II—final result. 67
- 5.10 Ontology alignment and debugging: OAEI Benchmark 2010—comparison between Run I and Run II 68
- 5.11 Ontology alignment and debugging: ToxOntology-MeSH—validation of mapping suggestions—initial alignment. 71
- 5.12 Ontology alignment and debugging: ToxOntology-MeSH—changes in the alignment (equivalence mapping ($\equiv$), ToxOntology term is-a MeSH term ($\rightarrow$), MeSH term is-a ToxOntology term ($\leftarrow$), related terms (R), wrong mapping (W), removed (rem)). 73
- 5.13 Ontology alignment and debugging: ToxOntology-MeSH—changes in the structure of ToxOntology. 74

Chapter 1

Introduction

1.1 Semantic Web

The Web today provides an immense variety of structured, semi-structured and, most often, completely unstructured information sources—databases, web pages, documents, figures, etc.—interconnected through an enormous number of links. Every minute different agents—both human and artificial—try to make sense out of the data, integrating different data sources in order to fulfill private and professional requirements.

In order to explore and employ the available data, the agents should be able to understand the message it conveys and formulate meaningful queries. Extracting the meaning, however, is a task that can only be performed by a human agent. Currently, computers only visualize and store the data without “understanding” the knowledge it conveys. The machines can do nothing to extract the semantics—they only “see” strings of symbols where people see words, phrases and sentences. Searching with search engines, until recently, was mainly based on string matching without considering the semantics of the input.

Making information machine-understandable is a key problem nowadays—for example, explaining to the computer what “rock” is. Terms should be considered in their context since it sometimes occurs that the same term is used to represent different concepts—for instance, *rock* as in rock music and *rock* as a geological concept. With time, the meanings of the terms change and new meanings for existing terms appear—for instance, *mouse* as a small mammal and *mouse* as a pointing device. Thus, in order to understand the intended meaning, the agents have to utilize matching definitions for the terms they use.

Information sources represent various domains, points of view and intended applications. They often overlap. For the purpose of different applications, for instance, data integration and agent communication, it is often necessary to know the relationship between the data available from separate

sources or between different versions of the same source. In order to figure out these relationships the agents must understand the meaning the data conveys.

The huge number of information sources at agents' disposal are often in different states—they may cover a topic area partially or may not be up to date—thus providing incomplete information for the area. Combining data from different sources, which have been developed to serve different applications, may lead to an inconsistent representation of an area. As a consequence the agents may use incomplete, inconsistent and erroneous data as input for their algorithms.

These problems have catalyzed the evolution of the Web towards the Semantic Web, where machines can “understand” and process data without human interaction. As a result the vision of the Semantic Web is coming into reality—just months ago Google introduced Google Knowledge Graph—enabling semantic search capabilities for their search engine. The rapid development of semantic technologies increasingly influences all aspects of our lives—with life sciences being one of the first domains to adopt the concept of ontologies and to benefit from their knowledge representation capabilities. Many large ontologies, such as SNOMED CT [11], Gene Ontology [15], MeSH [6], etc., have already been developed in this domain.

The concept of the Semantic Web encompasses a set of technologies that enable computers to “understand” the data they store. It is an extension of the Web, not its replacement. This vision was first introduced by Tim Berners-Lee, James Hendler and Ora Lassila in 2001 in a publication [21] in *Scientific American*. Through several examples the publication illustrates a world where intelligent agents explore the Web and collect and integrate relevant information from diverse data sources in order to fulfill complicated tasks without human guidance. By contrast, today machines can perform only simple tasks precisely specified in advance. Since they do not “understand” the meaning of the data they collect, they cannot combine the output of multiple tasks in a single functional output and draw conclusions (humans have to do that).

To illustrate the concept of the Semantic Web, consider the example of a sophisticated task, such as planning and scheduling a trip to a conference. The trip encompasses different aspects, such as:

- the traveler's daily schedule—available in the traveler's calendar—listing various appointments;
- flight schedules—the selected flights should fit the conference and personal schedule and should be compatible with different personal preferences and restrictions—transfer times on intermediate stops (compatible with the size of the airport/time for transfer), possession of a membership card for a particular airline, avoiding countries with transit visa requirements, etc.;
- hotel accommodation—it should be at a reasonable distance from the conference location, recommended by the conference organizers, with

available rooms for the conference period, avoiding neighbourhoods with high crime rates, etc.;

- transport between the airport, the hotel and the conference venue—possible delays and transfer times should be considered, etc.;
- entertainment/sightseeing during free time—finding cultural/sport/other activities that do not conflict with the conference schedule;
- food—finding high-rated restaurants meeting personal dietary requirements;
- etc.

The traveler can take all details into account, search and then integrate relevant information from different data sources to schedule the trip. However, this is still not the case for machines—each of the items in the list requires at least one search in various search engines where the inputs and outputs of the different searches are more or less connected. First, such an agent should locate the sources containing relevant information for the current task—plane tickets providers, hotels, restaurants guides, etc. The sources often have overlapping content and may contain outdated data, the sources appear and disappear. Then the data relevant for the current task should be retrieved. However, data coming from heterogeneous data sources have different formats and discrepancies in meaning that hinder the filtering of relevant data. Finally, the relevant information should be integrated in order to provide a complete trip and conference schedule. The key issue in all steps is interpreting every piece of data—something machines still cannot do autonomously.

1.2 Ontologies

How can the Semantic Web help a machine to autonomously schedule a trip?

The bullets in the list above are related to different data sources or agents providing the desired data. If an intelligent agent is doing the work on our behalf, it should be able to communicate with other agents regarding the data they possess or it should be able to query data sources with relevant queries. To fulfill these tasks the agents should have a shared understanding of the terms they use.

In this context **ontologies** are considered the “Silver bullet” for the Semantic Web. They provide mutual understanding of a domain, defining concepts, relations between concepts and rules for creating new concepts. For instance, the different aspects of the trip can be represented as different domain ontologies—accommodation ontology, restaurant ontology, transport ontology, etc. or a single travel ontology that includes all these concepts in an individual ontology. Thus, the ontologies enable the communication between the agents, providing common understanding of the domain in question. Applications, such as agent communication, that employ semantic technologies, in this case ontologies, are called **semantically-enabled applications**.

The ontologies are usually represented in ontology languages, such as OWL, RDF, etc. These languages often contain statements that can be used for logical inferences, for instance, in description logics (DL) systems, i.e., new knowledge (not explicitly recorded) can be inferred from the knowledge already stored.

1.2.1 Ontology alignment

It often happens, however, that agents employ different ontologies in the same domain, as they are developed by different organizations according to their needs and points of view. Similarly, the data sources could be annotated, i.e., their constructions could be labeled with terms from different, but similar ontologies. Thus, in order to communicate with each other and to formulate relevant queries the agents need to know how the concepts in the different ontologies are related. This is studied in the area of **ontology alignment**, which employs different techniques in order to find related concepts in different ontologies. A set with relations representing related concepts in two different ontologies is called an **alignment**. A single relation in the alignment is called a **mapping**. The alignments are usually created by ontology developers with or without the assistance of ontology alignment systems.

1.2.2 Ontology debugging

Furthermore, many ontologies are domain specific and are developed by domain experts who frequently lack proficiency in knowledge representation. For instance, it is very common that people who are not experts in knowledge representation confuse equivalence, is-a and part-of relations (e.g., [27]). Another common issue appears as ontologies grow in size, i.e., intended and unintended entailments become difficult to follow. As a consequence, in large ontologies, and in smaller ones, there are usually **defects**—incorrect (wrong), incomplete (missing) and contradictory (inconsistent) information. The same issues are also relevant to the development of alignments. Using ontologies and alignments with defects in semantically-enabled applications, such as agent communication or ontology-based search and data integration, may lead to incorrect conclusions while valid conclusions may be missed. Discovering and resolving defects in the ontologies and their alignments are the subjects of the **ontology debugging** area.

The following example highlights the influence of defects, in this case incomplete/incorrect results of an ontology-based search. The familiar string search only retrieves documents which contain the term(s) we are searching for. In comparison, an ontology-based search retrieves documents containing not only the term(s) in question but also documents containing relevant (often more specific) terms by exploring the structure of an ontology. Thus, the ontology-based search provides more relevant results. In the example here the MeSH thesaurus [6] is an ontology that is used for querying the

PubMed database [10]. According to the domain knowledge the Scleritis concept in MeSH is a sub-concept of the Scleral Diseases concept and it is included during a search for Scleral Diseases (1363 articles are retrieved). However, if the relation between Scleritis and Scleral Diseases were missing, only 613 articles would be retrieved, i.e., 55% of the results would be missed. If the relation were wrong (i.e., the relation between Scleritis and Scleral Diseases does not hold in the reality but exists in MeSH), incorrect results would be acquired.

There are different types of defects in ontologies [48]. Syntactic defects, such as wrong or missing tags, can be discovered and resolved by (XML) parsers. Semantic defects introduce contradictory information in the ontologies. They can be found by software programs called reasoners, for instance, DL reasoners. Modelling defects require domain knowledge to detect and resolve. For instance, missing and wrong structures in ontologies and their alignments are modelling defects. (Wrong structure could be also a semantic defect.) The example above demonstrates missing and wrong subsumption relations in the structure of an ontology and their consequences for semantically-enabled applications.

1.2.3 Ontology networks

Ontologies connected through their alignments can be seen as a network—an **ontology network**. The network itself provides more knowledge for the domain than an ontology or a pair of ontologies connected through an alignment since each ontology represents a different level of details reflecting the view and the interests of its developers and intended applications. This is available knowledge *intrinsic* to the network, which is a source of valuable domain information and provides a powerful automatic defect detection mechanism. It can be used for debugging modelling defects in single ontologies and pairs of ontologies and their alignments.

1.2.4 Benefits from the integration of ontology alignment and ontology debugging

This thesis focuses on debugging of modelling defects in the context of an ontology network. The algorithms presented rely heavily on the knowledge intrinsic to the network as a source of domain knowledge. However, it can sometimes occur that the network cannot be created due to the absence of alignments between the ontologies. In this case ontology alignment systems can be used to provide alignments.

In the context of an integration of ontology alignment and debugging, ontology alignment can be seen as a special kind of debugging of missing relationships between concepts in different ontologies, where alignment algorithms are employed to discover missing relationships. Both correct and incorrect relations obtained during the alignment process could then be used

for further debugging and alignment of the ontologies. In short, ontology alignment provides or extends (already available) alignments which are further necessary for ontology debugging.

Furthermore, some alignment algorithms, like those based on the structure of the ontology, depend on the correctness and completeness of the aligned ontologies. Ontology alignment preprocessing strategies also take advantage of knowledge of the structure of the alignments, if available. Debugging of modelling defects improves the structures of ontologies and their associated alignments. Another advantage is that the repairing algorithms used for ontology debugging can be adapted for the purposes of ontology alignment. This would provide alternatives to the process of creating alignments by simply adding the missing mappings, as is done in many pure ontology alignment systems.

Thus, integration of ontology alignment and debugging would provide additional benefits for both areas and would significantly improve the quality of both the ontologies and their alignments.

1.3 Problem formulation

The discussion above highlights the issues caused by defects in the ontologies and alignments and their consequences for the results of semantically-enabled applications. The quality and reliability of the results of such applications is directly dependent on the quality and reliability of the ontologies and alignments they employ. A key step towards achieving high-quality ontologies and alignments is discovering and resolving various defects. The modelling defects are particularly severe since domain knowledge is required for their debugging. This thesis considers **taxonomies**, as they are the most widely used kind of ontologies, connected through their alignments in **taxonomy networks**. It addresses two questions:

- *How to debug modelling defects, such as missing and wrong structure in taxonomies as well as their alignments, in the context of a taxonomy network?*

Since debugging usually consists of two phases, a detection and repairing phase, this question encompasses two more precise questions:

- *How to detect modelling defects without external knowledge?*—recognizing defects is the first step in their debugging;
- *How to repair modelling defects?*—After the defects are detected, they should be repaired. A trivial approach is to add or remove the missing or wrong structure. However, other approaches may contribute to a more complete representation of the domain in question and thus they could be preferred by domain experts as more beneficial;

In the process of exploring different possibilities for detecting modelling defects, the area of ontology alignment has come to our attention. Furthermore, we have found promising hints that the integration of ontology alignment and debugging will provide benefits for both areas. We have studied these expectations in the context of the following question:

- *What are the benefits from the integration of ontology alignment and debugging for*
 - *the ontology alignment?*
 - *the ontology debugging?*

1.4 Contributions

The main contribution of this thesis can be summarized in the following sentence: This is the first approach, to the best of our knowledge, which integrates ontology alignment and ontology debugging and allows debugging of modelling defects both in the structure of the ontologies as well as in their alignments. Below the contributions are listed in connection with the research questions:

How to debug modelling defects, such as missing and wrong structure in taxonomies as well as their alignments, in the context of a taxonomy network?

- **We have developed a unified approach for debugging modelling defects, such as missing and wrong structure, in taxonomies and their alignments without external knowledge.** A previous work, described in [67], considers debugging missing and wrong subsumption relations in taxonomies in the context of taxonomy networks. In this thesis we have extended the approach and framework, developing algorithms for debugging missing and wrong subsumption and equivalence mappings between taxonomies employing the knowledge intrinsic to the taxonomy network;
- **We have extended the system, described in [67],** implementing the algorithms for debugging missing and wrong subsumption and equivalence mappings;
- **We have performed experiments with existing real-world ontologies** using the extended system.

What are the benefits from the integration of ontology alignment and debugging?

- **We have developed a framework for integration of ontology alignment and ontology debugging.** Both areas take advantage of the integration—alignment algorithms are used to create a taxonomy network, or extend an existing one, where the knowledge intrinsic to

the network is used for detecting and repairing modelling defects in the taxonomies and their alignments. The debugging process improves the structure of the taxonomies and their alignments, which is important for some ontology alignment strategies. Further, in the integrated framework, alignment can be seen as a special kind of debugging and debugging using the knowledge intrinsic to the network can be seen as a special alignment algorithm;

- **We have, further, extended the system** to integrate ontology alignment algorithms. After the integration of the ontology alignment and debugging two components can be distinguished in our system—a debugging component and an alignment component. The system can be used as an integrated ontology alignment and debugging system or each of the components can be used independently as a separate system.
- **We have performed experiments with existing real-world ontologies** using our integrated ontology alignment and debugging system. These experiments demonstrate the benefits from the integration of ontology alignment and debugging.

1.5 Thesis structure

The thesis is structured as follows: Chapter 2 gives background on ontologies and provides more details on ontology alignment and ontology debugging. At the end of that chapter several definitions relevant to the subsequent presentation are given. Chapter 3 introduces our integrated framework with its two components—the debugging component and the alignment component—along with their algorithms and workflow. Chapter 4 presents our integrated ontology alignment and debugging system which is based on the framework discussed in Chapter 3. The experiments performed with the system and a discussion of their results are shown in Chapter 5. Recent issues in the fields of ontology alignment and debugging are discussed in Chapter 6. Chapter 7 provides concluding remarks and directions for future work.

1.6 List of publications

1.6.1 Thesis based on

Journal article

- Lambrix P, Ivanova V, A unified approach for debugging is-a structure and mappings in networked taxonomies, *Journal of Biomedical Semantics*, 4:10, 2013.

Conference articles

- Ivanova V, Lambrix P, A Unified Approach for Aligning Taxonomies and Debugging Taxonomies and Their Alignments, 10th Extended Semantic Web Conference—ESWC 2013, LNCS 7882, pages 1–15, Montpellier, France, 2013.
- Ivanova V, Lambrix P, A System for Aligning Taxonomies and Debugging Taxonomies and Their Alignments, 10th Extended Semantic Web Conference Satellite Events—ESWC 2013, pages 152–156, Montpellier, France, 2013. Demo.

Workshop articles

- Ivanova V, Laurila Bergman J, Hammerling U, Lambrix P, Debugging Taxonomies and their Alignments: the ToxOntology-MeSH Use Case, 1st International Workshop on Debugging Ontologies and Ontology Mappings—WoDOOM 2012, pages 25–36, Galway, Ireland, 2012.
- Ivanova V, Lambrix P, A System for Debugging Taxonomies and their Alignments, 1st International Workshop on Debugging Ontologies and Ontology Mappings—WoDOOM 2012, pages 37–42, Galway, Ireland, 2012. Demo.

Video journal publication

- Ivanova V, Lambrix P, A System for Aligning Taxonomies and Debugging Taxonomies and Their Alignments, *Video Journal of Semantic Data Management Abstracts*, volume 2, 2013.

1.6.2 Related publications

Book chapter

- Lambrix P, Ivanova V, Dragisic Z, Contributions of LiU/ADIT to Debugging Ontologies and Ontology Mappings, in Lambrix, (ed), *Advances in Secure and Networked Information Systems—The ADIT Perspective*, pages 109–120, LiU Tryck / LiU Electronic Press, 2012.

Conference article

- Lambrix P, Dragisic Z, and Ivanova V, Get My Pizza Right: Repairing Missing is-a Relations in *ALC* Ontologies, 2nd Joint International Semantic Technology Conference—JIST 2012, LNCS 7774, pages 17–32, Nara, Japan, 2012.

Workshop articles

- Lambrix P, Wei-Kleiner F, Dragisic Z, Ivanova V, Repairing missing is-a structure in ontologies is an abductive reasoning problem, 2nd International Workshop on Debugging Ontologies and Ontology Mappings—WoDOOM 2013, CEUR Workshop Proceedings volume 999, pages 33–44, Montpellier, France, 2013.
- Cuenca Grau B, Dragisic Z, Eckert K, Euzenat J, Ferrara A, Granada R, Ivanova V, Jiménez-Ruiz E, Kempf A O, Lambrix P, Nikolov A, Paulheim H, Ritze D, Scharffe F, Shvaiko P, Trojahn C, Zamazal O, Results of the Ontology Alignment Evaluation Initiative 2013, 8th International Workshop on Ontology Matching—OM 2013, CEUR Workshop Proceedings volume 1111, pages 61–100, Sydney, Australia, 2013.

1.6.3 Other publications

Journal article

- Strömbäck L, Ivanova V, Hall D, Using Statistical Information for Efficient Design and Evaluation of Hybrid XML Storage, *International Journal On Advances in Software* 4:3–4, pages 389–400, 2012.

Conference articles

- Ivanova V, Strömbäck L, Creating Infrastructure for Tool-Independent Querying and Exploration of Scientific Workflows, 7th IEEE International Conference on eScience, pages 287–294, Stockholm, Sweden, 2011.
- Strömbäck L, Ivanova V, Hall D, Exploring Statistical Information for Applications-Specific Design and Evaluation of Hybrid XML storage, 3rd International Conference on Advances in Databases, Knowledge, and Data Applications—DBKDA 2011, pages 108–113, St. Maarten, The Netherlands Antilles, 2011. Best paper award.

Chapter 2

Background

This chapter provides background in the areas relevant to this work. They are presented with the help of several examples.

Section 2.1 discusses the term *ontology* presenting several definitions in the scientific literature. It then lists their components and shows several applications of ontologies in areas different from the Semantic Web. Sections 2.2 and 2.3 give an overview of the areas of ontology alignment and debugging. Formal definitions relevant to the subsequent presentation of this work are given in Section 2.4.

2.1 Ontologies

The term *ontology* originates from philosophy, where it denotes a branch dealing with the questions of being and existence. In the 80's the term was borrowed and introduced to Computer Science by the Artificial Intelligence community. There are different definitions for ontologies available in the scientific literature and some of the most popular are:

- An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary [71];
- An ontology is an explicit specification of a conceptualization [38];
- An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base [86];
- An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base [20];
- An ontology is a formal, explicit specification of a shared conceptualization [85];

All definitions share the view that ontologies explicitly describe a topic area. They model the world around us (or someone's view of the world)

explicitly defining the meaning of its concepts, the existing relationships between them (for instance, *part-of*, *is-kind-of*, *is-located-in*, *is-not*) and rules for creating new concepts. The last definition supplies an additional important feature of ontologies, i.e., they provide a shared understanding of the area in question. Ontologies vary in their components and consequently in complexity and knowledge representation capabilities.

Figure 2.1 illustrates a real-world example from the Anatomy track at the Ontology Alignment Evaluation Initiative (OAEI) 2011, [8], which will be further used throughout the thesis. Two parts of ontologies are shown—on the left is a piece of the Adult Mouse Anatomy Dictionary (AMA), [1], which models the anatomy of an adult mouse and on the right is a piece of the NCI Thesaurus anatomy (NCI-A), [7], which models the human anatomy.

Figures 2.2 and 2.3 show parts of the Wine ontology [13]. It specifies terms and relations in the wine and food domains and provides information about the type of wine suitable for a particular food.

2.1.1 Components

There are different views for the components of the ontologies. According to [53] the components of the ontologies, from a knowledge representation point of view, are as listed below. The authors of [29] define a similar set with components which they call minimal set of components.

- **concepts (also known as classes)** represent a group of entities in a domain. All rectangles in Figures 2.1 and 2.2 and the rectangles with circles in front of the labels in Figure 2.3 depict concepts in the ontologies;
- **instances (also known as individuals)** represent the actual entities. However, they are often not represented in ontologies. The instances in the ontology in Figure 2.3 are depicted with rectangles with rhombuses in front of the labels;
- **relations (also known as roles, properties, slots)** represent different relationships between the entities in a domain, such as *part-of*, *is-kind-of*, *is-located-in*, *is-not*, etc. The concepts in an ontology connected through is-a relations form the is-a hierarchy in the ontology. Analogously, the part-of hierarchy in the ontology consists of all concepts connected through part-of relations. *Is-a* relations (known also as is-kind-of, subclass or subsumption relations) are the most often used in ontologies since they represent a common relationship that occurs in many domains. An is-a relation shows that one set of entities is a subset of another set of entities. For instance, the relation *limb bone is-a bone* in Figure 2.1 shows that a limb bone is a kind of bone. The directed solid edges in Figure 2.1 represent the is-a structures in the ontologies. The edges in Figure 2.2 illustrate the subclass (is-a) relations in the Wine ontology. Other relations depict different dependencies between the entities—the dashed edges in Figure 2.3 illustrate

Figure 2.1: (Part of an) Ontology network.

Figure 2.2: Part of the is-a hierarchy in the Wine ontology.

Figure 2.3: Part of the Wine ontology.

two relations—*locatedIn* between the concepts *Wine* and *Region*, and *hasMaker* between the concepts *Wine* and *Winery*;

- **axioms** represent facts that are always true in the area described by the ontology and are not represented by the other components. They are used to provide consistent representation of the domain. For instance (examples from the Wine ontology):

- domain restrictions (*adjacentRegion* has values from *Region*);
- cardinality restrictions (*VintageYear* can have at most one value);
- disjointness restrictions (*Fruit* is-not *Meat*).

2.1.2 Classification

The ontologies can be classified according to various criteria. Several one-dimensional classifications (utilizing only a single criterion) are shown in [78] in the context of a discussion regarding the usage of ontologies in software engineering and technology. Most of them consider how general the represented concepts are and the scope of the application of the ontologies—general, domain, task, application, etc. concepts/scopes. One of the classifications, given by [66] in a discussion regarding desirable and required features for ontology languages, considers the complexity of the relationships that can be depicted in the domain in question. This classification,

referred to as “richness of the internal structure”, and the classification in [90] referred to as “subject of conceptualization” are used as a foundation for the two-dimensional classification developed in [36]. Depending on the “richness of the internal structure”, i.e., the knowledge representation capabilities of an ontology, [36] defines eight categories of ontologies ranging from informally specified ontologies to ontologies precisely specified by formal languages. These eight categories can be further compacted to the four presented in [89] and [39] and listed here:

- **glossaries and data dictionaries** contain concepts with or without their definitions in a natural language;
- **thesauri and taxonomies** introduce, together with the concepts and their definitions, synonyms and relations such as narrower and broader;
- ontologies represented by **metadata, XML schemas, data models**. These models additionally provide properties and value restrictions. This category includes the so called strict is-a relations, which correspond to the is-a relations in our work;
- ontologies represented by **logical languages**. The ontologies represented by formal languages hold the most expressive knowledge representation capabilities.

Another categorization method, given in [53], takes into account the components and the information represented by them and arrives at a similar classification:

- **controlled vocabularies** contain only concepts;
- **taxonomies** contain concepts connected in a hierarchy through is-a relations (these is-a relations correspond to the so called strict is-a relations above);
- **thesauri** contain concepts and a set with predefined relations, e.g., WordNet [69], MeSH [6];
- ontologies represented by **data models**, for instance, EER and UML, include restricted forms of axioms, properties and cardinality constraints together with the concepts and relations. (This category corresponds to the metadata, XML schemas, data models category above.);
- ontologies represented by **logics**, e.g., description logics, are the most expressive kind of ontologies. They employ formal languages with their own syntax, semantics and inference mechanism along with the concepts, relations and axioms. Description logics vary in their expressivity. (This category corresponds to the logical languages above.).

Both classifications encompass the whole range of ontologies regarding their knowledge representation capabilities—from the so called lightweight to the heavyweight ontologies. The advantage of the former group is their simplicity at the price of reduced expressivity and high ambiguity. The advantage of the ontologies in the latter group is their powerful capabilities for expressivity and inference mechanism at the price of complex development.

2.1.3 Applications

The ontologies have a wide range of applications in the Semantic Web:

- provide mutual understanding of a domain enabling knowledge sharing and reuse, and facilitating autonomous communication between different intelligent agents as discussed in Tim Berners-Lee, James Hendler and Ora Lassila's publication, [21];
- serve as a repository of information [89];
- provide a query model for information sources explicitly structuring the domain knowledge [91], [70];
- data integration of heterogeneous information sources [91], [54], [70].

Ontologies are a key technology for the Semantic Web and are intensively employed in other areas as well:

- Artificial Intelligence—knowledge representation and reasoning;
- Software Engineering—in [25] two applications of ontologies in this area are discussed—sharing terminology and knowledge, and filtering knowledge in the process of definition of models and metamodels; [40] discusses the ontologies in the context of the Software Engineering life-cycle;
- Systems Engineering—ontologies are used for the purposes of re-usability, reliability and specification as pointed out in [88];
- Bioinformatics and Systems Biology—specification, ontology-based search, data integration and exchange as discussed in [53] and [64];
- E-commerce—such as GoodRelations [4].

2.2 Ontology alignment

In the fields pioneering ontology development, such as the life sciences, a number of ontologies have already been created by different organizations representing their needs and views of the domain. It may happen that the data sets are annotated with terms from different but overlapping ontologies, which is an obstacle for their integration. The communication between the intelligent agents using different ontologies is hindered as well.

A solution to these issues demands knowledge about the relationships between the concepts in the different ontologies. This is the field of research of the continuously growing **ontology alignment** community. The increased interest in the topic has led to the organization of an annual evaluation initiative—the Ontology Alignment Evaluation Initiative [8]—where the developers and researchers can evaluate their tools and algorithms in various tracks.

A set of relations showing the relationships between concepts in two different ontologies is called an **alignment**. Each relation in the set is called a **mapping**. We call the concepts that participate in mappings **mapped concepts**. Each mapped concept can participate in multiple mappings

and alignments. In our work we consider **equivalence** and **subsumption mappings**. The equivalence mappings connect two concepts which represent the same set of entities. The subsumption mappings are relations between two concepts, where one of the concepts represents a set of entities that is a subset of the other concept. Ontology alignment systems are used to facilitate the development of alignments.

The ontologies in Figure 2.1 are connected through an alignment, depicted with the dashed edges. It consists of 10 equivalence mappings. One of these mappings represents the fact that the concept *bone* in the first ontology is equivalent to the concept *bone* in the second ontology. The same applies for the concept *nasal bone* in the first ontology and the concept *nasal bone* in the second, and so on. As these four concepts appear in mappings, they are mapped concepts. An example of a subsumption mapping would be (*AMA:maxilla*, *NCI-A:irregular bone*) (not shown in Figure 2.1, but derivable through *NCI-A:maxilla*)—*AMA:maxilla* is *subsumed by* *NCI-A:irregular bone* and accordingly *NCI-A:irregular bone* *subsumes* *AMA:maxilla*.

A set of ontologies connected through their alignments form a network—an **ontology network**.

Figure 2.4: A general alignment framework.

Ontology alignment framework. With the increasing number of on-

tologies, their concepts and relations, the demand for automated or semi-automated ontology alignment systems grows stronger. Figure 2.4 shows a general semi-automated ontology alignment framework presented by Patrick Lambrix and Qiang Liu in 2009 in [58]. Many ontology alignment systems conform to it. The input for the system are two ontologies and the output is an alignment. The alignment process presented in the framework goes through two phases. In Phase I the system generates possible mappings that are presented to the user for a manual validation in Phase II. Phase I usually includes 3 steps:

Preprocessing step includes preliminary data processing, for instance, partitioning of the input ontologies or removing modifiers, such as definite and indefinite noun modifiers. [58] presents strategies for using **partial alignment (PA)** in this and the following steps.

Running matchers to compute similarity values between pairs of concepts in the different ontologies. The similarity values represent an estimate that two concepts are connected. The **matchers** employ various strategies as described in [63] and listed below:

- **linguistic strategies** explore the linguistic similarity of the concepts and relations labels. For instance, the labels are represented as sets of consecutive characters and then the similarity values between the concepts are calculated based on these sets. Another strategy counts the number of insertions, deletions and modifications needed in order to make one of the concepts identical to the other;
- **structure-based strategies** rely heavily on the structure of the ontologies. They are based on the heuristic that, given two ontologies and their alignment, if two regions in the different hierarchies are between pairs of concepts with high similarity values then there could be matching concepts between both regions;
- **constraint-based strategies** consider the concepts and properties data types and cardinalities. They are usually used to provide supplementary information, not as primary matchers;
- **instance-based strategies** assign similarity values based on the shared instances between the concepts in the different ontologies. The instances can be acquired from curated scientific resources (for instance, PubMed [10] in life sciences);
- **strategies based on auxiliary sources** use domain knowledge available from external sources, such as WordNet [69] and UMLS [14], to find additional information for the concepts (synonyms) and the relationships between them.

Combining and filtering the similarity values obtained from the different matchers—most often the similarity values are combined using a weighted-sum approach in which each matcher is given a weight and the final similarity value is the weighted sum of the similarity values divided by the sum of the weights of the matchers. Another approach uses the maximal similarity value obtained from the matchers.

Furthermore, those pairs of concepts with similarity values equal to or higher than a given threshold are retained in order to obtain the **mapping suggestions**. Another filtering strategy, presented in [26], uses two thresholds—those pairs equal to or above the higher threshold are directly retained as mapping suggestions while those between the two thresholds are filtered out with respect to the structure of the ontology and the pairs with similarity values above the higher threshold.

In Phase II the mapping suggestions are presented for validation to the user who can accept or reject them. The accepted suggestions become part of the final alignment. Both the accepted and the rejected mapping suggestions are further used in the alignment process to avoid unnecessary computations and validations. A conflict checker may be used to detect possible conflicts.

The alignment algorithms are evaluated mainly according to their **precision**, **recall** and **f-measure**. The precision measure reflects the ratio between the correct pairs and all pairs of concepts in the newly created alignment. The recall measure reflects the ratio between the pairs that should be retrieved by the alignment algorithms (it is known that they are correct according to, for instance, a reference alignment) and the correct pairs that have actually been retrieved. The f-measure connects precision and recall.

2.3 Ontology debugging

Developing ontologies and alignments is not a trivial task. As ontologies grow in size and complexity, the intended and unintended entailments become difficult to follow. As mentioned above, the ontologies are usually developed by domain experts who often are not expert in knowledge representation and may not have experience with the capabilities of the knowledge representation languages (good/bad practices). The same issues apply for developing alignments. Concept discrepancies between the different ontologies, for instance, using one term for different real-world entities, are also sources of defects during the alignment. The experiment in Section 5.2.3 presents such an example. During the alignment, the domain expert marked the *metabolism* concepts in both ontologies as equivalent. However, it was discovered that they are not equivalent during the following debugging process. As a consequence, the ontologies, alignments and integrated ontology network may be incorrect, incomplete or inconsistent. Using them in semantically-enabled applications may lead to entailment of incorrect conclusions or valid conclusions may be missed.

Recall the example from Subsection 1.2.2 regarding missing/wrong subsumption relations in the MeSH hierarchy. It clearly shows how substantial the influence of such defects for the semantically-enabled applications may be.

Another example demonstrates the way communication can be disrupted between two intelligent agents using two different ontologies in the medical

domain. For the same group of eye related illnesses, one of the ontologies uses the concept *Eye Diseases*, while the other uses the concept *Eye Disorders*. If a mapping between these two concepts is not available, the two agents will not be able to share data (understand each other) regarding these concepts. If the mapping were wrong they would exchange incorrect information.

To achieve highly reliable results from the semantically-enabled applications, it is necessary to have both high quality ontologies and high quality alignments. Debugging of the ontologies and alignments is a key step towards eliminating defects in them, which is essential for obtaining high-quality results in the semantically-enabled applications. The **ontology debugging** area deals with discovering and resolving defects in the structure of the ontologies and their alignments. To highlight the growing importance of the field the International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM) was founded in 2012.

2.3.1 Classification of defects

The defects differ [48] in nature and, consequently, in the complexity of their detection and repair.

- **syntactic defects**, such as incorrect format or a missing tag, are trivial to find and resolved using parsers;
- **semantic defects** have their origin in unintended inferences (the example in Figure 2.5 illustrates semantic defects in the Pizza ontology [12]):
 - **unsatisfiable concepts** are concepts that cannot have any instances. Figure 2.5 shows an unsatisfiable concept *CheeseyVegetableTopping*. It is defined as a *CheeseTopping* and as a *VegetableTopping* at the same time where *CheeseTopping* and *VegetableTopping* are disjoint concepts. Nothing can be *CheeseTopping* and *VegetableTopping* at the same time, i.e., the *CheeseyVegetableTopping* will not have any instances and it is an unsatisfiable concept;
 - **incoherent ontologies** are ontologies that contain unsatisfiable concepts. The Pizza ontology contains at least one unsatisfiable concept (*CheeseyVegetableTopping*), i.e., it is an incoherent ontology;
 - **inconsistent ontologies** contain inconsistencies, for example, an instance that belongs to an empty set. In this example if *CheeseyVegetableTopping* has instances the ontology would be inconsistent.

The semantic defects can be found using reasoners, which are software application programs that are able to derive logical consequences from a given set of asserted axioms—Pellet [9], Jena [2], FaCT++ [3], HermiT [5], etc.

Figure 2.5: An unsatisfiable concept in the Pizza ontology.

- **modelling defects**, such as missing and wrong relations, require domain knowledge to detect and resolve. With very few exceptions there is lack of system support for debugging such defects. The examples at the beginning of this section show modelling defects—**missing and wrong is-a relations and mappings**.

The missing is-a relations in Figure 2.1 are *(nasal bone, bone)*, *(maxilla, bone)*, *(lacrimal bone, bone)* and *(jaw, bone)* in the left ontology (AMA), and *(metatarsal bone, foot bone)* and *(tarsal bone, foot bone)* in the right ontology (NCI-A). The wrong is-a relations are *(upper jaw, jaw)* and *(lower jaw, jaw)* in the right ontology.

2.4 Definitions

This subsection presents several formal definitions that will be used throughout the thesis.

2.4.1 Ontologies and ontology networks

The focus of our work is on taxonomies, which are the most widely used kind of ontologies. ‘Taxonomy’ and ‘ontology’ are used interchangeably in the next chapters. The taxonomies consist of named concepts and subsumption (is-a) relations between the concepts. The following definition applies.

Definition 1 A **taxonomy** $\mathcal{O}$ is represented by a tuple $(\mathcal{C}, \mathcal{I})$ where $\mathcal{C}$ is its set of named concepts and $\mathcal{I} \subseteq \mathcal{C} \times \mathcal{C}$ is a set of asserted is-a relations, representing the is-a structure of the ontology.

The ontologies are connected into a network through alignments. We currently consider *equivalence* mappings ($\equiv$) and *is-a* mappings (*subsumed-by* ($\rightarrow$) and *subsumes* ($\leftarrow$)).

Definition 2 An **alignment** between ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ is represented by a set $\mathcal{M}_{ij}$ of pairs representing the mappings, such that for concepts $c_i \in \mathcal{O}_i$ and $c_j \in \mathcal{O}_j$: $c_i \rightarrow c_j$ is represented by (c_i, c_j) ; $c_i \leftarrow c_j$ is represented by (c_j, c_i) ; and $c_i \equiv c_j$ is represented by both (c_i, c_j) and (c_j, c_i) .¹

Definition 3 A **taxonomy network** $\mathcal{N}$ is a tuple $(\mathbb{O}, \mathbb{M})$ with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^n$ the set of the ontologies in the network and $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i < j}^n$ the set of representations for the alignments between these ontologies.

Without loss of generality, we assume that the sets of named concepts for the different ontologies in the network are disjoint.

A significant part of our approach relies on knowledge *intrinsic* to the network, i.e., knowledge logically derivable from the network. The domain knowledge of an ontology network is represented by its induced ontology.

Definition 4 Let $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ be an ontology network, with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^n$, $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i < j}^n$. Let $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$. Then the **induced ontology** for network $\mathcal{N}$ is the ontology $\mathcal{O}_N = (\mathcal{C}_N, \mathcal{I}_N)$ with $\mathcal{C}_N = \cup_{k=1}^n \mathcal{C}_k$ and $\mathcal{I}_N = \cup_{k=1}^n \mathcal{I}_k \cup_{i,j=1;i < j}^n \mathcal{M}_{ij}$.

2.4.2 Knowledge bases

In the algorithms we use the notion of knowledge base (KB). The notion that we define here is a restricted² variant of the notion as defined in description logics [16].

¹Observe that for every $\mathcal{M}_{ij}$ there is a corresponding $\mathcal{M}_{ji}$ such that $\mathcal{M}_{ij} = \mathcal{M}_{ji}$. Therefore, in the remainder of this thesis we will only consider the $\mathcal{M}_{ij}$ where $i < j$.

²We use only concept names and no roles. The axioms in the TBox are of the form $A \dot{\subseteq} B$ or $A \dot{=} C$, and the ABox is empty.

Definition 5 *Let $\mathcal{C}$ be a set of named concepts. A **knowledge base** is then a set of axioms of the form $A \rightarrow B$ with $A \in \mathcal{C}$ and $B \in \mathcal{C}$. A model of the knowledge base satisfies all axioms of the knowledge base.*

In the algorithms we initialize KBs with an ontology. This means that for ontology $\mathcal{O} = (\mathcal{C}, \mathcal{I})$ we create a KB such that $(A, B) \in \mathcal{I}$ iff $A \rightarrow B$ is an axiom in the KB.

For the KBs, we assume that they are able to do deductive logical inference. Furthermore, we need the following reasoning services. For a given statement the KB should be able to answer whether the statement is entailed by the KB.³ If a statement is entailed by the KB, it should be able to return the derivation paths (explanations) for that statement. The derivation paths, also called **justifications**, are used to show how a given statement is entailed. For a given named concept, the KB should return the super-concepts and the sub-concepts.

The KBs can be implemented in several ways. For instance, any description logic system could be used. In our setting, where we deal with taxonomies, we have used an efficient graph-based implementation. We have represented the ontologies using graphs where the nodes are concepts and the directed edges represent the is-a relations. The entailment of statements of the form $a \rightarrow b$ can be checked by transitively following edges starting at a . If b is reached, then the statement is entailed, otherwise not. If $a \rightarrow b$ is entailed, then the **derivation paths** are all the different paths obtained by following directed edges that start at a and end at b . The **super-concepts** of a are all the concepts that can be reached by following directed edges starting at a . The **sub-concepts** of a are all the concepts for which there is a path of directed edges starting at the concept and ending in a .

³In our setting, entailment by ontology can be reformulated as entailment by KB.

Chapter 3

Framework and Algorithms

This chapter presents our integrated ontology alignment and debugging framework with its two components—a debugging component and an alignment component. It is an extension of the framework in [67], which can be seen as the debugging component in this work. The extended framework introduces algorithms for debugging modelling defects in alignments and integrating ontology alignment and debugging of ontology networks. This is the first framework, to the best of our knowledge, that integrates ontology alignment and debugging in a unified approach. The interactions between them provide advantages for both areas.

This chapter is organized as follows: Section 3.1 gives an overview of the framework and introduces the three phases in its workflow—detection, validation and repairing phases. The first part of Section 3.2—Subsection 3.2.1—introduces two methods for detecting possible modelling defects in ontologies and their alignments. The second part—Subsection 3.2.2—explains the motivation for a set of requirements enforced during the repairing process and introduces four heuristics, initially defined in [61], in order to facilitate the repairing. The methods described in Section 3.2 are then applied and improved in the debugging and alignment components. Section 3.3 presents the algorithms for discovering and resolving wrong and missing is-a relations and mappings in the debugging component. Section 3.4 presents the algorithms in the alignment component, where the detection phase utilizes ontology alignment algorithms. The final section (3.5) illustrates the advantages of the interactions between the two components.

3.1 Framework and workflow

Our framework consists of two major components—a debugging component and an alignment component. They can be used completely independently, thus acting as two different systems, or in close interaction where each of the components benefits from the interaction. The alignment component detects and repairs missing and wrong mappings between ontologies using alignment algorithms, while the debugging component additionally detects and repairs missing and wrong is-a structure in ontologies employing the knowledge intrinsic to the network. Although we describe the two components separately, in our framework ontology alignment can be seen as a special kind of debugging.

The workflow in both components consists of three phases during which wrong and missing is-a relations/mappings are detected, validated and repaired in a semi-automatic manner by a domain expert (Figure 3.1).

In **Phase 1** possible modelling defects in ontologies and their alignments are detected. The debugging component detects possible defects for a selected ontology. Possible defects for a selected pair of ontologies can be detected from both components—when the debugging component is used, an initial alignment between the two ontologies is needed as well. In **Phase 2** the user validates the detected defects (possibly based on recommendations from the system) and categorizes each of them as a missing is-a relation/mapping or wrong is-a relation/mapping. The algorithms for detecting possible modelling defects and the validation procedure are explained in Subsection 3.3.1 for the debugging component and in Subsection 3.4.1 for the alignment component.

A naive way of repairing defects would be to compute all possible repairing actions¹ for the network with respect to the validated missing is-a relations and mappings for all the ontologies in the network (following the definition in Subsection 3.2.2). This is in practice infeasible as it involves all the ontologies and alignments and all the missing and wrong is-a relations and mappings in the network. It is also hard for domain experts to choose between large sets of repairing actions for all the ontologies and alignments. Moreover, functional visualization of such large sets may be complicated, if not impossible. Therefore, in our approach, we repair ontologies and alignments one at a time (**Phase 3**).

During **Phase 3** the validated missing and wrong is-a relations and mappings from the debugging component and the validated missing and (some of) the wrong mappings from the alignment component are repaired in similar ways. For the selected ontology (for repairing is-a relations) or for the selected alignment and its pair of ontologies (for repairing mappings), a user can choose to repair the missing or the wrong is-a relations/mappings (**Phase 3.1-3.4**). Although the algorithms for repairing

¹Is-a relations and/or mappings to add and/or remove in order to repair the validated defects.

Figure 3.1: Workflow.

are different for missing and wrong is-a relations/mappings, the repairing goes through the phases of generation of repairing actions, the ranking of is-a relations/mappings, the recommendation of repairing actions and finally, the execution of repairing actions.

In **Phase 3.1** repairing actions are generated. For missing is-a relations and mappings these are is-a relations or mappings to add, while for wrong is-a relations and mappings, these are is-a relations or mappings to remove.

In general, there will be many is-a relations/mappings that need to be repaired and some of them may be easier to start with, such as the ones with fewer repairing actions. We therefore rank them with respect to the number of possible repairing actions (**Phase 3.2**).

After this, the user can select an is-a relation/mapping to repair and choose among possible repairing actions. To facilitate this process, we use algorithms to recommend repairing actions (**Phase 3.3**).

Once the user decides on repairing actions, the chosen repairing actions are then removed (for wrong is-a relations/mappings) from or added (for missing is-a relations/mappings) to the relevant ontologies and alignments and the consequences are computed (**Phase 3.4**). For instance, by repairing one is-a relation/mapping some other missing or wrong is-a relations/mappings may also be repaired or their repairing actions may change. Furthermore, new modelling defects may be found.

Descriptions of our algorithms in the two components for **Phases 3.1-3.4** are found in Subsections 3.3.2 and 3.4.2.

The first two phases in the alignment component can be considered an instantiation of the general alignment framework presented in Subsection 2.2. The detection phase in the alignment component follows directly after Phase 1 in the general framework, applying ontology alignment algorithms. The validation phase in the alignment component corresponds to Phase 2 in the general framework. The third phase in the alignment component

can be seen as an extension of the alignment framework. While in the alignment framework the validation finalizes the alignment process, adding the correct mappings to the final alignment, in the alignment component we introduce a third phase where more possibilities for repairing missing and wrong mappings are presented to the domain expert.

We note that at any time during the debugging/alignment workflow, the user can switch between different ontologies, start earlier phases, or switch between the repairing of wrong is-a relations, the repairing of missing is-a relations, the repairing of wrong mappings and the repairing of missing mappings. The user can switch between the phases in the debugging and the alignment component as well. We also note that the repairing of defects often leads to the discovery of new defects, i.e., leading to additional debugging opportunities. Thus, several iterations are usually needed for completing the debugging/alignment process. The process ends when no more missing or wrong is-a relations and mappings are detected or need to be repaired.

In the following subsections we describe the components and their interactions, and present algorithms we have developed for the different components and phases.

3.2 Methods in the framework

This section presents methods and notions further implemented in the detection and repairing phases in both components. Subsection 3.2.1 presents two methods and related definitions for detecting modelling defects. Subsection 3.2.2 introduces the notion of *structural repair* during the repairing process and lists four heuristics used to facilitate the repairing.

3.2.1 Detect missing and wrong is-a relations and mappings

Two methods for discovering wrong and missing is-a relations and mappings are presented below. In the first method, given an ontology network, the domain knowledge represented by the network is utilized to detect the deduced is-a relations and mappings in the network (missing is-a relations and mappings). However, the ontology network may contain incorrect information and some of the detected missing is-a relations and mappings could be derived due to wrong is-a relations and mappings. Thus, the output of the method should be validated by a domain expert as missing structure (should be in the ontologies/alignments) and wrong structure (should not be in the ontologies/alignments). The method is presented together with examples and during its presentation related definitions are introduced. The second method employs different matchers for discovering modelling defects in alignments and its output (mapping suggestions) should be validated by a domain expert as well.

The possible defects in the structure of the ontologies, generated by detection methods prior to the validation, are called **candidate missing is-a relations (CMIs)**. The possible defects in the alignments, generated by detection methods prior to the validation, are called **candidate missing mappings (CMMs)**. The set of CMIs in the network is denoted as CMI and the set of CMMs in the network is denoted as CMM . Prior to repairing, the CMIs and CMMs should be validated by, e.g., a domain expert. During the validation the CMIs are divided into two sets—**wrong and missing is-a relations**, respectively denoted as $\mathcal{WI}$ and $\mathcal{MI}$. Similarly, the CMMs are divided into two sets as well—**wrong and missing mappings**, respectively denoted as $\mathcal{WM}$ and $\mathcal{MM}$. $\mathcal{MI}$, $\mathcal{WI}$, $\mathcal{MM}$, $\mathcal{WM}$ are not dependent on the origin of the CMIs and CMMs. After validation the relations in these sets are repaired.

Using knowledge intrinsic to an ontology network

Given an ontology network, **the set of candidate missing is-a relations logically derivable from the ontology network** (CMI_{LD}) consists of is-a relations between two concepts of an ontology, which can be inferred using logical derivation from the induced ontology of the network, but not from the ontology alone. Similarly, given an ontology network, **the set of candidate missing mappings logically derivable from the ontology network** (CMM_{LD}) consists of mappings between concepts in two ontologies, which can be inferred using logical derivation from the induced ontology of the network, but not from the two ontologies and their alignment alone.

Definition 6 Let $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ be an ontology network, with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^n$, $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1; i < j}^n$ and induced ontology $\mathcal{O}_N = (\mathcal{C}_N, \mathcal{I}_N)$. Let $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$. Then, we define the following:

- (1) $\forall k \in 1..n : CMI_{LD_k} = \{(a, b) \in \mathcal{C}_k \times \mathcal{C}_k \mid \mathcal{O}_N \models a \rightarrow b \wedge \mathcal{O}_k \not\models a \rightarrow b\}$ is the set of candidate missing is-a relations for $\mathcal{O}_k$ logically derivable from the network.
- (2) $\forall i, j \in 1..n, i < j : CMM_{LD_{ij}} = \{(a, b) \in (\mathcal{C}_i \times \mathcal{C}_j) \cup (\mathcal{C}_j \times \mathcal{C}_i) \mid \mathcal{O}_N \models a \rightarrow b \wedge (\mathcal{C}_i \cup \mathcal{C}_j, \mathcal{I}_i \cup \mathcal{I}_j \cup \mathcal{M}_{ij}) \not\models a \rightarrow b\}$ is the set of candidate missing mappings for $(\mathcal{O}_i, \mathcal{O}_j, \mathcal{M}_{ij})$ logically derivable from the network.
- (3) $CMI_{LD} = \bigcup_{k=1}^n CMI_{LD_k}$ is the set of candidate missing is-a relations logically derivable from the network.
- (4) $CMM_{LD} = \bigcup_{i,j=1; i < j}^n CMM_{LD_{ij}}$ is the set of candidate missing mappings logically derivable from the network.

Thus, $CMI_{LD} \subseteq CMI$ and $CMM_{LD} \subseteq CMM$.

As was mentioned, the structure of the ontologies and the mappings may contain wrong is-a relations and some of the CMI_{LD} and CMM_{LD} may be logically derived due to some wrong is-a relations and mappings. Therefore, we need to validate the CMI_{LD} and sort them out in one of the two sets $\mathcal{WI}$ or $\mathcal{MI}$. In this case we have that $\mathcal{MI} \supseteq \bigcup_{k=1}^n \mathcal{MI}_k$ with $\mathcal{MI}_k$ the

set of missing is-a relations in $\mathcal{O}_k$, and $\mathcal{WI} \supseteq \cup_{k=1}^n \mathcal{WI}_k$ with $\mathcal{WI}_k$ the set of wrong is-a relations in $\mathcal{O}_k$. Similarly, the CMM_{LD} should be validated and sorted out in one of the two sets $\mathcal{WM}$ or $\mathcal{MM}$. In this case we have that $\mathcal{MM} \supseteq \cup_{i,j=1;i < j}^n \mathcal{MM}_{ij}$ with $\mathcal{MM}_{ij}$ the set of missing mappings between $\mathcal{O}_i$ and $\mathcal{O}_j$, and $\mathcal{WM} \supseteq \cup_{i,j=1;i < j}^n \mathcal{WM}_{ij}$ with $\mathcal{WM}_{ij}$ the set of wrong mappings between $\mathcal{O}_i$ and $\mathcal{O}_j$.

As an example, in the network in Figure 2.1, we find 6 CMIs in the first ontology ((*nasal bone*, *bone*), (*maxilla*, *bone*), (*lacrimal bone*, *bone*), (*jaw*, *bone*), (*upper jaw*, *jaw*), (*lower jaw*, *jaw*)) and 2 CMIs in the second ontology ((*metatarsal bone*, *foot bone*), (*tarsal bone*, *foot bone*)). For instance, (*nasal bone*, *bone*) is a CMI as it cannot be logically derived in the first ontology, but it can be logically derived from the network as *nasal bone* in the first ontology is equivalent to *nasal bone* in the second ontology, *nasal bone* in the second ontology is a sub-concept of *bone* in the second ontology (via *flat bone*) and *bone* in the second ontology is equivalent to *bone* in the first ontology. After validation by a domain expert (*upper jaw*, *jaw*) and (*lower jaw*, *jaw*) will become wrong and the rest will become missing is-a relations.

As another example², in the experiment in Subsection 5.1.1 where we debug a network containing AMA and NCI-A, we find the CMIs (*lower respiratory system cartilage*, *cartilage*) and (*brain grey matter*, *white matter*). The former is a missing is-a relation, while the latter is a wrong is-a relation.

We note that each validation leads to a debugging opportunity. The wrong is-a relations and mappings are indications that there is incorrect information in the network. In order to repair the network some is-a relations or mappings need to be removed. In the case of a missing is-a relation or mapping, some is-a relations or mappings need to be added. This is a consequence and an advantage of our logic-based approach using the knowledge intrinsic to the network.

Using ontology alignment algorithms

While generating CMMs using the knowledge logically derivable from the network can be considered a special kind of ontology alignment, other alignment algorithms can be employed to detect CMMs. Since this method employs alignment algorithms it can only be used to detect **the set of candidate missing mappings from alignment algorithms** ($CMM_{Alignment}$).

Definition 7 Let $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ be an ontology network, with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^n$, $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i < j}^n$ and induced ontology $\mathcal{O}_N = (\mathcal{C}_N, \mathcal{I}_N)$. Let $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$ and $\mathcal{AA}$ is the set of available ontology alignment algorithms. Then, we define the following:

(1) $\forall i, j \in 1..n, i < j, CMM_{Alignment_{ij}}$ is the set of candidate missing mappings from alignment algorithms for $(\mathcal{O}_i, \mathcal{O}_j, \mathcal{M}_{ij}, \mathcal{AA})$.

²From OAEI 2010 Anatomy.

(2) $CMM_{Alignment} = \cup_{i,j=1; i < j}^n CMM_{Alignment_{ij}}$ is the set of candidate missing mappings from alignment algorithms for the network.

Thus, $CMM_{Alignment} \subseteq CMM$.

Analogously to the CMM_{LD} , $CMM_{Alignment}$ is presented to a domain expert for validation. As a result of the validation the members of $CMM_{Alignment}$ are sorted out in one of the two sets— $\mathcal{MM}$ and $\mathcal{WM}$, as shown above. In the previous detection method the CMI_{LD} and the CMM_{LD} are based on actually existing relations/mappings in the network and all of them will be repaired later. The detection using ontology alignment algorithms, however, does not employ existing knowledge, i.e., the $CMM_{Alignment}$ are not based on existing relations/mappings in the network. This fact leads to the following consequences during the repairing—all mappings in $\mathcal{MM}$ will be repaired. This is not the case with those in $\mathcal{WM}$ where only the mappings logically derivable from the network will be repaired. The rest will not be repaired since they are not based on existing relations/mappings in the network.

This method is particularly important when there is no network, i.e., no alignments between the ontologies. In such a case it is used to create an initial network enabling the detection of CMIs and CMMs with the detection algorithm, which employs the knowledge intrinsic to the network.

3.2.2 Repair missing and wrong is-a relations and mappings

Once missing and wrong is-a relations and mappings have been obtained, we need to repair them. We note that the theory for repairing does not require that the missing and wrong is-a relations and mappings are determined using the techniques for detection described above. They may have been generated using external knowledge and then validated by a domain expert or they may have been provided directly by a domain expert. The methods for repairing do not depend on and cannot distinguish the origin of the wrong and missing is-a relations/mappings.

We first present the notion of structural repair used to formalize a set of requirements enforced during the process of repairing of the defects. Then four heuristics, initially defined in [61] for missing is-a relations, are introduced with their extended definitions. They filter the possible repairing actions in order to assist the domain expert during the repairing process.

Structural repair

For each ontology in the network, we want to repair its is-a structure in such a way that (i) the missing is-a relations can be logically derived from their repaired host ontologies and (ii) the wrong is-a relations can no longer be logically derived from the repaired ontology network. In addition, for

each pair of ontologies, we want to repair its mappings in such a way that (iii) the missing mappings can be logically derived from the repaired host ontologies of their mapped concepts and the repaired alignment between the host ontologies of the mapped concepts and (iv) the wrong mappings can no longer be logically derived from the repaired ontology network. To satisfy requirement (i), we need to add a set of is-a relations to the host ontology. To satisfy requirement (iii), we need to add a set of is-a relations to the host ontologies of the mapped concepts and/or mappings to the alignment between the host ontologies of the mapped concepts. To satisfy requirements (ii) and (iv), a set of asserted is-a relations and/or mappings should be removed from the ontology network. The notion of **structural repair** formalizes this.

Definition 8 Let $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ be an ontology network, with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^n$, $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i < j}^n$ and induced ontology $\mathcal{O}_N = (\mathcal{C}_N, \mathcal{I}_N)$. Let $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$. Let $\mathcal{MI}_k$ and $\mathcal{WI}_k$ be the missing, respectively wrong, is-a relations for ontology $\mathcal{O}_k$ and let $\mathcal{MI} \supseteq \bigcup_{k=1}^n \mathcal{MI}_k$ and $\mathcal{WI} \supseteq \bigcup_{k=1}^n \mathcal{WI}_k$. Let $\mathcal{MM}_{ij}$ and $\mathcal{WM}_{ij}$ be the missing, respectively wrong, mappings between ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ and let $\mathcal{MM} \supseteq \bigcup_{i,j=1;i < j}^n \mathcal{MM}_{ij}$ and $\mathcal{WM} \supseteq \bigcup_{i,j=1;i < j}^n \mathcal{WM}_{ij}$. A **structural repair** for $\mathcal{N}$ with respect to $(\mathcal{MI}, \mathcal{WI}, \mathcal{MM}, \mathcal{WM})$, denoted by $(\mathcal{R}^+, \mathcal{R}^-)$, is a pair of sets of is-a relations and mappings, such that

- (1) $\mathcal{R}^- \cap \mathcal{R}^+ = \emptyset$
- (2) $\mathcal{R}^- = \mathcal{R}_M^- \cup \mathcal{R}_I^-$; $\mathcal{R}_M^- \subseteq \bigcup_{i,j=1,i < j}^n \mathcal{M}_{ij}$; $\mathcal{R}_I^- \subseteq \bigcup_{k=1}^n \mathcal{I}_k$
- (3) $\mathcal{R}^+ = \mathcal{R}_M^+ \cup \mathcal{R}_I^+$; $\mathcal{R}_M^+ \subseteq \bigcup_{i,j=1,i < j}^n ((\mathcal{C}_i \times \mathcal{C}_j) \setminus \mathcal{M}_{ij})$; $\mathcal{R}_I^+ \subseteq \bigcup_{k=1}^n ((\mathcal{C}_k \times \mathcal{C}_k) \setminus \mathcal{I}_k)$
- (4) $\forall k \in 1..n : \forall (a, b) \in \mathcal{MI}_k : (\mathcal{C}_k, (\mathcal{I}_k \cup (\mathcal{R}_I^+ \cap (\mathcal{C}_k \times \mathcal{C}_k)))) \setminus \mathcal{R}_I^- \models a \rightarrow b$
- (5) $\forall i, j \in 1..n, i < j : \forall (a, b) \in \mathcal{MM}_{ij} : ((\mathcal{C}_i \cup \mathcal{C}_j), (\mathcal{I}_i \cup ((\mathcal{C}_i \times \mathcal{C}_i) \cap \mathcal{R}_I^+) \cup \mathcal{I}_j \cup ((\mathcal{C}_j \times \mathcal{C}_j) \cap \mathcal{R}_I^+) \cup \mathcal{M}_{ij} \cup ((\mathcal{C}_i \times \mathcal{C}_j) \cap \mathcal{R}_M^+))) \setminus \mathcal{R}^- \models a \rightarrow b$
- (6) $\forall (a, b) \in \mathcal{WI} \cup \mathcal{WM} \cup \mathcal{R}^- : (\mathcal{C}_N, (\mathcal{I}_N \cup \mathcal{R}^+) \setminus \mathcal{R}^-) \not\models a \rightarrow b$

The definition states that (1) the added is-a relations and mappings cannot at the same time be removed, (2) the removed mappings come from the original alignments and the removed is-a relations come from the original asserted is-a relations in the ontologies, (3) the added mappings were not in the original alignments and the added is-a relations were not original is-a relations in the ontologies, (4) every missing is-a relation is logically derivable from its repaired host ontology, (5) every missing mapping is logically derivable from the repaired host ontologies of the mapped concepts and their repaired alignment, and (6) no wrong mapping, wrong is-a relation or removed mapping or is-a relation is logically derivable from the repaired network. The is-a relations and mappings contained in a structural repair are called **repairing actions**.

Preferences

As explained in [61] regarding missing is-a relations, there could be many structural repairs and not all of them are equally useful or interesting for a domain expert. For instance, four structural repairs for the set with missing is-a relations $M = \{(nasal\ bone, bone), (maxilla, bone)\}$ in the first ontology in Figure 2.1 are presented in the list below:

- $S_1 = \{(nasal\ bone, bone), (maxilla, bone)\}$ —the missing is-a relations are repaired by adding them;
- $S_2 = \{(nasal\ bone, bone), (maxilla, bone), (jaw, bone)\}$ —the missing is-a relations are repaired by adding them and one more is-a relation with no regard to the missing relations;
- $S_3 = \{(viscerocranium\ bone, bone)\}$ —adding this is-a relation will make the missing is-a relations logically derivable since $nasal\ bone \rightarrow viscerocranium\ bone$ and $maxilla \rightarrow viscerocranium\ bone$. It is also correct according to the domain and moreover it will repair $(lacrimal\ bone, bone)$ which is also a missing is-a relation;
- $S_4 = \{(viscerocranium\ bone, bone), (maxilla, bone)\}$ —same as the previous set plus one of the missing is-a relations. However, in the presence of $(viscerocranium\ bone, bone)$ in the taxonomy, adding $(maxilla, bone)$ will introduce redundancy, since $(maxilla, bone)$ will become logically derivable through $maxilla \rightarrow viscerocranium\ bone \rightarrow bone$.

Many others structural repairs can be created.

Four heuristics have been developed in [61] in order to assist the domain expert during the repairing process. They aim to reduce the number of structural repairs presented to the domain expert without excluding relevant repairing actions from them. We illustrate them with examples and present extended definitions here.

Definition 9 Pref1 Let S_1 and S_2 be structural repairs for the ontology O with respect to $(\mathcal{MI}, \mathcal{WI}, \mathcal{MM}, \mathcal{WM})$, then S_1 is axiom-preferred to S_2 (notation $S_1 \ll_A S_2$) iff $S_1 \subseteq S_2$.

The first heuristic states that we want to use repairing actions that contribute to the repairing. It corresponds to the notion of Subset Minimality given in [65]. For instance, consider the missing is-a relations $(nasal\ bone, bone)$ and $(maxilla, bone)$ in the first ontology in Figure 2.1. Two possible structural repairs are $S_1 = \{(nasal\ bone, bone), (maxilla, bone)\}$ and $S_2 = \{(nasal\ bone, bone), (maxilla, bone), (jaw, bone)\}$. According to this preference, to repair the missing is-a relations, we should choose S_1 over S_2 since using $(jaw, bone)$ in addition will not contribute to the repairing of the missing $(nasal\ bone, bone)$ and $(maxilla, bone)$. As another example consider structural repair $S_3 = \{(viscerocranium\ bone, bone)\}$ and $S_4 = \{(viscerocranium\ bone, bone), (maxilla, bone)\}$. In this case $S_3 \ll_A S_4$ since $(viscerocranium\ bone, bone)$ alone will repair both missing is-a relations and

adding (*maxilla*, *bone*) will introduce redundancy in the taxonomy and will not contribute to the repairing.

Definition 10 Pref2 We say that (x_1, y_1) is more informative than (x_2, y_2) iff $x_2 \rightarrow x_1$ and $y_1 \rightarrow y_2$. Let S_1 and S_2 be structural repairs for the ontology O with respect to $(\mathcal{MI}, \mathcal{WI}, \mathcal{MM}, \mathcal{WM})$. Then S_1 is information-preferred to S_2 (notation $S_1 \ll_I S_2$) iff $\exists (x_1, y_1) \in S_1, (x_2, y_2) \in S_2$: (x_1, y_1) is more informative than (x_2, y_2) .

Therefore, adding or removing more informative repairing actions adds or removes more knowledge than less informative repairing actions. According to this preference we want to repair with repairing actions that are as informative as possible. It is a special case of More Informative, as defined in [65]—adding more informative repairing actions for missing is-a relations to the set with asserted axioms in a taxonomy will always entail the missing is-a relations.

As an example, consider again the missing is-a relation (*nasal bone*, *bone*) in Figure 2.1. Knowing that *nasal bone* $\rightarrow$ *viscerocranium bone*, according to the definition of more informative, we know that (*viscerocranium bone*, *bone*) is more informative than (*nasal bone*, *bone*). As *viscerocranium bone* actually is a sub-concept of *bone* according to the domain, a domain expert would prefer to use the more informative repairing action for the given missing is-a relation.³

Definition 11 Pref3 Let S_1 and S_2 be structural repairs for the ontology $O = (C, I)$ with respect to $(\mathcal{MI}, \mathcal{WI}, \mathcal{MM}, \mathcal{WM})$. Then S_1 is strict-hierarchy-preferred to S_2 (notation $S_1 \ll_{SH} S_2$) iff $\exists A, B \in C$: $(C, I) \models A \rightarrow B$ and $(C, I) \not\models B \rightarrow A$ and $(C, I \cup S_1) \not\models B \rightarrow A$ and $(C, I \cup S_2) \models B \rightarrow A$.

The third heuristic prefers not to introduce equivalence relations between concepts when in the original ontology there is an is-a relation. For instance, consider the missing is-a relation (*metatarsal bone*, *foot bone*) in the second ontology in Figure 2.1. Two possible structural repairs are $\{(metatarsal\ bone, foot\ bone)\}$ and $\{(bone\ of\ the\ lower\ extremity, foot\ bone)\}$. Adding the latter will introduce an equivalence relation between (*bone of the lower extremity*, *foot bone*) which is not desirable with respect to this preference. Additionally, this is often not correct according to the domain.

Pref4 Finally, the *single relation heuristic* assumes that it is more likely that the ontology developers have failed to add single is-a relations, rather than a chain of is-a relations. For instance, consider again the missing is-a relation (*nasal bone*, *bone*). It is more likely that the developers have failed to add it, rather than missing a chain of relations, for example, *nasal bone* $\rightarrow x_1 \rightarrow x_2 \rightarrow \dots \rightarrow x_n \rightarrow bone$.

³We also note that using (*viscerocranium bone*, *bone*) as repairing action would also immediately repair the missing is-a relations (*maxilla*, *bone*) and (*lacrimal bone*, *bone*).

1. Initialize KB_N with ontology network $\mathcal{N}$;
2. For $k := 1 \dots n$:
 initialize KB_k with ontology $\mathcal{O}_k$;
3. For $i := 1 \dots n-1$: for $j := i+1 \dots n$:
 initialize KB_{ij} with ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$;
 for every mapping $(m, n) \in \mathcal{M}_{ij}$: add the axiom $m \rightarrow n$ to KB_{ij} ;

Figure 3.2: Initialization for detection.

3.3 Algorithms in the debugging component

Subsection 3.3.1 presents our algorithms for detecting (Phase 1) and validating (Phase 2) wrong and missing is-a relations and mappings employing knowledge intrinsic to the ontology network. The detection algorithm follows the definition for CMI_{LD} and CMM_{LD} given in Subsection 3.2.1 introducing an improvement of the method. Subsection 3.3.2 presents the process of repairing missing and wrong is-a relations/mappings (Phase 3) including our algorithms that calculate the structural repairs.

The input for the debugging component is a taxonomy network, i.e., a set of taxonomies and their alignments. The output is the set of repaired taxonomies and alignments.

3.3.1 Detect and validate candidate missing is-a relations and mappings

The detection phase (**Phase 1**) starts with initialization of a KB for the ontology network (KB_N), KBs for each ontology (KB_k) and for each pair of ontologies and their alignment (KB_{ij}). The algorithm for initialization of the different KBs is shown in Figure 3.2.

Then CMIs and CMMs that are logically derivable from the network could be found by directly applying the definition for CMI_{LD} and CMM_{LD} given in Subsection 3.2.1—using a brute-force method by checking each pair of concepts in the network. For each pair of concepts within the same ontology, we check whether an is-a relation between the pair can be logically derived from the KB of the network, but not from the KB of the ontology, and if so, it is a CMI. Similarly, for each pair of concepts belonging to two different ontologies, we check whether an is-a relation between the pair can be logically derived from the KB of the network, but not from the KB of the two ontologies and their alignment, and if so, it is a CMM.

However, for large ontologies or ontology networks, this is infeasible. Moreover, some of these CMIs and CMMs are redundant in the sense that they can be repaired by the repairing actions of other CMIs and CMMs. Therefore, instead of checking all pairs of concepts in the network we define a subset of the set of all pairs of concepts in the network that we will consider

for generating CMIs and CMMs logically derivable from the network. This subset will initially consist of all pairs of mapped concepts⁴ and we explain this choice below.

In the restricted setting where we assume that all existing is-a relations in the ontologies and all existing mappings in the alignments are correct (and thus the debugging problem does not need to consider wrong is-a relations and mappings), it can be shown that all CMIs and CMMs logically derivable from the network⁵ will be repaired when we repair the CMIs and CMMs between *mapped concepts*.

Proposition. *Let $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ be an ontology network with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^n$ the set of the ontologies in the network and $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i < j}^n$ the set of representations for the alignments between these ontologies. Further, assume that all is-a relations in the ontologies and all mappings in the alignments are correct. Then the following holds:*

(i) *For each logically derivable candidate missing is-a relation (a, b) in ontology $\mathcal{O}_i$, there exists a logically derivable candidate missing is-a relation (x, y) in ontology $\mathcal{O}_i$ where x and y are mapped concepts in alignments between $\mathcal{O}_i$ and other ontologies in the network, such that the repairing of (x, y) also repairs (a, b) .*

(ii) *For each logically derivable candidate missing mapping (a, b) such that $a \in \mathcal{O}_i$ and $b \in \mathcal{O}_j$ with $i \neq j$, there exists a logically derivable candidate missing mapping (x, y) such that $x \in \mathcal{O}_i$ and $y \in \mathcal{O}_j$, x is a mapped concept in an alignment between $\mathcal{O}_i$ and another ontology in the network and y is a mapped concept in an alignment between $\mathcal{O}_j$ and another ontology in the network, such that the repairing of (x, y) also repairs (a, b) .*

Proof. Assume (a, b) is a CMI logically derivable from the network in $\mathcal{O}_i$. According to the definition of CMI logically derivable from the network, the relation $a \rightarrow b$ is not logically derivable from $\mathcal{O}_i$ but logically derivable from the ontology network. So, there must exist at least one concept from another ontology in the network, for instance z , such that $\mathcal{O}_N \models a \rightarrow z \rightarrow b$. Because concepts a and z reside in different ontologies, the relation $a \rightarrow z$ must be supported by a mapping between a concept x in $\mathcal{O}_i$ and a concept x' in another ontology, e.g., $\mathcal{O}_r$, in the network, such that $(x, x') \in M_{ir}$ (if $i < r$) or $(x, x') \in M_{ri}$ (if $r < i$), and $\mathcal{O}_N \models a \rightarrow x \rightarrow x' \rightarrow z$. Likewise, for concepts z and b , the relation $z \rightarrow b$ must also be supported by a mapping between a concept y in $\mathcal{O}_i$ and a concept y' in another ontology, e.g., $\mathcal{O}_s$, in the network, such that $(y', y) \in M_{sj}$ (if $s < j$) or $(y', y) \in M_{js}$ (if $j < s$), such that $\mathcal{O}_N \models z \rightarrow y' \rightarrow y \rightarrow b$. We can then deduce that $x \rightarrow y$ is logically derivable from the ontology network because $\mathcal{O}_N \models a \rightarrow x \rightarrow$

⁴In the worst case scenario the number of mapped concept pairs is equal to the total number of concept pairs. In practice, the use of mapped concepts may significantly reduce the search space, e.g., when some ontologies are smaller than other ontologies in the network or when not all concepts participate in mappings. For instance, in the experiment in Section 5.1.1 the search space is reduced by almost 90%.

⁵In this setting all CMIs logically derivable from the network are also missing is-a relations, and all CMMs logically derivable from the network are also missing mappings.

$x' \rightarrow z \rightarrow y' \rightarrow y \rightarrow b$. Since $a \rightarrow b$ is not inferable from $\mathcal{O}_i$, the relation $x \rightarrow y$ cannot be inferred from $\mathcal{O}_i$ either. This means that (x, y) is also a CMI logically derivable from the network in $\mathcal{O}_i$, and the repairing of (x, y) also repairs (a, b) . This proves statement (i). A similar proof can be given for statement (ii). ♣

The proposition guarantees that for the part of the network for which the is-a structure and mappings are correct, we find all CMIs and CMMs logically derivable from the network when using the set of all pairs of mapped concepts. In addition, we may generate CMIs and CMMs that were logically derived using incorrect information. Thus, the CMIs and CMMs may later be validated as missing (these that are correct) or wrong (these that are incorrect). As our debugging approach is iterative, after repairing, larger and larger parts of the network will contain only correct is-a structure and mappings. When, finally, the entire network contains only correct is-a structure and mappings, the proposition guarantees that all defects that can be found using the knowledge intrinsic to the network have been found using our approach.

In the network in Figure 2.1 the CMIs are *(nasal bone, bone)*, *(maxilla, bone)*, *(lacrimal bone, bone)*, *(jaw, bone)*, *(upper jaw, jaw)* and *(lower jaw, jaw)* in the left ontology (AMA), and *(metatarsal bone, foot bone)* and *(tarsal bone, foot bone)* in the right ontology (NCI-A). Since the network contains only two ontologies and their alignment, CMMs cannot be detected in this example. In order to detect CMMs with this method at least three ontologies and two alignments are needed.

After the CMIs and CMMs have been generated, redundant ones are removed. The remaining CMIs and CMMs are then presented to a domain expert for validation (**Phase 2**).

We then use the recommendation algorithm for validation from [67]. As is-a and part-of are often confused, the user can ask for a recommendation based on existing part-of relations in the ontology or in external domain knowledge (WordNet). If a part-of relation exists between the concepts of a CMI, it is likely a wrong is-a relation. Similarly, the existence of is-a relations in external domain knowledge (WordNet and UMLS⁶) may indicate that a CMI is indeed a missing is-a relation.

In the network in Figure 2.1 *(upper jaw, jaw)* and *(lower jaw, jaw)* are validated as wrong since an upper/lower jaw is a *part-of* (not *is-a*) a jaw. The rest are validated as correct.

As noted before, every CMI or CMM that is generated using this approach also presents an opportunity for debugging. If a CMI or CMM that is logically derivable from the network is validated as correct, then informa-

⁶It is well-known that UMLS contains semantic and modelling defects (e.g., [52, 33]). Therefore, we only use the external resources in the *recommendation* of the validation of CMIs (and in Section 3.3.2 in the recommendation of repairing actions), but not in the *generation*. The validation (and in Section 3.3.2 the choice of repairing actions) is always the domain expert's responsibility and the recommendations should only be considered as an aid.

```

1. For k:= 1 .. n:
    for every missing is-a relation  $(a, b) \in \mathcal{MI}_k$ :
        add the axiom  $a \rightarrow b$  to  $KB_N$ ;
        add the axiom  $a \rightarrow b$  to  $KB_k$ ;
        for i := 1 .. k-1:
            add the axiom  $a \rightarrow b$  to  $KB_{ik}$ ;
        for i := k+1 .. n:
            add the axiom  $a \rightarrow b$  to  $KB_{ki}$ ;
2. For i := 1 .. n-1: for j := i+1 .. n:
    for every missing mapping  $(m, n) \in \mathcal{MM}_{ij}$ :
        add the axiom  $m \rightarrow n$  to  $KB_N$ ;
        add the axiom  $m \rightarrow n$  to  $KB_{ij}$ ;
3.  $\mathcal{MI} := \mathcal{MI}$ ;  $\mathcal{WI} := \mathcal{WI}$ ;  $\mathcal{MM} := \mathcal{MM}$ ;  $\mathcal{WM} := \mathcal{WM}$ ;
4.  $\mathcal{R}_I^+ := \emptyset$ ;  $\mathcal{R}_I^- := \emptyset$ ;  $\mathcal{R}_M^+ := \emptyset$ ;  $\mathcal{R}_M^- := \emptyset$ ;
5.  $\mathcal{CMI} := \emptyset$ ;  $\mathcal{CMM} := \emptyset$ ;
    
```

Figure 3.3: Initialization for repairing.

tion is missing and is-a relations or mappings need to be added; otherwise, some existing information is incorrect and is-a relations or mappings need to be removed. After repairing, new CMIs and CMMs may be logically derived from the network.

3.3.2 Repair missing and wrong is-a relations and mappings

In Phase 3 the missing and wrong is-a relations and mappings are repaired. The repairing process is different for the missing and wrong is-a relations/mappings but contains the same subphases of generation of structural repairs (Phase 3.1), ranking (Phase 3.2), recommendation (Phase 3.3) and execution (Phase 3.4) of repairing actions.

Initialization of the repairing phase

In our algorithm (Figure 3.3), at the start of the repairing phase we add all missing is-a relations and mappings to the relevant KBs (steps 1 and 2). Since these are validated as correct, this is extra knowledge that should be used in the repairing process. Adding the missing is-a relations and mappings essentially means that we have repaired these using the least informative repairing actions (see the definition of more informative in Section 3.2.2). In this subsection we try to improve on this and find more informative repairing actions.

We also initialize global variables for the current sets of missing ($\mathcal{MI}$) and wrong ($\mathcal{WI}$) is-a relations, the current sets of missing ($\mathcal{MM}$) and wrong

1. Compute $AllJust(w, r, \mathcal{O}_e)$
 where $\mathcal{O}_e = (\mathcal{C}_e, \mathcal{I}_e)$ such that $\mathcal{C}_e = \cup_{k=1}^n \mathcal{C}_k$ and
 $\mathcal{I}_e = ((\cup_{k=1}^n \mathcal{I}_k) \cup (\cup_{i,j=1; i < j}^n \mathcal{M}_{ij}) \cup \mathcal{MI} \cup \mathcal{MM} \cup \mathcal{R}_I^+ \cup \mathcal{R}_M^+) \setminus (\mathcal{R}_I^- \cup \mathcal{R}_M^-)$;
2. For every $\mathcal{I}' \in AllJust(w, r, \mathcal{O}_e)$:
 choose one element from $\mathcal{I}' \setminus (\mathcal{MI} \cup \mathcal{MM} \cup \mathcal{R}_I^+ \cup \mathcal{R}_M^+)$ to remove;

Figure 3.4: Algorithm for generating repairing actions for wrong is-a relations and mappings.

($\mathcal{WM}$) mappings in step 3, the added ($\mathcal{R}_I^+$ for is-a relations and $\mathcal{R}_M^+$ for mappings) and removed ($\mathcal{R}_I^-$ for is-a relations and $\mathcal{R}_M^-$ for mappings) repairing actions in step 4, and the current sets of candidate missing is-a relations ($\mathcal{CMI}$) and candidate missing mappings ($\mathcal{CMM}$) in step 5.

Repair wrong is-a relations and mappings

Figure 3.4 shows the algorithm for generating repairing actions (**Phase 3.1**) for a wrong is-a relation or mapping. This algorithm is run for all elements logically derivable from the network in $\mathcal{WI}$ and $\mathcal{WM}$. It computes all justifications for the wrong is-a relation or mapping in the current ontology network. The current network is the original network where the repairs up to now have been taken into account (i.e., all missing is-a relations have been repaired by adding them, and additionally some have been repaired using a more informative repairing action in $\mathcal{R}_I^+$, missing mappings have been repaired by adding them or by repairing actions in $\mathcal{R}_M^+$, and some wrong is-a relations and mappings have already been repaired by removing is-a relations and mappings in $\mathcal{R}_I^-$ and $\mathcal{R}_M^-$, respectively). A justification for a wrong is-a relation or mapping can be seen as an explanation for how this is-a relation or mapping is logically derivable from the network.

Definition 12 (similar definition as in [45]). Given an ontology $\mathcal{O} = (\mathcal{C}, \mathcal{I})$, and $(a, b) \in \mathcal{C} \times \mathcal{C}$ an is-a relation logically derivable from $\mathcal{O}$, then, $\mathcal{I}' \subseteq \mathcal{I}$ is a **justification** for (a, b) in $\mathcal{O}$, denoted by $Just(\mathcal{I}', a, b, \mathcal{O})$ iff (i) $(\mathcal{C}, \mathcal{I}') \models a \rightarrow b$; and (ii) there is no $\mathcal{I}'' \subsetneq \mathcal{I}'$ such that $(\mathcal{C}, \mathcal{I}'') \models a \rightarrow b$. We use $AllJust(a, b, \mathcal{O})$ to denote the set of all justifications for (a, b) in $\mathcal{O}$.

The algorithm to compute justifications initializes a KB taking into account the repairing actions up to now. To compute the justifications for $a \rightarrow b$ in our graph-based implementation, all the different paths obtained by following directed edges that start at a and end at b are collected. Among these the minimal ones (w.r.t $\subseteq$) are retained.

The wrong is-a relation or mapping can then be repaired by removing at least one element in every justification. However, missing is-a relations, missing mappings, and added repairing actions (is-a relations in ontologies

and mappings) cannot be removed. Using this algorithm structural repairs are generated that include only contributing repairing actions (preference $\ll_A$ in Section 3.2.2).

In the network in Figure 2.1 (*upper jaw, jaw*) in the left ontology (AMA) is validated as incorrect. Its justification is $AMA:upper\ jaw \equiv NCI-A:Upper_Jaw \rightarrow NCI-A:Jaw \equiv AMA:jaw$. To repair it (*Upper_Jaw, Jaw*) should be removed from NCI-A (on the right).

In **Phase 3.2** the wrong is-a relations and mappings are ranked with respect to the number of possible repairing actions. Those with fewer repairing actions are ranked higher.

We have also used the recommendation algorithm in [67] (**Phase 3.3**) that computes hitting sets for all the justifications of the wrong is-a relations and mappings under repair. Each hitting set contains the minimal set of is-a relations and mappings that must be removed to repair a wrong is-a relation/mapping (formal definition and algorithm in [76]). The recommendation algorithm then assigns a priority to each possible repairing action based on how often it occurs in the hitting sets and its importance in already repaired is-a relations and mappings. In the example⁷ in Figure 4.3 the highest priority is given to the mapping (*Brain.White_Matter, brain grey matter*), as this is the only way to repair more than one wrong is-a relation at the same time. (Both (*cerebellum white matter, brain grey matter*) and (*cerebral white matter, brain grey matter*) would be repaired.)

Once the user decides on repairing actions, the chosen repairing actions are then removed from the relevant ontologies and alignments and a number of updates need to be done (**Phase 3.4**). First, the wrong is-a relation (or mapping) is removed from WI (or WM). The chosen repairing actions that are is-a relations in an ontology are added to $\mathcal{R}_I^-$ and repairing actions that are mappings are added to $\mathcal{R}_M^-$. Some other wrong is-a relations or mappings may also have been repaired by repairing the current wrong is-a relation or mapping (update WI and WM). Also, some repaired missing is-a relations and mappings may end up missing again (update MI and MM). Additionally, new CMI and CMMs logically derivable from the network may appear (update CM_I and CM_M —and after validation update CM_I , MI , WI , CM_M , MM and WM). In other cases the possible repairing actions for wrong and missing is-a relations and mappings may change (update justifications and sets of possible repairing actions for missing is-a relations and mappings). We also need to update the knowledge bases.

Repair missing is-a relations and mappings

It was shown in [55] that repairing missing is-a relations (and mappings) can be seen as a generalized TBox abduction problem. Figure 3.5 shows our solution for the computation of repairing actions for a missing is-a relation or mapping (**Phase 3.1**). The algorithm, an extension of the algorithm

⁷From OAEI 2010 Anatomy.

Repair missing is-a relation (a, b) with $a \in \mathcal{O}_k$ and $b \in \mathcal{O}_k$:
Choose an element from $\text{GenerateRepairingActions}(a, b, KB_k)$;

Repair missing mapping (a, b) with $a \in \mathcal{O}_i$ and $b \in \mathcal{O}_j$:
Choose an element from $\text{GenerateRepairingActions}(a, b, KB_{ij})$;

$\text{GenerateRepairingActions}(a, b, KB)$:

1. $\text{Source}(a, b) := \text{super-concepts}(a) - \text{super-concepts}(b)$ in KB;
2. $\text{Target}(a, b) := \text{sub-concepts}(b) - \text{sub-concepts}(a)$ in KB;
3. $\text{Repair}(a, b) := \text{Source}(a, b) \times \text{Target}(a, b)$;
4. For each $(s, t) \in \text{Source}(a, b) \times \text{Target}(a, b)$:
 - if $(s, t) \in \mathcal{WI} \cup \mathcal{WM} \cup \mathcal{R}_I^- \cup \mathcal{R}_M^-$
then remove (s, t) from $\text{Repair}(a, b)$;
 - else if
 $\exists (u, v) \in \mathcal{WI} \cup \mathcal{WM} \cup \mathcal{R}_I^- \cup \mathcal{R}_M^- : (s, t)$
is more informative than (u, v) in KB
and
 $u \rightarrow s$ and $t \rightarrow v$ are logically derivable from validated
to be correct only is-a relations and/or mappings
then remove (s, t) from $\text{Repair}(a, b)$;
5. return $\text{Repair}(a, b)$;

Figure 3.5: Algorithm for generating repairing actions for missing is-a relations and mappings.

in [61], takes into consideration that all missing is-a relations and missing mappings will be repaired (using the least informative repairing action), but it does not take into account the consequences of the actual (possibly more informative) repairing actions that will be performed for other missing is-a relations and other missing mappings.

The main component of the algorithm ($\text{GenerateRepairingActions}$) takes a missing is-a relation or mapping (a, b) as input together with a knowledge base. For a missing is-a relation this is the knowledge base corresponding to the host ontology of the missing is-a relation; for a missing mapping this is the knowledge base corresponding to the host ontologies of the mapped concepts in the missing mapping and their alignment. In this component for a missing is-a relation or mapping we compute the more general concepts of the first concept a (Source) and the more specific concepts of the second concept b (Target) in the knowledge base. So as not to introduce non-validated equivalence relations where in the original ontologies and alignments there are only is-a relations, we remove the super-concepts of the second concept (b) from Source, and the sub-concepts of the first concept (a) from Target. Adding an element from $\text{Source} \times \text{Target}$ ($\text{Repair}(a, b)$) to the knowledge base makes the missing is-a relation or mapping logically derivable.

However, some elements in $\text{Source} \times \text{Target}$ may conflict with already known wrong is-a relations or mappings. Therefore, in Repair, we take the wrong is-a relations and mappings and the former repairing actions for wrong is-a relations and mappings into account. The missing is-a relation or mapping can then be repaired using an element in Repair. We note that for missing is-a relations, the elements in Repair are is-a relations in the host ontology for the missing is-a relation. For missing mappings, the elements in Repair can be mappings as well as is-a relations in each of the host ontologies of the mapped concepts of the missing mapping. Using this algorithm, structural repairs are generated that include only contributing repairing actions, and repairing actions of the form (a, t) or (s, b) for missing is-a relation or mapping (a, b) do not introduce non-validated equivalence relations (see `pref1` and `pref3` in Subsection 3.2.2). Furthermore, the solutions follow the single relation heuristic (`pref4`).

In the network in Figure 2.1 (*nasal bone, bone*) is validated as correct. The Source set for it contains $\{\textit{nasal bone}, \textit{viscerocranium bone}\}$ and the Target set contains $\{\textit{bone}, \textit{limb bone}, \textit{forelimb bone}, \textit{hindlimb bone}, \textit{foot bone}, \textit{metatarsal bone}, \textit{tarsal bone}, \textit{jaw}, \textit{maxilla}, \textit{lacrimal bone}\}$, i.e., Repair contains $2 \times 10 = 20$ possible repairing actions. Each of the repairing actions, when added to the first ontology, would make the missing is-a relation logically derivable from it. In this example a domain expert would select the more informative repairing action (*viscerocranium bone, bone*). As a consequence (*lacrimal bone, bone*) and (*maxilla, bone*) will become logically derivable (i.e., will be repaired as well).

As another example, for the missing is-a relation (*lower respiratory system cartilage, cartilage*) in AMA (experiment in Section 5.1.1 and Figure 4.4) a Source set of 2 elements and a Target set of 21 elements are generated and this results in 42 possible repairing actions. Each of the repairing actions, when added to AMA, would make the missing is-a relation logically derivable from AMA. In this example a domain expert would select the more informative repairing action (*respiratory system cartilage, cartilage*).

Similarly to the repairing of wrong is-a relations/mappings in **Phase 3.2**, we rank the is-a relations/mappings that need to be repaired with respect to the number of possible repairing actions.

In **Phase 3.3** a recommendation algorithm (as defined in [61] and [60]) computes for a missing is-a relation (a, b) the most informative repairing actions from $\text{Source}(a, b) \times \text{Target}(a, b)$ that are supported by domain knowledge (WordNet and UMLS).

When the selected repairing action is in $\text{Repair}(a, b)$, the repairing action is executed, and a number of updates need to be done (**Phase 3.4**). First, the missing is-a relation (or mapping) is removed from $\mathcal{MI}$ (or $\mathcal{MM}$) and the chosen repairing action is added to $\mathcal{R}_I^+$ or $\mathcal{R}_M^+$ depending on whether it is an is-a relation within an ontology or a mapping. In addition, new CMIs and CMMs logically derivable from the network may appear. Some other missing is-a relations or mappings may also have been repaired by repairing

the current missing is-a relation or mapping (as in the case of (*lacrimal bone*, *bone*) and (*maxilla*, *bone*) described above). Some repaired wrong is-a relations and mappings may also become logically derivable again. In other cases the possible repairing actions for wrong and missing is-a relations and mappings may change. We also need to update the knowledge bases.

3.4 Algorithms in the alignment component

Subsection 3.4.1 presents the algorithms for detection (Phase 1) and validation (Phase 2) in the alignment component. Only CMMs are detected in this component since the detection is based on alignment algorithms. The repairing phase (Phase 3) for the missing mappings in this component is the same—containing the same algorithms—as the repairing phase for missing is-a relations and mappings in the other component. The process of repairing the wrong mappings is different—only those logically derivable from the network are repaired. As the others are not based on existing relations/mappings in the network, they are not repaired.

The input for the alignment component consists of two taxonomies. The output is an alignment.

3.4.1 Detect and validate candidate missing mappings

As explained in Subsection 3.2.1, in ontology alignment mapping suggestions are generated that are essentially CMMs. In **Phase 1** in the alignment component we have currently used the linguistic matchers and the matchers based on auxiliary information (WordNet-based and UMLS-based) from the SAMBO system [62]. The matcher *n-gram* computes a similarity based on 3-grams. The matcher *TermBasic* uses a combination of n-gram, edit distance and an algorithm that compares the lists of words of which the terms are composed. The matcher *TermWN* extends *TermBasic* by using WordNet for looking up is-a relations. The matcher *UMLSM* uses the domain knowledge in UMLS to obtain similarity values. The results of the matchers are combined using a weighted-sum approach in which each matcher is given a weight and the final similarity value between a pair of concepts is the weighted sum of the similarity values divided by the sum of the weights of the used matchers. In addition, we use a single threshold for filtering. A pair of concepts is a mapping suggestion if the similarity value is equal to or higher than a given threshold value.

We note that in the alignment component the search space is not restricted to the *mapped concepts* only—similarity values are calculated for all pairs of concepts. KBs are initialized, in the same way as in the debugging component, for the taxonomy network and the pairs of taxonomies and their alignments. We also note that no initial alignment is needed for this component. Therefore, if alignments do not exist in the network (at all

or between specific ontologies) this component may be used before starting debugging.

The $CMM_{Alignment}$ (mapping suggestions) are presented to a domain expert for validation (**Phase 2**), which is performed in the same way as in the debugging component. The domain expert can use the recommendation algorithms during the validation as well. The $CMM_{Alignment}$ are partitioned into two sets—wrong mappings (WM) and missing mappings (MM). As mentioned, the wrong mappings in (WM) which are logically derivable from the network will be repaired. The others are not based on existing relations/mappings in the network and thus they will not be repaired. However, we store them in order to avoid recomputations, reducing the number of repairing actions, and for conflict checking/prevention. The missing mappings are repaired by adding mappings or is-a relations to the pair of ontologies and their alignment. The concepts in the missing mappings are added to the set of *mapped concepts* (if they are not already there), and they will be used the next time CMMs/CMIs are logically derived in the debugging component.

3.4.2 Repair missing and wrong mappings

Phase 3 in the alignment component uses the same algorithms as presented in Subsection 3.3.2. In the beginning the relevant KBs and sets are initialized, as shown in Figure 3.3.

Repair wrong mappings

The repairing actions for the wrong mappings that can be logically derived from the network are computed and the justifications are presented (**Phase 3.1**) to a domain expert. The repairing actions are ranked in **Phase 3.2** and recommendations, based on the hitting sets, are generated in **Phase 3.3**. In **Phase 3.4** the KBs and the proper sets are updated in correspondence with the repairing actions selected by the domain expert.

Repair missing mappings

Initially, the missing mappings are added to the KBs in the same way as in the debugging component and then we try to repair them using more informative repairing actions. To repair a missing mapping, Source and Target sets are generated using the same algorithms as in the debugging component (**Phase 3.1**) and the repairing process continues with the same actions described for the debugging workflow (**Phase 3.2** and **Phase 3.3**). In **Phase 3.4** the repairing actions are executed analogously to those in the debugging component and their consequences are computed. Additionally the concepts in the repairing actions are added to the set of *mapped concepts* (if not already there).

3.5 Interactions between the alignment component and the debugging component

The main difference between the components is in the detection phase, and this is the place they complement each other. The integration of ontology alignment and ontology debugging provides additional methods for both areas. The ontology alignment can be seen as a special kind of debugging providing detection methods for modelling defects. The alignment component generates CMMs that are validated in the same way as in the debugging component. The CMMs that are validated as correct are often missing mappings that are not found by the debugging component. These may lead to new mapped concepts that are used in the debugging component. The CMMs that are validated as wrong are used to avoid unnecessary recomputations and validations.

It is also the case that the detection of missing mappings using the knowledge intrinsic to the ontology network can be seen as an alignment algorithm. In general, ontology debugging repairs the structure of the ontologies and alignments, which provides better input for the alignment algorithms. For instance, the performance of structure-based matchers (e.g., [62]) and partial-alignment-based preprocessing and filtering methods [58] heavily depends on the correctness and completeness of the is-a structure. Also, the debugging of the alignments raises their quality.

The interaction between the components produces even greater benefits when alignments do not exist at all in the network (i.e., there is no network since the ontologies are not connected). In this case, debugging of the ontologies based on knowledge that is logically derivable from the network would not be possible. However, the alignment component can be used initially to create the necessary alignments (i.e., to create the network), thus providing opportunities for debugging the ontologies, and at the same time improving/debugging the newly created alignments. This means, in practice, that our debugging approach can be used for any two ontologies in a particular domain, regardless of whether an alignment between them is available.

The different phases in and between the components can also be interleaved. This allows for an iterative and modular approach, where, for instance, some parts of the ontologies can be fully debugged and aligned before proceeding to other parts.

Chapter 4

Implemented System

This chapter presents our system RepOSE, an extension of [67]. It is based on the framework presented in Chapter 3. The extended system can be seen from three points of view—as an ontology debugging system where ontology alignment algorithms are used for detecting modelling defects, as an ontology alignment system where various possibilities for adding mappings to the final alignment are presented, and as an integrated ontology alignment and debugging system with the aforementioned advantages.

Following the framework the system has two components—the debugging component and the alignment component. The user loads the ontologies and alignments (when available) in RepOSE. The input for the alignment component consists of two taxonomies while a taxonomy network is required in order to run the debugging process. The output from the debugging component is the set of repaired ontologies and alignments. The output from the alignment component is an alignment. The user can detect/validate/repair defects only in one ontology or one pair of ontologies and their alignment at a time because of the reasons discussed in Chapter 3.

One way to divide the interface components in the system is as components handling is-a relations (CMIs, wrong and missing is-a relations) and corresponding components managing mappings (CMMs, wrong and missing mappings). The debugging component utilizes all interface components, since it deals with both is-a relations and mappings. The alignment component shares the interfaces related to mappings with the debugging component, since the alignment component is only concerned with mappings.

There is no predetermined order in running the components of the framework. However, if the network is not available the alignment component should be run before the debugging process to create the necessary alignments. If alignments are available, running the alignment component first may lead to extending them and thus providing additional debugging opportunities. Running the debugging component first repairs the is-a structure of ontologies and alignments. The repaired ontologies and alignments can

be further used in the structure-based alignment algorithms.

Furthermore, the different phases—detection, validation and repairing—in and between the alignment and debugging components can be interleaved. However, currently, the user has to start with a detection phase, regardless of whether it is held in the debugging component or in the alignment component and whether it detects CMIs or CMMs. Although the framework allows externally generated CMIs/CMMs, the system currently does not support an external input yet.

4.1 Detect and validate candidate missing is-a relations and mappings

Subsection 4.1.1 illustrates the user interface for detecting and validating CMIs used by the debugging component. Subsection 4.1.2 presents the interface for detecting and validating CMMs shared by both framework components.

4.1.1 Detect and validate candidate missing is-a relations

The user can use the tab ‘Step1: Generate and Validate Candidate Missing is-a Relations’ (Figure 4.1) and choose an ontology for which the CMIs are computed. The **Generate Candidate Missing is-a Relations** button runs the detection algorithm. The user can validate all or some of the CMIs as well as switch to another ontology or another tab.

Showing all CMIs at once would lead to information overload and difficult visualization. Showing them one at a time has the disadvantage that the interactions with other is-a relations will not be disclosed. Therefore, as a trade-off we show the CMIs in groups where for each member of the group at least one of the concepts subsumes or is subsumed by a concept of another member in the group. The **Show Ontology** button shows the whole ontology, CMIs, CMMs and the repairing actions when needed.

The CMIs are presented as a directed graph where the nodes represent concepts and the edges represent is-a relations. The grey edges are existing asserted is-a relations, the blue edges are CMIs, and the orange edge (only one at a time) denotes a currently selected CMI. When a CMI is selected, its justification in the ontology network is shown as an extra aid for the user. For instance, in Figure 4.1 (*palatine bone*, *bone*) is selected and its justifications shown in the justifications panel. Concepts in different ontologies are presented with different background color. The brown edges denote mappings existing in the initial alignments.

Initially, CMIs are shown using edges labeled by ‘?’ (as in Figure 4.1 for (*acetabulum*, *joint*)) which the user can toggle to ‘W’ for wrong relations and ‘M’ for missing relations. We used the recommendation algorithm from [67],

4.1. DETECT AND VALIDATE CANDIDATE MISSING IS-A RELATIONS AND MAPPINGS

Figure 4.1: Generating and validating CMIs.

described in Subsection 3.3.1, in order to facilitate the validation process. If an is-a relation is likely to be wrong according to the recommendation algorithm the ‘?’ label is replaced by a ‘W?’ label as for *(upper jaw, jaw)*, if it is likely to be correct the ‘?’ label is replaced by a ‘M?’ label as for *(elbow joint, joint)*.

When a user decides to finalize the validation of a group of CMIs, pressing the **Validate** button, RepOSE checks for contradictions in the current validation as well as with previous decisions and if contradictions are found, the current validation will not be allowed and a message window is shown to the user.

4.1.2 Detect and validate candidate missing mappings

A similar tab ‘Step 2: Generate and Validate Candidate Missing Mappings’ is used to generate and validate CMMs. First, the user chooses the pair of ontologies for which the detection is run. Then the user can select one of the two detection methods—using knowledge intrinsic to the network, i.e., the debugging component or using alignment algorithms, i.e., the alignment component.

Figure 4.2: Aligning.

The **Generate Candidate Missing Mappings** button runs the detection algorithm, which uses the knowledge intrinsic to the network. The **Configure and Run Alignment Algorithms** button opens a configuration window (Figure 4.2) where the user can select the matchers, their weights and the threshold for the computation of the mapping suggestions. Clicking on the **Run** button starts the alignment process. The similarity values for all pairs of concepts belonging to the selected ontologies are computed, combined and filtered, and the resulting mapping suggestions are shown to the user for validation. The validation process continues in a manner that is similar to the process for the CMIs, regardless of the origin of the CMMs. During the validation a label on the edge shows the origin of the CMMs—logically derived from the network, computed by the alignment component or both. The CMMs computed only by the alignment algorithms do not have justifications since they were not logically derived. The rest of the process is as described above.

When alignments are not existing/available at all this tab should be used first in combination with the alignment algorithms in order to create the necessary alignments.

Figure 4.3: Repairing wrong is-a relations.

4.2 Repair missing and wrong is-a relations and mappings

After the detection and validation phases the CMIs and CMMs are divided into wrong and missing is-a relations and mappings. Subsection 4.2.1 presents the user interface for repairing wrong is-a relations and mappings while Subsection 4.2.2 presents the user interface for repairing missing is-a relations and mappings.

4.2.1 Repair wrong is-a relations and mappings

Figure 4.3 shows the RepOSE tab ('Step 3: Repair Wrong is-a Relations') for repairing wrong is-a relations. Clicking on the **Generate Repairing Actions** button results in the computation of repairing actions for each wrong is-a relation of the ontology under repair. The algorithm for these computations is presented in Subsection 3.3.2.

The wrong is-a relations are then ranked in ascending order according to the number of possible repairing actions and shown in a drop-down list. Then, the user can select a wrong is-a relation and repair it using an interactive display. The user can choose to repair all wrong is-a relations in groups or one by one. The display shows a directed graph representing the

justifications. The nodes represent concepts. As mentioned before, concepts in different ontologies are presented with different background colors. The concepts in the is-a relation under repair are shown in red. The edges represent is-a relations in the justifications. These is-a relations may be existing asserted is-a relations (shown in grey), existing asserted mappings (shown in brown), unrepaired missing is-a relations/mappings (shown in blue) and the added repairing actions for the repaired missing is-a relations/mappings (shown in black).

For the wrong is-a relations under repair, the user can choose, by clicking, multiple existing asserted is-a relations and mappings on the display as repairing actions and click the **Repair** button. RepOSE ensures that only existing asserted is-a relations and mappings are selectable, and when the user finalizes the repair decision, RepOSE ensures that the wrong is-a relations under repair and every selected is-a relation and mapping will not be logically derivable from the ontology network after the repairing. Additionally, all consequences of the repair are computed (such as changes in the repairing actions of other is-a relations and mappings and changes in the lists of wrong and missing is-a relations and mappings).

In Figure 4.3 the user has chosen to repair several wrong is-a relations at the same time, i.e., (*brain grey matter*, *white matter*), (*cerebellum white matter*, *brain grey matter*), and (*cerebral white matter*, *brain grey matter*). In this example¹ we can repair these wrong is-a relations by removing the mappings between *brain grey matter* and *Brain.White.Matter*. We note that, when removing these mappings, all these wrong is-a relations will be repaired at the same time.

During the repairing, the user can choose to use the recommendation feature, described in Subsection 3.3.2, by enabling the **Show Recommendation** check box. In the example in Figure 4.3 the highest priority (indicated by pink labels marked ‘Pn’, where n reflects the priority ranking) is given to the mapping (*Brain.White.Matter*, *brain grey matter*), as this is the only way to repair more than one wrong is-a relation at the same time. (Both (*cerebellum white matter*, *brain grey matter*) and (*cerebral white matter*, *brain grey matter*) would be repaired.) Upon the selection of a repairing action, the recommendations are recalculated and the labels are updated. As long as there are labels, more repairing actions need to be chosen.

A similar tab (‘Step 4: Repair Wrong Mappings’) is used for repairing wrong mappings.

4.2.2 Repair missing is-a relations and mappings

Figure 4.4 shows the RepOSE tab (‘Step 5: Repair Missing is-a Relations’) for repairing missing is-a relations. Clicking on the **Generate Repairing Actions** button results in the computation of repairing actions for the missing is-a relations of the ontology under repair. The algorithm for these

¹From OAEI 2010 Anatomy.

4.2. REPAIR MISSING AND WRONG IS-A RELATIONS AND MAPPINGS

Figure 4.4: Repairing missing is-a relations.

computations is presented in Subsection 3.3.2.

They are shown to the user as Source and Target sets (instead of Repair) for easy visualization. Once the Source and Target sets are computed, the missing is-a relations are ranked with respect to the number of possible repairing actions. The first missing is-a relation in the list has the fewest possible repairing actions, and may therefore be a good starting point. When the user chooses a missing is-a relation, its Source and Target sets are displayed on the left and right, respectively, within the **Repairing Actions** panel (Figure 4.4). Both have zoom control and can be opened in a separate window.

Similarly to the displays for wrong is-a relations and mappings, concepts in the missing is-a relations are highlighted in red, existing asserted is-a relations are shown in grey, unrepaired missing is-a relations in blue and added repairing actions for the missing is-a relations in black. For instance, Figure 4.4 shows the Source and Target sets for the missing is-a relation (*lower respiratory tract cartilage*, *cartilage*), which contain 2 and 21 concepts, respectively. The Target panel shows also the unrepaired missing is-a relation (*nasal septum*, *nasal cartilage*). The **Justifications of current relation** panel is a read-only panel that displays the justifications of the current missing is-a relation as an extra aid.

For the selected missing is-a relation, the user can also ask for recom-

mended repairing actions by clicking the **Recommend** button. In general, the system presents a list of recommendations. By selecting an element in the list, the concepts in the recommended repairing action are identified by round boxes in the panels. For instance, for the case in Figure 4.4, the recommendation algorithm proposes to add (*respiratory system cartilage, cartilage*). Using the recommendation algorithm we recommend structural repairs that try to use as informative repairing actions as possible (pref2 in Subsection 3.2.2).

The user can repair the missing is-a relation by selecting a concept in the Source panel and a concept in the Target panel and clicking on the **Repair** button. When the selected repairing action is not in $Repair(a, b)$, the repairing will not be allowed and a message window is shown to the user. Additionally, all consequences of a chosen repair are computed (such as changes in the repairing actions of other is-a relations and mappings and changes in the lists of wrong and missing is-a relations and mappings).

The tab ‘Step 6: Repair Missing Mappings’ is used for repairing missing mappings. The main differences between this tab and the one for repairing missing is-a relations are that we deal with two ontologies and their alignment, and that the repairing actions can be is-a relations within an ontology as well as mappings. The missing mappings found from the alignment component, but not from the debugging component, do not have justifications.

Chapter 5

Experiments and Discussions

Several experiments were performed with our implemented system. This chapter presents them together with our experiences and reflections on their results. Using the experiments we not only demonstrate the benefits from our unified approach for ontology alignment and debugging, but we also show the essential need for a system during this process. Without a dedicated system, reliable alignment and debugging is tedious if not infeasible, especially for large ontologies.

Section 5.1 presents in detail one experiment focused only on debugging of an ontology network. Section 5.2 presents three experiments exploring the advantages of the integration of the ontology alignment and debugging. Each experiment is followed by a subsection that discusses it and its results. A general discussion in Section 5.3 summarizes the experiments and provides general reflections on the approach and the system.

5.1 Ontology debugging

The experiment presented in the next subsection employs only the detection algorithms in the debugging component, i.e., only knowledge intrinsic to the network.

5.1.1 OAEI Anatomy 2010

Experiment setup

In this experiment a domain expert ran a complete debugging session on a network consisting of the two ontologies and the alignment from the Anatomy track in OAEI 2010—Adult Mouse Anatomy Dictionary (AMA), the NCI Thesaurus anatomy (NCI-A) and the partial reference alignment

	concepts	asserted is-a relations	asserted equivalence mappings	asserted is-a mappings
AMA	2744	1807	-	-
NCI-A	3304	3761	-	-
Alignment	-	-	986	1

Table 5.1: Ontology debugging: OAEI Anatomy 2010—ontologies and alignment.

	candidate missing: all/ non-redundant	missing	wrong	added: is-a relations/ more informative	removed: is-a relations/ mappings
AMA	200/123	102	21	85/22	13/-
NCI-A	127/80	61	19	57/8	12/-
Alignment	-	-	-	-	-/12

Table 5.2: Ontology debugging: OAEI Anatomy 2010—final result.

(PRA). These ontologies as well as the alignment were developed by domain experts. For the 2010 version of OAEI, AMA contains 2,744 concepts and 1,807 asserted is-a relations, while NCI-A contains 3,304 concepts and 3,761 asserted is-a relations. The alignment contains 986 equivalence and 1 subsumption mapping between AMA and NCI-A. This information is summarized in Table 5.1. The experiment was performed on an Intel Core i7-950 Processor 3.07GHz with 6 GB DDR2 memory under Windows 7 Ultimate operating system and Java 1.7 compiler. The domain expert completed debugging this network within 2 days. Since the system provided nearly immediate response in most cases, much of this time was spent making decisions for validation and repairing (essentially looking up and analyzing information to make decisions) and interactions with RepOSE.

Results

Table 5.2 summarizes the results of the detection and repairing of defects in the is-a structures of the ontologies and the mappings. The system detected 200 CMIs¹ in AMA of which 123 were non-redundant. Of these non-redundant CMIs 102 were validated as missing is-a relations and 21 were validated as wrong is-a relations. For NCI-A 127 CMIs, of which 80 non-redundant, were detected. Of these non-redundant CMIs 61 were validated as missing is-a relations and 19 were validated as wrong is-a relations. To repair these defects 85 is-a relations were added to AMA and 57 to NCI-A, 13 is-a relations were removed from AMA and 12 from NCI-A, and

¹As was explained earlier in Subsection 3.3.1, in order to detect CMs with the debugging component at least three ontologies and two alignments are needed. Since the network in this example contains only two ontologies and their alignment, CMs cannot be detected.

	CMI missing: accept/reject	CMI wrong: accept/reject	repair missing: accept/reject
AMA	81/8	7/13	69/16
NCI-A	27/2	6/2	43/14

Table 5.3: Ontology debugging: OAEI Anatomy 2010—recommendations.

12 mappings were removed from the alignment. In 22 cases in AMA and 8 cases in NCI-A a missing is-a relation was repaired using a more informative repairing action, thereby adding new knowledge to the network.

The ranking and recommendations seemed useful. Table 5.3 summarizes the recommendation results. Regarding CMIs, 81 and 27 recommendations that the relation should be validated as a missing is-a relation were accepted for AMA and NCI-A, respectively, while 8 and 2 were rejected. When the system recommended that a CMI should be validated as a wrong is-a relation, the recommendation was accepted in 7 out of 20 cases for AMA and 6 out of 8 cases for NCI-A. The recommendations regarding repairing missing is-a relations were accepted in 69 out of 85 cases for AMA and 43 out of 57 cases for NCI-A. We note that the system may not always give a recommendation. This is the case, for instance, when there is no information about the is-a relation under consideration in the external sources. In the remainder of this subsection we discuss the experimental session, the results, and our experience with the system in more detail.

Detecting and validating candidate missing is-a relations for the first time. After loading AMA, NCI-A and the alignment, it took less than 30 seconds to detect all CMIs for each of the ontologies. As a result, RepOSE found 192 CMIs in AMA and 122 in NCI-A. Among these CMIs, 115 in AMA and 75 in NCI-A are displayed in 24 groups and 18 groups, respectively, for validation, while the remaining 77 in AMA and 47 in NCI-A are redundant and thus ignored. With the help of the recommendations, the domain expert identified 20 wrong is-a relations and 95 missing is-a relations in AMA. For NCI-A the domain expert identified 17 wrong and 58 missing is-a relations. These results are summarized in Table 5.4. As for the recommendation, the use of asserted part-of relations in ontologies together with WordNet recommended 20 possible wrong is-a relations in AMA and 8 in NCI-A, of which 7 in AMA and 6 in NCI-A were accepted as decisions. WordNet and UMLS recommended 84 possible missing is-a relations in AMA and 29 in NCI-A, of which 77 in AMA and 27 in NCI-A were accepted as decisions.

Repairing wrong is-a relations for the first time. After the validation phase, the domain expert continued with the repairing of wrong is-a relations. In this experiment, for the 20 wrong is-a relations in AMA and 17 in NCI-A, each wrong is-a relation has only one justification, consisting of two or more mappings and one or more asserted is-a relations in the other ontology. Therefore, the repairing is done by removing the involved asserted is-a relations and/or mappings (Table 5.4). For example, for the wrong is-a

	candidate missing: all/ non-redundant	missing	wrong	repair wrong removed	repair missing: self/ more informative/ other
AMA	192/115	95	20	12	59/19/17
NCI-A	122/75	58	17	11	49/5/4
Alignment	-	-	-	11	-

Table 5.4: Ontology debugging: OAEI Anatomy 2010—first iteration results.

relation (*Ascending Colon*, *Colon*) in NCI-A (which actually is a part-of relation), its justification contains two equivalence mappings (between *Ascending Colon* and *ascending colon*, and between *Colon* and *colon*) and an asserted is-a relation (*ascending colon*, *colon*) in AMA. The repairing was done by removing (*ascending colon*, *colon*) from AMA. As shown before in Figure 4.3 in Subsection 4.2.1, the wrong is-a relation (*brain grey matter*, *white matter*) in AMA was repaired by removing the mappings between *Brain.White.Matter* and *brain grey matter*.

We note that 11 mappings were removed, 8 of them as a result of wrong is-a relations in AMA and 3 as a result of the debugging of NCI-A. Additionally, several wrong is-a relations were repaired by repairing other wrong is-a relations.

Repairing missing is-a relations in AMA and NCI-A for the first time. As the next step, the domain expert proceeded with the repairing of missing is-a relations in AMA. At this point there were 95 missing is-a relations to repair, and it took less than 10 seconds to generate the repairing actions for them. Almost all Source and Target sets were small enough to allow a good visualization. For 59 missing is-a relations, the domain expert used the missing is-a relation itself as the repairing action (i.e., the least informative repairing actions). For 19 missing is-a relations, the domain expert used more informative repairing actions, which also repaired 17 other missing is-a relations. These results are summarized in the last column of Table 5.4. The recommendation algorithm was used in 78 cases. In 63 of them the selected repairing action was among the recommended repairing actions and in 9 of them the recommendation algorithm suggested more informative repairing actions.

The domain expert then continued with the repairing of missing is-a relations in NCI-A. Out of the 58 missing is-a relations to be repaired, 49 missing is-a relations were repaired using themselves as the repairing actions, 5 were repaired using more informative repairing actions, and 4 were repaired by the repairing of others (Table 5.4). For example, for the repairing of missing is-a relation (*Epiglottic Cartilage*, *Laryngeal Connective Tissue*) in NCI-A, the domain expert used more information repairing action (*Laryngeal Cartilage*, *Laryngeal Connective Tissue*), where *Laryngeal Cartilage* is a super-concept of *Epiglottic Cartilage* in NCI-A. This repairing also repaired

3 other missing is-a relations, i.e., (*Cricoid Cartilage*, *Laryngeal Connective Tissue*), (*Arytenoid Cartilage*, *Laryngeal Connective Tissue*) and (*Thyroid Cartilage*, *Laryngeal Connective Tissue*), where *Cricoid Cartilage*, *Arytenoid Cartilage* and *Thyroid Cartilage* are sub-concepts of *Laryngeal Cartilage* in NCI-A. The recommendation algorithm was used in 54 cases. In 42 of them the selected repairing action was among the recommended repairing actions and in 3 of them the recommendation algorithm suggested more informative repairing actions.

The subsequent debugging process. The repairing of the wrong and the missing is-a relations in both ontologies resulted in 6 non-redundant new CMIs in AMA and 4 in NCI-A. In each ontology 1 of those was validated as wrong and the others as missing. 2 of the 5 missing is-a relations in AMA were repaired by themselves and 3 using more informative repairing actions. The wrong is-a relation was repaired by removing an is-a relation in NCI-A. The 3 missing is-a relations in NCI-A were repaired by using more informative repairing actions. The wrong is-a relation was repaired by removing a mapping from the alignment. The repairing of these newly found relations led to two more CMIs in AMA, which were validated as correct and repaired by themselves, and one CMI in NCI-A, which was validated as wrong and repaired by removing an is-a relation in AMA. At this point there were no more CMIs to validate, and no more wrong or missing is-a relations to repair.

Discussion

Apart from showing the need for ontology debugging, this experiment highlights the benefits from our system during the detection phase where manual detection is out of the question. Even if we assume that all asserted, more than 6000, is-a relations and mappings in the network can be checked manually in order to find the wrong ones, this is simply infeasible for the missing is-a relations and mappings (where $\sum \frac{n(n-1)}{2}$ pairs should be checked² in order to find all missing is-a relations and mappings in the network). Our approach took around 30 seconds and explores the domain knowledge intrinsic to the network. To reduce the number of CMIs for validation and to show them in their context the CMIs are presented to the domain expert in groups where the redundant are excluded.

Furthermore, our system provides support for the domain expert during the repairing phase—calculating and presenting the justifications of the wrong is-a relations and calculating and presenting the possible repairing actions for the missing is-a relations. A trivial way to repair a missing is-a relation is to add it to the ontology (i.e., the least informative repairing action). However, our system calculates all possible repairing actions for a missing is-a relation and thus provides the domain expert with the possibility of adding different repairing actions (i.e., more informative as explained

²n is the number of concepts in the ontology network

in Subsection 3.2.2). We observe that during this experiment for 19 missing is-a relations in AMA and 5 in NCI-A, the domain expert has used repairing actions that are more informative than the missing is-a relation itself. This means that for each of these the domain expert has added knowledge that was not intrinsic to (i.e., logically derivable from) the network. Thus, the knowledge represented by the ontologies and the network has increased. Our system also calculates the consequences of user actions and keeps track of them. If the user actions contradict themselves or previous user actions, a warning message describing the contradiction appears.

5.2 Integration of ontology debugging and ontology alignment

This subsection presents three experiments showing the benefits from the integration of the ontology alignment and debugging. Each experiment consists of several smaller experiments (called *runs* in the text) focusing on different aspects of the integration. Each experiment is presented with its setup, detailed description of the different runs, an explanation for each of the iterations in the runs and a follow-up discussion. Both components are used in all experiments presented in this subsection.

Subsections 5.2.1 and 5.2.2 present experiments with the ontologies from the Anatomy track in OAEI 2011 and the Benchmark track in OAEI 2010 respectively. Subsection 5.2.3 presents a use case that is performed together with the Swedish National Food Agency³. In this collaboration we applied our approach to the ontology they have developed—ToxOntology and MeSH [6].

5.2.1 OAEI Anatomy 2011

Experiment setup

This experiment consists of three runs where each run is a complete experiment on its own and demonstrates different use cases of our system. As input for Run I and II we used the two ontologies from the Anatomy track of OAEI 2011—AMA contains 2,737 concepts and 1,807 asserted is-a relations, and NCI-A contains 3,298 concepts and 3,761 asserted is-a relations. The input for the last run contained the reference alignment (1516 equivalence mappings between AMA and NCI-A) along with the two ontologies. The reference alignment was used indirectly as external knowledge during the validation phase in the first two runs. The runs were performed on an Intel Core i7-2620M Processor 2.7GHz with 4 GB memory under Windows 7 Professional operating system and Java 1.7 compiler.

³Livsmedelsverket—slv.se

5.2. INTEGRATION OF ONTOLOGY DEBUGGING AND ONTOLOGY ALIGNMENT

	candidate missing mappings	missing: $\equiv/\leftarrow,\rightarrow$	wrong: $\equiv/\leftarrow,\rightarrow$	repair missing: $\equiv/\leftarrow/\rightarrow$ / derivable/ more informative	repair missing is-a relations
Alignment	1384	1286/39	59/39	1286/21/8/5/5	-
AMA	-	-	-	-	3
NCI-A	-	-	-	-	2

Table 5.5: Ontology alignment and debugging: OAEI Anatomy 2011—Run I results—debugging of the alignment.

Run I

The first run demonstrates a complete debugging and alignment session where the input is a set comprised of the two ontologies. Since a network did not exist, we first employed the alignment component—after loading the ontologies mapping suggestions were computed using matchers TermWN and UMLSM, weight 1 for both and threshold 0.5. This resulted in 1384 mapping suggestions. The 1233 mapping suggestions that are also in the reference alignment were validated as missing equivalence mappings (although, as we will see, there are defects in the reference alignment) and repaired by adding them to the alignment. The others were validated manually and resulted in missing mappings (53 equivalence and 39 is-a) and wrong mappings (59 equivalence and 39 is-a). These missing mappings were repaired by adding 53 equivalence and 29 is-a mappings (5 of them more informative is-a mappings) and 5 is-a relations (3 to AMA and 2 to NCI-A). 5 of these missing mappings were repaired by repairing others. Among the wrong mappings there were 3 that were logically derivable in the network. These were repaired by removing 2 is-a relations from NCI-A. Table 5.5 summarizes the results. This sequence of actions can be considered a procedure of debugging missing mappings.

The generated alignment was then used in the debugging of the network created by the ontologies and the alignment. Two iterations with the debugging component were performed, since the repairing of wrong and missing is-a relations in the first iteration led to the detection of new CMIs which had to be validated and repaired. Over 90% of the CMIs for both ontologies were detected during the first iteration, the detection of CMIs took less than 30 seconds per ontology. Table 5.6 summarizes the results.

In total the system detected 263 non-redundant (410 in total) CMIs for AMA and 183 non-redundant (355 in total) CMIs for NCI-A. The non-redundant CMIs were displayed in groups, 45 groups for AMA and 31 for NCI-A. Among the 263 non-redundant CMIs in AMA 224 were validated as correct and 39 as wrong. In NCI-A 166 were validated as correct and 17 as wrong. The 39 wrong is-a relations in AMA were repaired by removing 30 is-a relations from NCI-A, and 8 equivalence and 1 is-a mapping from the alignment. The 17 wrong is-a relations in NCI-A were repaired by removing

	candidate missing: all/ non-redundant	missing	wrong	repair wrong removed	repair missing: self/ more informative/ other
AMA	410/263	224	39	30	144/57/23
NCI-A	355/183	166	17	17	127/13/26
Alignment	-	-	-	8 $\equiv$ and 1 $\rightarrow$	-

Table 5.6: Ontology alignment and debugging: OAEI Anatomy 2011—Run I results—debugging of the ontologies.

17 is-a relations in AMA. The missing is-a relations in AMA were repaired by adding 201 is-a relations—in 144 cases the missing is-a relation itself and in 57 cases a more informative is-a relation. 23 of the 224 missing is-a relations became logically derivable after repairing some of the others. To repair the missing is-a relations in NCI-A 140 is-a relations were added—in 127 cases the missing is-a relation itself and in 13 cases a more informative is-a relation. 26 out of the 166 missing is-a relations were repaired while other is-a relations were repaired.

We observe that for 57 missing is-a relations in AMA and 13 in NCI-A the repairing actions are more informative than the missing is-a relation itself. This means that for each of these, knowledge that was not logically derivable from the network before was added to it. Thus, the knowledge represented by the ontologies and the network has increased.

Run II

For this run the alignment process was carried out twice and at the end the alignments were compared. This run used the same matchers, weights and threshold as in Run I. During both runs of the alignment process the CMMs (mapping suggestions) were computed and validated in the same manner; to this point the results for Run II are the same as the respective results in Run I and they can be seen in the first three columns in Table 5.5. The difference between the two runs is in the repairing phase. When the alignment process was carried out for the first time the missing mappings were repaired by directly adding them to the final alignment without benefiting from the repairing algorithms, in the same way most of the alignment systems do. The final alignment contained 1286 equivalence and 39 is-a⁴ mappings.

During the repairing phase, when the alignment process was carried out for the second time, the debugging component was used to provide alternative repairing actions to those available in the initial set of mapping suggestions. The results can be seen in the last two columns in Table 5.5. The

⁴5 of these are repaired in the second run by adding is-a relations in the ontologies.

final alignment then contained 1286 equivalence mappings from the mapping suggestions, 24 is-a mappings from the mapping suggestions and 5 more informative is-a mappings, thus adding knowledge to the network. 5 further mapping suggestions were repaired adding is-a relations (3 in AMA and 2 in NCI-A) and thus adding more knowledge to each of the ontologies. 5 more mapping suggestions became logically derivable from the network as a result of the repairing actions for other CMMs.

Run III

In this run the detection phase with the debugging component was carried out twice and the detected CMMs were compared between the runs. The input for the first run was the set of the two ontologies and their alignment from the Anatomy track in OAEI 2011. The network was loaded in the system and the CMMs were detected. 496 CMMs were detected for AMA, of which 280 were non-redundant. For NCI-A 365 CMMs were detected of which 193 were non-redundant. The same input was used in the second run. However, the alignment algorithms were used to extend the set with mappings prior to generating the CMMs. The set-up for the aligning was the same as in Run I and the mapping suggestions were computed, validated and repaired in the same way as well. Then CMMs were generated—638 CMMs were detected for AMA of which 357 were non-redundant, and 460 CMMs for NCI-A, of which 234 were non-redundant. In total 145 new CMMs were detected for AMA—120 were validated as missing and 25 validated as wrong⁵. For NCI-A 103 new CMMs were detected—53 were validated as missing and 50 as wrong.

Discussion

Run I shows the usefulness of the system through a complete session where an alignment was generated and many defects in the ontologies were repaired. Some of the repairs added new knowledge. As a side effect, we have shown that the ontologies that are used by the OAEI contain over 200 and 150 missing is-a relations, respectively, and 39 and 17 wrong is-a relations, respectively. We have also shown that the alignment is not complete and contains incorrect information. We also note that our system allows validation and allows a domain expert to distinguish between equivalence and is-a mappings. Most ontology alignment systems do not support this.

Run II shows the advantages for ontology alignment when a debugging component is added. The debugging component allowed more informative mappings to be added and reduced redundancy in the alignment, as well as debugging the ontologies leading to further reduced redundancy in the

⁵The sum of the newly generated CMMs and those in the first run is not equal to the number of the CMMs in the second run because some of the CMMs generated in the first run are logically derivable in the second run.

Ontologies and Alignments	concepts	asserted is-a relations	asserted equivalence mappings	asserted is-a mappings
Ontologies:				
101	36	25	-	-
301	15	16	-	-
302	13	11	-	-
303	56	47	-	-
304	39	31	-	-
Alignments:				
101 - 301	-	-	14	8
101 - 302	-	-	11	12
101 - 303	-	-	16	2
101 - 304	-	-	28	2

Table 5.7: Ontology alignment and debugging: OAEI Benchmark 2010—ontologies and alignments.

alignment. New knowledge was added that had not been found when only aligning. In general, this results in higher quality alignments and ontologies.

Run III shows that the debugging process can take advantage of the alignment component even when an alignment is available. The alignment algorithms can provide additional mapping suggestions thus extending the alignment. More mappings between two ontologies means higher coverage and possibly more defects detected and repaired. In the experiment more than 100 CMIs (of which many were correct) were detected for each ontology using the extended set of mappings. We also note that the initial alignment contained many mappings (1516). In the case that an alignment contains fewer mappings the benefit to the debugging process will be even more significant.

5.2.2 OAEI Benchmark 2010

Experiment setup

This subsection presents an experiment that consists of two parts (runs) performed on a taxonomy network from the Benchmark track in the Ontology Alignment Evaluation Initiative 2010. As in the previous subsection, each run can be considered an experiment on its own. Details regarding the network are available in Table 5.7. The network consists of 5 small ontologies connected in a star layout through four sets with mappings, i.e., alignments do not exist between all pairs of ontologies. The five ontologies are called 101, 301, 302, 303 and 304. They contain 36, 15, 13, 56 and 39 concepts and 25, 16, 11, 47 and 31 asserted is-a relations, respectively. Alignments are only available between 101-301, 101-302, 101-303 and 101-304 and contain 22, 23, 18 and 30 mappings, respectively. The experiment was performed on an Intel Core i7-2620M Processor 2.70GHz with 4 GB memory, running

Ontologies and Alignments	candidate missing: all/non- redundant	missing/ derivable after	wrong/ repaired by others	added: is-a relations/ mappings/ more informative	removed: is-a relations/ mappings
Ontologies:					
101	7/7	2/-	5/-	2/-/-	1/-
301	1/1	1/-	-/-	1/-/-	-/-
302	1/1	1/-	-/-	1/-/-	-/-
303	1/1	1/-	-/-	1/-/-	-/-
304	8/7	6/-	1/-	6/-/3	5/-
Alignments:					
101 - 301	-/-	-/-	-/-	-/-/-	-/-
101 - 302	-/-	-/-	-/-	-/-/-	-/5
101 - 303	-/-	-/-	-/-	-/-/-	-/1
101 - 304	1/1	1/-	-/-	-/1/-	-/3
301 - 302	60/28	25/4	3/1	-/21/-	-/-
301 - 303	57/38	38/11	-/-	-/27/-	-/-
301 - 304	71/37	36/10	1/-	-/26/1	-/-
302 - 303	61/28	25/4	3/3	-/21/-	-/-
302 - 304	78/28	26/5	2/1	-/21/1	-/-
303 - 304	74/40	39/13	1/-	-/26/1	-/-

Table 5.8: Ontology alignment and debugging: OAEI Benchmark 2010—Run I—final result.

the Windows 7 Professional operating system and Java 1.7 compiler. Each experiment took around two and a half hours.

Run I presents a complete debugging session and it is compared with Run II, which presents a session that combines ontology alignment and debugging. Both runs contain five iterations that are described in detail. Their results are compared and discussed at the end of the subsection.

Run I

This run demonstrates a complete debugging session on the network. Five iterations were needed to complete the session—three for detection, validation and repairing of the CMIs and two for the CMMs. This subsection presents the iterations one by one. The summarized results from all iterations in Run I are at the beginning of the Discussion subsection. Most of the CMIs and CMMs were detected during the first detection (the first iteration during the experiment for the CMIs and the third for the CMMs). Table 5.8 presents the final results from this experiment.

CMIs were detected, validated and repaired for each ontology during the first iteration. Their repairing actions led to the detection of a few more CMIs during the second iteration. They were validated and repaired as

well. During the two iterations 15 non-redundant (16 redundant) CMIs were detected in total. 9 of the non-redundant CMIs were validated as missing and the remaining 6 as wrong is-a relations. The wrong is-a relations were repaired by removing 4 mappings and 4 is-a relations from the network. The missing is-a relations were repaired adding is-a relations to the respective ontologies. In 2 cases the added is-a relations were more informative than the missing is-a relations under repair.

CMMs were detected only with the debugging component during the third iteration. The system derived CMMs for all pairs of ontologies for which alignments were not available. No CMMs were detected for the available alignments since one of the ontologies participated in all alignments, and for detecting CMMs from the network at least one alignment where this ontology does not participate was required. 198 non-redundant CMMs were detected and 189 of them were validated as correct. The other 9 were validated as wrong and repaired by removing 4 existing mappings and 2 is-a relations in total. Some of the wrong mappings were repaired by the repairing actions for other wrong mappings. These are shown in the fourth column in Table 5.8, under the heading ‘repaired by others’. The missing mappings were added to the corresponding alignments in 142 cases. In 47 cases the missing mappings became logically derivable after other missing mappings were repaired (the ‘derivable after’ label in the third column in Table 5.8). In 3 cases the added mappings were more informative than the missing mappings themselves.

After all CMMs were repaired the system detected two more CMIs (fourth iteration), both validated as correct. One of them was repaired by adding it to the corresponding ontology and the other was repaired by a more informative repairing action. Then CMMs were generated and validated again (fifth iteration) which resulted in 1 correct and 1 wrong CMM. The correct one was repaired by adding it and the wrong one was repaired by removing a mapping. The debugging session ended at that point since no more CMIs and CMMs were detected and those previously detected had been already repaired.

Run II

In this run CMMs were detected initially with the alignment component and then with the debugging component. The final results are summarized in Table 5.9. Five iterations were performed in this experiment as well. Since the alignment component is only involved in the CMM detection the first two iterations for detecting and repairing CMIs were the same as in Run I. This subsection presents the iterations one by one. The summarized results from all iterations in Run II are at the beginning of the Discussion subsection.

In the third iteration CMMs were detected not with the debugging component but with the alignment component instead. We used the TermWN matcher with threshold 0.5 and weight 1. Mapping suggestions were gen-

5.2. INTEGRATION OF ONTOLOGY DEBUGGING AND ONTOLOGY ALIGNMENT

Ontologies and Alignments	candidate missing: all/non-redundant	missing/derivable before/derivable after	wrong/ repaired by others	added: is-a relations/ mappings/ more informative	removed: is-a relations/ mappings
Ontologies:					
101	6/6	1/-/-	5/-	1/-/-	-/-
301	1/1	1/-/-	-/-	1/-/-	-/-
302	1/1	1/-/-	-/-	1/-/-	-/-
303	1/1	1/-/-	-/-	1/-/-	-/-
304	7/6	5/-/-	1/-	5/-/2	4/-
Alignments:					
101 - 301	16/-	2/2/0	14/-	-/-/-	-/-
101 - 302	17/-	2/1/0	15/-	-/1/-	-/5
101 - 303	34/-	4/2/0	30/-	-/2/-	-/2
101 - 304	43/-	8/4/0	35/-	-/4/-	-/3
301 - 302	33/-	22/0/0	11/-	-/22/-	-/-
301 - 303	45/-	31/0/3	14/-	-/28/-	-/-
301 - 304	50/-	30/0/2	20/-	-/28/-	-/-
302 - 303	44/-	27/0/3	17/3	-/24/1	-/-
302 - 304	49/-	28/0/3	21/1	-/25/-	-/-
303 - 304	84/-	47/0/9	37/-	-/38/1	-/-

Table 5.9: Ontology alignment and debugging: OAEI Benchmark 2010—Run II—final result.

erated for all pairs of ontologies. Every mapping suggestion is presented to the user as two is-a relations with opposite directions. All mappings suggestions were shown to the user although some of them were logically derivable from other suggestions in the network, i.e., they were redundant. The redundant CMMs were shown as well since their derivation paths contain non-validated (possibly wrong) is-a relations/mappings, which could be removed later during the repairing phase (if validated as wrong), and thus the logically derivable ones would be no longer derivable.

During this experiment CMMs for the pairs of ontologies for which the alignments were available in advance were found as well. Most of them (93 out of 107) were validated as wrong. 14 were validated as correct. The mapping suggestions validated as wrong were only stored for future validations and were not repaired since they did not actually exist in the network (if they are logically derivable from the network they should be repaired as well). 5 out of 14 validated as correct were repaired by adding them while the remaining 9 were already logically derivable from the pair of ontologies under repair and its alignment (the ‘derivable before’ label in the third column in the table). 4 of the 5 repaired were not found in the previous experiment. 276 mapping suggestions were calculated for the pairs of ontologies for which alignments were not available in advance. 165 were

Ontologies and Alignments	candidate missing: all/non- redundant	missing/ derivable before/ derivable after	wrong/ repaired by others	added: is-a relations/ mappings/ more informative	removed: is-a relations/ mappings
Ontologies:					
Experiment I	18/17	11/-/-	6/-	11/-/3	6/-
Experiment II	16/15	9/-/-	6/-	9/-/2	4/-
Alignments:					
Experiment I	402/200	190/-/47	10/5	-/143/3	-/9
Experiment II	415/-	201/9/20	214/4	-/172/2	-/10

Table 5.10: Ontology alignment and debugging: OAEI Benchmark 2010—comparison between Run I and Run II

validated as correct, 148 were repaired by adding them and 1—by adding a more informative repairing action, while 16 became logically derivable after repairing the others (the ‘derivable after’ label in the third column in the table). 111 were validated as wrong. 23 of the 149 repaired were not found in the previous experiment.

The next (fourth) iteration performed with the debugging component led to the detection of 31 CMMs in total for almost all pairs of ontologies. 22 were validated as correct and the remaining 9 as wrong. 5 mappings were removed to repair the wrong ones since these actually existed in the network. In 17 cases the missing ones were repaired by adding them, in 1 case by adding a more informative mapping and in 4 cases the mappings became logically derivable after repairing other missing mappings.

One last (fifth) iteration to detect CMMs from the network was done, which resulted in 1 CMM validated as wrong (1 mapping was removed to repair it). No more CMIs and CMMs were found at that point and all those previously detected had been already repaired.

Discussion

Here we compare and discuss the results from both runs. Their final results are summarized in the next two paragraphs and Table 5.10.

Run I shows a complete debugging session with the network using only the debugging component. 17 non-redundant CMIs and 200 CMMs were detected in total (18 and 402 redundant respectively). 11 CMIs and 190 CMMs were validated as correct. They were repaired adding 11 is-a relations (3 of them more informative) and 143 mappings (3 of them more informative). In 47 cases the missing mappings become logically derivable from the network after repairing others and thus they were not repaired since repairing them would lead to redundancies. The wrong CMIs (6) and CMMs (10) were repaired by removing 6 is-a relations and 9 mappings. Sometimes the re-

pairing actions for a wrong is-a relation/mapping include more than one is-a relation/mapping. 5 wrong mappings were repaired while repairing others.

In Run II the alignment component was used prior to the debugging component. During this run 15 non-redundant CMIs were detected (16 in total), 9 validated as correct and 6 as wrong. The correct CMIs were repaired by adding them in 7 cases and by adding more informative repairing actions in 2 cases. 415 redundant CMMs were calculated from both components in total and presented to the user. 201 were validated as correct (9 of them were logically derivable from the pairs of ontologies and their alignments as well) and 214 were validated as wrong. To repair the validated as correct CMMs 172 missing mappings were added (2 more informative) and 20 became logically derivable from the network after adding the others. Most of the validated wrong mappings came from the alignment component and did not actually exist in the network and thus they were not repaired. The others were repaired removing 4 is-a relations and 10 mappings. Sometimes the repairing actions for a wrong is-a relation/mapping include more than one is-a relation/mapping. 4 wrong mappings were repaired while repairing others.

As mentioned above in Run II all CMMs, including those that were redundant, were shown to the user, i.e., the CMMs for validation were doubled. In Run I, when only the debugging component was used, only the non-redundant CMMs were shown to the user. The redundant CMMs in Run I are logically derivable if those shown to the user are validated as correct. If they are validated as wrong those that were redundant will be no longer redundant, but they will still be logically derivable from the network the next time the detection is run. In the case when the alignment component was used the redundant ones are not logically derivable and thus they will not be derived if the user validates the others as wrong (the alignment algorithms should be run again in order to show them).

During Run II the alignment algorithms were run only at the beginning to create/extend the initial alignments. Since our alignment algorithms currently do not employ any structure-based strategies, running them again would not lead to discovering new mapping suggestions. If such strategies were employed the alignment process could benefit from the repaired structure of the ontologies and possibly generate new mapping suggestions. On the other hand, the debugging component could be run as long as it detects new CMIs and CMMs.

The high number of wrong mappings in Run II can be explained with the selected alignment algorithm and threshold. In this run the threshold was 0.5 in order to get more mapping suggestions.

Direct comparison of the results does not show a considerable advantage from the interaction between the two components (presented in Run II)—almost the same number of CMIs (11 versus 9) and CMMs (190 versus 201) were detected. However, the missing mappings in Run II were repaired by adding 172 mappings while in Run I—by adding only 143 mappings, i.e.,

more mappings were added in the second run. 29 mappings became logically derivable in Run II and 47 in Run I after repairing the missing mappings. 27 mappings that were not found and were not logically derivable in Run I were added in Run II. The concepts in these 27 mappings were added to the set with mapped concepts (if not already there) and were later used when CMMs were detected from the network.

It should be noted that the number of removed mappings and is-a relations is different in each experiment. In Run II one additional mapping was removed. This happened because after aligning ontologies 304 and 303 a mapping between another pair of ontologies became logically derivable (it was not derivable from the network before aligning these ontologies). The derivable mapping was validated as wrong, which led to the removal of the additional mapping. During Run I two more is-a relations were removed. Since we first detected and repaired CMMs with the alignment component, the mapping causing their removal was not found in the second experiment because it became logically derivable (from the pair of ontologies and its alignment) after the alignment process. It should be noted that if the detection phases in each of the two components were run one after the other (prior to any repairing) these is-a relations would be found and removed in the second experiment as well.

The removal of the two is-a relations described in the previous paragraph led to discovering two more CMIs. This is the reason for the difference in the number of the CMIs in Run I and II.

5.2.3 ToxOntology-MeSH use case

This experiment was conducted in collaboration with the Swedish National Food Agency (SNFA). An alignment between an ontology created by SNFA—ToxOntology—and an already curated index, in this case MeSH, was deemed necessary. In this context our integrated ontology alignment and debugging framework was very suitable for their needs—an initial alignment was created by the alignment component and then the ontology and the alignment were further refined through debugging. Since our integrated system was not fully implemented at that time the work has been done by two systems—the old version of RepOSE (as debugging component) and a version of the SAMBO system (for creating the alignment) that was further integrated in RepOSE.

Both systems require input in RDF or OWL format, however, MeSH is not available in either of these. Thus, the first step in our work was to translate MeSH into OWL.

The size and the setting of the experiment provide us with the possibility of comparing the repairing process carried out with our system RepOSE with the repairing process carried out manually by domain experts. We performed two runs—Run I and Run II. In the first run we used the validated alignment obtained from SAMBO as input. In order to observe the repairing

similarity value	suggestions	equivalence/ ToxOntology isa MeSH/ MeSH isa ToxOntology	related	wrong
≥ 0.8	41	29/2/2	1	7
$\geq 0.5, < 0.8$	419	9/18/31	42	319
$\geq 0.4, < 0.5$	906	2/21/14	83	786
$\geq 0.35, < 0.4$	146	1/2/2	117	24

Table 5.11: Ontology alignment and debugging: ToxOntology-MeSH—validation of mapping suggestions—initial alignment.

process in RepOSE during the second run we used a nonvalidated alignment as input.

Experiment setup

ToxOntology is an OWL2 ontology, encompassing 263 concepts and 266 asserted is-a relations. ToxOntology appeared after a merge of classification systems covering concepts within toxicology used by ACToR [47] and an implementation of the OpenTox API [42]. The merge was further refined and expanded manually by toxicology experts at the SNFA, end-users of ToxOntology. The overall design principle can be summarized as follows: it is broad enough to cover almost any aspect of interest in the field, but small enough to be used as an interactive tool in users’ daily search for toxicology information.

MeSH [6] consists of sets of terms naming descriptors in a 12-level hierarchical structure. The 2011 version of MeSH contains 26,142 descriptors. As MeSH contains many descriptors not related to the domain of toxicology, we used parts from the Diseases [C], Analytical, Diagnostic and Therapeutic Techniques and Equipment [E] and Phenomena and Processes [G] branches of MeSH. The resulting ontology contained 9,878 concepts and 15,786 asserted is-a relations. A Java program was written to parse (using the SAX parser) the XML file, filter the selected elements and create the OWL file (using Jena 2.1). We note that the MeSH hierarchy is not based on subsumption relations only, thus interpreting all structural relations as is-a relations may lead to unintended results.

Results

Aligning ToxOntology and MeSH. Our first step was to create an initial alignment between ToxOntology and MeSH. In order to create the alignment we used SAMBO (e.g., [62], [87], [58]), an ontology alignment system based on the framework described in Subsection 2.2. It implements different strategies for preprocessing, matching, combining and filtering. Due to a preference for a high-quality alignment that was as complete as possible, preprocessing to reduce the search space was excluded from the proce-

ture. We used different types of matchers—TermBasic (linguistic approach), TermWN (approach using WordNet [69]), UMLSM (approach using domain knowledge—UMLS [14]) and NaiveBayes (instance-based approach using scientific literature), and as a combination strategy we used the maximum-based strategy. We generated the similarity values for all pairs of terms. We used single threshold filtering with threshold 0.35 for the filtering strategy. These choices would lead to a high recall, although there would be many mapping suggestions to validate.

During the validation phase the domain expert classified the mapping suggestions into: equivalence mapping, is-a mapping (ToxOntology term is-a MeSH term and MeSH term is-a ToxOntology term), related terms mapping and wrong mapping. The mapping suggestions were shown to the domain expert in different steps based on the similarity values. The results are summarized in Table 5.11. The validated alignment consists of 41 equivalence mappings, 43 is-a mappings between a ToxOntology term and a MeSH term, 49 is-a mappings between a MeSH term and a ToxOntology term and 243 related terms mappings. There is also information about 1,136 wrong mappings.

The steps described above are similar to the detection and validation phases in the alignment component. The difference is in the repairing phase—in SAMBO the validated correct mapping suggestions are directly added to the final alignment, while in our framework different options for repairing them are presented to the domain experts.

Run I—Debugging using validated alignment

The debugging process started after the alignment was created. It was not considered feasible to identify defects manually. Therefore, we used the detection mechanisms of RepOSE. RepOSE computed CMIs, which were then validated by domain experts. As there were initially only 29 CMIs, we decided to repair the ontologies and their alignment independently in two ways. First, the CMIs and their justifications were given to the domain experts who manually repaired the ontologies and their alignment. Second, the repairing mechanisms of RepOSE were used. A summary of the changes in the alignment and in ToxOntology as a result of the debugging sessions are summarized in Table 5.12 column ‘original/final alignment’⁶, and Table 5.13 column ‘final’, respectively. There are also 5 missing is-a relations for MeSH. In the remainder of this subsection we describe the detection and repairing in more detail and compare the manual repairing with the repairing using RepOSE.

Detection using RepOSE. As input to RepOSE we used ToxOntology and MeSH. We additionally used the validated part of the alignment created by SAMBO, which contains the 41 equivalence mappings, the 43

⁶The final alignment contains changes from the two debugging sessions and is the one that is now used.

ToxOntology	MeSH	original/ final alignment	final alignment: manual/ RepOSE
metabolism	metabolism	$\equiv/\rightarrow$	$\rightarrow/\text{rem } \leftarrow$
photosensitisation	photosensitivity disorders	$\equiv/R$	$R/\text{rem } \leftarrow, \rightarrow$
phototoxicity	dermatitis phototoxic	$\equiv/R$	$R/\text{rem } \leftarrow, \rightarrow$
inhalation	administration inhalation	$\equiv/W$	$W/\text{rem } \leftarrow, \rightarrow$
urticaria	urticaria pigmentosa	$\leftarrow/W$	$W/\text{rem } \leftarrow$
autoimmunity	diabetes mellitus type 1	$\leftarrow/R$	$R/\text{rem } \leftarrow$
autoimmunity	hepatitis autoimmune	$\leftarrow/R$	$R/\text{rem } \leftarrow$
autoimmunity	thyroiditis autoimmune	$\leftarrow/R$	$R/\text{rem } \leftarrow$
gastrointestinal metabolism	carbohydrate metabolism	$\leftarrow/W$	$W/\text{rem } \leftarrow$
gastrointestinal metabolism	lipid metabolism	$\leftarrow/W$	$W/\text{rem } \leftarrow$
cirrhosis	fibrosis	$\equiv/R$	$R/\text{rem } \leftarrow, \rightarrow$
cirrhosis	liver cirrhosis	$\leftarrow/\equiv$	$\equiv/-$
metabolism	biotransformation	$\leftarrow/\equiv$	$\equiv/-$
metabolism	carbohydrate metabolism	$\leftarrow/W$	$W/-$
metabolism	lipid metabolism	$\leftarrow/W$	$W/-$
hepatic porphyria	porphyrias	$\equiv/\rightarrow$	$W/\text{rem } \leftarrow$
hepatic porphyria	drug induced liver injury	$\rightarrow/R$	$-/\text{rem } \rightarrow$

Table 5.12: Ontology alignment and debugging: ToxOntology-MeSH—changes in the alignment (equivalence mapping ($\equiv$), ToxOntology term is-a MeSH term ($\rightarrow$), MeSH term is-a ToxOntology term ($\leftarrow$), related terms (R), wrong mapping (W), removed (rem)).

is-a mappings between a ToxOntology term and a MeSH term and the 48 is-a mappings between a MeSH term and a ToxOntology term.⁷

RepOSE generated 12 non-redundant CMIs for ToxOntology (34 in total) of which 9 were validated by the domain experts as missing and 3 as wrong. For MeSH, RepOSE generated 32 redundant CMIs. 17 out of the 32 were non-redundant CMIs (2 out of the 17 relations represented one equivalence relation) where 5 were validated as missing and the rest as wrong.

Manual repair. The domain experts focused on the repair of ToxOntology and the alignment. Regarding the 9 missing is-a relations in ToxOntology, all were added to the ontology. Furthermore, another is-a relation, *asthma* $\rightarrow$ *respiratory toxicity*, was added, in addition to *asthma* $\rightarrow$ *hypersensitivity*, based on an analogy of this case with the already existing *urticaria* $\rightarrow$ *dermal toxicity* and added *urticaria* $\rightarrow$ *hypersensitivity*. This is summarized in Table 5.13 column ‘manual’. The domain experts also removed two asserted is-a relations (*asthma* $\rightarrow$ *immunotoxicity* and *subcutaneous absorption* $\rightarrow$ *absorption*) for reasons of redundancy. These is-a relations are valid and they are logically derivable in ToxOntology.

⁷The related term mappings cannot be used in logical derivation related to the is-a structure of the ontologies and are therefore not included in the alignment used in RepOSE.

added is-a relations	final	manual	RepOSE
absorption → physicochemical parameter	Yes	Yes	Yes
hydrolysis → metabolism	Yes	Yes	Yes
toxic epidermal necrolysis → hypersensitivity	Yes	Yes	Yes
urticaria → hypersensitivity	Yes	Yes	Yes
asthma → hypersensitivity	Yes	Yes	Yes
asthma → respiratory toxicity	Yes	Yes	No
allergic contact dermatitis → hypersensitivity	Yes	Yes	Yes
subcutaneous absorption → dermal absorption	Yes	Yes	Yes
oxidation → metabolism	Yes	Yes	Yes
oxidation → physicochemical parameter	Yes	Yes	Yes

Table 5.13: Ontology alignment and debugging: ToxOntology-MeSH—changes in the structure of ToxOntology.

The wrong is-a relations for MeSH and ToxOntology were all repaired by removing mappings in the alignment (Table 5.12 column ‘final alignment manual/RepOSE’). In 5 cases a mapping was changed from equivalence or is-a into related. In one of the cases (concerning *cirrhosis* in ToxOntology and *fibrosis* and *liver cirrhosis* in MeSH) a further study also led to the change of *cirrhosis* ← *liver cirrhosis* into *cirrhosis* ≡ *liver cirrhosis*.

The wrong is-a relations involving *metabolism* in ToxOntology invoked a deeper study of the use of this term in ToxOntology and in MeSH. The domain experts concluded that the ToxOntology term *metabolism* is equivalent to the MeSH term *biotransformation* and a sub-concept of the MeSH term *metabolism*. This observation led to a repair of the mappings related to *metabolism*.

Furthermore, some mappings were changed from an equivalence or is-a mapping to a wrong mapping.⁸ In these cases (e.g., between *urticaria* in ToxOntology and *urticaria pigmentosa* in MeSH) the terms were syntactically similar and were initially validated wrongly during the alignment phase.

Repairing using RepOSE. For the 3 wrong is-a relations for ToxOntology and the 12 wrong is-a relations for MeSH, the justifications were shown to the domain experts. The justifications for a wrong is-a relation contained at least 2 mappings and 0 or 1 is-a relations in the other ontology. In each of these cases the justification contained at least one mapping that the domain expert validated as wrong or related and the wrong is-a relations were repaired by removing these mappings (see Table 5.12 column ‘final alignment manual/RepOSE’, except last row). In some cases repairing one wrong is-a relation also repaired others (e.g., removing mapping *hepatic porphyria* ← *porphyrias* repairs two wrong is-a relations in MeSH: *porphyrias* → *porphyrias hepatic* and *porphyrias* → *drug induced liver injury*).

For the 9 missing is-a relations in ToxOntology and the 5 missing is-a

⁸So the domain experts changed their original validation based on the reasoning support provided by RepOSE.

relations in MeSH, possible repairing actions (using Source and Target sets) were generated. For most of these missing is-a relations the Source and Target sets were small, although for some there were too many elements in the set to provide for good visualization. For all these missing is-a relations, repairing them consisted of adding the missing is-a relations themselves (Table 5.13 column ‘RepOSE’). In all but three cases this is what RepOSE recommended based on external knowledge from WordNet and UMLS. In 3 cases the system recommended adding additional is-a relations, that were not considered correct by the domain experts (and thus wrong or based on the external domain knowledge taking a different view of the domain).

After this repairing, we detected one new CMI in MeSH. This was validated as a wrong is-a relation and resulted in the removal of one more mapping (see Table 5.12 column ‘final alignment manual/RepOSE’ last row).

Run II—Debugging using non-validated alignment

In Run I the validated alignment was used as input. As a domain expert validated the mappings, they could be considered of high quality, although we showed that defects in the mappings were detected. In this subsection we perform an experiment with a non-validated alignment; we use the 41 mapping suggestions with a similarity value higher than or equal to 0.8 and use them initially as equivalence mappings.⁹

Using RepOSE (in 2 iterations) 16 non-redundant CMIs (27 in total), were computed for ToxOntology of which 6 were also computed in the debugging session in Run I. For MeSH 6 non-redundant CMIs (10 in total) were computed, of which 2 were also computed earlier. As expected, the newly computed CMIs were all validated as wrong is-a relations and their computation was a result of wrong mappings. During the repairing 5 of the 7 wrong mappings were removed, and 2 initial mappings were changed into is-a mappings.

Discussion

As the set of CMIs in Run I was relatively small, it was possible for domain experts to perform a manual repair. They could focus on the pieces of ToxOntology that were related to the missing and wrong is-a relations. This allowed us to compare results of manual repair with those of repairs done using RepOSE.

Regarding the changes in the alignment, for 11 term pairs the mapping was removed or changed in both approaches. For 2 term pairs the manual approach changed an is-a relation into an equivalence and for 2 other term pairs an is-a relation was changed into a wrong relation. These changes were not logically derivable and could not be found by RepOSE. For 3 of

⁹From the validation we know that these actually contain 29 equivalence mappings, 2 is-a mappings between a ToxOntology term and a MeSH term, 2 is-a mappings between a MeSH term and a ToxOntology term, 1 related term mapping and 7 wrong mappings.

these term pairs the change came after the domain experts realized (using the justifications of the CMIs) that *metabolism* in MeSH has a different meaning than *metabolism* in ToxOntology. For 1 term pair (second to last row in Table 5.12) the equivalence mapping was changed into wrong by the domain experts, while using RepOSE it was changed into an is-a relation. In the final alignment the RepOSE result was used. Additionally, using RepOSE an additional wrong mapping was detected and repaired through a second round of detection. It was not found in the manual approach.

Regarding the addition of is-a relations to ToxOntology, the domain experts added one more is-a relation in the manual approach than in the approach using RepOSE. It could not be logically derived that *asthma* $\rightarrow$ *respiratory toxicity* was missing, but it was added by the domain experts in connection with the repairing of another missing is-a relation.

In some cases, when using RepOSE, the justification for a missing is-a relation was removed after a wrong is-a relation was repaired by removing a mapping. For instance, after removing *metabolism* (*ToxOntology*) $\leftarrow$ *metabolism* (*MeSH*), there was no more justification for the missing is-a relation *hydrolysis* $\rightarrow$ *metabolism*. However, an advantage of RepOSE is that once a relation is validated as missing, RepOSE requires that it be repaired and thus this knowledge will be added even if there is no justification.

Another advantage of RepOSE is that, for repairing a wrong is-a relation, it allows the removal of multiple is-a relations and mappings in the justification, even though it may be sufficient to remove one. This was used, for instance, in the repair of the wrong is-a relation *phototoxicity* $\rightarrow$ *photosensitisation* in ToxOntology where *photosensitisation* $\equiv$ *photosensitivity disorders* and *phototoxicity* $\equiv$ *dermatitis phototoxic* were removed. Furthermore, the repairing of one defect can lead to other defects being repaired. For instance, the removal of these two mappings also repaired the wrong is-a relation *photosensitivity disorders* $\rightarrow$ *dermatitis phototoxic* in MeSH. In general, RepOSE facilitates the computation and understanding of the consequences of repairing actions.

Comparing Run I and II we confirm that RepOSE can be helpful in the validation of non-validated alignments—a domain expert will be able to detect and remove wrong mappings that lead to the logical derivation of wrong is-a relations, but wrong mappings that do not lead to logical derivation of wrong is-a relations may not be found.

5.3 Discussion

The discussion here is carried out mainly in two directions—highlighting the benefits from our integrated ontology debugging and alignment approach on one side and showing that a dedicated system to support ontology alignment and debugging is extremely necessary on the other side.

All three experiments in Subsection 5.2 clearly demonstrate the advantages of the integration of ontology alignment and debugging. Our inte-

grated approach improves the quality of the alignments by providing different alternatives for repairing them. It also leads to discovering more possible modelling defects in ontologies and alignments by extending the sets with the alignments. We note that the experiments presented in this chapter do not completely explore the benefits from the integration. Since structure-based matchers, preprocessing and filtering strategies were not employed in the alignment component we were not able to explore the benefits from the repaired structure of the ontologies and alignments over the alignment process.

Our integrated approach is universal and can be applied to any two ontologies. To detect modelling defects in ontologies the network is an important source of domain knowledge, while defects in alignments can be detected by both alignment algorithms and intrinsic knowledge. The detected defects can also be repaired by different, sometimes more informative, repairing actions. The number of detected defects and their repairing actions depend on the correctness and completeness of the structure of the input ontologies and alignments. For instance, in the ToxOntology-MeSH use case only mappings were removed to repair wrong is-a relations. This indicates that the ontology developers modeled the is-a structure decently. This kind of repair is not, however, a consistent outcome. For instance, in the experiment outlined in Subsection 5.1.1 and [56] involving debugging the two ontologies (AMA and NCI-A) and their alignment from the Anatomy track in OAEI 2010, 14 is-a relations were removed from AMA and 11 from NCI-A, as well as 5 mappings. Furthermore, in ToxOntology all missing is-a relations were repaired by adding them. In the experiment in Subsection 5.1.1 in 27 cases in AMA and 11 cases in NCI-A a missing is-a relation was repaired using a more informative repairing action, thereby adding new knowledge that was not logically derivable from the ontologies and their alignment. More informative repairing actions are also used in the two experiments in Subsections 5.2.1 and 5.2.2.

Generally, detecting defects in ontologies without the support of a dedicated system is cumbersome and unreliable. In all cases outlined in this chapter RepOSE clearly provided necessary support. Additionally, visualization of the justifications for possible defects was very helpful to have at hand, as was a graphical display of the possible defects within their contexts in the ontologies being addressed. During the entire debugging and alignment processes, and not only through the detection phase, the system provides proper visualization, thus assisting the user in understanding the defects and the available options for repairing actions. During the repairing phase the system generates different repairing actions for the defects, thus providing an opportunity to add more knowledge to the ontologies and alignments. Moreover, RepOSE stored information about all changes made and their consequences as well as the remaining defects needing amendment. It prevents contradictory information from being added to the ontology network as well.

An identified constraint of RepOSE pertains to the fact that adding and removing is-a relations and mappings not appearing in the computations in RepOSE can be a demanding undertaking. Currently, these changes need to be conducted in the ontology files, but it would be useful to allow a user to do this via the system. For instance, in the ToxOntology-MeSH use case, it would have been useful to add *asthma* $\rightarrow$ *respiratory toxicity* via RepOSE.

Although the system has good responsiveness, the number of the user-system interactions for large ontologies and alignments, such as validations and repairings, is very high. Thus, proper approaches to lower this number are desirable. For instance, it was observed that in many cases there is just one possible repairing action for a defect. Thus, instead of showing the defect and the repairing action to the user, the system could execute it automatically.

Our system provides contextual visualization and most of the time the display is easily readable and not cluttered with objects. In some cases, however, there are too many objects to allow good visualization. We have implemented a number of techniques to manage such cases, for instance, visualizing the defects and their repairing actions in groups, zoom in/out, and the option to open the current display in a separate larger window that can be resized to fit the screen dimensions. However, in order to further facilitate better understanding of the presented information, other grouping heuristics and visualization techniques should be explored.

Chapter 6

Related work

This chapter discusses related work in the areas of ontology alignment and debugging and compares other approaches with our approach. At the end an overview is given of the available approaches that can be considered to some extent related to the integration of ontology alignment and debugging.

6.1 Ontology debugging

This section focuses on two of the three types of defects identified in [48]—semantic and modelling defects in ontologies and ontology networks. Syntactic defects are not considered since they are not signs of misinterpretation of a domain but are rather caused by mistyping. They can be found and resolved using parsers.

6.1.1 Debugging modelling defects

The approach for debugging modelling defects presented in this thesis is an extension of [61], [60] and [59]. The problem of repairing missing is-a relations in a single taxonomy was initially discussed in [61] with the assumption that the is-a structure of the taxonomy is correct. The authors in [61] present two algorithms for computing repairing actions for missing is-a relations. The first, which is similar to the one presented in this thesis, only computes solutions for a single missing is-a relation. The second extends it by taking into account the influence of the repairing actions of other missing is-a relations during the computation of the Source and Target sets. The results of the experimental evaluation of the extended algorithm show that such influences are not negligible and in some cases the repairing actions for different missing is-a relations influence each other. The work in [59] is continuation of [61] where the authors consider wrong is-a relations in the structure of the taxonomies. In contrast to [61] where the focus is on a single taxonomy, the context in [59] is a taxonomy network with the assump-

tion that the mappings in the network are correct. Missing is-a relations in the context of a taxonomy network with correct mappings are discussed in [60]. This thesis takes the approach even further, considering wrong and missing subsumption and equivalence mappings in a taxonomy network and employing ontology alignment algorithms as an additional method for detecting missing mappings.

The work presented in [61], [60], [59] and this thesis, only considers missing and wrong is-a relations in taxonomies, which are a simple kind of ontologies from a knowledge representation point of view. The work in [55] extends the scope to deal with repairing missing is-a relations in the structure of $\mathcal{ALC}$ ontologies, which can be represented using acyclic terminologies. The problem of repairing missing is-a relations is formulated as a generalized version of the TBox abduction problem. In [65] the authors define properties for the ontologies, the set of missing is-a relations, the domain expert and preferences for the solutions for the problem in [55]. Finally, in [93], complexity results for the existence, relevance and necessity decision problems for the generalized TBox abduction problem for $\mathcal{EL}^{++}$ ontologies are presented.

Debugging in general has two phases—discovering defects and resolving them. Often a validation phase performed prior to the repairing phase is also presented. The detection of modelling defects, especially missing structure, is not trivial since it requires, among other competencies, domain knowledge. Manual inspection, apart from being error-prone, is, of course, possible, however, tedious and even infeasible for very large ontologies. In our approach we utilize the knowledge intrinsic to an ontology network and, additionally, ontology alignment algorithms for detecting missing mappings. Other approaches, such as those in the area of ontology learning are available as well. They allow automatic creation of ontologies from a large set of texts.

An insight into the state of the art in the ontology learning area is provided in [24]. It contains three parts focusing on methods, evaluation and learning methodology. This field takes advantage of methods developed in already established areas, such as knowledge acquisition and natural language processing. The methods presented within this book can be seen as supporting methods (to those presented in this thesis) for detection of modelling defects.

The work in [41] is of particular interest since it deals with discovering missing is-a relations from large texts corpora. The authors describe a method for automatic acquisition of hyponyms, which are lexical relations of the kind *something is a (kind-of) something*. Beneficial features of hyponyms include easy recognition, high frequency of occurrence and relevance across domains. Hyponyms can also be employed for the purpose of identifying instances of concepts, as in the example *hyponym(author, Shakespeare)*. An important side effect is discovering of pairs, such as *(broken bone, injury)*, which are not common dictionary entries. Ontology learning approaches can be used as detection methods on their own or as additional external infor-

mation for suggesting recommendations during the processes of detection, validation and repairing. Querying external sources, such as WordNet, to determine existing relations between concepts can be used in both cases as well.

Based on their experience, the authors of [19] propose a set of ten requirements that should be fulfilled as a basic step for a valid and reusable reference alignment. They can be seen as patterns for debugging. The first half of the requirements covers mainly technical and versioning issues in the ontologies and alignments. The second part is focused on the content and completeness of the alignments taking into account subsumption and equivalence relations from structural and linguistic points of view. Closest to the approach presented in this thesis are two requirements that deal with structural completeness and resemble part of our detection phase. According to one of them, for instance, if there are equivalence mappings between a particular concept from one of the ontologies and more than one concept in the other, the concepts in the second ontology should be connected through equivalence relations. In the other requirement, a given equivalence mapping is checked to see if the subclasses of one of the concepts are connected through subsumption mappings with the superclasses of the other and vice versa. Another pair of requirements can be seen as alignment algorithms where (part of) the labels (or local names) are compared. These requirements can help with the identification of missing and wrong mappings and also missing and wrong subsumption relations in the ontologies. The approach was tested on the OAEI Anatomy 2010 dataset and the results incorporated in the dataset used in the OAEI Anatomy 2011. We have compared the results of our approach (the experiment in Subsection 5.1.1) with the results in [19] regarding wrong mappings. Of the 25 wrong mappings identified by [19], our approach can identify 21 wrong mappings using the full reference alignment. Our approach also identified 8 additional wrong mappings.

Another approach for detecting defects, [28], assumes that the ontology specification does not change over time and explores the modifications in it during its evolution on an axiom level. Having different versions of an ontology, the authors propose to compare them in order to identify suspicious editing patterns, such as consecutive additions and removals of the same axioms in the different versions. This approach can be employed to detect both semantic and modelling defects, however, its limitation is that several versions of the ontology in question should be available.

The closest approach to the one in this thesis regarding detecting missing is-a relations is discussed in [17] where the authors describe a method for identifying nonalignments (essentially missing subsumptions) between Open Biomedical Ontologies (OBOs). The nonalignments discussed in [17] can be seen as the CMIs and CMMs in this thesis. The nonalignments are detected based on properties while in our work we only use subsumption and equivalence relations. Similarly to the framework in this thesis, three phases can

be distinguished in [17]—a phase for detecting nonalignments, an examination which resembles our validation phase and a repairing phase. During the examination, the nonalignments that should be aligned are separated (they are called discrepancies). The authors suggest two approaches for rectifying the discrepancies—either adding the missing subsumptions or removing the existing subsumptions. The nonalignments that are not discrepancies are indicators of inconsistencies in the ontologies. They are resolved by upward propagation of the corresponding concepts to the superclass levels. When a discrepancy is repaired by adding the missing subsumption, other possibilities for repairing which would make it derivable are not considered. This approach does not consider nonalignments based on incorrect information in the ontologies or alignments. In both approaches the search space during the detection phase is reduced—in our approach we only employ mapped concepts and in [17] only pairs of assertions with an already existing subsumption relation are checked.

Borrowing the concept of design patterns from software engineering, the authors of [30] apply patterns and antipatterns for the purpose of debugging semantic and modelling defects. The (anti)patterns are based on description logics constructions that do not necessarily exist in taxonomies, for instance, existential and universal quantifiers etc. The patterns do not aim to change the semantics of the ontologies, they are guidelines for restructuring the ontologies in order to make them more understandable for the developers. The antipatterns represent common errors in the ontologies developed by domain experts originating in misuse and misunderstanding of logical constructions. The authors also suggest actions for resolving the antipatterns that go beyond simply removing axioms or exchanging a class with one of its superclasses. In order to avoid changes in the intended meaning of the ontologies during debugging, the actions for resolving the antipatterns should be validated by a domain expert.

6.1.2 Debugging semantic defects

More work is available in the field of debugging semantic defects. The detection of the semantic defects is usually done by a reasoner and the focus is on computing diagnoses and repairing actions.

One of the works that does not use a reasoner for detection of semantic defects is [77]. It deals with the detection of one of the antipatterns in [30], *Onlyness Is Loneliness*, in OWL ontologies. The authors propose an approach where candidates of the antipattern are identified without a reasoner, since in large ontologies with many complex axioms and defects the reasoners do not scale well. The detection process goes through the following two steps—applying transformation rules in a predetermined order for the purpose of simulating inference and to avoid use of a reasoner. These rules do not remove original axioms, they only add new ones. The second step is to execute one or more SPARQL queries in their ontology patterns detec-

tion tool—PatOMat. The query returns the candidates of the Onlyness Is Loneliness antipattern. The authors suggest that the same approach with suitable transformation rules can also be applied for the other antipatterns in [30].

Computing diagnoses and repairing actions is the focus in [80] and [81] where a method for repairing the axioms in an incoherent TBox is proposed. It identifies a minimal set of axioms that should be removed from the TBox in order to make it coherent. The method creates minimal subsets of the TBox, called MUPs (Minimal Unsatisfiability-Preserving sub-TBox), where the unsatisfiability of each unsatisfiable concept is preserved. Then, based on the MUPs, MIPSs (Minimal Incoherence-Preserving sub-TBox) are created—they are the smallest subsets of the TBox that make it incoherent. A set of axioms occurring in several MIPSs is called a core. More occurrences of a core leads to higher probability that it is the cause of the incoherency and it should be removed from the TBox. This is similar to our single relation heuristic.

Another popular work in the field of debugging semantic defects is [51] where the authors focus on debugging unsatisfiable concepts. For the purposes of ontology debugging, two techniques from the software testing area are utilized—the glass box and the black box. The authors have developed and integrated methods and algorithms for them in their ontology editor Swoop. The glass box approach relies on extra information from the reasoner, extended with additional data structures, to identify the causes for unsatisfiable concepts. Two forms of this technique were presented—presentation of the root cause of the contradiction (clash) and computation of relevant sets with axioms responsible for the clash (sets of support). The sets of support are computed for each unsatisfiable concept, and when minimally determined they coincide with the Minimal Unsatisfiability-Preserving sub-TBox (MUP) presented in [80] and [81]. A common set of support represents the repairing actions for all unsatisfiable concepts, however, it is not easy to obtain such a set with the glass box approach since it may not scale for many unsatisfiable concepts. To resolve this problem the authors explore the black box technique where the reasoner is only used as an oracle for answering queries in order to determine dependencies between concepts. The unsatisfiable concepts are divided into root concepts (with unsatisfiable concept definitions) and derived concepts (which depend on the unsatisfiability of other concepts).

The authors of [51] continued their work regarding explaining the causes for unsatisfiable concepts, and developing strategies to rectify them in [50]. One of the focuses in [50] is providing precise explanations of the causes for unsatisfiability by identifying the smallest parts of axioms responsible for them, thus aiding the users' understanding. Similar to the idea of arity in [80] and to the single relation heuristic in our work, the authors propose a simple ranking criterion based on the frequency of occurrence of axioms across the MUPs. Other ranking strategies based on the impact and fre-

quency of usage of the axioms across the ontology, user-driven test cases and provenance information were presented as well. The solutions for the unsatisfiable concepts are generated using a modified version of Reiter's hitting set tree algorithm [76] extended to take into account the ranking of the axioms. The users can choose between three granularity levels during the repairing—to repair a single unsatisfiable concept, to repair all root or all unsatisfiable concepts. Similarly, our tool has two modes for repairing wrong is-a relations and mappings—repairing them one by one or repairing all of them at once (in a single taxonomy). The authors propose methods for rewriting axioms, instead of removing them, for known modeling pitfalls.

Providing an explanation for a given entailment is the key to understand why a concept is unsatisfiable and how it can be repaired. The authors of [51] apply their glass box and black box techniques to find all justifications for an entailment in [49]. Their definition of a justification is similar to the one in our work. They have developed two methods for finding a single justification based on each of the two techniques. Using the black box technique, the axioms from the initial ontology are copied one by one to a new ontology until the given unsatisfiable concept becomes unsatisfiable. Then the new ontology is pruned in order to exclude those axioms that are not part of the justification. The other method is based on an extension of the glass box technique from [51] and applies the same pruning method at the end. Having a single justification and benefiting from the duality of the hitting set trees from [76], the algorithm in [49] computes all justifications—instead of computing minimal hitting sets from the tree, the algorithm uses a single justification as a root of the tree and at each step creates new branches reusing the methods for computing a single justification. Known hitting set trees optimization techniques are utilized to reduce the number of calls to the algorithm that computes a single justification.

Another work that addresses understanding of entailments is [73], in which the authors discuss the steps to generate easily understandable explanations of OWL inferences in English. They have developed a probabilistic model for estimating the understandability of a justification composed of multiple inferences based on a measure of the understandability of a single inference. The measure of understandability of a single inference is called the Facility Index and its development is described in [72]. The Facility Index is the result of an empirical study of a set of deduction rules collected from a corpus of around 500 ontologies. For every entailment with multiple inferences a proof tree is built, where the entailment is the root of the tree and the deduction rules are its leaves. Then the understandability of the entailment is estimated by multiplying the Facility Indexes for the deduction rules in the tree. For entailments with multiple proof trees this method can be used to rank them according to their understandability.

Understanding justifications of multiple entailments is the focus in [18]. The authors have observed that often sets of justifications are similar, containing axioms with similar, sometimes even identical, structure, which differ

only in class names, properties and relations. The length of the justifications also varies. In these cases the same type of reasoning is required from a human user. This observation can be used to help the user in the process of understanding the justifications and thus significantly reduce the number of justifications to grasp. The notion of structural similarity was called justification isomorphism and appeared in the authors' previous work. In this paper they define three types of isomorphism—strict isomorphism (same number and type of axioms in the two justifications), subexpression-isomorphism (different concept expressions requiring the same reasoning but the same number of axioms) and lemma-isomorphism (the same type but different number of axioms). An experiment, performed with the ontologies from NCBO BioPortal, shows that the isomorphism can reduce the number of justifications that must be understood by 90% and that most of the justifications are strictly isomorphic, i.e., they use the same number and type of axioms.

In [84] the authors generalize the ontology debugging problem, introducing weighted ontologies where weights are assigned to the axioms. The problem is transformed into an optimization problem for computing subontologies with the maximum sum of weights. The axioms that are not part of the maximum sum are then removed. The approach is promising for very large ontologies with a large number of inconsistencies, however, it is not clear how the weights will be assigned.

Semantic defects in ontology networks are also an area of interest. However, all of these approaches consider the ontologies correct and only debug the alignments. By comparison, our approach considers defects in both the ontologies and alignments. In [92] the authors detect four patterns of frequently occurring defects in mappings and propose repairing methods that are either automatic or user-driven. They focus on equivalence and subsumption mappings and define four types of defects: redundant mappings, imprecise mappings, inconsistent mappings and abnormal mappings.

The authors of [68] propose a completely automatic method for debugging ontology mappings, detecting and repairing inconsistencies caused by erroneous mappings. The method deals with equivalence and subsumption mappings and relies on the assumption that the mappings model semantic relationships without causing inconsistencies. Distributed description logics is used to formalize the problem—the domain knowledge is represented by a distributed ontology (similar to the induced ontology in this thesis) and the mappings are represented as a set of bridge rules. For diagnosis they rely on the classic Reiter's definition from [76]. An inconsistency is resolved by removing a bridge rule, thus the method for selection of the rule is important. Instead of applying the classical hitting set tree algorithm the authors propose a simple heuristic that selects the rule for removal by its confidence value or the WordNet distance between the concepts in it if the confidence value is not available. This resembles the rank approach in [50]. At the end the authors discuss the problem of incorrect mappings that do not cause

inconsistencies and propose the notion of instable mappings to deal with it. They suggest that a mapping which makes a previously non existing subsumption relation in a single ontology derivable may indicate inconsistency. The idea of the instable mappings is quite similar to our approach for detecting CMIs, however, we interpret it differently—as a possible missing is-a relation.

In [75] a conflict-based operator for mapping revision is proposed that considers subsumption and equivalence mappings. The operator is based on the notion of “conflict sets”, which are the minimal sets of mappings causing logical contradictions between the ontologies. It is defined by two postulates adapted from the belief-based revision theory.

The authors of [74] discuss the relationships between inconsistency and incoherency in the ontologies and categorize the reasons for the inconsistency in three groups—inconsistency due to terminology axioms, inconsistency due to assertional axioms and inconsistency caused by both terminology and assertional axioms. They propose a general integrated approach for dealing with inconsistency and incoherency in ontology evolution and give several suggestions for how the different phases of the approach can be instantiated by revisiting several concrete approaches.

The authors of [43] implement their algorithms in the RaDON system. They propose an efficient relevance-directed algorithm for computing MUPs in subontologies adapted from [49] and based on Reiter’s hitting set trees [76]. The user can choose to compute one, all or some MUPs and hitting sets for an unsatisfiable concept. An element of the system’s functionality is reasoning in an inconsistent setting based on four-valued semantics.

In [82], reasoning with multiple ontologies connected through directional mappings is presented. Distributed description logics is used to formalize the knowledge in the ontologies and their alignments (the alignments are represented with sets of bridge rules; in this work only subsumption and equivalence mappings are considered). In this setting the knowledge propagation only occurs in one direction (directionality property) and inconsistency in one of the ontologies would not lead to inconsistency in the whole distributed ontology (localized inconsistency property).

6.2 Ontology alignment

After years of substantial research effort in the field of ontology alignment, the authors of [83] seek new promising directions for its future development. After sharing an observation that the field is slowing down, they give an overview of the state of the art and identify eight challenges for the alignment community. These challenges are united around the issue of scalability, both in terms of matchers strategies and evaluation, as well as user involvement and supporting infrastructure.

With two contradictory tendencies in place—increasing the size of the

matching task (demanding scalability techniques) and broadening the range of the applications performing it (including devices with limited resources)—the *efficiency of matching techniques* in terms of both computational time and memory consumption is becoming more and more important. Possible solutions to this problem include parallelization, distribution, modularization, etc. The increasing size of the alignment task also demands *large scale matching evaluation*, which is not possible without automatic methods for developing high-quality reference alignments. In the context of evaluation more accurate (in addition to precision and recall) as well as application specific evaluation measures are required. There are different matchers available but none of them is considerably better than the others for a specific application. As a result a combination of matchers is usually used in order to obtain more reliable results. Those combinations could be tailored to application areas or datasets’ features or both, which is why strategies for *matcher selection, combination and tuning* are highly desirable. Some matchers utilize *background knowledge* during the alignment process, for instance, curated resources such as WordNet and UMLS. In the future other resources, including resources that are not curated such as linked open data, can be utilized as well.

The scalability problem should be addressed in the area of user interactions as well. Given an alignment the end user, who is not necessarily an ontology alignment expert, should be able to understand it and how it was obtained in order to better utilize and edit it. Analogously to the justifications in the area of ontology debugging, easily comprehensible yet clear and precise *explanations of matching results* are needed. *User involvement* is crucial for the success of each task and ontology alignment is not an exception. Increasing the number of tools supporting user interface and various user interactions will foster user engagement in the process. Higher quality alignments will be the product of better user interfaces with good scalability features, rather than more accurate matchers [22]. The user involvement in the process can be encouraged through *social and collaborative matching* as well. The manual curation of large alignments is a demanding task for a single user. It can be relaxed by involving several users who can discuss together problematic mappings. Such collaborative effort will demand metadata standards and proper *alignment management* frameworks providing *infrastructure and support* during all phases of the process—storage, version control, etc.

Comparing our system with these challenges, we have already made initial steps towards addressing three of them. Many matchers have been proposed¹, and most systems use similar combination and filtering strategies as in this thesis². However, there are still not many alignment systems that explore background knowledge. Since the alignment algorithms in our system are reused from the SAMBO system [62], we have already been addressing

¹e.g., many papers at <http://ontologymatching.org/>

²For an overview we refer to [83].

the challenge of matching with background knowledge. We employ external, curated resources with well-known structure and reliability—WordNet and UMLS. Our system is one of the few supporting user validation of the mappings, the others being SAMBO [62], COGZ [34] for PROMPT, and COMA++ [31]. RepOSE also has a unique feature as it provides different options for repairing the missing mappings, rather than just directly adding the mapping suggestions. The whole repairing phase is supported by a user interface. Moreover it provides debugging of the alignment during the process of its development. These features can be considered steps in the direction of user involvement and explanation of the matching results. It was mentioned in [83] that very few systems support mappings other than equivalence—RepOSE is among them, it supports subsumption in addition to the equivalence mappings.

6.3 Integration of ontology alignment and ontology debugging

There are a few systems that could be considered to integrate ontology alignment and debugging to some extent. They are usually focused on ontology alignment and perform ontology debugging (considering semantic defects) only as a means of providing coherent alignments. In contrast our system, RepOSE, is an integrated ontology alignment and debugging system. It can be used as such or as a separate alignment or debugging system. Moreover RepOSE allows debugging of both the structure of the ontologies as well as the alignments, while most of the other systems assume that the ontologies are correct and only debug the alignments. Generally, debugging of modelling defects, such as missing is-a structure, requires domain knowledge. A unique feature of our system is that it detects missing is-a relations without external domain knowledge.

One of the first works to make a connection between ontology alignment and debugging is [23]. Its authors compare two approaches for aligning earlier versions of AMA and NCI Thesaurus—manual and lexical. The manual and lexical alignments are used to create a final alignment and a structural validation was performed in order to remove pairs of concepts without structural similarity from it. The structural validation was performed employing the pairs of concepts with lexical similarity, called anchors. The relations in which the anchors participate are examined and the existence of at least one common hierarchical relation among the concepts in the anchors across the ontologies is taken as positive structural evidence. This approach can be used for detection of CMIs.

Based on their experience, the authors of [46] identify several requirements for an ontology alignment system (partially covering different aspects of the challenges from [83]). According to their view such requirements are interactivity (user interactions during the alignment process instead of post

curation of the alignments), scalability (both in terms of the size of the ontologies, but also in terms of user interactions) and reasoning-based error diagnosis (detecting and repairing unsatisfiable concepts). They present LogMap 2, an ontology alignment system that implements scalable reasoning and diagnosis algorithms. The ontologies and mappings are encoded in Horn propositional representation, which allows scalable detection and repairing of unsatisfiable concepts performed on modules extracted from the ontologies. More details about the detection and repairing of logical contradictions can be found in LogMap [44] implementing Dowling-Gallier algorithm [32] for Horn propositional satisfiability. Comparing RepOSE and LogMap 2, both deal with subsumption mappings. However, LogMap 2 only simulates user interactions while RepOSE has a fully functional user interface.

Evaluation of the coherence of the alignments generated by the systems in the Ontology Alignment Evaluation Initiative has recently started—in the 2011 campaign in the Anatomy track and in the 2011.5 campaign in the Large Biomedical Ontologies track. It shows that in most cases the generated alignments are incoherent. The authors of [79] have found that the incoherences in the alignments generated by the systems in the Large Biomedical Ontologies track are most often caused by disjointness restrictions between concepts. They propose a method for detecting incoherence (caused only by disjointness restrictions) in ontologies employing ontology modularization techniques. Their method creates core fragments (modules) that contain concepts and relations from the two ontologies and their alignment needed for resolving all conflicts caused by the disjointness restrictions. They have also developed a repairing method and a heuristic (similar to our single relation heuristic) that minimizes incoherence in the final alignment and the number of removed mappings from the initial alignment. Their system AML is among the best performing systems in terms of runtime in the Anatomy track in the OAEI 2013 [37]. In the 2013 campaign, AML-bk, an extension of AML that uses background knowledge, achieved the best result in the Anatomy track in terms of f-measure.

Chapter 7

Conclusions and Future Work

This chapter concludes the thesis and presents several possible directions for long-term future work and improvements of the work presented so far.

7.1 Conclusions

The vision of the Semantic Web is coming into reality and ontologies play a key role in it. They model the world around us by defining the semantics of entities and their relationships. Ontologies provide mutual understanding of a domain and facilitate applications such as agent communication and data integration. Now, in the era of Big Data, the demand for data integration will grow even stronger and more complicated. Other areas take advantage of ontologies as well. Many ontologies in various domains have already been developed and more will be developed in the near future. Often several overlapping ontologies are employed in order to fulfill a specific task, for instance, integration of several data sources annotated with different ontologies. Thus, an understanding of the relationships between the concepts in the different ontologies is essential.

The development of ontologies and alignments is not a trivial task, for various reasons—domain experts are not proficient in knowledge representation, the intended and unintended entailments become more difficult to follow with the increasing size and complexity of the ontologies, concept discrepancies, etc. As a consequence defects in the structure of the ontologies and their alignments may be introduced.

In this context debugging of ontologies and their alignments is a key step towards obtaining highly reliable results from a wide range of applications employing ontologies. Debugging aims at detecting and repairing different types of defects. Modelling defects are some of the most complex to detect

and resolve since they require domain knowledge. While for syntax and semantic defects there is tool support, such support, with few exceptions, is missing for modelling defects.

The manual detection of modelling defects, if it is possible at all, is infeasible, especially in ontologies with many concepts and complex relations. Thus, automatic detection methods for modelling defects are highly desirable. Once detected, the defects should be repaired. A modelling defect such as wrong structure should be repaired by removing it or modifying it. Regarding missing structure, the obvious solution is to directly add the missing information. However, it was observed that other repairing actions exist that add more knowledge to the ontologies and alignments. Since domain experts might prefer actions of this type, methods are required that can provide nontrivial repairing actions.

7.1.1 Debugging of ontologies and alignments

The focus of this work is on taxonomies since they are the most widely used kind of ontologies and, in general, the structure of ontologies is often based on subsumption relations between their concepts. We also considered modelling defects, such as missing and wrong is-a relations in taxonomies and mappings in alignments, which require domain knowledge to detect and repair. The taxonomies themselves, connected through alignments in a taxonomy network, can provide the necessary domain knowledge. We have shown algorithms for debugging modelling defects in alignments employing knowledge intrinsic to the network.

However, alignments are not always available, and in some cases they do not exist, thus the network cannot be created. In order to create alignments and, consequently, a network, we utilize ontology alignment algorithms.

We extended the framework in [67] with algorithms for debugging modelling defects in alignments and integrated ontology alignment and ontology debugging. The framework has two components—a debugging component and an alignment component. In each component, the workflow consists of phases for detection, validation and repairing of modelling defects in the ontologies and the corresponding alignments.

Using only the debugging component we were able to detect a significant number of wrong and missing is-a relations in the ontologies from the Anatomy track in the OAEI 2010 (details in Subsection 5.1.1).

7.1.2 Benefits from the integration of ontology alignment and ontology debugging

The integration of ontology alignment and debugging led to the exploration of their interactions. Ontology alignment can be seen as a special kind of debugging of missing mappings, and ontology debugging using the knowledge

intrinsic to the network can be seen as a special, structure-based alignment algorithm.

Exploring the integration of ontology alignment and debugging we found that it provides advantages for both and raises the quality of the ontologies and alignments. Since our debugging approach is based on the knowledge intrinsic to an ontology network, the existence of such network is required. Using the ontology alignment algorithms we are able to create alignments and consequently a network between any number of ontologies. Even if a network already exists the alignment algorithms can be applied to extend the set with available alignments and thus provide more information for debugging of modelling defects (as shown in Run III in Subsection 5.2.1). These observations are relevant to our debugging approach, which relies heavily on the knowledge intrinsic to the network. However, ontology alignment algorithms, in general, can be applied in cases when domain knowledge is required.

As was pointed out, the repairing phase in our debugging approach provides different options for repairing modelling defects in addition to directly adding the missing structure and removing the wrong structure. The alignment component in our framework follows the general alignment framework, as described in Subsection 2.2, and extends it with a repairing phase. Furthermore, the debugging repairs the structure of the ontologies and alignments and provides higher quality input for the structure-based alignment algorithms, preprocessing and filtering strategies.

7.1.3 Implemented system

We extended the system in [67], implementing algorithms for detecting and repairing modelling defects in alignments and integrating ontology alignment and ontology debugging. We also performed several experiments and analyzed their results.

During the experiments it was observed that our implemented system clearly provided the necessary support through the phases of detection, validation and repairing. The possible defects and their repairing actions are visualized in their context during the validation and repairing phases, helping the user to understand them and their causes and providing repair options that add as much new knowledge as possible to the network. The system had good responsiveness to user actions at any given moment during the experiments. It also keeps track of the whole process—stores the defects, computes the consequences of the repairing actions and prevents the usage of contradictory repairing actions.

7.2 Future work

In this subsection we outline our ideas for improving the system and lay out long-term future work.

7.2.1 Extending the system

Reflecting on the experiments and their results, several directions for improvements to the system were identified. They are focused on extending the functionality of the system and reducing user-system interactions.

As was noted earlier, our repairing approach does not depend on the origin of the defects, i.e., whether they are detected by the system or provided by external sources. Thus, supporting an external input will allow our repairing methods to resolve defects detected by methods other than those presented in this thesis. During the experiments, it was noticed that adding/removing is-a relations or mappings that do not appear as defects or their justifications is not possible. This functionality could be helpful in cases such as the one described in Subsection 5.2.3, and it could be achieved by integrating a simple ontology editor.

Currently, our method for detecting modelling defects by employing the knowledge intrinsic to the network considers the subsumption relations between the concepts in one or more taxonomies. One immediate step is to extend the set with relations (for instance, *is-located-in*, *is-part-of*) in a single ontology combined with equivalence and subsumption relations between the ontologies. For instance, let us assume there are two geographic ontologies (*o1* and *o2*) and one of them contains the relation *Stockholm is-located-in Sweden*. It is missing in the other ontology. The alignment between them contains two mappings—*o1:Stockholm* $\equiv$ *o2:Stockholm* and *o1:Sweden* $\equiv$ *o2:Sweden*. Thus, adapting our approach we can infer *Stockholm is-located-in Sweden* in the second ontology, i.e., to detect a candidate missing *is-located-in* relation. A similar idea is presented in [17] in the context of ontology enrichment where its authors use properties between the ontologies in order to identify nonalignments (essentially missing subsumptions) in the ontologies.

Furthermore, the set of alignment algorithms in the system can be extended by implementing structure-based matchers, partial-alignment filtering and preprocessing strategies.

When the input ontologies contained thousands of concepts and many defects were detected the system maintained good responsiveness. However, the number of the interactions between the user and the system was high. For instance, during the repairing phase some of the defects had only one repairing action. Instead of showing it to the user, the system could add it automatically, thus reducing the number of user-system interactions. Another direction is to reduce the interactions during the validation phase—this will lead to fewer CMIs and CMMs to validate and fewer missing and wrong is-a relations and mappings to repair. Reducing the number of CMMs can be achieved by utilizing the approach for computing minimal mappings between lightweight ontologies (their structure is based on subsumption relations) as presented in [35]. In their paper the authors propose an efficient algorithm for computing the minimal alignment and observe that such an alignment is unique and always exists.

7.2.2 Long-term future work

Three directions for long-term future work were identified—improving the scalability of the approach, developing new visualization techniques for large datasets and extending the presented approach to ontologies represented in more expressive languages. The subsections below discuss each direction in more detail.

Improving the scalability of the approach

During the ToxOntology-MeSH use case, presented in Subsection 5.2.3, our implemented system had good responsiveness even with 10000 concepts and more than 15000 asserted is-a relations and mappings. The same applies for the experiments with the Anatomy track ontologies from the OAEI. In all experiments, the detection of the defects and the computation of their repairing actions took approximately 30 seconds each. However, the system required between 4 and 6 GB of memory. This fact limits the usage of our approach and system to medium-size ontologies (several thousand concepts) and prevents its application for ontologies like SNOMED (approximately 400 000 classes). Thus, close inspection of the algorithms is necessary in order to reduce memory consumption.

The ontology alignment algorithms in our system are another area that needs attention in the context of scalability since they currently run for hours with high memory consumption. For comparison, the best performing ontology alignment systems, that participated in the OAEI 2012, run for less than a minute with less than 3 GB of memory for the same input. Thus, for a scalable, competitive system the run time and the memory consumption should be reduced. To achieve reduced run time and memory usage two directions can be explored—optimization of the existing algorithms or developing new approaches. For instance, one option is to develop or reuse different heuristics and (structure-based) preprocessing strategies in order to reduce the pairs of concepts for which similarity values are calculated since, currently, the alignment algorithms compute similarity values for all pairs of concepts between the ontologies. We could also investigate in more detail the usage of the mapping suggestions validated as wrong in all phases in the debugging and alignment components.

Another possibility to address the scalability issue of the system is to introduce session-based alignment and debugging, similarly to the methods used in [57]. The session-based framework, described in this paper, addresses almost all of the challenges discussed in [83]. It presents three types of sessions that can be interrupted in order to provide partial results and can then be resumed—computation, validation and recommendation sessions. During the computation session mapping suggestions are generated that are accepted or rejected during the validation session. The recommendation session is used to recommend combinations of alignment algorithms for future computation sessions. Adapting the session-based approach together with

enhanced algorithms will improve scalability and user interaction not only during alignment but also during debugging where the scalability is also an issue.

Visualization techniques for large data sets

Data visualization is another issue, especially when large data structures are involved. The visualization techniques employed in software systems have an important influence on the user perception of the presented data and the ease of use of the system.

It was shown that our system provides contextual visualization, facilitating the understanding of the defects and their repairing actions. Using our grouping techniques the visualized sets were small enough to not clutter the display in most cases. In some cases, however, there were too many objects on the display, which hindered the perception of the visualized information. This observation does not consider what happens when the entire ontology network is visualized at once. Adequate visualization of 10000 concepts (as in some of the experiments described in Chapter 5) with their asserted is-a relations at the same time is currently not possible with our system.

In the above cases we consider only the subclass relations. However, in ontologies with predefined relations, for example, there will be even more, and at the same time diverse, relations between the concepts to visualize. These observations demand further improvement of the available visualization techniques or development of new ones. Moreover, improving the scalability of the approach will allow its application to large ontologies and therefore comprehensive visualization is extremely important.

Ontologies in more expressive languages

The work presented in this thesis is in the context of taxonomies—the simplest kind of ontologies from a knowledge representation point of view. The components of taxonomies are named concepts and is-a relations. Limited to these two components, only simple relations in a domain can be expressed—for instance, recall the earlier example, *maxilla is-a bone*. However, other relations, such as *bone is-not-a blood vessel* and *Stockholm is-located-in Sweden*, cannot be expressed with taxonomies. Thus, extending the scope of this work to ontologies represented in more expressive languages is highly desirable in order to represent more complex relationships in the domain of interest.

A step in this direction is to look at the debugging of is-a relations in ontologies represented in more expressive languages and to investigate the limitations and possible extensions of the current approach in this setting. Regarding the detection phase—the knowledge intrinsic to an ontology network can be employed using techniques similar to those described in this thesis. Other approaches, such as those discussed in Chapter 6.1.1, can be utilized as well. Some of the works described in Subsection 6.1.2 discuss

repairing of wrong is-a relations in the context of ontologies represented in more expressive languages. When it comes to the algorithm for repairing a single missing is-a relation—in the context of the taxonomies all possible solutions can be found and consist of the is-a relations between the sub-concepts and super-concepts of the concepts in the missing is-a relation. However, in the extended setting our repairing algorithm may not be able to find all solutions. The more expressive languages allow complex concept definitions including different logical connectives and quantifiers. Thus, a missing is-a relation can be repaired by adding an is-a relation that is not in the hierarchy of the concepts in the missing is-a relation.

Some work has already been done in this area. In [55] the problem of repairing missing is-a relations is formulated as a generalized version of the TBox abduction problem. In [65] we define properties for the ontologies, the set of missing is-a relations, the domain expert and preferences for the solutions of the problem in [55]. Also, in [93], complexity results for the existence, relevance and necessity decision problems for the generalized TBox abduction problem for $\mathcal{EL}^{++}$ ontologies are presented.

Bibliography

- [1] Adult Mouse Anatomy. http://www.informatics.jax.org/searches/AMA_form.shtml. Accessed: 2013-10-01.
- [2] Apache Jena project. <http://jena.apache.org/>. Accessed: 2013-08-26.
- [3] FaCT++. <http://owl.man.ac.uk/factplusplus/>. Accessed: 2013-08-26.
- [4] GoodRelations. <http://www.heppnetz.de/projects/goodrelations/>. Accessed: 2013-08-26.
- [5] HermiT OWL Reasoner. <http://hermit-reasoner.com>. Accessed: 2013-08-26.
- [6] MeSH: Medical Subject Headings. www.nlm.nih.gov/mesh/. Accessed: 2013-08-26.
- [7] NCI-A. <http://ncit.nci.nih.gov/ncitbrowser/>. Accessed: 2013-10-01.
- [8] Ontology Alignment Evaluation Initiative. <http://oei.ontologymatching.org>. Accessed: 2013-08-26.
- [9] Pellet OWL 2 Reasoner. <http://clarkparsia.com/pellet>. Accessed: 2013-08-26.
- [10] PubMed. www.ncbi.nlm.nih.gov/pubmed/. Accessed: 2013-08-26.
- [11] SNOMED-CT. <http://www.ihtsdo.org/snomed-ct/>. Accessed: 2013-08-26.
- [12] The Pizza ontology. <http://owl.cs.manchester.ac.uk/co-ode-files/ontologies/pizza.owl>. Accessed: 2013-08-26.
- [13] The Wine ontology. w3.org/TR/owl-guide/wine.rdf. Accessed: 2013-08-26.

- [14] Unified Medical Language System. <http://www.nlm.nih.gov/research/umls/>. Accessed: 2013-08-26.
- [15] M Ashburner, C A Ball, J A Blake, D Botstein, H Butler, J M Cherry, A P Davis, K Dolinski, S S Dwight, J T Eppig, M A Harris, D P Hill, L Issel-Tarver, A Kasarskis, S Lewis, J C Matese, J E Richardson, M Ringwald, G M Rubin, and G Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics*, 25(1):25–29, 2000.
- [16] F Baader, D Calvanese, D L McGuinness, D Nardi, and P F Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. 2003.
- [17] M Bada and L Hunter. Identification of OBO Nonalignments and Its Implications for OBO Enrichment. *Bioinformatics (Oxford, England)*, 24(12):1448–1455, 2008.
- [18] S Bail, B Parsia, and U Sattler. Declutter Your Justifications: Determining Similarity Between OWL Explanations. In *Proceedings of the 1st International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2012)*, volume 79 of *LECP*, pages 13–24, 2012.
- [19] E Beisswanger and U Hahn. Towards valid and reusable reference alignments – ten basic quality checks for ontology alignments and their application to three different reference data sets. *Journal of Biomedical Semantics*, 3(Suppl 1), 2012.
- [20] A Bernaras, I Laresgoiti, and J Corera. Building and Reusing Ontologies for Electrical Network Applications. In *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI 1996)*, pages 298–302, 1996.
- [21] T Berners-Lee, J Hendler, and O Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
- [22] P A Bernstein and S Melnik. Model Management 2.0: Manipulating Richer Mappings. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD 2007, pages 1–12, 2007.
- [23] O Bodenreider, T Hayamizu, M Ringwald, S D Coronado, and S Zhang. Of mice and men: aligning mouse and human anatomies. In *Proceedings of the American Medical Informatics Association (AIMA) Annual Symposium*, pages 61–65, 2005.
- [24] P Buitelaar, P Cimiano, and B Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence and Applications Series*. July 2005.

- [25] C Calero, F Ruiz, and M Piattini, editors. *Ontologies for Software Engineering and Software Technology*. 2006.
- [26] B Chen, H Tan, and P Lambrix. Structure-Based Filtering for Ontology Alignment. In *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE 2006*, pages 364–369, 2006.
- [27] C Conroy, R Brennan, D O’Sullivan, and D Lewis. User Evaluation Study of a Tagging Approach to Semantic Mapping. In *The Semantic Web: Research and Applications*, volume 5554 of *LNCS*, pages 623–637. 2009.
- [28] M Copeland, R S Gonçalves, B Parsia, U Sattler, and R Stevens. Finding fault: detecting issues in a versioned ontology. In *Proceedings of the 2nd International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2013)*, volume 999 of *CEUR Workshop Proceedings*, pages 9–20, 2013.
- [29] O Corcho, M Fernández-López, and A Gómez-Pérez. Ontological Engineering: Principles, Methods, Tools and Languages. In *Ontologies for Software Engineering and Software Technology*, pages 1–48. 2006.
- [30] O Corcho, C Roussey, L M V Blazquez, and I Perez. Pattern-based OWL Ontology Debugging Guidelines. In *Proceedings of the Workshop on Ontology Patterns (WOP 2009)*, volume 516 of *CEUR Workshop Proceedings*, 2009.
- [31] H Do and E Rahm. Matching large schemas: Approaches and evaluation. *Information Systems*, 32(6):857–885, 2007.
- [32] W F Dowling and J H Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, 1(3):267–284, 1984.
- [33] H Erdogan, O Bodenreider, and E Erdem. Finding Semantic Inconsistencies in UMLS Using Answer Set Programming. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*, pages 1927–1928, 2010.
- [34] S M Falconer and M Storey. A Cognitive Support Framework for Ontology Mapping. In *The Semantic Web*, volume 4825 of *LNCS*, pages 114–127. 2007.
- [35] F Giunchiglia, V Maltese, and A Autayeu. Computing minimal mappings between lightweight ontologies. *International Journal on Digital Libraries*, 12(4):179–193, 2012.
- [36] A Gómez-Pérez, M Fernández-López, and O Corcho. *Ontological Engineering*. 2004.

- [37] B Cuenca Grau, Z Dragisic, K Eckert, J Euzenat, A Ferrara, R Granada, V Ivanova, E Jiménez-Ruiz, A O Kempf, P Lambrix, A Nikolov, H Paulheim, D Ritze, F Scharffe, P Shvaiko, C Trojahn, and O Zamazal. Results of the Ontology Alignment Evaluation Initiative 2013. In *Proceedings of the 8th International Workshop on Ontology Matching (OM 2013)*, volume 1111 of *CEUR Workshop Proceedings*, pages 61–100, 2013.
- [38] T R Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
- [39] N Guarino, D Oberle, and S Staab. What Is an Ontology? In *Handbook on Ontologies*, International Handbooks on Information Systems, pages 1–17. Second edition, 2009.
- [40] H Happel and S Seedorf. Applications of Ontologies in Software Engineering. In *2nd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006)*, 2006.
- [41] M A Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th Conference on Computational Linguistics*, volume 2 of *COLING 1992*, pages 539–545, 1992.
- [42] N Jeliaskova and V Jeliaskov. AMBIT RESTful web services: an implementation of the OpenTox application programming interface. *Journal of Cheminformatics*, 3(1):1–18, 2011.
- [43] Q Ji, P Haase, G Qi, P Hitzler, and S Stadtmüller. RaDON—Repair and Diagnosis in Ontology Networks. In *Proceedings of the 6th European Semantic Web Conference (ESWC 2009)*, volume 5554 of *LNCS*, pages 863–867, 2009.
- [44] E Jiménez-Ruiz and B Cuenca Grau. LogMap: Logic-Based and Scalable Ontology Matching. In *International Semantic Web Conference (ISWC 2011)*, volume 7031 of *LNCS*, pages 273–288, 2011.
- [45] E Jiménez-Ruiz, B Cuenca Grau, I Horrocks, and R Berlanga. Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences. In *Proceedings of the 6th European Semantic Web Conference (ESWC 2009)*, volume 5554 of *LNCS*, pages 173–187, 2009.
- [46] E Jiménez-Ruiz, B Cuenca Grau, Y Zhou, and I Horrocks. Large-scale Interactive Ontology Matching: Algorithms and Implementation. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, pages 444–449, 2012.
- [47] R Judson, A Richard, D Dix, K Houck, F Elloumi, M Martin, T Cathey, T R Transue, R Spencer, and M Wolf. ACToR—Aggregated Computational Toxicology Resource. *Toxicology and Applied Pharmacology*, 233(1):7–13, 2008.

- [48] A Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, 2006.
- [49] A Kalyanpur, B Parsia, M Horridge, and E Sirin. Finding All Justifications of OWL DL Entailments. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference, ISWC 2007/ASWC 2007*, pages 267–280, 2007.
- [50] A Kalyanpur, B Parsia, E Sirin, and B Cuenca Grau. Repairing Unsatisfiable Concepts in OWL Ontologies. In *Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications, ESWC 2006*, pages 170–184, 2006.
- [51] A Kalyanpur, B Parsia, E Sirin, and J Hendler. Debugging Unsatisfiable Classes in OWL Ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4), 2005.
- [52] A Kumar and B Smith. The Unified Medical Language System and the Gene Ontology: Some Critical Reflections. In *Proceedings of the 26th German Conference on Artificial Intelligence*, volume 2821 of *LNAI*, pages 135–148, 2003.
- [53] P Lambrix. Ontologies in Bioinformatics and Systems Biology. In *Artificial Intelligence Methods And Tools For Systems Biology*, volume 5 of *Computational Biology*, pages 129–145. 2004.
- [54] P Lambrix. Towards a semantic Web for bioinformatics using ontology-based annotation. In *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005.*, pages 3–7, 2005.
- [55] P Lambrix, Z Dragisic, and V Ivanova. Get My Pizza Right: Repairing Missing is-a Relations in *ALC* Ontologies. In *The 2nd Joint International Semantic Technology Conference (JIST 2012)*, volume 7774 of *LNCS*, pages 17–32. 2012.
- [56] P Lambrix and V Ivanova. A unified approach for debugging is-a structure and mappings in networked taxonomies. *Journal of Biomedical Semantics*, 4(1), 2013.
- [57] P Lambrix and R Kaliyaperumal. A Session-Based Approach for Aligning Large Ontologies. In *Proceedings of the 10th European Semantic Web Conference (ESWC 2013)*, volume 7882 of *LNCS*, pages 46–60. 2013.
- [58] P Lambrix and Q Liu. Using partial reference alignments to align ontologies. In *Proceedings of the 6th European Semantic Web Conference (ESWC 2009)*, volume 5554 of *LNCS*, pages 188–202, 2009.

- [59] P Lambrix and Q Liu. Debugging Is-a Structure in Networked Taxonomies. In *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, SWAT4LS 2011, pages 58–65, 2012.
- [60] P Lambrix and Q Liu. Debugging the missing is-a structure within taxonomies networked by partial reference alignments. *Data & Knowledge Engineering*, 86(0):179–205, 2013.
- [61] P Lambrix, Q Liu, and H Tan. Repairing the Missing is-a Structure of Ontologies. In *Proceedings of the 4th Asian Semantic Web Conference (ASWC 2009)*, volume 5926 of *LNCS*, pages 76–90, 2009.
- [62] P Lambrix and H Tan. SAMBO - A system for aligning and merging biomedical ontologies. *Journal of Web Semantics*, 4(3):196–206, 2006.
- [63] P Lambrix and H Tan. Ontology Alignment and Merging. In *Anatomy Ontologies for Bioinformatics*, volume 6 of *Computational Biology*, pages 133–149. 2008.
- [64] P Lambrix, H Tan, V Jakoniene, and L Strömbäck. Biological Ontologies. In *Semantic Web: Revolutionizing Knowledge Discovery in Life Sciences*, pages 85–99. 2007.
- [65] P Lambrix, F Wei-Kleiner, Z Dragisic, and V Ivanova. Repairing missing is-a structure in ontologies is an abductive reasoning problem. In *Proceedings of the 2nd International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2013)*, volume 999 of *CEUR Workshop Proceedings*, pages 33–44, 2013.
- [66] O Lassila and D L McGuinness. The Role of Frame-Based Representation on the Semantic Web. Technical report, 2001.
- [67] Q Liu and P Lambrix. A System for Debugging Missing Is-a Structure in Networked Ontologies. In *Data Integration in the Life Sciences*, volume 6254 of *LNCS*, pages 50–57. 2010.
- [68] C Meilicke, H Stuckenschmidt, and A Taminin. Repairing Ontology Mappings. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, volume 2 of *AAAI 2007*, pages 1408–1413, 2007.
- [69] G A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
- [70] S Mukherjea, B Bamba, and P Kankar. Information retrieval and knowledge discovery utilizing a biomedical Semantic Web. *IEEE Transactions on Knowledge and Data Engineering*, 17:1099–1110, 2005.

- [71] R Neches, R Fikes, T Finin, T Gruber, R Patil, T Senator, and W P Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, 1991.
- [72] T A T Nguyen, R Power, P Piwek, and S Williams. Measuring the understandability of deduction rules for OWL. In *Proceedings of the 1st International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2012)*, volume 79 of *LECP*, pages 1–12, 2012.
- [73] T A T Nguyen, R Power, P Piwek, and S Williams. Predicting the Understandability of OWL Inferences. In *Proceedings of the 10th European Semantic Web Conference (ESWC 2013)*, volume 7882 of *LNCS*, pages 109–123. 2013.
- [74] G Qi and A Harth. Reasoning with Networked Ontologies. In *Ontology Engineering in a Networked World*, pages 363–380. 2012.
- [75] G Qi, Q Ji, and P Haase. A Conflict-Based Operator for Mapping Revision. In *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, volume 5823 of *ISWC 2009*, pages 521–536, 2009.
- [76] R Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.
- [77] C Roussey and O Zamazal. Antipattern detection: how to debug an ontology without a reasoner. In *Proceedings of the 2nd International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2013)*, volume 999 of *CEUR Workshop Proceedings*, pages 45–56, 2013.
- [78] F Ruiz and J R Hilera. Using Ontologies in Software Engineering and Technology. In *Ontologies for Software Engineering and Software Technology*, pages 49–102. 2006.
- [79] E Santos, D Faria, C Pesquita, and F M Couto. Ontology alignment repair through modularization and confidence-based heuristics. *CoRR*, abs/1307.5322, 2013.
- [80] S Schlobach. Debugging and Semantic Clarification by Pinpointing. In *Proceedings of the 2nd European Conference on The Semantic Web: Research and Applications (ESWC 2005)*, volume 3532 of *LNCS*, pages 27–44, 2005.
- [81] S Schlobach and R Cornet. Non-standard Reasoning Services for the Debugging of Description Logic Terminologies. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, IJCAI 2003, pages 355–360, 2003.

- [82] L Serafini, A Borgida, and A Tamin. Aspects of Distributed and Modular Ontology Reasoning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, IJCAI 2005, pages 570–575, 2005.
- [83] P Shvaiko and J Euzenat. Ontology Matching: State of the Art and Future Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):158–176, 2013.
- [84] H Stuckenschmidt. Debugging weighted ontologies. In *Proceedings of the 2nd International Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM 2013)*, volume 999 of *CEUR Workshop Proceedings*, pages 1–8, 2013.
- [85] R Studer, V R Benjamins, and D Fensel. Knowledge Engineering: Principles and Methods. *Data & Knowledge Engineering*, 25(1–2):161–197, 1998.
- [86] B Swartout, R Patil, K Knight, and T Russ. Toward Distributed Use of Large-Scale Ontologies. In *Ontological Engineering, AAAI-97 Spring Symposium Series*, pages 138–148, 1997.
- [87] H Tan, V Jakoniene, P Lambrix, J Aberg, and N Shahmehri. Alignment of Biomedical Ontologies Using Life Science Literature. In *Knowledge Discovery in Life Science Literature*, volume 3886 of *LNCS*, pages 1–17. 2006.
- [88] M Uschold and M Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11:93–136, 1996.
- [89] M Uschold and M Gruninger. Ontologies and Semantics for Seamless Connectivity. *SIGMOD Record*, 33(4):58–64, 2004.
- [90] G van Heijst, A T Schreiber, and B J Wielinga. Using Explicit Ontologies in KBS Development. *International Journal Human-Computer Studies*, 46(2–3):183–292, 1997.
- [91] H Wache, T Vögele, U Visser, H Stuckenschmidt, G Schuster, H Neumann, and S Hübner. Ontology-based integration of information—a survey of existing approaches. In *Proceedings of the International Joint Conference on Artificial Intelligence—01 Workshop: Ontologies and Information Sharing*, pages 108–117, 2001.
- [92] P Wang and B Xu. Debugging Ontology Mappings: A Static Approach. *Computing and Informatics*, 27(1):21–36, 2008.
- [93] F Wei-Kleiner, Z Dragisic, and P Lambrix. Abduction Framework for Repairing Incomplete $\mathcal{EL}$ Ontologies: Complexity Results and Algorithms. under review.

Department of Computer and Information Science
Linköpings universitet

Licentiate Theses

Linköpings Studies in Science and Technology
Faculty of Arts and Sciences

- No 17 **Vojin Plavsic:** Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory. (Available at: FOA, Box 1165, S-581 11 Linköping, Sweden. FOA Report B30062E)
- No 28 **Arne Jönsson, Mikael Patel:** An Interactive Flowcharting Technique for Communicating and Realizing Algorithms, 1984.
- No 29 **Johnny Eckerland:** Retargeting of an Incremental Code Generator, 1984.
- No 48 **Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching, 1985.
- No 52 **Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems, 1985.
- No 60 **Johan Fagerström:** Simulation and Evaluation of Architecture based on Asynchronous Processes, 1985.
- No 71 **Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System, 1987.
- No 72 **Tony Larsson:** On the Specification and Verification of VLSI Systems, 1986.
- No 73 **Ola Strömfors:** A Structure Editor for Documents and Programs, 1986.
- No 74 **Christos Levcopoulos:** New Results about the Approximation Behavior of the Greedy Triangulation, 1986.
- No 104 **Shamsul I. Chowdhury:** Statistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology, 1987.
- No 108 **Rober Bilos:** Incremental Scanning and Token-Based Editing, 1987.
- No 111 **Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments, 1987.
- No 113 **Ralph Rönquist:** Network and Lattice Based Approaches to the Representation of Knowledge, 1987.
- No 118 **Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs, 1987.
- No 126 **Dan Strömberg:** Transfer and Distribution of Application Programs, 1987.
- No 127 **Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems, 1987.
- No 139 **Christer Bäckström:** Reasoning about Interdependent Actions, 1988.
- No 140 **Mats Wirén:** On Control Strategies and Incrementality in Unification-Based Chart Parsing, 1988.
- No 146 **Johan Hultman:** A Software System for Defining and Controlling Actions in a Mechanical System, 1988.
- No 150 **Tim Hansen:** Diagnosing Faults using Knowledge about Malfunctioning Behavior, 1988.
- No 165 **Jonas Löwgren:** Supporting Design and Management of Expert System User Interfaces, 1989.
- No 166 **Ola Petersson:** On Adaptive Sorting in Sequential and Parallel Models, 1989.
- No 174 **Yngve Larsson:** Dynamic Configuration in a Distributed Environment, 1989.
- No 177 **Peter Åberg:** Design of a Multiple View Presentation and Interaction Manager, 1989.
- No 181 **Henrik Eriksson:** A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.
- No 184 **Ivan Rankin:** The Deep Generation of Text in Expert Critiquing Systems, 1989.
- No 187 **Simin Nadjm-Tehrani:** Contributions to the Declarative Approach to Debugging Prolog Programs, 1989.
- No 189 **Magnus Merkel:** Temporal Information in Natural Language, 1989.
- No 196 **Ulf Nilsson:** A Systematic Approach to Abstract Interpretation of Logic Programs, 1989.
- No 197 **Staffan Bonnier:** Horn Clause Logic with External Procedures: Towards a Theoretical Framework, 1989.
- No 203 **Christer Hansson:** A Prototype System for Logical Reasoning about Time and Action, 1990.
- No 212 **Björn Fjellborg:** An Approach to Extraction of Pipeline Structures for VLSI High-Level Synthesis, 1990.
- No 230 **Patrick Doherty:** A Three-Valued Approach to Non-Monotonic Reasoning, 1990.
- No 237 **Tomas Sokolnicki:** Coaching Partial Plans: An Approach to Knowledge-Based Tutoring, 1990.
- No 250 **Lars Strömberg:** Postmortem Debugging of Distributed Systems, 1990.
- No 253 **Torbjörn Näslund:** SLDDFA-Resolution - Computing Answers for Negative Queries, 1990.
- No 260 **Peter D. Holmes:** Using Connectivity Graphs to Support Map-Related Reasoning, 1991.
- No 283 **Olof Johansson:** Improving Implementation of Graphical User Interfaces for Object-Oriented Knowledge- Bases, 1991.
- No 298 **Rolf G Larsson:** Aktivitetsbaserad kalkylering i ett nytt ekonomisystem, 1991.
- No 318 **Lena Srömbäck:** Studies in Extended Unification-Based Formalism for Linguistic Description: An Algorithm for Feature Structures with Disjunction and a Proposal for Flexible Systems, 1992.
- No 319 **Mikael Pettersson:** DML-A Language and System for the Generation of Efficient Compilers from Denotational Specification, 1992.
- No 326 **Andreas Kågedal:** Logic Programming with External Procedures: an Implementation, 1992.
- No 328 **Patrick Lambrix:** Aspects of Version Management of Composite Objects, 1992.
- No 333 **Xinli Gu:** Testability Analysis and Improvement in High-Level Synthesis Systems, 1992.
- No 335 **Torbjörn Näslund:** On the Role of Evaluations in Iterative Development of Managerial Support Systems, 1992.
- No 348 **Ulf Cederling:** Industrial Software Development - a Case Study, 1992.
- No 352 **Magnus Morin:** Predictable Cyclic Computations in Autonomous Systems: A Computational Model and Implementation, 1992.
- No 371 **Mehran Noghabai:** Evaluation of Strategic Investments in Information Technology, 1993.
- No 378 **Mats Larsson:** A Transformational Approach to Formal Digital System Design, 1993.

- No 380 **Johan Ringström:** Compiler Generation for Parallel Languages from Denotational Specifications, 1993.
- No 381 **Michael Jansson:** Propagation of Change in an Intelligent Information System, 1993.
- No 383 **Jonni Harrius:** An Architecture and a Knowledge Representation Model for Expert Critiquing Systems, 1993.
- No 386 **Per Österling:** Symbolic Modelling of the Dynamic Environments of Autonomous Agents, 1993.
- No 398 **Johan Boye:** Dependency-based Groudnness Analysis of Functional Logic Programs, 1993.
- No 402 **Lars Degerstedt:** Tabulated Resolution for Well Founded Semantics, 1993.
- No 406 **Anna Moberg:** Satellitkontor - en studie av kommunikationsmönster vid arbete på distans, 1993.
- No 414 **Peter Carlsson:** Separation av företagsledning och finansiering - fallstudier av företagsledarutköp ur ett agent-teoretiskt perspektiv, 1994.
- No 417 **Camilla Sjöström:** Revision och lagreglering - ett historiskt perspektiv, 1994.
- No 436 **Cecilia Sjöberg:** Voices in Design: Argumentation in Participatory Development, 1994.
- No 437 **Lars Viklund:** Contributions to a High-level Programming Environment for a Scientific Computing, 1994.
- No 440 **Peter Loborg:** Error Recovery Support in Manufacturing Control Systems, 1994.
- FHS 3/94 **Owen Eriksson:** Informationssystem med verksamhetskvalitet - utvärdering baserat på ett verksamhetsinriktat och samskapande perspektiv, 1994.
- FHS 4/94 **Karin Pettersson:** Informationssystemstrukturer, ansvarsfördelning och användarinflytande - En komparativ studie med utgångspunkt i två informationssystemstrategier, 1994.
- No 441 **Lars Poignant:** Informationsteknologi och företagsetablering - Effekter på produktivitet och region, 1994.
- No 446 **Gustav Fahl:** Object Views of Relational Data in Multidatabase Systems, 1994.
- No 450 **Henrik Nilsson:** A Declarative Approach to Debugging for Lazy Functional Languages, 1994.
- No 451 **Jonas Lind:** Creditor - Firm Relations: an Interdisciplinary Analysis, 1994.
- No 452 **Martin Sköld:** Active Rules based on Object Relational Queries - Efficient Change Monitoring Techniques, 1994.
- No 455 **Pär Carlshamre:** A Collaborative Approach to Usability Engineering: Technical Communicators and System Developers in Usability-Oriented Systems Development, 1994.
- FHS 5/94 **Stefan Cronholm:** Varför CASE-verktyg i systemutveckling? - En motiv- och konsekvensstudie avseende arbetssätt och arbetsformer, 1994.
- No 462 **Mikael Lindvall:** A Study of Traceability in Object-Oriented Systems Development, 1994.
- No 463 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av Sandviks förvärv av Bahco Verktyg, 1994.
- No 464 **Hans Olsén:** Collage Induction: Proving Properties of Logic Programs by Program Synthesis, 1994.
- No 469 **Lars Karlsson:** Specification and Synthesis of Plans Using the Features and Fluents Framework, 1995.
- No 473 **Ulf Söderman:** On Conceptual Modelling of Mode Switching Systems, 1995.
- No 475 **Choong-ho Yi:** Reasoning about Concurrent Actions in the Trajectory Semantics, 1995.
- No 476 **Bo Lagerström:** Successiv resultatavräkning av pågående arbeten. - Fallstudier i tre byggföretag, 1995.
- No 478 **Peter Jonsson:** Complexity of State-Variable Planning under Structural Restrictions, 1995.
- FHS 7/95 **Anders Avdic:** Arbetsintegrerad systemutveckling med kalkylprogram, 1995.
- No 482 **Eva L Ragnemalm:** Towards Student Modelling through Collaborative Dialogue with a Learning Companion, 1995.
- No 488 **Eva Toller:** Contributions to Parallel Multiparadigm Languages: Combining Object-Oriented and Rule-Based Programming, 1995.
- No 489 **Erik Stoy:** A Petri Net Based Unified Representation for Hardware/Software Co-Design, 1995.
- No 497 **Johan Herber:** Environment Support for Building Structured Mathematical Models, 1995.
- No 498 **Stefan Svenberg:** Structure-Driven Derivation of Inter-Lingual Functor-Argument Trees for Multi-Lingual Generation, 1995.
- No 503 **Hee-Cheol Kim:** Prediction and Postdiction under Uncertainty, 1995.
- FHS 8/95 **Dan Fristedt:** Metoder i användning - mot förbättring av systemutveckling genom situationell metodkunskap och metodanalys, 1995.
- FHS 9/95 **Malin Bergvall:** Systemförvaltning i praktiken - en kvalitativ studie avseende centrala begrepp, aktiviteter och ansvarsroller, 1995.
- No 513 **Joachim Karlsson:** Towards a Strategy for Software Requirements Selection, 1995.
- No 517 **Jakob Axelsson:** Schedulability-Driven Partitioning of Heterogeneous Real-Time Systems, 1995.
- No 518 **Göran Forslund:** Toward Cooperative Advice-Giving Systems: The Expert Systems Experience, 1995.
- No 522 **Jörgen Andersson:** Bilder av småföretagares ekonomistyrning, 1995.
- No 538 **Staffan Flodin:** Efficient Management of Object-Oriented Queries with Late Binding, 1996.
- No 545 **Vadim Engelson:** An Approach to Automatic Construction of Graphical User Interfaces for Applications in Scientific Computing, 1996.
- No 546 **Magnus Werner :** Multidatabase Integration using Polymorphic Queries and Views, 1996.
- FiF-a 1/96 **Mikael Lind:** Affärsprocessinriktad förändringsanalys - utveckling och tillämpning av synsätt och metod, 1996.
- No 549 **Jonas Hallberg:** High-Level Synthesis under Local Timing Constraints, 1996.
- No 550 **Kristina Larsen:** Förutsättningar och begränsningar för arbete på distans - erfarenheter från fyra svenska företag. 1996.
- No 557 **Mikael Johansson:** Quality Functions for Requirements Engineering Methods, 1996.
- No 558 **Patrik Nordling:** The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
- No 561 **Anders Ekman:** Exploration of Polygonal Environments, 1996.
- No 563 **Niclas Andersson:** Compilation of Mathematical Models to Parallel Code, 1996.

- No 567 **Johan Jenvald:** Simulation and Data Collection in Battle Training, 1996.
- No 575 **Niclas Ohlsson:** Software Quality Engineering by Early Identification of Fault-Prone Modules, 1996.
- No 576 **Mikael Ericsson:** Commenting Systems as Design Support—A Wizard-of-Oz Study, 1996.
- No 587 **Jörgen Lindström:** Chefers användning av kommunikationsteknik, 1996.
- No 589 **Esa Falkenroth:** Data Management in Control Applications - A Proposal Based on Active Database Systems, 1996.
- No 591 **Niclas Wahllöf:** A Default Extension to Description Logics and its Applications, 1996.
- No 595 **Annika Larsson:** Ekonomisk Styrning och Organisatorisk Passion - ett interaktivt perspektiv, 1997.
- No 597 **Ling Lin:** A Value-based Indexing Technique for Time Sequences, 1997.
- No 598 **Rego Granlund:** C³Fire - A Microworld Supporting Emergency Management Training, 1997.
- No 599 **Peter Ingels:** A Robust Text Processing Technique Applied to Lexical Error Recovery, 1997.
- No 607 **Per-Arne Persson:** Toward a Grounded Theory for Support of Command and Control in Military Coalitions, 1997.
- No 609 **Jonas S Karlsson:** A Scalable Data Structure for a Parallel Data Server, 1997.
- FiF-a 4 **Carita Åbom:** Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.
- FiF-a 6 **Tommy Wedlund:** Att skapa en företagsanpassad systemutvecklingsmodell - genom rekonstruktion, värdering och vidareutveckling i T50-bolag inom ABB, 1997.
- No 615 **Silvia Coradeschi:** A Decision-Mechanism for Reactive and Coordinated Agents, 1997.
- No 623 **Jan Ollinen:** Det flexibla kontorets utveckling på Digital - Ett stöd för multiflex? 1997.
- No 626 **David Byers:** Towards Estimating Software Testability Using Static Analysis, 1997.
- No 627 **Fredrik Eklund:** Declarative Error Diagnosis of GAPLog Programs, 1997.
- No 629 **Gunilla Ivefors:** Krigsspel och Informationsteknik inför en oförutsägbart framtid, 1997.
- No 631 **Jens-Olof Lindh:** Analysing Traffic Safety from a Case-Based Reasoning Perspective, 1997
- No 639 **Jukka Mäki-Turja:** Smalltalk - a suitable Real-Time Language, 1997.
- No 640 **Juha Takkinen:** CAFE: Towards a Conceptual Model for Information Management in Electronic Mail, 1997.
- No 643 **Man Lin:** Formal Analysis of Reactive Rule-based Programs, 1997.
- No 653 **Mats Gustafsson:** Bringing Role-Based Access Control to Distributed Systems, 1997.
- FiF-a 13 **Boris Karlsson:** Metodanalys för förståelse och utveckling av systemutvecklingsverksamhet. Analys och värdering av systemutvecklingsmodeller och dess användning, 1997.
- No 674 **Marcus Bjärelund:** Two Aspects of Automating Logics of Action and Change - Regression and Tractability, 1998.
- No 676 **Jan Håkegård:** Hierarchical Test Architecture and Board-Level Test Controller Synthesis, 1998.
- No 668 **Per-Ove Zetterlund:** Normering av svensk redovisning - En studie av tillkomsten av Redovisningsrådets rekommendation om koncernredovisning (RR01:91), 1998.
- No 675 **Jimmy Tjäder:** Projektledaren & planen - en studie av projektledning i tre installations- och systemutvecklingsprojekt, 1998.
- FiF-a 14 **Ulf Melin:** Informationssystem vid ökad affärs- och processorientering - egenskaper, strategier och utveckling, 1998.
- No 695 **Tim Heyer:** COMPASS: Introduction of Formal Methods in Code Development and Inspection, 1998.
- No 700 **Patrik Hägglund:** Programming Languages for Computer Algebra, 1998.
- FiF-a 16 **Marie-Therese Christiansson:** Inter-organisatorisk verksamhetsutveckling - metoder som stöd vid utveckling av partnerskap och informationssystem, 1998.
- No 712 **Christina Wennestam:** Information om immateriella resurser. Investeringar i forskning och utveckling samt i personal inom skogsindustrin, 1998.
- No 719 **Joakim Gustafsson:** Extending Temporal Action Logic for Ramification and Concurrency, 1998.
- No 723 **Henrik André-Jönsson:** Indexing time-series data using text indexing methods, 1999.
- No 725 **Erik Larsson:** High-Level Testability Analysis and Enhancement Techniques, 1998.
- No 730 **Carl-Johan Westin:** Informationsförsörjning: en fråga om ansvar - aktiviteter och uppdrag i fem stora svenska organisationers operativa informationsförsörjning, 1998.
- No 731 **Åse Jansson:** Miljöhänsyn - en del i företags styrning, 1998.
- No 733 **Thomas Padron-McCarthy:** Performance-Polymorphic Declarative Queries, 1998.
- No 734 **Anders Bäckström:** Värdeskapande kreditgivning - Kreditriskhantering ur ett agentteoretiskt perspektiv, 1998.
- FiF-a 21 **Ulf Seigerroth:** Integration av förändringsmetoder - en modell för välgrundad metodintegration, 1999.
- FiF-a 22 **Fredrik Öberg:** Object-Oriented Frameworks - A New Strategy for Case Tool Development, 1998.
- No 737 **Jonas Mellin:** Predictable Event Monitoring, 1998.
- No 738 **Joakim Eriksson:** Specifying and Managing Rules in an Active Real-Time Database System, 1998.
- FiF-a 25 **Bengt E W Andersson:** Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsarenor i samverkan om utbyte av information, 1998.
- No 742 **Pawel Pietrzak:** Static Incorrectness Diagnosis of CLP (FD), 1999.
- No 748 **Tobias Ritzau:** Real-Time Reference Counting in RT-Java, 1999.
- No 751 **Anders Ferntoft:** Elektronisk affärskommunikation - kontaktkostnader och kontaktprocesser mellan kunder och leverantörer på producentmarknader, 1999.
- No 752 **Jo Skåmedal:** Arbete på distans och arbetsformens påverkan på resor och resmönster, 1999.
- No 753 **Johan Alvehus:** Mötets metaforer. En studie av berättelser om möten, 1999.

- No 754 **Magnus Lindahl:** Bankens villkor i låneavtal vid kreditgivning till högt belånade företagsförvärv: En studie ur ett agentteoretiskt perspektiv, 2000.
- No 766 **Martin V. Howard:** Designing dynamic visualizations of temporal data, 1999.
- No 769 **Jesper Andersson:** Towards Reactive Software Architectures, 1999.
- No 775 **Anders Henriksson:** Unique kernel diagnosis, 1999.
- FiF-a 30 **Pär J. Ågerfalk:** Pragmatization of Information Systems - A Theoretical and Methodological Outline, 1999.
- No 787 **Charlotte Björkegren:** Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.
- No 788 **Håkan Nilsson:** Informationsteknik som drivkraft i granskningsprocessen - En studie av fyra revisionsbyråer, 2000.
- No 790 **Erik Berglund:** Use-Oriented Documentation in Software Development, 1999.
- No 791 **Klas Gäre:** Verksamhetsförändringar i samband med IS-införande, 1999.
- No 800 **Anders Subotic:** Software Quality Inspection, 1999.
- No 807 **Svein Bergum:** Managerial communication in telework, 2000.
- No 809 **Flavius Gruian:** Energy-Aware Design of Digital Systems, 2000.
- FiF-a 32 **Karin Hedström:** Kunskapsanvändning och kunskapsutveckling hos verksamhetskonsulter - Erfarenheter från ett FOU-samarbete, 2000.
- No 808 **Linda Askenäs:** Affärssystemet - En studie om teknikens aktiva och passiva roll i en organisation, 2000.
- No 820 **Jean Paul Meynard:** Control of industrial robots through high-level task programming, 2000.
- No 823 **Lars Hult:** Publika Gränssytor - ett designexempel, 2000.
- No 832 **Paul Pop:** Scheduling and Communication Synthesis for Distributed Real-Time Systems, 2000.
- FiF-a 34 **Göran Hultgren:** Nätverksinriktad Förändringsanalys - perspektiv och metoder som stöd för förståelse och utveckling av affärsrelationer och informationssystem, 2000.
- No 842 **Magnus Kald:** The role of management control systems in strategic business units, 2000.
- No 844 **Mikael Cäker:** Vad kostar kunden? Modeller för intern redovisning, 2000.
- FiF-a 37 **Ewa Braf:** Organisationers kunskapsverksamheter - en kritisk studie av "knowledge management", 2000.
- FiF-a 40 **Henrik Lindberg:** Webaserade affärsprocesser - Möjligheter och begränsningar, 2000.
- FiF-a 41 **Benneth Christiansson:** Att komponentbasera informationssystem - Vad säger teori och praktik?, 2000.
- No. 854 **Ola Pettersson:** Deliberation in a Mobile Robot, 2000.
- No 863 **Dan Lawesson:** Towards Behavioral Model Fault Isolation for Object Oriented Control Systems, 2000.
- No 881 **Johan Moe:** Execution Tracing of Large Distributed Systems, 2001.
- No 882 **Yuxiao Zhao:** XML-based Frameworks for Internet Commerce and an Implementation of B2B e-procurement, 2001.
- No 890 **Annika Flycht-Eriksson:** Domain Knowledge Management in Information-providing Dialogue systems, 2001.
- FiF-a 47 **Per-Arne Segerkvist:** Webaserade imaginära organisations samverkansformer: Informationssystemarkitektur och aktörssamverkan som förutsättningar för affärsprocesser, 2001.
- No 894 **Stefan Svarén:** Styrning av investeringar i divisionaliserade företag - Ett koncernperspektiv, 2001.
- No 906 **Lin Han:** Secure and Scalable E-Service Software Delivery, 2001.
- No 917 **Emma Hansson:** Optionsprogram för anställda - en studie av svenska börsföretag, 2001.
- No 916 **Susanne Odar:** IT som stöd för strategiska beslut, en studie av datorimplementerade modeller av verksamhet som stöd för beslut om anskaffning av JAS 1982, 2002.
- FiF-a-49 **Stefan Holgersson:** IT-system och filtrering av verksamhetskunskap - kvalitetsproblem vid analyser och beslutsfattande som bygger på uppgifter hämtade från polisens IT-system, 2001.
- FiF-a-51 **Per Oscarsson:** Informationssäkerhet i verksamheter - begrepp och modeller som stöd för förståelse av informationssäkerhet och dess hantering, 2001.
- No 919 **Luis Alejandro Cortes:** A Petri Net Based Modeling and Verification Technique for Real-Time Embedded Systems, 2001.
- No 915 **Niklas Sandell:** Redovisning i skuggan av en bankkris - Värdering av fastigheter. 2001.
- No 931 **Fredrik Elg:** Ett dynamiskt perspektiv på individuella skillnader av heuristisk kompetens, intelligens, mentala modeller, mål och konfidens i kontroll av mikrovärlden Moro, 2002.
- No 933 **Peter Aronsson:** Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, 2002.
- No 938 **Bourhane Kadmiry:** Fuzzy Control of Unmanned Helicopter, 2002.
- No 942 **Patrik Haslum:** Prediction as a Knowledge Representation Problem: A Case Study in Model Design, 2002.
- No 956 **Robert Sevenius:** On the instruments of governance - A law & economics study of capital instruments in limited liability companies, 2002.
- FiF-a 58 **Johan Petersson:** Lokala elektroniska marknadsplatser - informationssystem för platsbundna affärer, 2002.
- No 964 **Peter Bunus:** Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.
- No 973 **Gert Jervan:** High-Level Test Generation and Built-In Self-Test Techniques for Digital Systems, 2002.
- No 958 **Fredrika Berglund:** Management Control and Strategy - a Case Study of Pharmaceutical Drug Development, 2002.
- FiF-a 61 **Fredrik Karlsson:** Meta-Method for Method Configuration - A Rational Unified Process Case, 2002.
- No 985 **Sorin Manolache:** Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, 2002.
- No 982 **Diana Szentiványi:** Performance and Availability Trade-offs in Fault-Tolerant Middleware, 2002.
- No 989 **Iakov Nakhimovski:** Modeling and Simulation of Contacting Flexible Bodies in Multibody Systems, 2002.
- No 990 **Levon Saldamli:** PDEModelica - Towards a High-Level Language for Modeling with Partial Differential Equations, 2002.
- No 991 **Almut Herzog:** Secure Execution Environment for Java Electronic Services, 2002.

- No 999 **Jon Edvardsson:** Contributions to Program- and Specification-based Test Data Generation, 2002.
- No 1000 **Anders Arpteg:** Adaptive Semi-structured Information Extraction, 2002.
- No 1001 **Andrzej Bednarski:** A Dynamic Programming Approach to Optimal Retargetable Code Generation for Irregular Architectures, 2002.
- No 988 **Mattias Arvola:** Good to use! : Use quality of multi-user applications in the home, 2003.
- FiF-a 62 **Lennart Ljung:** Utveckling av en projektivitetsmodell - om organisationers förmåga att tillämpa projektarbetsformen, 2003.
- No 1003 **Pernilla Qvarfordt:** User experience of spoken feedback in multimodal interaction, 2003.
- No 1005 **Alexander Siemers:** Visualization of Dynamic Multibody Simulation With Special Reference to Contacts, 2003.
- No 1008 **Jens Gustavsson:** Towards Unanticipated Runtime Software Evolution, 2003.
- No 1010 **Calin Curescu:** Adaptive QoS-aware Resource Allocation for Wireless Networks, 2003.
- No 1015 **Anna Andersson:** Management Information Systems in Process-oriented Healthcare Organisations, 2003.
- No 1018 **Björn Johansson:** Feedforward Control in Dynamic Situations, 2003.
- No 1022 **Traian Pop:** Scheduling and Optimisation of Heterogeneous Time/Event-Triggered Distributed Embedded Systems, 2003.
- FiF-a 65 **Britt-Marie Johansson:** Kundkommunikation på distans - en studie om kommunikationsmediets betydelse i affärstransaktioner, 2003.
- No 1024 **Aleksandra Tešanovic:** Towards Aspectual Component-Based Real-Time System Development, 2003.
- No 1034 **Arja Vainio-Larsson:** Designing for Use in a Future Context - Five Case Studies in Retrospect, 2003.
- No 1033 **Peter Nilsson:** Svenska bankers redovisningsval vid reservering för befarade kreditförluster - En studie vid införandet av nya redovisningsregler, 2003.
- FiF-a 69 **Fredrik Ericsson:** Information Technology for Learning and Acquiring of Work Knowledge, 2003.
- No 1049 **Marcus Comstedt:** Towards Fine-Grained Binary Composition through Link Time Weaving, 2003.
- No 1052 **Åsa Hedenskog:** Increasing the Automation of Radio Network Control, 2003.
- No 1054 **Claudiu Duma:** Security and Efficiency Tradeoffs in Multicast Group Key Management, 2003.
- FiF-a 71 **Emma Eliason:** Effekttanalys av IT-systems handlingsutrymme, 2003.
- No 1055 **Carl Cederberg:** Experiments in Indirect Fault Injection with Open Source and Industrial Software, 2003.
- No 1058 **Daniel Karlsson:** Towards Formal Verification in a Component-based Reuse Methodology, 2003.
- FiF-a 73 **Anders Hjalmarsson:** Att etablera och vidmakthålla förbättringsverksamhet - behovet av koordination och interaktion vid förändring av systemutvecklingsverksamheter, 2004.
- No 1079 **Pontus Johansson:** Design and Development of Recommender Dialogue Systems, 2004.
- No 1084 **Charlotte Stoltz:** Calling for Call Centres - A Study of Call Centre Locations in a Swedish Rural Region, 2004.
- FiF-a 74 **Björn Johansson:** Deciding on Using Application Service Provision in SMEs, 2004.
- No 1094 **Genevieve Gorrell:** Language Modelling and Error Handling in Spoken Dialogue Systems, 2004.
- No 1095 **Ulf Johansson:** Rule Extraction - the Key to Accurate and Comprehensible Data Mining Models, 2004.
- No 1099 **Sonia Sangari:** Computational Models of Some Communicative Head Movements, 2004.
- No 1110 **Hans Nässla:** Intra-Family Information Flow and Prospects for Communication Systems, 2004.
- No 1116 **Henrik Sällberg:** On the value of customer loyalty programs - A study of point programs and switching costs, 2004.
- FiF-a 77 **Ulf Larsson:** Designarbete i dialog - karaktärisering av interaktionen mellan användare och utvecklare i en systemutvecklingsprocess, 2004.
- No 1126 **Andreas Borg:** Contribution to Management and Validation of Non-Functional Requirements, 2004.
- No 1127 **Per-Ola Kristensson:** Large Vocabulary Shorthand Writing on Stylus Keyboard, 2004.
- No 1132 **Pär-Anders Albinsson:** Interacting with Command and Control Systems: Tools for Operators and Designers, 2004.
- No 1130 **Ioan Chisalita:** Safety-Oriented Communication in Mobile Networks for Vehicles, 2004.
- No 1138 **Thomas Gustafsson:** Maintaining Data Consistency in Embedded Databases for Vehicular Systems, 2004.
- No 1149 **Vaida Jakonić:** A Study in Integrating Multiple Biological Data Sources, 2005.
- No 1156 **Abdil Rashid Mohamed:** High-Level Techniques for Built-In Self-Test Resources Optimization, 2005.
- No 1162 **Adrian Pop:** Contributions to Meta-Modeling Tools and Methods, 2005.
- No 1165 **Fidel Vascós Palacios:** On the information exchange between physicians and social insurance officers in the sick leave process: an Activity Theoretical perspective, 2005.
- FiF-a 84 **Jenny Lagsten:** Verksamhetsutvecklande utvärdering i informationssystemprojekt, 2005.
- No 1166 **Emma Larsdotter Nilsson:** Modeling, Simulation, and Visualization of Metabolic Pathways Using Modelica, 2005.
- No 1167 **Christina Keller:** Virtual Learning Environments in higher education. A study of students' acceptance of educational technology, 2005.
- No 1168 **Cécile Åberg:** Integration of organizational workflows and the Semantic Web, 2005.
- FiF-a 85 **Anders Forsman:** Standardisering som grund för informationssamverkan och IT-tjänster - En fallstudie baserad på trafikinformationstjänsten RDS-TMC, 2005.
- No 1171 **Yu-Hsing Huang:** A systemic traffic accident model, 2005.
- FiF-a 86 **Jan Olsson:** Att modellera uppdrag - grunder för förståelse av processinriktade informationssystem i transaktionsintensiva verksamheter, 2005.
- No 1172 **Petter Ahlström:** Affärsstrategier för seniorbostadsmarknaden, 2005.
- No 1183 **Mathias Cöster:** Beyond IT and Productivity - How Digitization Transformed the Graphic Industry, 2005.
- No 1184 **Åsa Horzella:** Beyond IT and Productivity - Effects of Digitized Information Flows in Grocery Distribution, 2005.
- No 1185 **Maria Kollberg:** Beyond IT and Productivity - Effects of Digitized Information Flows in the Logging Industry, 2005.
- No 1190 **David Dinka:** Role and Identity - Experience of technology in professional settings, 2005.

- No 1191 **Andreas Hansson**: Increasing the Storage Capacity of Recursive Auto-associative Memory by Segmenting Data, 2005.
- No 1192 **Nicklas Bergfeldt**: Towards Detached Communication for Robot Cooperation, 2005.
- No 1194 **Dennis Maciuszek**: Towards Dependable Virtual Companions for Later Life, 2005.
- No 1204 **Beatrice Alenljung**: Decision-making in the Requirements Engineering Process: A Human-centered Approach, 2005.
- No 1206 **Anders Larsson**: System-on-Chip Test Scheduling and Test Infrastructure Design, 2005.
- No 1207 **John Wilander**: Policy and Implementation Assurance for Software Security, 2005.
- No 1209 **Andreas Käll**: Översättningar av en managementmodell - En studie av införandet av Balanced Scorecard i ett landsting, 2005.
- No 1225 **He Tan**: Aligning and Merging Biomedical Ontologies, 2006.
- No 1228 **Artur Wilk**: Descriptive Types for XML Query Language Xcerpt, 2006.
- No 1229 **Per Olof Pettersson**: Sampling-based Path Planning for an Autonomous Helicopter, 2006.
- No 1231 **Kalle Burbeck**: Adaptive Real-time Anomaly Detection for Safeguarding Critical Networks, 2006.
- No 1233 **Daniela Mihailescu**: Implementation Methodology in Action: A Study of an Enterprise Systems Implementation Methodology, 2006.
- No 1244 **Jörgen Skågeby**: Public and Non-public gifting on the Internet, 2006.
- No 1248 **Karolina Eliasson**: The Use of Case-Based Reasoning in a Human-Robot Dialog System, 2006.
- No 1263 **Misook Park-Westman**: Managing Competence Development Programs in a Cross-Cultural Organisation - What are the Barriers and Enablers, 2006.
- FiF-a 90 **Amra Halilovic**: Ett praktikperspektiv på hantering av mjukvarukomponenter, 2006.
- No 1272 **Raquel Flodström**: A Framework for the Strategic Management of Information Technology, 2006.
- No 1277 **Viacheslav Izosimov**: Scheduling and Optimization of Fault-Tolerant Embedded Systems, 2006.
- No 1283 **Håkan Hasewinkel**: A Blueprint for Using Commercial Games off the Shelf in Defence Training, Education and Research Simulations, 2006.
- FiF-a 91 **Hanna Broberg**: Verksamhetsanpassade IT-stöd - Designteori och metod, 2006.
- No 1286 **Robert Kaminski**: Towards an XML Document Restructuring Framework, 2006.
- No 1293 **Jiri Trnka**: Prerequisites for data sharing in emergency management, 2007.
- No 1302 **Björn Hägglund**: A Framework for Designing Constraint Stores, 2007.
- No 1303 **Daniel Andreasson**: Slack-Time Aware Dynamic Routing Schemes for On-Chip Networks, 2007.
- No 1305 **Magnus Ingmarsson**: Modelling User Tasks and Intentions for Service Discovery in Ubiquitous Computing, 2007.
- No 1306 **Gustaf Svedjemo**: Ontology as Conceptual Schema when Modelling Historical Maps for Database Storage, 2007.
- No 1307 **Gianpaolo Conte**: Navigation Functionalities for an Autonomous UAV Helicopter, 2007.
- No 1309 **Ola Leifer**: User-Centric Critiquing in Command and Control: The DKExpert and ComPlan Approaches, 2007.
- No 1312 **Henrik Svensson**: Embodied simulation as off-line representation, 2007.
- No 1313 **Zhiyuan He**: System-on-Chip Test Scheduling with Defect-Probability and Temperature Considerations, 2007.
- No 1317 **Jonas Elmqvist**: Components, Safety Interfaces and Compositional Analysis, 2007.
- No 1320 **Håkan Sundblad**: Question Classification in Question Answering Systems, 2007.
- No 1323 **Magnus Lundqvist**: Information Demand and Use: Improving Information Flow within Small-scale Business Contexts, 2007.
- No 1329 **Martin Magnusson**: Deductive Planning and Composite Actions in Temporal Action Logic, 2007.
- No 1331 **Mikael Asplund**: Restoring Consistency after Network Partitions, 2007.
- No 1332 **Martin Fransson**: Towards Individualized Drug Dosage - General Methods and Case Studies, 2007.
- No 1333 **Karin Camara**: A Visual Query Language Served by a Multi-sensor Environment, 2007.
- No 1337 **David Broman**: Safety, Security, and Semantic Aspects of Equation-Based Object-Oriented Languages and Environments, 2007.
- No 1339 **Mikhail Chalabine**: Invasive Interactive Parallelization, 2007.
- No 1351 **Susanna Nilsson**: A Holistic Approach to Usability Evaluations of Mixed Reality Systems, 2008.
- No 1353 **Shanai Ardi**: A Model and Implementation of a Security Plug-in for the Software Life Cycle, 2008.
- No 1356 **Erik Kuiper**: Mobility and Routing in a Delay-tolerant Network of Unmanned Aerial Vehicles, 2008.
- No 1359 **Jana Rambusch**: Situated Play, 2008.
- No 1361 **Martin Karresand**: Completing the Picture - Fragments and Back Again, 2008.
- No 1363 **Per Nyblom**: Dynamic Abstraction for Interleaved Task Planning and Execution, 2008.
- No 1371 **Fredrik Lantz**: Terrain Object Recognition and Context Fusion for Decision Support, 2008.
- No 1373 **Martin Östlund**: Assistance Plus: 3D-mediated Advice-giving on Pharmaceutical Products, 2008.
- No 1381 **Håkan Lundvall**: Automatic Parallelization using Pipelining for Equation-Based Simulation Languages, 2008.
- No 1386 **Mirko Thorstensson**: Using Observers for Model Based Data Collection in Distributed Tactical Operations, 2008.
- No 1387 **Bahlol Rahimi**: Implementation of Health Information Systems, 2008.
- No 1392 **Maria Holmqvist**: Word Alignment by Re-using Parallel Phrases, 2008.
- No 1393 **Mattias Eriksson**: Integrated Software Pipelining, 2009.
- No 1401 **Annika Öhgren**: Towards an Ontology Development Methodology for Small and Medium-sized Enterprises, 2009.
- No 1410 **Rickard Holmsmark**: Deadlock Free Routing in Mesh Networks on Chip with Regions, 2009.
- No 1421 **Sara Stymne**: Compound Processing for Phrase-Based Statistical Machine Translation, 2009.
- No 1427 **Tommy Ellqvist**: Supporting Scientific Collaboration through Workflows and Provenance, 2009.
- No 1450 **Fabian Segelström**: Visualisations in Service Design, 2010.
- No 1459 **Min Bao**: System Level Techniques for Temperature-Aware Energy Optimization, 2010.
- No 1466 **Mohammad Saifullah**: Exploring Biologically Inspired Interactive Networks for Object Recognition, 2011

No 1468 **Qiang Liu**: Dealing with Missing Mappings and Structure in a Network of Ontologies, 2011.

No 1469 **Ruxandra Pop**: Mapping Concurrent Applications to Multiprocessor Systems with Multithreaded Processors and Network on Chip-Based Interconnections, 2011.

No 1476 **Per-Magnus Olsson**: Positioning Algorithms for Surveillance Using Unmanned Aerial Vehicles, 2011.

No 1481 **Anna Vapen**: Contributions to Web Authentication for Untrusted Computers, 2011.

No 1485 **Loove Broms**: Sustainable Interactions: Studies in the Design of Energy Awareness Artefacts, 2011.

FiF-a 101 **Johan Blomkvist**: Conceptualising Prototypes in Service Design, 2011.

No 1490 **Håkan Warnquist**: Computer-Assisted Troubleshooting for Efficient Off-board Diagnosis, 2011.

No 1503 **Jakob Rosén**: Predictable Real-Time Applications on Multiprocessor Systems-on-Chip, 2011.

No 1504 **Usman Dastgeer**: Skeleton Programming for Heterogeneous GPU-based Systems, 2011.

No 1506 **David Landén**: Complex Task Allocation for Delegation: From Theory to Practice, 2011.

No 1507 **Kristian Stavåker**: Contributions to Parallel Simulation of Equation-Based Models on Graphics Processing Units, 2011.

No 1509 **Mariusz Wzorek**: Selected Aspects of Navigation and Path Planning in Unmanned Aircraft Systems, 2011.

No 1510 **Piotr Rudol**: Increasing Autonomy of Unmanned Aircraft Systems Through the Use of Imaging Sensors, 2011.

No 1513 **Anders Carstensen**: The Evolution of the Connector View Concept: Enterprise Models for Interoperability Solutions in the Extended Enterprise, 2011.

No 1523 **Jody Foo**: Computational Terminology: Exploring Bilingual and Monolingual Term Extraction, 2012.

No 1550 **Anders Fröberg**: Models and Tools for Distributed User Interface Development, 2012.

No 1558 **Dimitar Nikolov**: Optimizing Fault Tolerance for Real-Time Systems, 2012.

No 1582 **Dennis Andersson**: Mission Experience: How to Model and Capture it to Enable Vicarious Learning, 2013.

No 1586 **Massimiliano Raciti**: Anomaly Detection and its Adaptation: Studies on Cyber-physical Systems, 2013.

No 1588 **Banafsheh Khademhosseini**: Towards an Approach for Efficiency Evaluation of Enterprise Modeling Methods, 2013.

No 1589 **Amy Rankin**: Resilience in High Risk Work: Analysing Adaptive Performance, 2013.

No 1592 **Martin Sjölund**: Tools for Understanding, Debugging, and Simulation Performance Improvement of Equation-Based Models, 2013.

No 1606 **Karl Hammar**: Towards an Ontology Design Pattern Quality Model, 2013.

No 1624 **Maria Vasilevskaya**: Designing Security-enhanced Embedded Systems: Bridging Two Islands of Expertise, 2013.

No 1627 **Ekhiotz Vergara**: Exploiting Energy Awareness in Mobile Communication, 2013.

No 1644 **Valentina Ivanova**: Integration of Ontology Alignment and Ontology Debugging for Taxonomy Networks, 2014.