Toward Shared Understanding-

An Argumentation Based Approach for Communication in open Multi-Agent Systems

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree
of Doctor in Philosophy

By

Loredana Laera

Department of Computer Science

February 2008

Contents

Abstract x

Acknowledgements xii

I Background and Context 1

1 Introduction 2

1.1 Background and Motivation 2

1.2 Research Aims and Contributions 6

1.3 Thesis Structure 7

2 Autonomous Agents and Agent Communication 11

2.1 What is an Agent? 11

2.2 Multi-Agent Systems 14

2.3 Open Multi-Agent Systems 16

2.3.1 Agents on the Semantic Web 17

2.4 Agent Communication 20

2.4.1 Agent Communication Languages 21

2.4.2 Communication Protocols 24

2.4.3 Middle Agents 25

2.5 Conclusions 26

3 Ontologies 27

3.1 What is an Ontology? 27

3.2 The Components and Types of Ontologies 30

3.3 Representing an Ontology 32

3.3.1 Description Logics 33

3.3.2 OWL 34

3.4 Conclusions 37

4 Tackling Semantic Heterogeneity - Ontology Alignment **38**

4.1 Ontologies for Agent Communication 38

4.2 Communication using Heterogeneous Ontologies 41

4.3 Forms of Ontology Heterogeneity 43

4.3.1 Language Level Mismatches 43

4.3.2 Ontology Level Mismatches 44

4.4 Ontology Alignment 45

4.4.1 Alignment Methods 47

4.5 Agents and Heterogeneous Ontologies: Current Approaches 48

4.5.1 Ontology Alignment 48

4.5.2 Meaning Negotiation in Open MAS 50

4.6 Conclusions 52

II An Argumentation Based Approach for Communication in MAS **53**

5 Agents Arguing over Mappings **54**

5.1 Ontology Alignment in Open MAS - Issues and Challenges 54

5.1.1 Agent Autonomy over Ontology Alignments 55

5.1.2 Ontology Alignment is Context-Dependent 55

5.1.3 The Dynamics of Ontology Alignment 57

5.1.4 Diverse Approaches to Ontology Alignment 58

5.1.5 Ontology Alignment Reuse 58

5.2 Key Assumptions 59

5.3 A Mapping-oriented Argumentation Framework for Open MAS 61

5.3.1 Argumentation Framework 62

5.3.2 Arguments about Mappings 65

5.3.3 Agreed and Agreeable Alignments 70

5.4 An Illustrative Example 75

5.5 An Ontology for the Semantics of Arguments over Mappings 78

5.5.1 Agents 80

5.5.2 Alignment 80

5.5.3 Argument 81

5.6 Conclusions 82

6 Mapping-oriented Argumentation
- Architecture and Protocol **83**
6.1 A MAS Architecture for the Argumentation Based Approach 83
6.1.1 Communication Mechanism 86
6.2 Argumentative Agent 89
6.3 Argumentation Protocol 91
6.3.1 Conditions on the Argumentation Acts 95
6.4 Conclusions 97

III Evaluation and Application **98**

7 Implementation and Evaluation **99**
7.1 An Implementation for the Mapping-oriented Argumentation Framework 100
7.1.1 Alignment Module 101
7.1.2 Alignment API 102
7.1.3 Argumentation Module 104
7.1.4 Agents 105
7.2 Empirical Evaluation 107
7.2.1 Experimental Setup 108
7.2.2 Experiments and Results 111
7.3 Further Experiments in the Web Services Domain 116
7.4 An Application in Ontology Engineering Process for Multi-Agent Systems 122
7.4.1 A Brief Introduction to Ontology Engineering for MAS 122
7.4.2 Using the Mapping-oriented Argumentation framework into DINO 124
7.4.3 A Working Through Example in an eHealthCare Scenario 126
7.5 Conclusions 130

IV Synopsis **133**

8 Conclusions and Future Work **134**
8.1 Review of Contributions 134
8.2 Future Work 137

V Appendicies **139**

A OWL – Web Ontology Language **140**

A.1 OWL Lite Synopsis 140
A.2 OWL DL Synopsis 141

B Argumentation Ontology **142**

C An Overview on Argumentation in MAS **150**
C.1 Argumentation Theory 150
C.2 Argumentation for Agent Communication 151
C.3 Argumentation-Based Negotiation 152

D DINO – The Dynamic Ontology Lifecycle Scenario **154**
D.1 Dynamic Integration of New Learned Knowledge in the DINO Framework 154
D.1.1 Ontology Learning Wrapper 154
D.1.2 Ontology Collaborative Development Portal 155
D.1.3 Ontology Reasoning/Management Wrapper 156
D.1.4 Ontology Difference Wrapper 157
D.1.5 Triple Sorter 157
D.1.6 Natural Language Suggestions Generator 158

Bibliography **160**

To my family with love and gratitude

List of Figures

2.1 The Semantic Web's Layered Architecture. 19
2.2 Basis Communication Model. 21
2.3 Layer Organization of a General Agent Communication Language. 23

4.1 Single Ontology Approach. 39
4.2 Multiple Ontology Approach. 40
4.3 Hybrid Ontology Approach. 40
4.4 Agent Sharing Worlds but Using Different Ontologies. 42
4.5 Pairwise Mappings between Ontologies. 43
4.6 Mappings to a Single Common Ontology. 43
4.7 Multiple Ontology Clusters with Inter-Claster Mappings. 43
4.8 Alignment between Two Ontologies (only the classes involved in mappings are displayed). 46

5.1 Layers of Ontology Alignment. 56
5.2 Mapping-Oriented Argumentation Architecture. 61
5.3 Example of Arguments. 67
5.4 Excerpts of O_1 and O_2 Ontologies. 75
5.5 Value-Based Argumentation Frameworks. 77
5.6 Argumentation Ontology 79

6.1 Open MAS Architecture. 86
6.2 Ontology Alignment Protocol. 88
6.3 Argumentative Agent Structure. 89
6.4 Argumentation Unit. 90
6.5 Argumentation Protocol for Two Agents. 94

7.1 Mapping-Oriented Argumentation Prototype. 101
7.2 Displaying Mappings. 102
7.3 Displaying Arguments. 102

7.4 Alignment Server. 105
7.5 Displaying Agreed Mappings. 106
7.6 GUI to Set Preferences, Ontologies and Threshold of an Agent. 107
7.7 F-measure with and without argumentation. 112
7.8 F-measure with and without argumentation for Ag_1. 113
7.9 F-measure with and without argumentation for Ag_2. 113
7.10 Comparison between F-measure with argumentation for both agents and for each single one. 114
7.11 (a) F-measures for Ag_1. (b) F-measures for Ag_2. 115
7.12 Plot demonstrating the effects of varying the preferences for the test case 301. 116
7.13 Processing time with and without argumentation. 117
7.14 Processing cost for the argumentation over a single mapping. 117
7.15 Communication cost for the argumentation over a single mapping. 118
7.16 Plots showings the comparison of our argumentation approach wrt. current alignment tools (OAEI'05). 119
7.17 Plots showings the comparison of our argumentation approach wrt. current alignment tools (OAEI'06). 120
7.18 Two ontologies from a service-oriented travel domain for Ag_1 and Ag_2. 121
7.19 Precision and Recall Measurements for Web Service Ontologies. 122
7.20 DINO Architecture. 125
7.21 A Text Sample and the Learned Ontology. 127
7.22 A Master Ontology Sample. 128
7.23 Candidate Mappings and Arguments. 129
7.24 Negotiated Mappings. 129

D.1 Dynamic Integration Scheme. 155

List of Tables

5.1 Argument Scheme for OWL Ontological Alignments (continued in Table 5.2.) 69

5.2 (continued) Argument Scheme for OWL Ontological Alignments. 70

5.3 Arguments For and Against the Mappings m_1, m_2, m_3, m_4, m_5 and m_6. 76

5.4 Preferred Extensions. 78

6.1 Communication Path between Agents. 85

7.1 Servives Provided by the Ontology Alignment Service. 107

7.2 Test Cases. 110

7.3 Ag_1 and Ag_2's Preferences for each Test. 110

7.4 Except of Arguments and Counter-Arguments 128

7.5 Extension Triples and the respective NL Suggestions 130

Abstract

Open distributed computing applications are becoming increasingly commonplace nowadays. In many cases, these applications are composed of multiple autonomous agents, each with its own aims and objectives. In such complex systems, communication between these agents is usually essential for them to perform their task, to coordinate their actions and share their knowledge. However, successful and meaningful communication can only be achieved by a *shared understanding* of each other's messages. Therefore efficient mechanisms are needed to reach a mutual understanding when exchanging expressions from each other's world model and background knowledge. We believe the de facto mechanisms for achieving this are *ontologies*, and this is the area explored in this thesis [88].

However, supporting shared understanding mechanisms for open distributed applications is a major research challenge. Specifically, one consequence of a system being open is the heterogeneity of the agents. Agents may have conflicting goals, or may be heterogeneous with respect to their beliefs or their knowledge. Forcing all agents to use a common vocabulary defined in one or more shared ontologies is, thus, an oversimplified solution, particularly when these agents are designed and deployed independently of each other.

This thesis proposes a novel approach to overcome vocabulary heterogeneity, where the agents dynamically negotiate the meaning of the terms they use to communicate. While many proposals for aligning two agent ontologies have been presented in the literature as the current standard approaches to resolve heterogeneity, they are lacking when dealing with important features of agents and their environment. Motivated by the hypothesis that ontology alignment approaches should reflect the characteristics of autonomy and rationality that are typical of agents, and should also be tailored to the requirements of an open environment, such as dynamism, we propose a way for agents to define and agree upon the semantics of the terms used at run-time, according to their interests and preferences. Since agents are autonomous and represent different stakeholders, the process by which they come to an agreement will necessarily only come through negotiation. By using argumentation theory, agents generate and exchange different arguments, that support or reject possible mappings between vocabularies, according

to their own preferences. Thus, this work provides a concrete instantiation of the *meaning negotiation* process that we would like agents to achieve, and that may lead to *shared understanding*. Moreover, in contrast to current ontology alignment approaches, the choice of a mapping is based on two clearly identified elements: (i) the argumentation framework, which is common to all agents, and (ii) the preference relations, which are private to each agent.

Despite the large body of work in the area of semantic interoperability, we are not aware of any research in this area that has directly addressed these important requirements for open Multi-Agent Systems as we have done in this thesis.

Acknowledgements

First of all, I would like to express my gratitude to my supervisors, Dr. Valentina Tamma and Prof. Trevor Bench-Capon for their introduction to the areas of Multi-Agent Systems, Semantic Web and Argumentation and for their support during my PhD research.

This work has also been possible thanks to the financial support of the Knowledge Web (FP6- IST 2004-507482), PIPS (FP6-IST 2004-507019) and Ontogrid (FP6-511513) EU IST projects.
I would also like to thank all the people involved in these projects that made contributions towards this work. In particular I would like to thanks Jérôme Euzenat for sharing his insightful knowledge in the field of ontology interoperability.

A very special thank you goes to all my friends and colleagues at the University of Liverpool. In particular, I wish to thank to all members of the Semantic Web Lab: Ian Blacoe, Luigi Iannone, Ignazio Palmisano, Paul Doran and Ben Lithgow Smith. It was a great pleasure to working and socialise with you all.
Thanks also to Tim Miller and all the players of departmental friday football. I really enjoyed it!

A particular thank you goes to my family and Ian's family: my parents Angelo and Giovanna, my brother Gennaro, his wife Rosa, my sister Paola, her husband Renzo, Ian's parents Peter and Diana, Helen and Chris and my fantastic nieces Elena, Greta and Amelie and my nephew Logan. I cannot express how very grateful I am to them for the love and support they have given to me and for the opportunities that I have been presented with which stem from this.

Last, but not least, I would like to thank Ian, for your love, thoughtfulness, and support (and patience) through all these years.

Part I

Background and Context

Chapter 1

Introduction

"Communicate, communicate, and then communicate some more."

-Bob Nelson

The aim of this thesis is to investigate how to achieve shared understanding between agents in an open Multi-Agent System, using ontologies, in a way that conforms to the characteristics of agents and is tailored to the requirement of their environment.

The rest of this chapter is structured as follows. We start with an extended outline of the background and scope of the thesis in Section 1.1, focusing on the open problems and challenges that motivated us. In this context, we summarize the main goals and point out the main contributions of the thesis in Section 1.2. In Section 1.3, the structure of this thesis is presented as a guide for readers.

1.1 Background and Motivation

A wide variety of computer applications are open distributed systems in which components are spread throughout a network, in a decentralised control regime, and are often subject to continuos change throughout the system's lifetime. Examples include the Grid [62], Peer-to-Peer computing [103], the Semantic Web [17] and pervasive computing environments [106]. Such open distributed systems are typically composed of various stakeholders, each with their own, possibly conflicting, interests and preferences. Therefore, there is a need to have autonomous components, that represent these stakeholders, and act and interact in flexible ways in order to achieve their design objectives in uncertain and dynamic environments. Given this, agent paradigm has been emerged as the natural computation model for such systems [68]. More specifically, the agent paradigm allows the decomposition of large, complex, and distributed systems into a number of autonomous entities that can interact with each other in order to

achieve their individual objectives [68]. Thus, open distributed systems can be modelled as open multi-agent systems [91], that are composed of autonomous agents that can join and leave at any time, undergo continual change and can be developed by different parties. In such systems, it is usually not possible for agents to possess complete a priori information about other agents within the system, simply because such information will initially be unavailable or too costly to obtain. As was pointed out by Hewitt and de Jong [60] the only thing that the components of an open system have in common is their ability to *communicate*.

Achieving successful communication in open multi-agent systems requires that the agents share a standard agent communication language, e.g., KQML [48] and FIPA ACL [1]. Citing Luck, McBurney and Preist [79]:

> *"Powerful agents need to be able to communicate with users, with customers, with system resources and with each other if they are to cooperate, collaborate, negotiate and so on. Common agent languages hold the promise of diverse agents communicating to provide more complex functions across the networked world. Indeed, as agents grow more powerful, their need for communication increases."*

However, sharing a common communication language is not enough. Agents should also share a common understanding of the vocabularies to be used to represent the knowledge being communicated. Thus, agents need a language for representing the knowledge, i.e. the *content language*. It has been long accepted [54, 114] that a such particular role is designed to be played by a common *ontology*, which is expected to provide the definitions of the vocabularies used by agents to describe the world. A common ontology would guarantee that any concept, object, or entity has a uniform meaning across agents even if different agents use different names to refer to them. Therefore, an ontology will provide a common *understanding* - the conceptual framework of knowledge - to be shared between agents.
The use of explicit ontologies in the multi-agent systems research community has become widely recognized as having great benefits for the integration of disparate and distributed information sources to form an open, extensible and loosely coupled agent system. Industry initiatives such as OMG Model Driven Architecture[1], UDDI[2] and BizTalk[3] provide both an opportunity and a challenge to the agent research community to demostrate the benefits that ontology and knowledge-level communication bring and the potential business applications of their technology. In addition, the development of research fields such as the Semantic Web (SW), will bring a renewed focus within multi-agent research community on the development of explicitly modelled ontologies and tools and the infrastructure to support their use. For this reason, the Semantic Web is the enviroment we focus on primarily in this research.

Achieving shared understanding for open distributed applications is a major research challenge.

[1] http://www.omg.org/mda/.
[2] http://www.uddi.org/.
[3] http://biztalk.org.

Specifically, in an open multi-agent system, agents are often committed to different ontologies which are designed for different purposes and owned by different stakeholders, who profit from their interactions in the virtual world. The pre-existence of such a shared vocabulary cannot be guaranteed or even expected, not only because it would result in assuming a standard content vocabulary for all agents (and thus violate the dynamics of open environments); but also because it does not take into account the conceptual requirements of agents that could appear in future. Thus, as pointed out by Uschold [122], a fundamental problem to take into consideration in agent communication is that caused by the *heterogeneity* of the underlying knowledge sources, or ontologies. Moreover, apart from different ontologies, each agent has its own set of goals, which this can lead to self-interested behaviour, and its own business logic.

Consider for instance electronic market places in which agents from different origins are put together to interact with each other in business environments. Agents in such systems might have been implemented in different institutions by different programmers, and will typically employ different ontologies to represent information, such as different currencies to denote prices and so on.

Against this background, this thesis deals with the problems of *how to achieve, in a dynamic fashion, shared understanding that conforms to the characteristics of agents and is applicable to open and dynamic environments, where it is expected that agents' ontologies are different.*

Here, shared understanding between agents is concerned with having a common meaning attached to the elements of an entity as well as to their relationships in their ontologies. The semantics of this common meaning should be sufficient for the agents to carry out their tasks, even when their ontologies are different.

Typically, the interoperation between these agents is based on the reconciliation of their heterogeneous views, which is accomplished by aligning the diverse ontologies associated with the agents composing the system. An *alignment* between the ontologies is a set of mappings between the concepts, properties, and relationships in the agents' ontologies. Citing Sampson [105]:

> *"Ontology alignment is one prerequisite towards enabling interoperability between agents or services that use different ontologies. It is necessary that these agents align their ontologies to enable data exchange at the syntactic and semantic level".*

In the context of open multi-agent systems - where agents cannot have a priori knowledge about the ontologies of other agents in the system, and where these ontologies are often private - a middle component that provides the alignments is often indispensable to agent systems.

The use of external alignment services that agents can invoke to obtain and store possible mappings between two ontologies has been proposed by Euzenat in [42]. The mappings might originate from independent mapping engines that employ differing algorithms to calculate them.

However, the availability of such external services and the approaches to ontology alignments themselves provide only a partial solution to achieving shared understanding between open agents, since

they are lacking when dealing with important features of agents and their environment.

First, ontology mappings may only be suitable in a certain context. For a given context, agents might have different and conflicting perspectives; i.e. interests and preferences; on the acceptability of a candidate mapping, each of which may be rationally acceptable. This may be due to the subjective nature of ontologies, to the circumstances and the requirements of the alignment and so on. For example, an agent may be interested in accepting only those mappings that have linguistic similarities, since its ontology is too structurally simple to realise any other type of mismatch.

In addition, in most of the approaches to ontology alignment, the generation of ontology mappings is often a step-wise process that includes the definition of an initial alignment, the training of some examples, and that invariably involves some form of interpretation of preliminary results. Therefore, they are most effective when used to support semantic interoperation at design time, and in closed or partially open environments, where the actors involved are often already known, ontology changes are controlled and thus the alignments can be established before any agent interaction. However, in open environment the above conditions are not satisfied. In these systems, the communication pattern is highly dynamic, thus any decision on the acceptability of these mappings should be made dynamically (at run time), due to the fact that the agents have no prior knowledge of the existence or constraints of other agents.

Given these lacunae, the motivating hypothesis behind this thesis is that approaches to overcome the problem of communication between heterogeneous agents should also reflect the characteristics of autonomy and rationality that are typical of agents [141] and take into account the requirements of the environment. Thus, it is desirable for agents to have the ability to define and agree upon the semantics of the terms used at run-time, according to their interests and preferences. Since the agents are autonomous and represent different stakeholders, a way by which they can come to an agreement is through a negotiation process [3].

For this reason, the main contribution of this thesis is to advocate an approach in which agents dynamically negotiate the meaning of the terms they use to communicate. Particularly because the agents are autonomous and cannot be assumed to be benevolent, agents must try to convince each other, by arguments, to accept one mapping rather than another, and negotiation is thus critical for managing such inter-agent dependencies. We term the framework *mapping-oriented argumentation*, since it involves negotiation about the mappings between vocabularies, by argumentation theory. In our argumentation approach, agents generate and exchange different arguments, that support or reject possible ontology mappings, according to its preferences and interests. The set of potential arguments are clearly identified and grounded on the underlying ontology language, and the types of mappings supported by any such arguments are clearly specified. A value-based argumentation framework [15] is used to express agents' preferences between the categories of arguments built on the types of mappings and which are specific to the matching task and on the ontology semantics.

Thus, this work provides a concrete instantiation of the *meaning negotiation* process that we would like agents to achieve, and that may lead to *shared understanding*.

1.2 Research Aims and Contributions

In this section we summarize the aims and objectives of this thesis and the contributions to the state of the art that were made to achieve them.

The intent of this thesis is *to provide solutions that enable mutually consistent understanding of shared information between agents in an open and distributed environment.*

To this end, we set out to achieve the following objectives:

- A prerequisite for improving the state of the art in semantic interoperability in MAS is to have a good understanding of this process. In particular, it is important to identify the requirements that these approaches should fulfill.

- Develop a comprehensive model for open multi-agent systems that overcomes the problems concerning the communication between agents committed to different ontologies; the model needs to reflect the characteristics of autonomy and rationality of agents; and should be tailored to the requirements of an open environment.

According these objectives, the main research direction investigates an approach in which agents dynamically negotiate the meaning of the terms they use to communicate. Thus, the novel contributions of the thesis are the following:

- We identify the requirements for achieving shared understanding between agents in open MAS. Despite the large body of work in the area of semantic interoperability in MAS and SW, few efforts are directed towards identifying requirements for these environments. We are only aware of the requirements stated in [125]. Based on our experiences when working with open and distributed systems, we identified a set of requirements that the mechanisms for semantic interoperability should fulfil. While non-exclusive, our list of requirements complements the requirements brought forward by the community so far.

- We extend the notion of *reaching agreement* through automated argumentation-based negotiation (i.e. without human intervention) over ontology mappings between heterogeneous agents. Thus, rather than assuming that agents are restricted to using traditional mapping algorithms that are only effective when used at design time, we argue that an ontology mapping should be decided at run-time, so that they can be tailored to reflect the agent's context and tasks.

- We present a way by which agents can express their preferences over the types of mappings they use when aligning the ontologies. The approach uses an argumentation framework that allows

the agents to reach an agreement on those mappings that are *mutually acceptable*, because they are mappings that cannot be refuted by any agents involved in the negotiation. In contrast to current ontology alignment approaches, the choice of alignment is based on two clearly identified elements: (i) the argumentation framework, which is common to all agents, and (ii) the preference relations which are private to each agent.

- We implement a proof of concept of an argumentation-based mapping framework in order to demonstrate its applicability and effectiveness in inter-agent communication.

- We demonstrate the benefits of using argumentation-based negotiation for choosing mappings and show that they enable agents to communicate more efficiently than using the current alignment approaches.

Despite the large body of work in the area of semantic interoperability, very few efforts have directly addressed these important requirements for open MAS as we did in this thesis.

Note that the work presented in this thesis is confined to a problem in Artificial Intelligence (AI), specifically: the supporting of interoperability in distributed and open systems. Even though we draw on concepts from agent negotiation and the theory of argumentation, we do not make a contribution to these areas.

1.3 Thesis Structure

This thesis describes the techniques and tools developed in meeting the aims described previously. The thesis is divided into 4 parts, which are further divided into 8 chapters and there are appendices. Part I presents the background and the context of our research, which is relevant to the contributions we present in this thesis. Part II describes the proposed argumentation framework and explains how it is going to be applied for distributed heterogeneous agents settings. Part III describes the implemented tool and evaluates its applicability. Finally, Part IV underlines the main results, concludes the thesis and discusses future work. The appendices provides technical information.

A more detailed explanation of the structure of this thesis is depicted as follows:

Chapter 1 is this chapter in which we have defined the motivation of this thesis and the background which on this thesis is based.

Chapter 2 addresses the wide field of Multi Agent Systems. We review the definitions of Agent and Multi-Agent System and discuss their main features. It gives an overview of agent interoperation in open and dynamic environments, while emphasizing the relation with the area of the Semantic Web. It further discusses agent communication, introducing the reader to the connection between agents and knowledge sharing in order to understand the role of ontology for the next chapter.

Chapter 3 contains an introduction to the ontologies and the most important related concepts that have been considered in this work. First, it reviews the different meanings that the term "ontology" takes in Artificial Intelligence and Computer Science. Then, it discusses suitable representations of ontologies.

Chapter 4 outlines in details the problem of semantic heterogeneity between different ontologies. First, it introduces the role of ontology for agent communication, introducing the problem of semantic heterogeneity in open and dynamic Multi-Agent Systems. Then, it discusses how ontologies may differ by a classification of the different types of mismatches. Further on, it sketches *ontology alignment* as the current proposal in the literature for achieving interoperability. Ontology alignment is based on the ability to find the set of mappings between the concepts, properties, and relationships in the ontologies. It briefly surveys a general classification of different ontology alignment methods. Finally, it reviews a selection of proposed solutions, showing how the importance of reconciling heterogeneous ontologies has evolved in recent years, with a particular emphasis on their applicability in open MAS.

Chapter 5, following the considerations from the previous chapter, introduces and motivates our argumentation based approach. While many proposals for aligning ontologies have been presented in the literature, most of them do not address important requirements and issues inherent for the agent community. The resulting alignment may not be satisfactory to the agents, and thus can become the object of further negotiation between them. The chapter proffers the use of argumentation over ontology mappings as an approach that may lead to shared understanding between heterogeneous agents. It introduces the arguments and how to reason about these arguments in an specific argumentation framework - the Value-Based Argumentation framework, designed to accommodate participants with different preferences. It defines the notion of agreed and agreeable alignments for agents, and proposes a procedure to find them. The chapter concludes with the description of an argumentation ontology used to represent the arguments.

Chapter 6 discusses in detail the MAS architecture, the argumentative agents and their capabilities for argument generation to support our argumentation framework. Moreover, it proposes a protocol for evaluating the acceptability of a mapping.

Chapter 7 discusses the implementation of the proposed framework, followed by the evaluation and applicability of the framework. A number of experiments are presented and discussed. The implementation is evaluated on pre-defined test cases by comparing the results of the methods to human judgement. The chapter concludes with a presentation of a pilot study. The pilot study describes the application of our framework in a specific scenario: the *lifecycle for Ontology development process*.

Chapter 8 presents some conclusions and identifies some open issues that are believed to deserve future work.

The thesis also includes four additional elements as appendices. Appendix A provides a summary of the OWL Lite and OWL DL language features. Appendix B shows the OWL version of the Argumentation Ontology, that models arguments for and against potential ontology mappings. Appendix C presents an outline of argumentation in MAS. Appendix D presents more details of the DINO framework, its architecture, and functional requirements.

Some of the content presented in this thesis has been previously published, or accepted for publication, as follows:

- V. Novacek, L. Laera and S. Handschuh. *Aiding the Data Integration in Medicinal Settings by Means of Semantic Technologies.* In Proceedings of the International Workshop on Making Semantics Work for Business at ESTC 2007, Vienna, Austria, 2007.

- V. Novacek, L. Laera and S. Handschuh. *Semi-Automatic Integration of Learned Ontologies into a Collaborative Framework.* In Proceedings of the First International Workshop on Ontology Dynamics at ESWC'07. June 2007 Innsbruck, Austria.

- V. Novacek, L. Laera and S. Handschuh. *Dynamic Integration of Medical Ontologies in Large Scale.* In Proceedings of the First International Workshop on Health Care and Life Sciences Data Integration for the Semantic Web at WWW20007. May 8, 2007 Banff, Alberta, Canada.

- L. Laera, I. Blacoe, V. Tamma, T. Payne, J. Euzenat and T. Bench-Capon. *Argumentation over Ontology Correspondences in MAS.* In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 07). Honolulu, Hawaii, pages 1285-1292, 2007 .

- V. Novacek, S. Handschuh, L. Laera, D. Maynard and M. Voelkel. *Dynamic Ontology Lifecycle Scenario in Translational Medicine.* In Proceedings of the Fifth European Conference of Computational Biology (ECCB 2006) - Book of Abstracts, pages 31-35. Eilat, Israel. 2006

- L. Laera, V. Tamma, J. Euzenat, T. Bench-Capon, and T. Payne. *Agents arguing over Ontology Alignments.* In Proceedings of the Fourth European Workshop on Multi-Agent Systems (EUMAS 06), Lisbon, Portugal.

- L. Laera, V. Tamma, J. Euzenat, T. Bench-Capon, and T. Payne. *Arguing over Ontology Alignments.* In Proceedings of the Workshop on Ontology Matching (OM-2006) at ISWC'06. Athens, GA, USA, November 2006.

- L. Laera, V. Tamma, J. Euzenat, T. Bench-Capon, and T. Payne. *Reaching Agreement over Ontology Alignments.* In Proceedings of the Fifth International Semantic Web Conference (ISWC 2006), LNCS 4273, pages 371-384, Athens, GA, USA, November 2006.

- L. Laera, V. Tamma, T. Bench-Capon, and J. Euzenat. *Agent-based Argumentation for Ontology Alignment.* In Proceedings of the International Workshop on Computational Models of Natural Argument (CMNA 2006) at ECAI'06. Riva del Garda, Italy, September 2006.

- L.Laera. *Resolving Semantic Heterogeneity in Rule-Enabled Ontologies.* In Proceedings of KnowledgeWeb PhD Symposium 2006 (KWEPSY2006) Budva, Montenegro, 17th June 2006.

- L. Laera and V. Tamma. *The Hitchhiker's Guide to Ontology Editors.* AgentLink Newsletter Issue 18.

- L. Laera, V. Tamma, T. Bench-Capon. *SweetProlog: A system to integrate Ontologies and rules.* In Proceedings of the Fourth International Semantic Web Conference (ISWC 2005) (Poster), Galway, Ireland, November 2005.

- L. Laera, V. Tamma, T. Bench-Capon, and G. Semeraro. *SweetProlog: A System to Integrate Ontologies and Rules.* In Proceedings of the third Workshop on Rules and Rule Markup Languages for the Semantic Web, LNCS 3323, pages 188-193, Hiroshima, Japan, November 2004.

Chapter 2

Autonomous Agents and Agent Communication

In this chapter we introduce the fundamental principles behind Agents, Multi-Agent Systems and Agent Communication. It is not our intention to give a complete overview of the wide field. Rather, we will restrict ourselves to those aspects that are relevant for the current investigation.

First, we review the definitions of Agent and Multi-Agent System in Sections 2.1 and 2.2. We focus on Open Multi-Agent Systems in Section 2.3. This section is especially relevant for this thesis as it deals with agent interoperation in large scale open and distributed systems such as the Semantic Web. Section 2.4 discusses agent communication and introduces the reader to the connection between agents and knowledge sharing in order to understand the role of ontologies discussed in the next chapter.

2.1 What is an Agent?

The increasing complexity of distributed computer systems has led researchers to utilise various tools of abstractions in order to improve the software engineering process. However, the requirements of an increasing number of computing scenarios go beyond the capabilities of traditional computer science and software engineering abstractions. In particular, four main features distinguish future software systems from traditional ones:

1. *Situatedness*. Software components execute in the context of an environment, which they can influence and be influenced by;

2. *Openness*. Software systems are subject to decentralised management and can dynamically change their structure, and thus be *heterogeneous*;

3. *Locality in control.* Software systems components represent autonomous and proactive loci of control;

4. *Locality in interactions.* Despite living in a fully connected world, software components interact according to local (geographical or logical) patterns.

Examples of these distributed systems include the Grid [62], peer-to-peer computing [103] and the Semantic Web [17]. Their characteristics have led to the emergence of a new paradigm in computing: the agent paradigm. Indeed, an increasing number of computer systems are being viewed in terms of multiple, interacting autonomous agents. This is because the multi-agent paradigm offers a powerful set of metaphors, concepts and techniques for conceptualising, designing, implementing and verifying complex distributed systems.

So, what are these agents and what is all this fuss about?

During the past years, quite a few researchers have attempted to define the notion of an agent [49]. Indeed, as Shoham points out in [108], there are so many diverse definitions of "agent" that the term is almost meaningless without reference to a particular concept of agent.

In this section, we review the most relevant definitions proposed in the literature, and we will discuss the main features of an agent that are relevant for the purpose and context of this work. The section ends with our definition of agent that will be used throughout in this thesis.

The most influential definitions are probably the *"weak notion"* and the *"strong notion"* of an agent, defined by Wooldridge and Jennings in [141]. The weak notion defines an agent as a hardware or software based computer that is a *autonomous*, *reactive*, *pro-active* and has *social ability*. The most important feature that differentiates an agent system from other computer systems is the *autonomy*: an agent is able to set its own goals, based on its motivations [28], and to choose a way to achieve those goals. Informally, autonomous action involves carrying out action without human intervention and autonomous decision involves adapting their responses (e.g., accept or reject incoming messages) to dynamically changing environment conditions and other agent's behaviors. This implies that an agent system is expected to have more robust behaviour such as recovery from failure and more reliable behaviour such as coping with an uncertain and changing environment. Autonomy is essential for *situated* agents, that is agents situated in an environment shared by other agents; which are able to sense their environment and carry out actions in that environment. Examples of environments in which agents may be situated include the physical world, a user, a society of agents or the Semantic Web.

The properties *reactivity*, *pro-activeness* and *social ability* together determine the degree of flexibility of an agent:

- *Reactivity* of an agent refers to its ability to perceive their environment; and to react, presumably in a sensible way, to unexpected situations arising in its environment;

- While an agent is, by its autonomy, able to set its own goals, its *pro-activeness* ensure that it will actually do so. Thus, an agent takes the initiative and create opportunities to pursue its goals instead of merely reacting to its environment;

- An agent's *social ability* enables it to interact and cooperate with other agents in its environment, in order to satisfy its design goals. The social ability goes beyond exchanging binary code, but is more related to the human world, in terms of sharing resource and goals, and thus it includes situations where an agent can negotiate and cooperate with others.

The reader may have noticed that the weak notion of agent applies to human beings to a large extent: humans are generally assumed to be autonomous, situated in their society and socially able. This is of interest to our investigation since the same concept can be theoretically used both for the analysis of (exiting) human procedures and the analysis of automated systems consisting of software agents[1].

Wooldridge and Jennings have also defined, in the same paper, a stronger notion of agent:

> "*.. an agent to be a computer system that, in addition to having the properties identified above, is either conceptualised or implemented using concepts that are more usually applied to humans. For example, it is quite common in AI to characterise an agent using mentalistic notions, such as knowledge, belief, intention, and obligation.* "

Under the strong notion, an agent is ascribed mental states, including beliefs, knowledge, intentions, goals, commitments and obligations. The behaviors of the agent are described using these mentalistic notions. Depending on the application only a subset of these will be employed.

In its more common and simpler version, an agent can be thought of as a state machine [66]. Every state describes the situation in which the agent currently is in each time and the agent decides the action to fulfill on the basis of its current state. The execution of an action causes a state transition. The actions can be *inside actions* or *communicative acts* [102]. The *inside actions* are directed to modify the state of the same agent, to interact with not-agent software (the so-called sensors and actuators), and to interact with human consumers. The *communicative acts* are related to the messages exchanged with the other agents (dispatch or receipt of messages). As a consequence, to describe an agent means to describe its state machine, i.e., its states, its actions and the messages, which it can receive and send. An agent, in its more general version, is able to work in parallel, i.e., it can exchange messages with more than one agent simultaneously. This slightly complicates the formal description of an agent, but the intuition is substantially unchanged.

Before concluding this section, it is worth presenting the definition of agent of Frankling and Graesser [49]:

[1]Chapter 7 shows that the main ideas presented in this thesis can be equivalently applied to humans and software agents.

> *"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."*

This definition emphasises situatedness and continuity of an agent and requires that there is some kind of feedback: the action of an agent should affect future observations of the agent. Both continuity as well as the feedback requirement help to distinguish agents from 'ordinary programs'.

The definitions presented here have shown that the term "agent" can be understood differently depending on the focus of research and the context. However, in this thesis, the definition of an agent that we will throughout is adapted from Wooldridge and Jennings [141], which ensures the ability of the agent to communicate, negotiate and pursue a specific goal:

> *"An agent is a computer system, situated in an heterogeneous computing environment, capable of flexible autonomous decisions in this environment in order to meet its design objectives, and able to interact and negotiate with other agents according to some interaction mechanism as a means by which agents are able to exchange information, resolve their conflicts and reach agreements. "*

2.2 Multi-Agent Systems

Agents are abstraction tools and as such they are increasingly seen as a most well established and promising approach to develop complex distributed computing systems, known as Multi-Agent Systems (MAS). A Multi-Agent System is defined as a loosely coupled network of agents that work together as a society aiming at solving problems that would generally be beyond the reach of any individual agent [38]. This definition is the general definition that is usually given in the major part of articles and inherited from the field of distributed planning. More recently, the term "Multi Agent System" has come to have a more general meaning, and it is now used to refer to all types of systems composed of multiple (semi-) autonomous components [68]. In general, the MAS approach advocates decomposing problems in terms of autonomous agents that can engage in flexible, high level interactions, and this way of decomposing a problem aids the process of engineering complex systems [67].
According to [117], the main characteristics of a Multi-Agent System are that:

- each agent has incomplete information or capabilities for solving the overall problem tackled by the system and, thus, has a limited viewpoint;
- there is no global control in the system: the collective behaviour is the result of social rules and interactions and not of a supervising central authority;

- resources are *decentralised*: resources needed for the completion of the tasks assigned to the system are divided and distributed.

Thus leads to the recognition that a MAS is not a simple set of agents put together in a common environment, but a real organization with social rules and interactions allowing cooperation and collaboration to resolve problems that centralized systems (as intelligent as they can) would not have resolved.

But *what advantages does MAS offer and in what circumstances is it useful?* In agreement with Sycara [117], the most important reason to use MAS is to solve problems that are too large for a centralized agent because of resource limitations or the sheer risk of having one centralized system that could be a performance bottleneck or could fail at critical times. Moreover, different and distributed organizations with different (possibly conflicting) goals, proprietary information and expertise might benefit from a multiagent system in order to handle their interactions. However, there are also several reasons to use MAS even in domains that could conceivably use systems that are not distributed. Having multiple agents could speed up a system's operation by providing a method for parallel computation, for instance. Furthermore, the parallelism of MAS can help deal with limitations imposed by time-bounded reasoning requirements.

Applications built upon MAS currently cover a wide number of areas including electronic commerce, information management, health care, process control, electronic games, manufacturing , and so on. For example, their application within industrial scenarios include tasks such as [13]:

- automated trading in online marketplaces;
- simulation and training applications in defence domains;
- network management in utilities networks;
- user interface and local interaction management in telecommunication networks;
- schedule planning and optimisation in logistics and supply-chain management;
- control system management in industrial plants (such as steel works);
- simulation modeling to guide decision-makers in public policy domains.

A common feature for the success of these applications is that agents have to interact with other agents to perform their tasks.

Before moving on to the details of how agents interact, first we need to distinguish between two main classes of MAS [144]:

- distributed problem solving systems in which the agents are explicitly designed to cooperatively achieve a given common goal in a benevolent fashion;

- open systems in which agents are not necessarily co-designed to share a common goal and can dynamically leave and enter the system and may not be benevolent.

In the former case, all agents are known a priori and benevolent to each other and therefore, they can trust one another during interactions. This class is also known as *closed Multi-Agent System* in which the set of agents is predefined by the entity that sets up and controls the system. However, the real utility of Multi-Agent Systems lies in the potential of *open* agent systems.

In order to identify the problem areas handled in these systems, which is one of the requirements of this thesis, the next step is to further investigate open MAS. This is the subject of the next section, which establishes the characteristics of these systems, and examinates how these characteristics are suited to the Semantic Web environment.

2.3 Open Multi-Agent Systems

In this section we briefly summarize the main features of agents in open MAS, before turning to a detailed description of a specific open enviroment: the Semantic Web.

Open MASs are defined as systems in which agents can freely join and leave at any time and where agents operate independently towards their own goals.
In these systems, not all relevant properties of the agents are known at design time (e.g., agent architecture, personal beliefs and goals), and their properties might be (intentionally) hidden or dynamic (black-box character). Agents can change over time and, usually they exhibit *self-interested* behavior. Here, the notion of self-interest means that an agent will act in a manner which maximises its own utility given its own goals. In addition, agents are often not designed and developed by the same group, nor do they represent the same stakeholders, and may follow different policies or objectives. All these features imply that an open MAS has to deal with the following three possible modifications of its composition:

- Addition of an agent to the Multi-Agent System. For example, an agent can be added to provide new capabilities or for supporting a complex task (in this case a new agent could be created to solve one or more subtasks of a given main task).

- Removal of an agent from the Multi-Agent System. For example, an agent can leave the community when its goals have been satisfied.

- Evolution of an agent in the Multi-Agent System. For example, an agent gains and shares new capabilities and/or unregisters some of its old capabilities (for instance, because obsolete).

It is worth noting that all these modifications are alterations of the Multi-Agent System which have a great influence on the possibility of interoperation between agents. Indeed, the addition of an agent

to a Multi-Agent System creates new possibilities of interoperation between this agent and some other agents in the system, whereas the removal of an agent causes some of these possibilities to become obsolete. Moreover, when an agent evolves, its possibilities of interoperation also change because it has a different set of capabilities and different needs. Thus, a consequence of a MAS being open is the *heterogeneity* of these agents. Heterogeneity means agents can rely on different architectures, programming languages, or mechanisms to take part into the system. For example, agents may differ on their internal architecture since they incorporate distinct properties [65]. Agents may be heterogeneous with respect to their beliefs, as discussed by Lebbink in [75]. Moreover, agents may have heterogeneous knowledge, which is the subject of this thesis. Agents may also have heterogeneous representation of this knowledge, which is not discussed in this thesis but the interested reader is referred to [29] for an approach to translation with knowledge preservation.

The introduction of open systems of this kind is thus likely to lead to a new set of problems relating to the effects of interactions between them. In fact, given the high heterogeneity and dynamic structure of a open MAS, a number of issues have to be addressed. In this thesis we specifically address one of them: how to ensure the interoperability of agents with different knowledge.

The next section further analyses a specific open and distributed environment: the Semantic Web[2].

2.3.1 Agents on the Semantic Web

The Semantic Web (SW) is an open and complex distributed computing environment and, having discussed the use of agents in open and distributed environments, it can be said that multi-agent systems exhibit characteristics that are well suited to the domain of the SW [17].

The Semantic Web proposes an evolution of the current web, from a web of documents to a distributed and decentralised, global knowledge-base, in which *"information is given well-defined meaning, better enabling computers and people to work in cooperation"* [17].

The Semantic Web extends the current web, making available a semantically enriched layer of machine accessible content on top of the existing infrastructure. This environment will provide a much richer habitat for agents than the current World Wide Web (WWW). This shift in intended use is reflected by the encoding of Web pages using visual markup languages (such as HTML), designed for use by human users, and represented using machine-processable languages. Agents can access and exploit this knowledge by using different reasoning mechanisms, and other entailed knowledge can be inferred and utilised to offer services.

[2] Although the Semantic Web is a prominent example of an open environment, we believe that many of the insights we develop will equally well transfer to other open systems, such as peer-to-peer computing, pervasive computing, the Grid, etc.

In order to show the promise of agents and their impact within such a knowledge-rich and large-scale, open and distributed environment, Berners-Lee and colleagues give in [17] the following every-day life example. *The siblings Pete and Lucy need to arrange medical treatments for their mother; each of them has a personal agent that performs tasks automatically. In order to fix appointments at the doctor and organize transportation, Pete's and Lucy's agents interact and coordinate their actions; they determine a suitable doctor and arrange appointments automatically, they elaborate transportation plans with regard to Pete and Lucys time schedules, and exchange information on several issues. All information applied by the agents is exchanged over the Internet; also, computational facilities as well as information repositories used by the agents are accessed over the Internet.*

The Semantic Web can then be viewed, in part, as a worldwide infrastructure for automated communication support. Agents interact with each other to achieve complex objectives, where automated support along with information interchange and processing over the Web is envisioned.

Thus, the 4 key concepts for realizing this vision of the SW are:

- **Languages**, that are suitable for the open nature of the Web and allow powerful and efficient knowledge representation;
- **Ontologies** to conceptualize shared views upon knowledge domains, as well as the means to manage them;
- **Agents** to access information and to autonomously perform actions, often on behalf of their users; and
- **Reasoning mechanisms** that allow agents to access and utilise the entailed knowledge represented within ontologies.

These aspects are illustrated in Figure 2.1 that shows the layered infrastructure of the Semantic Web[3]. This infrastructure is based on six pillars:

- The ability to univocally identify all the tokens of the universe of discourse: URIs - Universal resource identifiers. URIs permit to refer to any token irrespective of where it is published;
- A common data model that allows the description of and access to information: RDF - Resource description framework [59];
- A number of possibly interconnected conceptual models that define the vocabularies and provide the underlying semantics to the data model: Ontologies expressed in formalisms such as RDF(S) or OWL [96] (see Chapter 3);

[3]www.w3.org/2005/ 12/31-daml-final.html

- Some reasoning mechanisms that allow to reason about ontologies, to establish the consistency and correctness of specific concepts or to infer new concepts that are not explicitly stated. Rules are examples of complex axioms to perform inferences;
- A mechanism to query the data model: for example, SPARQL - Query language for RDF[4];
- A toolkit for building Semantic Web applications that provides a development environment for RDF, RDF(S) and OWL, the possibility to query knowledge bases using SPARQL, and that supports rule-based inferences.

One fundamental assumption of the SW is *decentralisation*, with respect to both information and authority [33]. This dictates that no single agent can claim ownership of any ontological definition, or any statement, and as a consequence, such ontological definitions can be extended by anyone. Thus, as with multi-agent systems, the only way to achieve authority is though consensus, rather than via a centralised control mechanism. This is the mechanism explored in this thesis by using argumentation theory (see chapter 5).

The Semantic Web offers a compelling vision, but it also raises many difficult challenges. Although ontology is the key factor for enabling interoperability in the Semantic Web, allowing applications and agents to agree on the terms that they use when communicating, ontologies themselves can be heterogeneous and some work has to be done to achieve interoperability.

Summarizing, agent technology provides methodologies and architectural approaches that reflect the epistemology for communication in the real world, while emerging Semantic Web technologies will provide a world-wide infrastructure for communication and information exchange with support for semantically enhanced information processing and distributed computing over the Web.

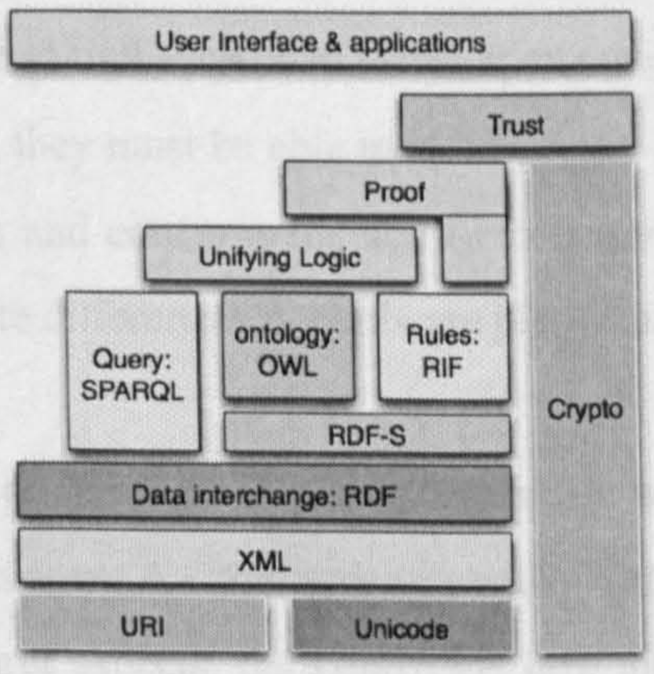

Figure 2.1: The Semantic Web's Layered Architecture.

Moreover, the inherently heterogeneous distributed nature of the Semantic Web, and the presence

[4]www.w3.org/TR/rdf-sparql-query/

of many individual problem-solving agents, strengthens the problem of heterogeneity discussed above: agents have to interact with other agents that have their own aims, objectives and knowledge. In addition, agents can exhibit different behaviours, and will themselves be faced with a choice of interaction partners showing different characteristics.

As pointed out by Hewitt and de Jong [60], the only thing that agents, situated in these environments, have in common is their ability to *communicate*.

Agent communication is discussed in the next section.

2.4 Agent Communication

The definitions of agents, presented in Section 2.1, emphasize the fact that an agent should be able to interact with its environment and with other agents either to solve common problems or either simply reach its goal. The means for achieving interoperability is focused on their ability to *communicate*. Thus, interoperability is akin to effectively exchanging the information and knowledge content of the agents by *communication*. Using its communicative abilities, an agent can work together cooperatively to accomplish its goal, act on its own initiative, use local information and knowledge to manage local resources, and handle requests from agents and user and out-source parts of its task. This entails that it must be able to talk to other agents, that is, try to call upon other agents to assist it in accomplishing its goal. Note that this goes much further than merely exchanging information or running processes in some distributed manner, since agents do not need to comply with requests, or can engage in some form of negotiation.

The importance of agent communication is especially highlighted in the context of Semantic Web, where the agents' tasks are accomplished by the cooperation of personal agents, service agents and Semantic Web servers. Agents that would like to cooperate with each other have to face two major challenges. First, they must be able to find each other (in an open environment, agents might appear and disappear unpredictably). Second, they must be able to interoperate. Interoperation is thus an essential property for the success of agents and concerns the ability to communicate effectively and exchange knowledge with one another despite differences in hardware platforms, operating systems, architectures and programming languages.

At this stage, it is worth considering how agents can communicate with one another. The simple communication model assumes that an agent A communicates with another agent B, by sending a message M[5]. The message M deals with three aspects: the *syntax*, i.e, how the symbols in M are structured, the *semantics*, i.e., what they denote, and *pragmatics*, i.e., how they are interpreted and used. The agent

[5]In this thesis, the communication we are interested in is *direct* mode, i.e. communication is achieved via direct message passing between agents. In the indirect mode, agents communicate via the environment (the world where they live and evolve) and thus limit the potential of both coordination and cooperation.

B interprets the meaning of the message M using a combination of semantics and pragmatics. The pragmatics includes considerations external to the message M, such as the mental states of the communicating agents and the environment, i.e. the *context*. The "context" of the agent B determines M' as the interpretation of the message M and then determines an appropriate response (see Figure 2.2).

It is generally accepted that the key elements to assure communication in open MAS include:

- A shared language to express the message contents sent by an agent and to define the types of messages that agents may exchange;
- A language for representing the knowledge of an agent, i.e. the content language;
- A shared set of prescriptive conversation policies to provide a structure for basic agent dialog, i.e. the communication protocol.

In the rest of this chapter we will discuss the first and third point, while the second will be discussed in depth in the next chapter.

In the context of open MAS, it is worth mentioning middle agents that have been proposed to enable interoperability among agents in open environments. Middle agents are the subject of Section 2.4.3.

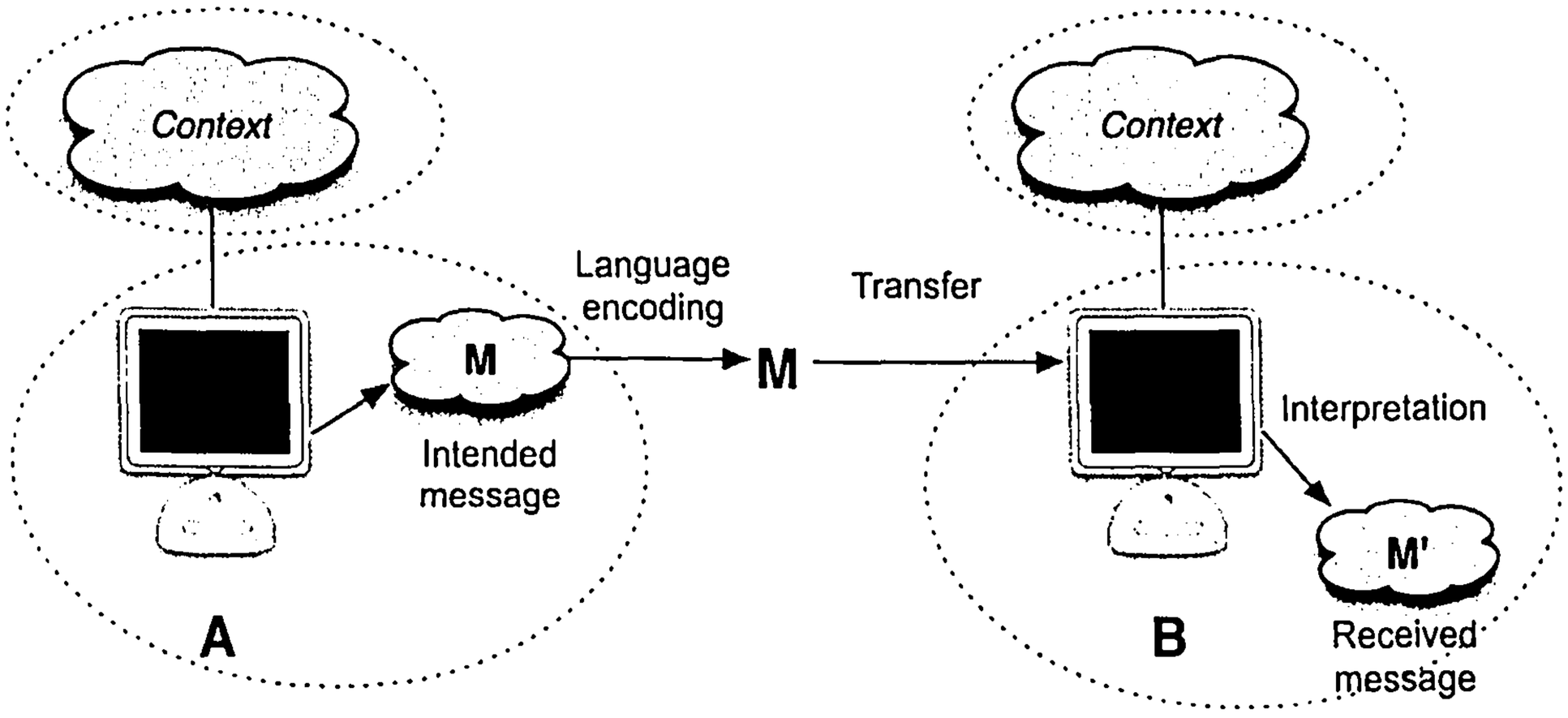

Figure 2.2: Basis Communication Model.

2.4.1 Agent Communication Languages

Agents, often designed by different people and using different representations of knowledge, must share a communication language to be able to interoperate and to enable effective knowledge-level communication between them. A language used by the agents for this exchange is an "Agent Communication Language (ACL)". An ACL has a crucial role in the solution to the interoperability problem and defines a communication channel for the reliable exchange of messages over a computer network (i.e. the

lower level aspects of the communication). An ACL stems from the need to coordinate the actions of an agent with that of the other agents. It can be used to share information and knowledge among agents in Multi-Agent Systems but also to request the performance of a task. The main objective of an ACL is to provide an interface that allows heterogeneous agents to interact, to communicate with meaningful statements that convey information about their environment or knowledge.

There are two main approaches to agent communication languages [50]: the procedural approach and the declarative approach. The procedural approach to communication is based on executable content, which can be accomplished by using programming and scripting languages. Since procedural languages are difficult to control, coordinate and merge, declarative languages are preferred for the design of agent communication languages, especially in open environments. Most declarative communication languages, e.g., FIPA [1], KQML [48] are based on speech act theory [107].

KQML was designed originally as part of the *Knowledge Sharing Effort (KSE)*, whose purpose was to develop a formalism and technical structure to enable the sharing of knowledge between knowledge base systems, humans and software systems [88]. Its model of communication, a declarative message content representation language and the use of explicit ontologies defining the domain of discourse, has become widely recognized as having great benefits for the integration of desperate and distributed information sources to form an open, extensible and loosely coupled system.

In FIPA and KQML, communication is modelled through illocutionary acts called *performatives* (e.g. request, inform, agree), which are conceptualized as action intending to produce some effect on the receiver, such as performing some task (request) or giving some information (query). The use of a standard set of speech acts in open systems provides a structure to messages in which the intended meaning of the content of the speech act (what is being said) can be interpreted more easily, and provides a semantic structure to the messages intuitive to the human users of a system.

In the FIPA (Foundation for Intelligent Physical Agents) ACL specification [1], for example, the semantics of each performative is specified in terms of a set of feasibility *preconditions* -i.e., those conditions that should hold before the action is performed, and a set of *rational effects* -i.e., the reasons for the agent to performe that action. While, the semantics of KQML [48] are defined in terms of preconditions and postconditions that govern the use of a performative, along with conditions that suggest the final state for the successful performance of the speech act (performative).

An example of a KQML message is given below.

```
(tell
:sender Ag1
:receiver Ag2
:language prolog
:ontology Travel
```

```
:content price(hotel-Hilton,200))
```

The first line in the message is the performative, which specifies the illocutionary force, in this case *tell*. Other examples of performatives in KQML are *replay*, *ask* and *subscribe*. The last line describes the actual content of the message. The other lines in a KQML message specify the sender and the receiver, the language that is used to describe the content and the ontology that underlies it.

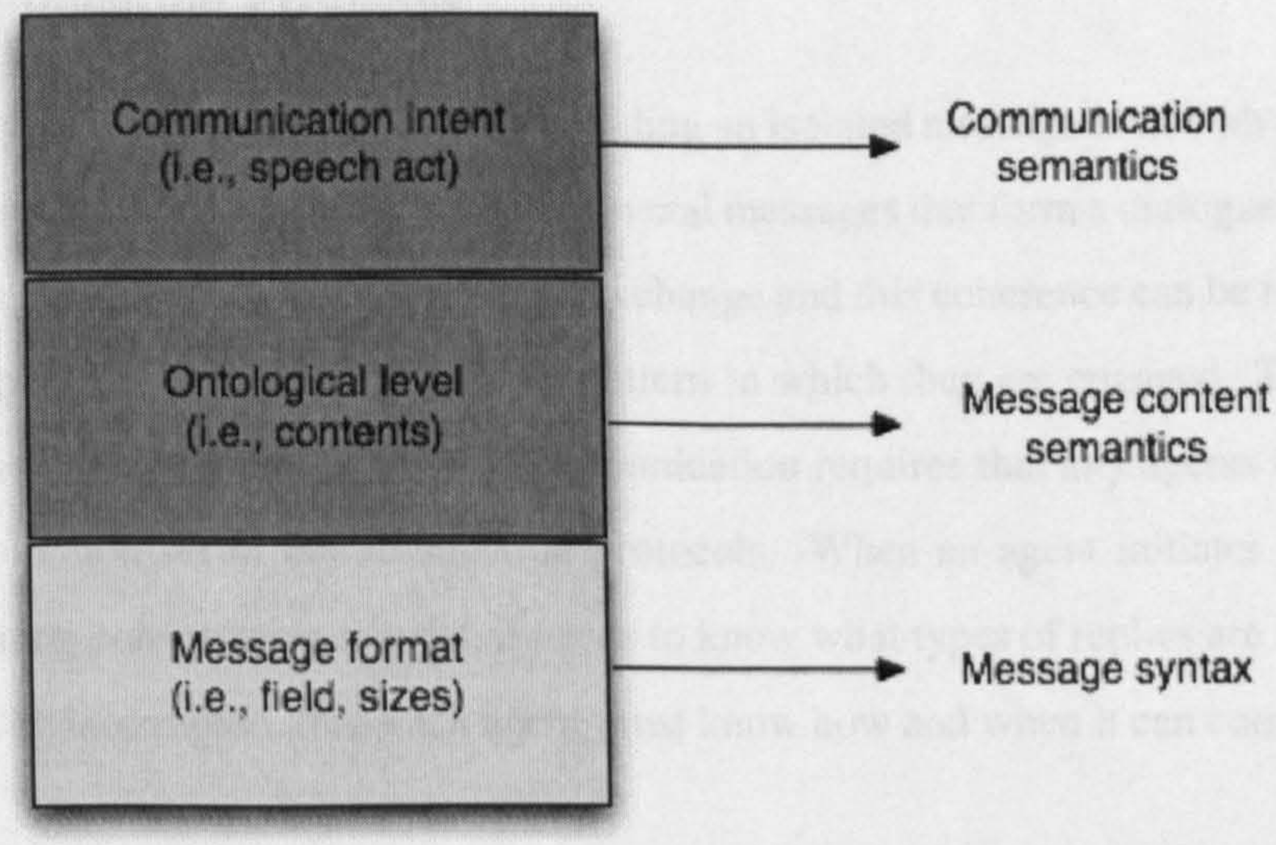

Figure 2.3: Layer Organization of a General Agent Communication Language.

Figure 2.3 sketches a three-layer organization of a general agent communication language which separates the semantics of the content from those one related to the message as a part of a conversation. The lower layer deals with technical details of message syntax and effective transport. Generally, this layer defines a specific format for messages, indicating what kind of information must be specified (e.g. message sender and recipient, message identifier, and so on) and how much memory is devoted to these fields. Moreover, lower level protocols are often adopted and exploited in order to implement communication channels between agents. The message content is generally opaque to the previous layer and is the focus of the second layer, which has to deal with ontological issues (i.e. what agents may talk about) and with the possible expressions that can be stated with reference to elements specified by the ontology (i.e. what can they say about objects)[6]. In other words, this layer deals with the message content, that must be referred to the shared ontology and may thus specify propositions on domain objects. According to several approaches these issues are (to a certain degree) independent from the other layers [74]. Generally, they are delegated to a specific content language, which regulates how propositions are expressed, rather than managing low level technical details or capturing propositional attitudes. Messages are associated to a speech act or performative (third layer) which indicates the intention of the sender, the meaning of the message in the communication (mentalistic approach to

[6]Ontology is the topic of the next chapter.

communication semantics definition). For instance, in KQML an agent could inform another one about a new fact P (that should thus be added to the knowledge base of the receiver), or could ask whether a fact is true. Thus $tell(P)$ and $ask(P)$ have the same content but the different speech acts elicit different responses.

We now turn our attention to a discussion of communication protocols.

2.4.2 Communication Protocols

Communication typically consists of more than sending an isolated message even with the correct syntax and semantics. Communication usually involves several messages that form a dialogue. These dialogues follow patterns or policies that ensure a coherent exchange and this coherence can be reached only if the two agent are explicitly aware of the particular pattern in which they are engaged. This aspect is well known as *communication protocols*. Agent communication requires that any agents that communicate should share a common set of communication protocols. When an agent initiates a conversation or receives an incoming conversation request, it needs to know what types of replies are appropriate. If the conversation can be interrupted, then each agent must know how and when it can correctly interrupt the conversation.

A protocol thus describes in terms of the high level of communication languages, the sequences of messages that can occur, and their meaning in order to ensure the coherence required to initiate, carry out and terminate a conversation.

> "*A protocol can be viewed as an institutionalised pattern of interaction. That is, a pattern of interaction that has been formally defined and abstracted away from any particular sequence of execution steps. Viewing interactions in this way means that attention is focused on the essential nature and purpose of the interaction, rather than on the precise ordering of particular message exchanges.* " [139]

One of the most well-known protocols is the contract-net: agents in need of services distribute calls for proposals (CfP) to other agents. The recipients of these messages evaluate those requests and submit bids to the originating agents. The originators use these bids to decide which agents to choose and then award contracts to those agents.

However, in open MAS, communication protocols present a few drawbacks, especially regarding interoperability, composition and verification of protocols which depend on a common semantics. One of the primary drawbacks of a communication protocol, though, is that agents are obliged to have fixed conversations, since the protocol is previously determined (before a conversation) and cannot change dynamically. Instead, we believe that heterogeneous agents cannot be governed by fixed protocol but a protocol should dynamically accommodate all forms of agents and interactions. However, this issue is quite beyond the scope of this thesis. Interested readers are referred to [98] for further details.

2.4.3 Middle Agents

A common approach to overcome the interoperability problem of agents in open environments is the introduction of a middle-ware layer between the agents. Having a middle-ware layer, for example, enables agents to advertise their capabilities, and thus enables agent to look for agents with some specific capabilities appropriate for a given problem at hand. The middle-ware layer can be realized by an agent, known as *middle agent*, often specialized in supporting the activities of other agents and reasoning. The idea of middle agents was born in the work on mediators, which was initially proposed in the field of Information Systems [135] and then applied in the field of Agent Information Integration [136] and Information Brokering [83], as a way to locate and combine information coming from multiple and heterogeneous sources.

Middle agents can also be seen as mediators between agent requesters and a set of agent providers, in which the information being mediated is constituted by the description of available services while allowing the participants in a transaction to preserve their anonymity.
The literature presents the following types of middle agents:

- *Facilitors:* agents to which other agents surrender their autonomy in exchange of the facilitator's services. They can also coordinate the activity of the agents and can satisfy requests on behalf of their subordinated agents [50].

- *Mediators:* agents that exploit encoded knowledge to create services for a higher level of applications.

- *Matchmakers* and *yellow pages.* They provide a service to find agent providers based on advertised capabilities [31].

- *Brokers:* agents that receive requests and are able to contact the appropriate providers on behalf of the requester. The main difference between brokers and matchmakers is that the latter only introduces matching agents to each other, whereas a broker handles all the communication with the capability providers [31];

- *Blackboards:* repository agents that receive and hold requests for other agents to process [90].

Although, middle agents are not fundamental for agent communication, they provides lookup services that facilitate the discovery of agents with specific capability descriptions, and to mediate communication between them.

In this thesis, we are using a mediator infrastructure in order to overcome the problem of reconciling heterogeneous agents' ontologies. The mediator infrastructure is responsible to provide specific services to a multi-agent system, such as to align ontologies, to store the mappings, to translate expressions and so on. Note that this infrastructure is not fundament for the success of our approach. Our approach could

also be apply when each agent is able to generate itself the mappings and the arguments. However, we believe that the use of a middle agent to provide the mappings presents several advantages. First of all, the mediator avoid the agents to access to other agents' ontologies, which are often private. Second, it provides a common place to store and retrieve alignments as well as providing them on the fly.

2.5 Conclusions

In this chapter we have presented an overview of the wide area of agents, multi-agent systems and agent communication. The chapter began by defining an Agent and a Multi-Agent System. We have argued how the real utility of Multi-Agent Systems lies on the open nature of agent systems and in this context, we have sketched the basic principle of one specific open environment: the Semantic Web. The Semantic Web provides a rich habitat for agents by offering technologies for representing and using ontologies to create a Web of machine understandable information. We have seen that, in such systems, an important element is the agents' ability to communicate. This then motivated an overview of agent communication. We have discussed how the agents' ability to communicate is achieved through speech-act inspired languages, which express how the message content is to be understand. The actual content of a messages is expressed in an ontology. However, an agent does not only commit to a common agent communication language, but must also agree to use a particular protocol when it interacts. Communication protocols were then discussed.
The concept of mediator, as a common approach to overcome the interoperability problems agents in open environments, has also been introduced.

The aim of the next chapter is to move on to aspects of an ontology as a means to provide the definitions of the vocabularies used by agents to describe the world.

Chapter 3

Ontologies

In the previous chapter we have presented agent communication as a key aspect of open Multi-Agent Systems, without which we would be unaware of the abilities, knowledge and use of the agents around us. In order to communicate, agents need a conceptualization of the domain of interest and a shared vocabulary to communicate facts related to this domain. This conceptualization can be expressed by ontologies.

This chapter contains an introduction to ontologies and the most important related concepts that have been considered in our work. First, we review in Section 3.1 the different meanings that the term "ontology" takes in Artificial Intelligence (AI) and Computer Science. Section 3.2 identifies the main components and the different types of ontologies. Section 3.3 discusses suitable representations of ontologies. The chapter ends by presenting the conclusions.

3.1 What is an Ontology?

It is widely and long accepted that an ontology is an effective approach that can be used to tackle the many issues involved in the generalisation of low-level heterogenous data to relatively high-level concepts for the purposes of communication, system interoperation and software reuse [88].

Recent years have seen a promising surge in both research and practice of ontologies. Ontologies, especially for the World Wide Web (WWW) have been used successful and are now heavily relied upon by e-commerce [30], e-business [77] and bioinformatics [10] applications. In the next generation, ontologies will facilitate the evolution of the WWW into Berners-Lee's vision of the Semantic Web [17] (see Section 2.3.1), which makes information available in machine-readable format. With information being machine-readable, software agents can be employed to perform knowledge gathering, reasoning, economic optimising and even bidding on behalf of humans. Moreover, research into the use of ontologies in agent systems has rapidly gained importance over the past few years, corresponding to a convergence

between research on explicitly modeled ontologies and the systems that are expected to use them, and demonstrated by the increasing number of major conferences and workshops on the intersection of the area of ontologies and multi-agent systems.

But what does the term ontology mean and what is an ontology about?

The meaning of the term ontology has different connotations in Philosophy and in various areas in Artificial Intelligence (AI) and Computer Science. From the philosophical field, the term "ontology" is used to show a systematic description of existence. Guarino [56] gave a characterisation of the philosophical account for the term ontology as a particular system of categories accounting for a certain vision of the world. In the context of AI the term "ontology" is used with different meanings. Gruber [53] provides one of the most cited definitions of ontology, which is adopted in this thesis, as

> "*an explicit specification of a conceptualisation*"

Here, a conceptualization refers to people's conceptual understanding of a certain domain. While being very general, this definition captures the essence of what ontology means, regardless of potential application areas one might have in mind.

Borst [20] has extended Gruber's definition:

> "*An ontology is a formal specification of a shared conceptualisation.*"

This definition introduces the idea that an ontology provides a way to specify content-specific agreements, and thus to share such conceptualizations. In terms of this definition an ontology is characterized by the subsequent features:

- *formal* and *explicit*: this means that the ontology is represented using a formal language;
- *shared*, i.e., the ontology mirrors a common understanding of the modelled domain, being the result of a consensus achieved within a (potential) community of ontology users;
- it specifies a *conceptualization*: the ontology is used to model some domain of the world.

The following is the definition given by Studer combining and expanding the definitions of Gruber and Borst [114].

> "*An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine- readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group.*"

An ontology can therefore be viewed as a means for providing a shared and common understanding of a domain that can be communicated among people and heterogeneous application systems. The emphasis is thus on the capture of consensual knowledge for use (and reuse) across agents, software applications and among groups of people.

Campbell and Shapiro [24] consider that ontologies

> *"...consist of a representational vocabulary with precise definitions of the meanings of the terms of this vocabulary plus a set of formal axioms that constrain interpretations and well-formed use of these terms".*

This definition emphasizes the formal logic aspects of ontology, the intention of wanting to reason with ontological constructs. In contrast to this viewpoint, Uschold and Jasper in [124] give a very loose definition and state:

> *"An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning."*

An alternative definition of conceptualisation is given by Guarino in [55] that states that an ontology is *"not a specification of a conceptualization, but a (possibly incomplete) agreement about a conceptualization."* He further states in [56] that:

> *"An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models. "*

According to Guarino, an ontological commitment should be made explicit when applying the ontology, in order to facilitate its accessibility, maintainability and integrity. This will lead to increase of trasparency for the application software which is based on that ontology.

Although, the definitions presented above highlight a specific aspect of a role played by ontologies, all share the idea that an ontology provides a description of a particular viewpoint about a domain and that such a description, explicitly or implicitly, states a vocabulary that an agent commits to.

In this thesis, the term "ontology" describes an explicit specification of a conceptualisation that an agent commit to so that it can communicate about a specific domain [53]. In particular, we say that an agent *commits* to an ontology if its observable actions, in term of messages, and according to the intended meaning with, are consistent with the definitions in the ontology. Thus, a ontology *explicitly* defines the vocabulary with which message are exchanged among agents.

3.2 The Components and Types of Ontologies

Ontologies can formalize the knowledge of a domain by means of different components: concepts, relations, functions, axioms and instances [54]:

- *Concepts.* A concept, also named class, represents the abstractions used to describe an object of the world; where a concept can be abstract or tangible, simple or complex, real or fictitious. Formally, a concept is described by a term (generally a symbol), an extension, and an intension. The extension of a concept is the set of objects (i.e., instances) that the concept can be applied to: for instance, the extension of `car` includes: "`the red car parked at the end of the street`" or "`the fiat panda`". On the other hand, the intension of a concept is the set of properties, features, and attributes specifying the semantics of the concept, that is the set of features shared by these objects. For instance, the intension of the concept `car` includes the features of `a street vehicle with engine, generally with four wheels`. Moreover, in the description of a concept we can also use specific elements, such as: *Metaclasses* that are classes which have classes as their instances; *Slots/Attributes* which belong to a specific concept; for instance, the attribute `age` may belong to concept `Person`; *Facets* that allow the specification of an attribute; e.g. default value, type, cardinality, operational definition. Concepts in the ontology are usually organized in taxonomies. *Taxonomies* are widely used to organize ontological knowledge in the domain using generalization/specialization relationships through which simple/multiple inheritance can be applied. Their semantics may be based on the definitions of *subclass of*, *partitions*, *disjoint composition*, etc.

- *Relations.* Represent a kind of linkage among domain concepts. A relation is described by a term, an extension, and an intension. The extension of a relation is the set of possible tuples of the relation instantiated by objects. For instance, the extension of the relation `parent` can include: "`Giovanna and Angelo are the parents of Loredana`". The intension specifies the types of the linked concepts. For instance, the intension of the relation `parent` can be: "`the raising of children and all the responsibilities and activities that are involved in it`". Examples of binary relations are `is-a` and `links to`.

- *Function.* Functions are particular relations which are defined on the set of concepts and return a concept. For example, the function `Price-of-flat` is function of the concepts `Year` and `Location`.

- *Axioms.* They are assertions that are always true and specify the semantics of the concepts. They generally describe how the vocabulary (concepts and relations) can be used to reason about the

domain. They are included in an ontology as information constraints, correctness checking, or knowledge inference. Particularly, they may express the kind of relation among concepts, the signature and the cardinality of a relation, algebraic properties of a relation (symmetry, transitivity), and other conceptual properties, such as exclusivity, generality, or identity.

- *Instances.* The instances are the elements of the domain, the actual objects of the world. The instances of a concept are also called *individuals*.

Often, ontologies can be also complemented by *rules*, which underline the business logic of the specific application:

- *Rules*: Rules follow an `if-then` structure and are used to express a set of actions or heuristics, to represent decision making, logic, business logic and to facilitate enhanced representation and reasoning capabilities. As we have briefly introduced in the previous chapter, rules play an integral role in the Semantic Web, where the rule layer is meant to allow further means to deduce knowledge and combine information.

Depending on the representation language and the scope of ontology, only a subset of the components presented above will be used.

The literature presents different types of ontologies, which can be classified according to different dimensions, which range from the level of generality of the concepts they describe, to the type of knowledge they model (be it related to the domain or the task). According to [56], ontologies can be classified into four categories:

- *Upper-level/top-level ontologies.* They describe very general concepts or common-sense knowledge, such as space, time, etc., which are independent of a particular problem or domain.
- *Domain ontologies.* They are used to model specific domains such as medicine or academia.
- *Task ontologies.* They describe generic or domain-specific activities, such as diagnosis or selling.
- *Application ontologies.* They describe concepts depending both on a particular domain and on a particular task. They are often a specialisation of both domain and task ontologies and correspond to the roles played by domain entities when they perform certain activities.

A last category of ontologies, which was not covered by the classifications mentioned so far, are the so-called *meta-ontologies* or (knowledge) representation ontologies. They describe the primitives which are used to formalize knowledge in conformity with a specific representation paradigm.

Ontologies differ also in the degree of formality by which the terms and their meaning are expressed in the ontology. In [123], for example, the authors classify ontologies as:

- *Highly informal*: are those ontologies expressed in natural language. Term definitions might be ambiguous due to the inherent ambiguity of natural language.

- *Semi-informal*: these ontologies are expressed in a restricted and structured form of natural language. Restricting and structuring natural language achieves improvement in clarity and reduction in ambiguity.

- *Semi-formal*: these are ontologies expressed in artificial languages which are formally defined.

- *Rigorously formal*: these are ontologies whose terms are precisely defined with formal semantics, theorems and proofs of desired properties such as soundness and completeness.

Along the same line, McGuinness [85] defines an "*ontological continuum*" specifying a total order between common types of models. This basically divides ontologies (or ontology-like structures) in informal and formal as follows:

- *Informal models* are ordered in ascending order of their formality degree as controlled vocabularies, glossaries, thesauri and informal taxonomies.

- *Formal models* are ordered in the same manner: starting with formal taxonomies, which precisely define the meaning of the specialization/generalization relationship, more formal models are derived by incrementally adding formal instances, properties/frames, value restrictions, general logical constraints, disjointness, etc.

In the first category we usually encounter thesauri such as WordNet [86], taxonomies such as the Open Directory[1] and the ACM classification[2] or various eCommerce standard [47]. Many of the available Semantic Web ontologies can be localized at the lower end of the formal continuum (i.e. as formal taxonomies), a category which overlaps with the semi-formal level in the previous categorizations.

We now turn to a description of recent formalisms for representing ontologies.

3.3 Representing an Ontology

All the definitions presented in Section 3.1 highlight the fact that ontologies are not bound to any particular formalism but to the aspect of sharing a model. However, a fundamental requirement for the latter aspect requires that ontologies are represented in some formal language, in order that "*detailed, accurate, consistent, sound and meaningful distinctions can be made*" [64]. Ontologies have typically been represented using frames (e.g. FLogic [70] and OKBC [25]), conceptual graphs, first-order logic

[1] http://www.dmoz.org
[2] http://www.acm.org/

or description logics. Currently, the most dominant are the representation schemes based on description logic languages, such as OWL [96].

We will start with a short introduction to Description Logics and will use that as a foundation for the presentation of OWL, which is one of the standard ontology languages.

3.3.1 Description Logics

Description Logics (DLs) [9] are a family of logic-based Knowledge Representation (KR) formalisms devised for the representation of and reasoning about the knowledge on an application domain in a structured and unambiguous way. For such a purpose, DLs are equipped with a well-defined semantics, which provides each of its constructs with a precise logical meaning. DLs structure the knowledge about an application domain by defining the relevant atomic *concepts* and *roles* of the domain, and then using these concepts and roles to specify properties of objects and individuals occurring in that domain. A DL provides a set of operators, called constructors, which allow the formation of complex concepts and roles from atomic ones. Concepts are sets of individuals and roles are binary relationships between individuals. For example, by applying the concept disjuncton constructor $\sqcup$ on the atomic concepts $UndergraduateStudent$ and $GraduateStudent$, the set of all college students can be represented by the following complex concept:

$$UndergraduateStudent \sqcup GraduateStudent$$

The Boolean Concept Constructors are, apart from concept disjunction ($\sqcup$), concept conjunction ($\sqcap$), and concept negation ($\neg$). A Description Logic that provides, either implicitly or explicitly, all the boolean operators is called *propositionally closed*.

Moreover, DLs typically provide concept constructors that use roles to form complex concepts. The basic constructors of this kind are *existential* and *universal* restrictions operators, which represent restricted forms of quantification. For example, the following concepts would describe the researchers whose only affiliation is to a university and the persons who have obtained at least one bachelors degree.

$$Researcher \sqcap \forall affiliatedWith.University$$

$$Person \sqcap \exists hasDegreeBachelors$$

A set of concept and role assertions constitute an ABox. The statements about how the concepts and roles are related to each other are defined by *terminological axioms*. A set of terminological axioms constitute a TBox. A TBox describes the structure of a domain in terms of classes (concepts) and properties (roles). This means that in a description logic, concepts are defined intentionally in terms of descriptions that specify what properties objects must have to belong to a certain class. In its simplest

form, a TBox consists of a restricted form of concept inclusion axioms called concept definitions: sentences of the form $C_1 \sqsubseteq C_2$ or $C_1 \equiv C_2$, for C_2 atomic, which describe necessary or necessary and sufficient conditions respectively for individuals members of C_1 to be members of C_2. For example, the axiom:

$$CollegeStudent \sqsubseteq UndergraduateStudent \sqcup GraduateStudent$$

introduces the atomic concept *CollegeStudent* and states that a *CollegeStudent* is necessarily either an *UndergraduateStudent* or a *GraduateStudent* or both. However, TBox axioms can also be used to describe more complex sentences. For example, the axiom: $\forall studiesAt.University \sqsubseteq UndergraduateStudent \sqcup GraduateStudent$ states that everybody studying at a university must be either an undergraduate or a graduate student, or both.

The ABox on the other hand, consists of assertions about individuals, using the concepts and roles defined in the TBox, i.e. terminological axioms. Thus, an ABox containing axioms of the form $C(a)$, called concept assertions and $R(a, b)$, called role assertions, where a, b are object names, R is a role and C a concept. For example, the axiom $University(UNILIV)$ states that the University of Liverpool is a university. On the other hand, a role assertion is used to state that two objects are related by a role. For example, the axiom $locatedIn(UNILIV, UK)$ would state that the University of Liverpool is located in UK.

DLs are used as the basis for ontology languages such as OWL, presented in the next section.

3.3.2 OWL

The Web Ontology Language (OWL) [96] is the World Wide Web Consortium (W3C) standard for representing ontologies on the Semantic Web. As a Web Ontology Language, OWL has influences from both the Web and Knowledge Representation communities. On the one hand, as a Semantic Web language, OWL had to fit into the Semantic Web vision of a stack of languages (see Section 2.31), and hence it was designed to be compatible with the Extensible Markup Language (XML) [143] and the Resource Description Framework (RDF) [59]. On the other hand, being an ontology language, OWL had to be compatible with its most immediate predecessor, namely DAML+OIL [97], which was deeply influenced by Description Logics and Frame systems.

OWL has three sublanguages: OWL Full, OWL DL and OWL Lite:

- OWL Full is a same syntax, extended semantics extension of RDF. It is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the predefined (RDF or OWL) vocabulary. Although OWL Full provides some interesting capabilities,

it is undecidable and thus no reasoning software is able to support complete reasoning for every feature of OWL Full.

- OWL DL includes all the OWL Full language constructs with some restrictions, such as type separation (a class cannot be treated as an individual or a property, for example) or the inability to use transitive roles on number restrictions. It aims to support those users who want maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL does not extend the syntax of RDF, since not every RDF document is a valid OWL DL document. However, it is possible to prove that, for every RDF document that is valid OWL DL, the DL-based direct model theoretic semantics and the RDF semantics are equivalent [96]. Moreover, it is possible to design tableau-based algorithms, well suited for implementation and optimization, for OWL DL.

- OWL Lite provides a subset of the constructs permitted in OWL DL. It aims to support those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1.

From its relationship with Description Logics come the main modeling constructs in OWL as well as the semantics for those constructs. Ontologies written in two of the species of OWL, OWL Lite and OWL DL, correspond to Description Logics knowledge bases and hence can be processed by DL-based reasoners, such as Racer[3], Pellet [57] and Kaon2[4]. These tools support classical Description Logic reasoning tasks, including:

- *Instance checking:* verifying that an instance is an individual in an ontology concept;
- *Concept subsumption:* checking the existence of an hierarchical relation between two concepts in the ontology;
- *Concept satisfiability:* verifying that the definition of a concept in the ontology is satisfiable, i.e. that it can admit instances;
- *Ontology satisfiability:* verifying the satisfiability of an ontology and its instances.

All the work presented in this thesis is grounded on the assumption that the agent's ontologies are represented in OWL DL / OWL Lite.

In the rest of this section we briefly discuss the main modelling primitives provided by OWL DL.

[3]pellet.owldl.com

[4]kaon2.semanticweb.org/

Classes (Concepts) in OWL DL

A class defines a group of individuals that belong together because they share some properties. There are two built-in classes: *Thing* is the class of all individuals and is a superclass of all OWL classes, and *Nothing* is the class that has no instances and is a subclass of all OWL classes.

New classes are introduced with either complete or partial descriptions. Complete descriptions are introduced by axioms stating equivalence of classes, while partial descriptions specify that a class is a *subclass* of another class. Furthermore, the *oneOf* class axiom gives a complete definition by enumerating all individuals belonging to this class. Additionally, classes can be specified to be disjoint with other classes.

Classes can also be specified via restrictions. Cardinality restrictions specify a lower or upper bound or an exact number of properties that must be present. More precisely, this means that an individual belongs to such a class if and only if the number of individuals that it is related to (via some property, which is a binary relation) meets the specified bounds. The restrictions *allValuesFrom* and *someValuesFrom* are stated on a property with respect to a class. The first one means that this property on this particular class has a local range restriction associated with it. The second one means that a particular class may have a restriction on a property that at least one value for that property is of a certain type.

Properties (Roles) in OWL DL

Properties are used to state relationships between individuals (*object properties*) or from individuals to data values (*datatype properties*). Property hierarchies may be created by making one or more statements that a property is a *subproperty* of one or more other properties. Both types of properties can be restricted to a certain domain and range. Without the definition of a domain and range, every class can be related with every other class or datatype by a given property. A domain of a property limits the individuals to which the property can be applied. The range of a property limits the individuals that a property may have as its value.

OWL DL also includes primitives related to equality or inequality of properties. Equivalent properties relate one individual to the same set of other individuals. There are special identifiers that are used to provide information concerning properties and their values. One property may be stated to be the inverse of another property. Properties may be stated to be transitive and/or symmetric. Moreover, properties may be stated to have a unique value. This feature is shorthand for stating that the propertys minimum cardinality is zero and its maximum cardinality is 1. Properties may be stated to be inverse functional, which has also been referred to as an unambiguous property.

We conclude the overview of the OWL language here, as the account given above suffices for the purposes required in this thesis. The readers who wish to see all constructs of OWL Lite and OWL DL should consult Appendix A.

3.4 Conclusions

Agents, in order to communicate in an effective way, must share concepts and their representation. These definitions should be represented in an artifact shared by the agents which mean to interact. In current systems based on an Agent Communication Languages approach to agent interaction, this artifact is represented by an ontology, made up of expressions in a formal language.

This chapter has presented an overview of theoretical foundations of ontologies. We have particularly focussed on those aspects which are relevant for the thesis: the definition of an ontology, the different types of ontologies, and, finally, the languages to represent ontologies.

Ontologies are embraced by the agent community to enable knowledge sharing as it provides a way to specify content-specific agreements. Once the ground rules for modeling a domain are specified in an ontology, agents can agree to adhere to that specification. If they do that, it would enable them to share knowledge, as their knowledge bases would all have the same underlying structure. However, in the context of multi-agent systems, the ontologies of the agents are often not shared. Different agents may view their world differently and thus, they may use heterogeneous ontologies. Heterogeneity and interoperability among agents' ontologies are discussed in the next chapter.

Chapter 4

Tackling Semantic Heterogeneity - Ontology Alignment

Within an open multi-agent system, agents are characterised by different views of the world, explicitly defined by ontologies. These ontologies are often diverse in nature and might have been developed for different purposes. Semantic heterogeneity among these agents needs to be resolved to enable meaningful information exchange and interoperation among them. In order to enable interoperation, their ontologies need to be reconciled. Reconciliation here means using two or more different ontologies developed for a different task in which their mutual relation is relevant [71]. The reconciliation of heterogeneous ontologies usually takes the form of an alignment between the ontologies, that is a set of mappings between the concepts, properties, and relationships in the agents ontologies.

This chapter aims at providing an overview of the field of ontology alignment. First, Section 4.1 discusses the role of ontologies in MAS, introducing the problem of semantic heterogeneity, to be presented in detail in the rest of this chapter. in Section 4.2 the problem of how agents can "understand" messages from other agents that use a different ontology. Section 4.3 discusses how ontologies may differ, presenting an overview of the different types of ontology mismatches. A formal definition of an ontology alignment and a general classification of the current methods to find them is discussed in Section 4.4. Section 4.5 presents the state of the art that addresses the problem of dealing with agents with heterogeneous ontologies. Section 4.6 completes this chapter with a short discussion of its contents.

4.1 Ontologies for Agent Communication

It has been largely acknowledged that, for agents to understand each other's messages, they must in some way share a common body of background knowledge to ensure that the communicated concept,

object, or entity has a uniform interpretation across agents even if different "names" are used to refer to it. As we have argued in this thesis, the technical term for this body of knowledge is "ontology". Ontologies provide the terms used to describe a domain's knowledge and the semantics of a language, which are made available to all the components active in a multi-agent system.

In a multi-agent system, common ontologies specify the ontological commitments of a set of participating agents, i.e. an agreement to use a vocabulary in a way that is consistent with an ontology. The agent committed to the same ontology can share knowledge among themselves with some confidence that they share an underlying understanding of what is being said.

It is worth making the point that commitment to a common ontology is a guarantee of consistency, but not of completeness, with respect to queries and assertions using the vocabulary defined in the ontology. A common ontology defines the vocabulary using which queries and assertions are exchanged between agents, but this does not imply that agents share a knowledge base - each agent can know facts that others do not. Thus, the answer to a query, formulated in the shared vocabulary, may differ when posed to different agents, even though they commit to the same ontology.

Agents may use different ontologies to represent their view of a domain. Each domain may be specified in many different ways. Watche and et al. in [131], have identified three different ways to use ontologies in a multi-agent system: single ontology approach, multiple ontology approach and hybrid ontology approach.

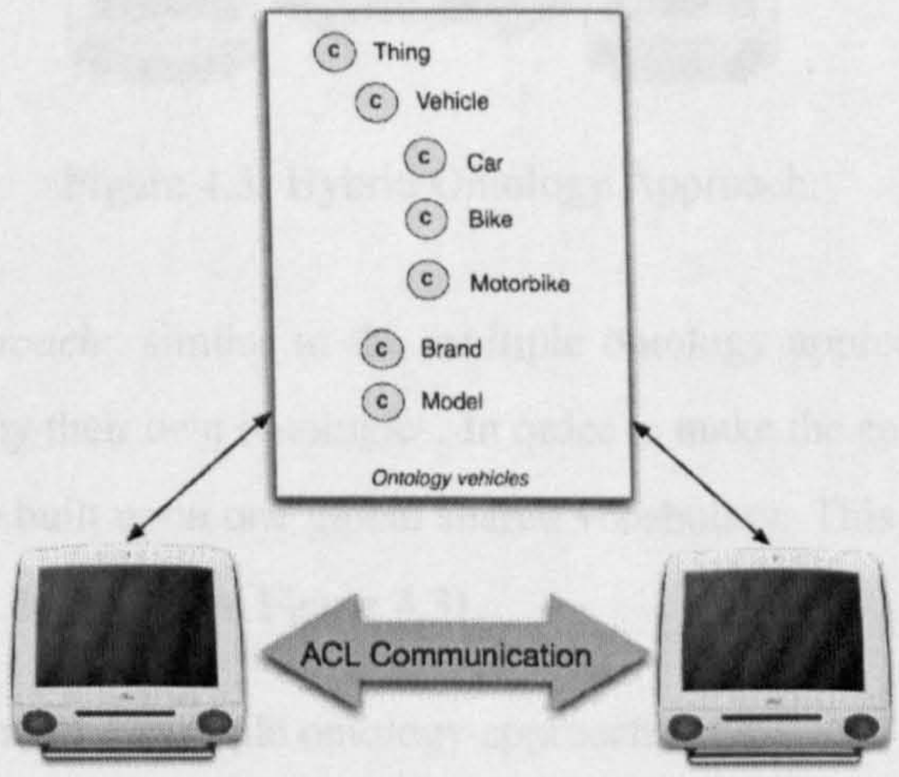

Figure 4.1: Single Ontology Approach.

1. *Single ontology approach*: uses a global ontology providing a shared vocabulary for the specification of the domain semantics. All information sources are related to a global ontology. The global ontology can also be formed as a combination of several specialized ontologies (See Figure 4.1).
2. *Multiple ontology approach*: each information source is described by its own ontology. In principle, the source ontology can be a combination of several other ontologies, but it cannot be assumed that the different source ontologies share the same vocabulary (See Figure 4.2).

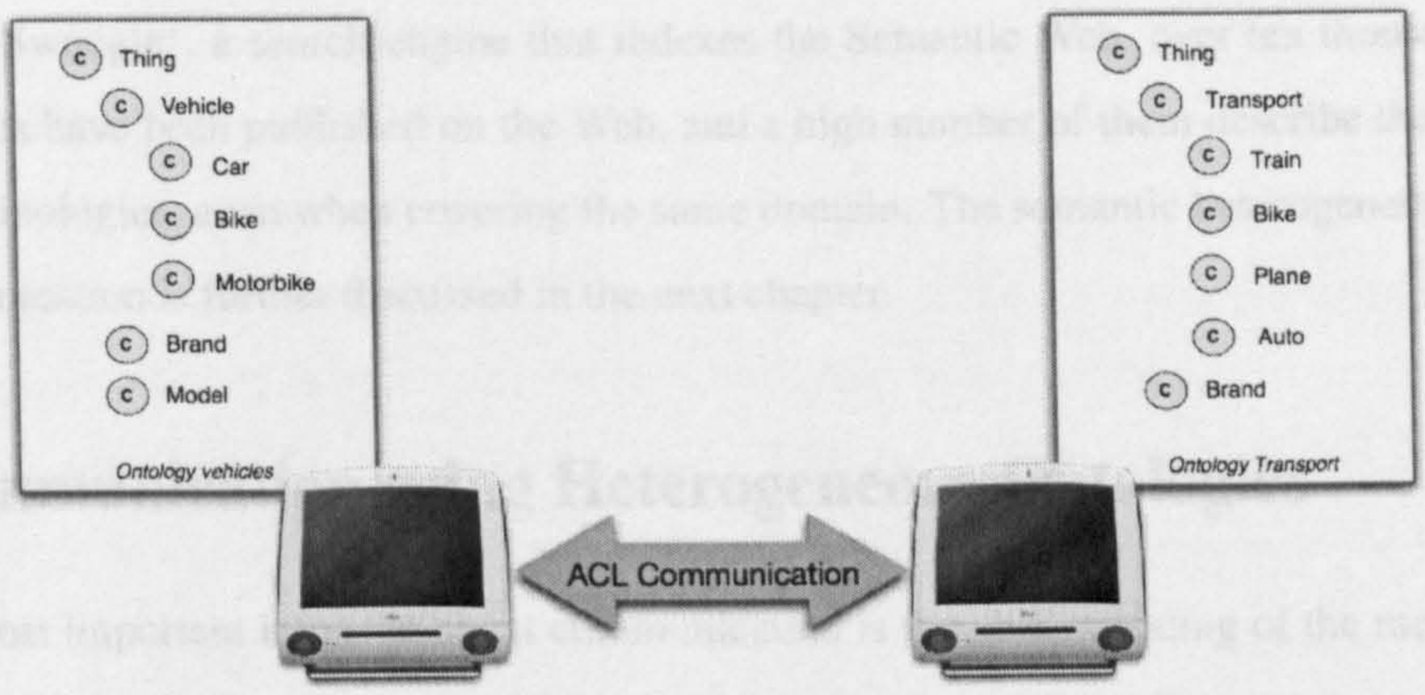

Figure 4.2: Multiple Ontology Approach.

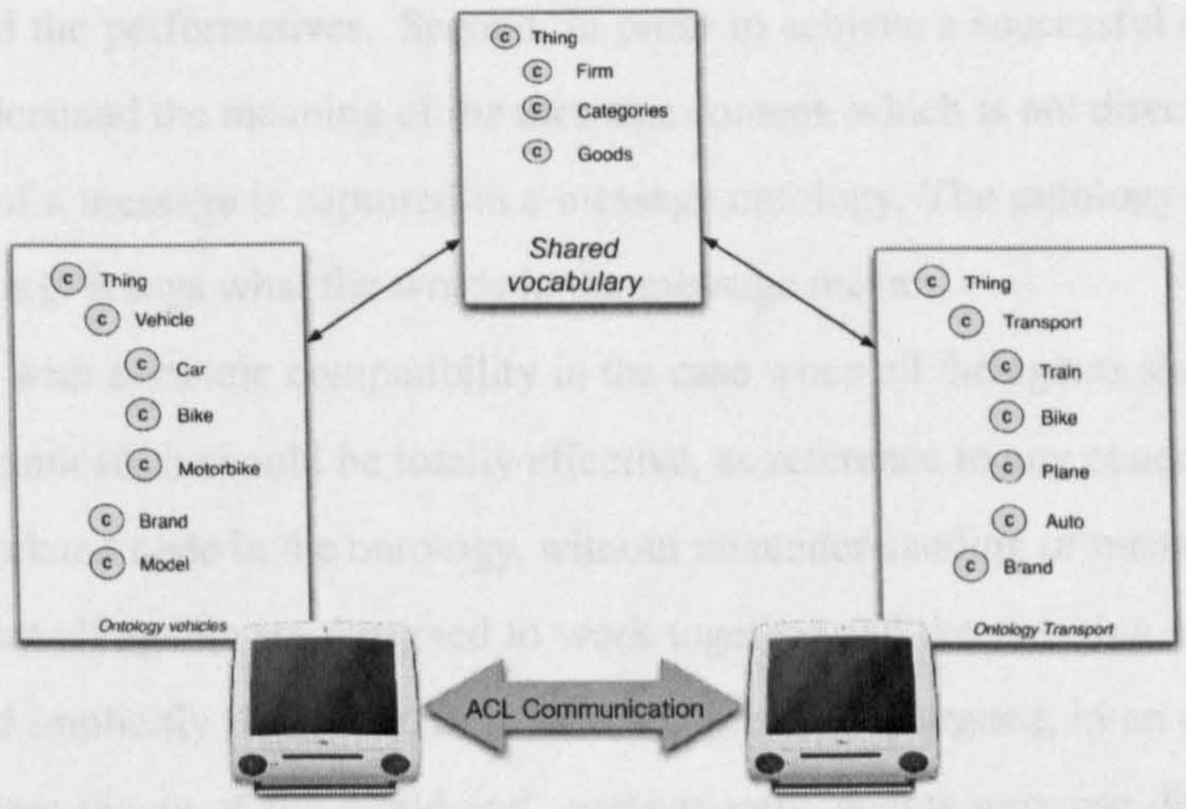

Figure 4.3: Hybrid Ontology Approach.

3. *Hybrid ontology approach*: similar to the multiple ontology approach, the semantics of each source are described by their own ontologies. In order to make the source ontologies comparable to each other they are built upon one global shared vocabulary. This shared vocabulary contains the basic terms of the domain (See Figure 4.3).

In this thesis, we are using the multiple ontology approach, where each agent has its own ontology.

The use of ontologies to facilitate agent-to-agent communication has been well established since the late 90s, with systems like Infosleuth [12], where ontologies enabled the integration of information in open and distributed environments. Moreover, the definition of Semantic Web languages such as RDF(S) and OWL have facilitated the uptake of ontology-based agent architectures, such as GraniteNights [52].

However, few efforts have attempted to address the problem of supporting agent communication in open, distributed environments, where agents commit to several, partially overlapping ontologies.

According to Swoogle[1], a search engine that indexes the Semantic Web, over ten thousand Semantic Web ontologies have been published on the Web, and a high number of them describe their domains in different terminologies, even when covering the same domain. The semantic heterogeneity problems in agent communication is further discussed in the next chapter.

4.2 Communication using Heterogeneous Ontologies

One of the most important issues in agent communication is the understanding of the meaning of messages. Several layers can be distinguished in the process of understanding when agents want to exchange information by ACL - first agent must be able to understand the structure of the exchanged messages as well as to understand the performatives. Second, in order to achieve a successful communication, the agents must also understand the meaning of the message content, which is not directly supported by the ACL. The meaning of a message is captured in a message ontology. The ontology provides interpretation of the message, e.g. it says what the words in the message mean.

There is no problem with semantic compatibility in the case when all the agents share one *common ontology*. Agent communication should be totally effective, as reference to any concept could be realized by direct linking to related node in the ontology, without misunderstanding or mistakes. This is the case of closed MAS where all agents are designed to work together and the meaning of ontology elements is usually hardcoded implicitly. However, as it has been previously argued, in an open MAS, different agents have a different vision of the world and, consequently, agents may use different terms for the same meaning or may use the same term to mean different things. Indeed agents' ontologies can differ since they might be developed by different people, at different times, or in different contexts. Moreover, different agents can have different tasks, and bearing in mind that ontologies are task-dependent [23], agents will commit to the different ontologies which best suit the tasks they wish to perform. Similar problems occur when a newly created agent joins some already existing community and it is not aware of the ontology used. An typical example is a supply chain where new agents may enter an existing community at any time.

So, the question is how do agents understand messages from agents that use a different vocabulary? The answer is simply illustrated in Figure 4.4, where two agents, committed to two different ontologies, the ontology *vehicles* and the ontology *transport*, want to communicate each other. The ontology *vehicles* contains the concept `Car`, while the ontology *transport* contains the concept `Automobile` - the two agents use different terminologies even to express the same entity. Imagine that the message of the first agent contain the word *car*. In order the second agent to understand that message and, eventually, reply, "a reconciliation" of their ontology is required. A variety of alternative architectures to reconcile these ontologies have been proposed. The simplest, "bottom up" approach (see Figure 4.2), is

[1] http://swoogle.umbc.edu

merely to map between terms and expressions defined in a source ontology *vehicles* into terms and expressions defined in a target ontology *transport*. This process is here known as an ontology alignment[2]. The advantages of this approach is its simplicity and flexibility: the ontologies are mapped only to the extent that is required. Moreover, it allows the interoperability while maintaining the agents' autonomy. In our example, we simply map that "`car` means exactly the same thing of an `automobile`".

In contrast to the "bottom-up" approach, Figure 4.3 illustrates an approach where a single common, standard ontology is used as a basis for reconciling the individual ontologies - we map from *vehicles* ontology to the common ontology, then from the common ontology to *transport* ontology. Figure 4.4 shows a variant of the common ontology approach, where there are a number of common ontologies, forming *clusters* of inter-related ontologies [130]. Each individual ontology maps to the common ontology for its cluster, and the common ontologies are mapped to allow the exchange of information and knowledge between the clusters. However, we believe that the identification of common conceptualisations, proposed in the last two approaches, are not flexible and very far from realisation within open multi agents systems: developing and maintaining common ontologies to map all the individual ontologies is too costly; and in some cases, may not even be possible, so parts of some individual ontologies may still need to be mapped directly. For these reasons, they are not considered by this thesis.

In order to align ontologies, first we need to analyze the different forms of ontology heterogeneity, i.e. the mismatches between ontologies. This is the subject of the next section.

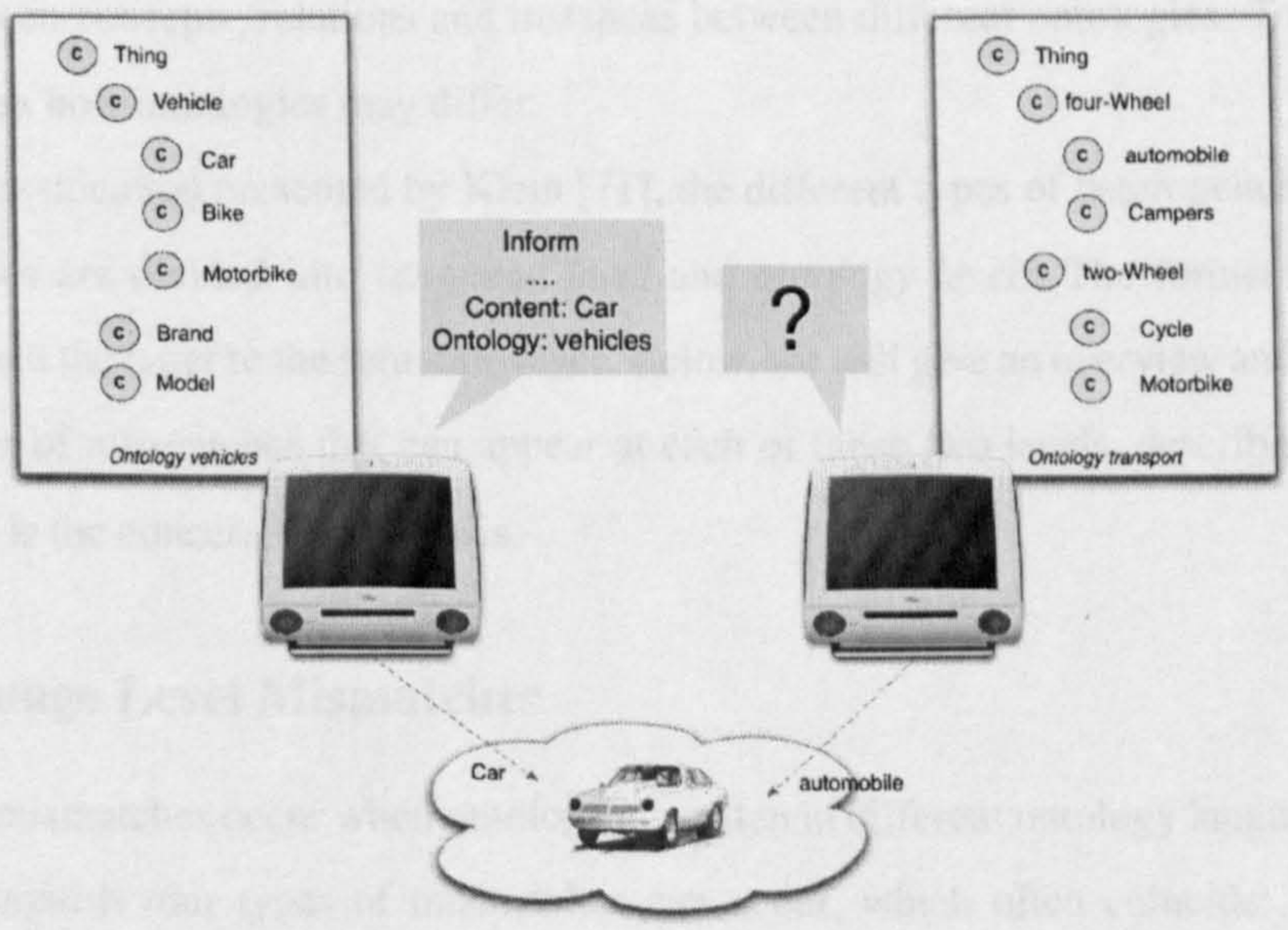

Figure 4.4: Agent Sharing Worlds but Using Different Ontologies.

[2]Ontology alignment is also known as ontology mapping. The two terms are often used interchangeably in the literature.

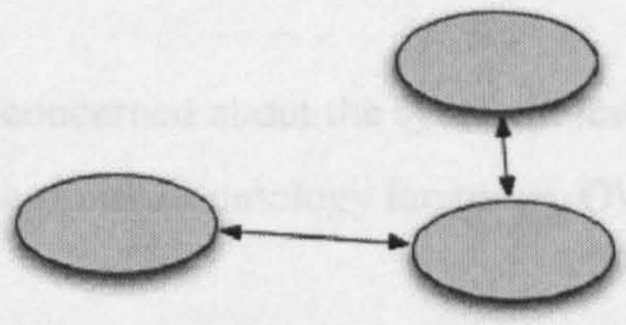

Figure 4.5: Pairwise Mappings between Ontologies.

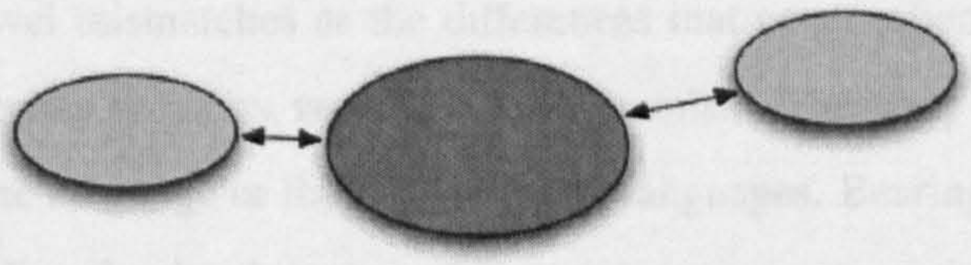

Figure 4.6: Mappings to a Single Common Ontology.

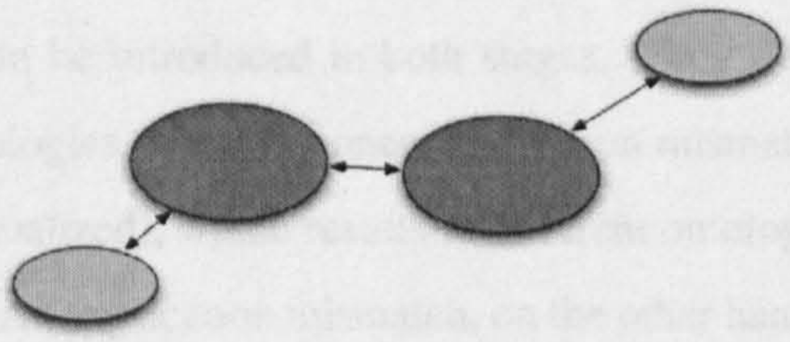

Figure 4.7: Multiple Ontology Clusters with Inter-Claster Mappings.

4.3 Forms of Ontology Heterogeneity

A main issue in the field of ontology alignment is the location and the specification of the overlap and differences between concepts, relations and instances between different ontologies. To this end, in this section we discuss how ontologies may differ.

Following the classification presented by Klein [71], the different types of heterogeneity that can affect agents' ontologies are divided into *language level* and *ontology level*. The former conforms to the syntactic layer, and the latter to the semantic layer. Below, we will give an overview and characterization of different types of mismatches that can appear at each of those two levels, describing in more detail the latter, which is the concern of this thesis.

4.3.1 Language Level Mismatches

Language level mismatches occur when ontologies, written in different ontology languages, are aligned. Klein [71] distinguish four types of mismatches can occur, which often coincide: *syntax* (different ontology languages often use different syntaxes), *logical representation* (difference in representation of logical notions), *semantics of primitives* (difference on the semantics of languages constructors) and *expressivity of the languages* used to represent the ontologies (some languages are able to express things that are not expressible in other languages). An example of language expressivity mismatch is the difference between OWL and DAML+OIL, for example in ability of the former to directly state that

properties can be symmetric.

However, in this thesis, we are not concerned about the syntactic level since we assume that the agents' ontologies are represented using the standard ontology language, OWL.

4.3.2 Ontology Level Mismatches

Klein defines ontology level mismatches as the differences that occur when two or more ontologies, describing (partly) overlapping domains, are aligned. These mismatches may occur whether the ontologies are written in the same language or they use different languages. Bearing in mind the definition of Gruber (see Section 3.1) that the development of an ontology consists of the conceptualisation of the domain and the subsequent explication of this conceptualisation, Visser et al. [129] introduce the idea that ontology heterogeneity can be introduced in both stages. They distinguish between *conceptualization* and *explication* of ontologies, where a conceptualization mismatch is a difference in the way a domain is interpreted (conceptualized), which results in different ontological concepts or different relations between those concepts. An explication mismatch, on the other hand, is a difference in the way the conceptualization is specified. Klein further subdivided the *conceptualisation mismatches* as follows:

- *Concept scope.* Two ontologies may differ from each other as the extensions (that is the set of their instances) related to the same concept (or relation) are not the same, although they are not disjoint. The standard example is the class "employee": several administrations use slightly different concepts of employee, as mentioned by Wiederhold [137].

- *Model coverage and granularity.* An ontology may differ from another as it covers different portion of the domain or it provides a more (or less) detailed description of the same entities. This type of mismatch also occurs when an ontology provides a view point on the some domain, which is different from the view point adopted in another ontology. For example, an ontology about public transport might or might not include taxis (difference in coverage), or might distinguish many different types of trains or not (difference in granularity).

Explication mismatches, on the other hand, can be divided as follows:

- *Paradigm.* This type of mismatch depends on different representation paradigms used to model the same domain. Example of different paradigms that can be used to represent concepts as time, action, plans, causality, propositional attitudes, etc. A reason for this type of mismatches could be the adoption of different knowledge representation paradigms. For example, one model might use temporal representations based on interval logic while another might use a representation based on time points.

- *Concept description.* This type of mismatch depends on different way to model the same concept. For example, a concept can be modeled using a qualifying attribute or by introducing a separate

class. These choices are often influenced by the intended inference system or how the hierarchy is built.

- *Synonym terms*. It occurs when the same concept, attribute, or relation are referred by different terms and/or described by different definitions, although semantically equivalent. A trivial example is the use of the term `car` in one ontology and the term `automobile` in another ontology.

- *Homonym terms*. It occurs when different concepts, attributes, or relations are referred by using the same terms and/or described by same definitions, although semantically different in different contexts. For example, the term `conductor` has a different meaning in a music domain from its meaning in an electric engineering domain.

- *Encoding Values*. It occurs when different ontologies encode values in different ways. A simple example is a date that may be represented as `dd/mm/yyyy` or as `mm-dd-yy`, or distance may be described in miles or kilometers, etc.

The above classification has presented the different types of heterogeneity associated with ontologies, that need to be detected and reconciled in order to align them.

We now turn to the definition of an ontology alignment.

4.4 Ontology Alignment

An Ontology alignment allows agents to preserve their individual ontologies, leaving the agents with full control of their own ontologies. The communication between heterogeneous agents is achieved by a set of defined mappings, which specify the semantic relations between their ontologies.

A more restricted conception of the term is used in [45], that conceive alignments as pairs of elements of the ontologies, together with information on the type of the relation and the confidence in its correctness. Formally, an ontology alignment is defined as follows:

Definition Given two ontologies O and O', an alignment between O and O' is a set of mappings. A mapping is described as a 4-tuple: $m = \langle e, e', n, r \rangle$, where e and e' are the entities (concepts, relations or individuals) between which a relation is asserted by the mapping; r is the relation (e.g., equivalence, more general, etc.) holding between e and e' asserted by the mapping; and n is a degree of confidence in that mapping.

In the literature, an ontology alignment is also defined as a set of morphisms from one ontology to the other i.e. a collection of functions assigning the symbols used in one vocabulary to the symbols of the other (see [145]).

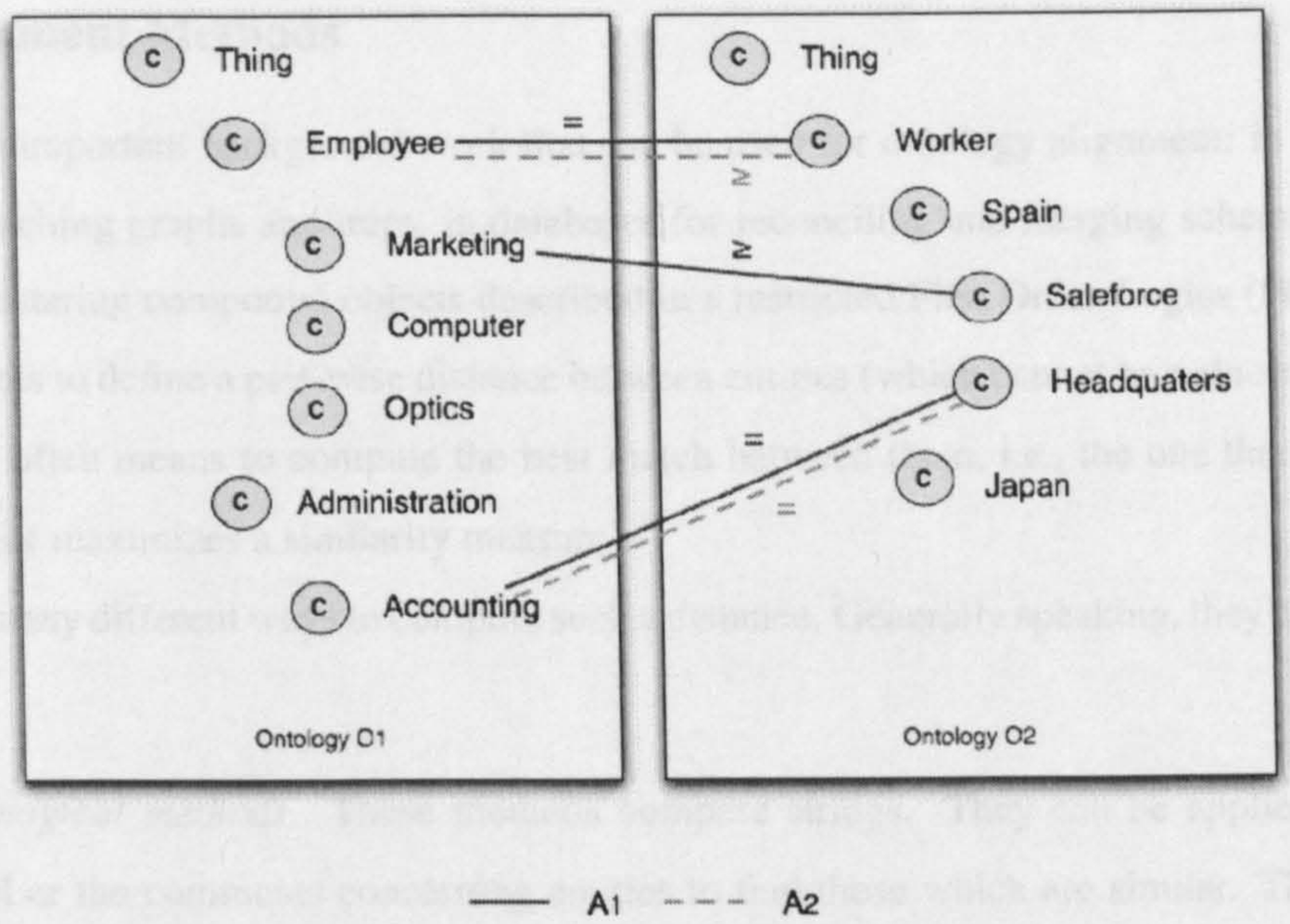

Figure 4.8: Alignment between Two Ontologies (only the classes involved in mappings are displayed).

Note that, since many alignment methods compute a degree of confidence of the relation between entities, n can be provided as a normalized measure. This measure is by no mean characterizing the relationship (e.g., as a fuzzy relation which should be expressed in the relation attribute), but reflects the confidence of the alignment provider in the relation holding between the entities. Currently, we restrict this value to be a float value between 0. and 1.. Moreover, the entities e and e' are identified by an URI and corresponds to some discrete entity of the representation language.

In open environments, no assumptions can be made about the ontologies and their content, thus establishing these mappings cannot rely on the existence of common concepts that can be used as points of reference between ontologies. The mappings that we are considering here are then based solely on the ontological information, that is, on the terms (i.e. the similarity between the labels), the topology (i.e. the ontology structure), and on similarities between the instances.

Figure 4.8 presents two ontologies together with two alignments A_1 and A_2. The alignment A_1 includes the mappings: $m_1 = \langle Employee, Worker, 0.9, \equiv \rangle$, $m_2 = \langle Marketing, Saleforce, 0.73, \supseteq \rangle$, and $m_3 = \langle Accounting, Headquaters, 0.4, \equiv \rangle$.
In contrast, the alignment A_2 includes the mappings: $m_1 = \langle Employee, Worker, 0.68, \supseteq \rangle$ and $m_3 = \langle Accounting, Headquaters, 0.6, \equiv \rangle$.
Often, a mechanism to validate and select the alignments is needed to obtain those mappings that are suitable for an individual agent.

The mappings between different entities of the two ontologies are typically expressed using some axioms expressed in a specific mapping language.

4.4.1 Alignment Methods

There has been important background work that can be used for ontology alignment: in discrete mathematics for matching graphs and trees, in databases for reconciling and merging schemas, in machine learning for clustering compound objects described in a restricted First Order Logics (FOL). Basically, aligning amounts to define a pair-wise distance between entities (which cannot be reduced to an equality predicate) and often means to compute the best match between them, i.e., the one that minimizes the total distance (or maximizes a similarity measure).

There are many different ways to compute such a distance. Generally speaking, they can be classified as in [45]:

- *Terminological methods.* These methods compare strings. They can be applied to the name, the label or the comments concerning entities to find those which are similar. This can be used for comparing class names and/or URI, where the comparison depends on the consideration of character strings only or using some linguistic knowledge. The methods range from finding a common substring in two strings to the use of Wordnet [86].

- *Internal structure methods.* These methods calculate the similarity between entities comparing their internal structure. Thus, they look for criteria such as the value range of their properties (attributes and relations), their cardinality, and their transitivity and/or symmetry. Internal structure based methods are sometimes referred to as constraint based approaches in the literature [99].

- *External structure methods.* These methods compare the relations of the entities with other entities; principally, the comparison can be based on the position of the entities within a taxonomy: if two entities from two ontologies are similar, their neighbours might also be somehow similar.

- *Extensional methods.* They compare the known extension of entities, i.e. the set of other entities that are attached to them (i.e., instances of classes);

- *Semantic methods.* They compare the interpretations (or more exactly the models of the entities). For example, subsumption or satisfiability tests are used, which are well-studied reasoning tasks in description logic. The limitations of these methods are that they only become usable after a preprocessing phase in which a number of concept mappings are declared. When no relations are known to exist between ontologies, these techniques cannot be used to derive ontology alignments.

Therefore, ontology alignments can been derived using different methods and based on different backgrounds. This explains why a huge number of representative approaches for ontology alignment have been proposed in the last few years.

In the next section, we present the state of the art that addresses the problem of dealing with agents with heterogeneous ontologies.

4.5 Agents and Heterogeneous Ontologies: Current Approaches

In recent times, significant research has been performed that addresses the problem of dealing with agents with heterogeneous ontologies. Two main approaches have been proposed in the literature: the first is based on establishing the mappings between the meanings of a set of terms in different ontologies, namely *ontology alignment*; the second is *meaning negotiation*, i.e. solving semantic conflicts by negotiating the meaning of the terms.

This section reviews a selection of notable solutions, dividing our discussion into the two above areas, and shows how the importance of reconciling heterogeneous ontologies has evolved in recent years, with a particular emphasis on their applicability in open MAS.

Note that, since we are dealing with autonomous agents, without human intervention, we restrict our survey on ontology alignment to those works that are labelled as *automatic* ontology mapping tools.

4.5.1 Ontology Alignment

Recently research has been undertaken using supervised machine learning techniques for ontology mapping. The DOGGIE approach [138], which stands for Distributed Ontology Gathering Group Integration Environment, focuses on the machine learning aspects of ontology exchange. The agents learn representations of their own ontologies using a machine learning algorithm and then seek to locate and/or translate semantic concepts by using examples of their concepts to query each other. Thus, the agents are able to teach each other what their concepts mean using their own conceptualization of the world.

HICAL (Hierarchical Concept Alignment system) [63] provides concept hierarchy management for ontology alignment, allowing one concept in a concept hierarchy to align with a concept in another concept hierarchy. HICAL uses a machine-learning method for aligning multiple concept hierarchies, and exploits the data instances in the overlap between the two taxonomies to infer mappings. It uses hierarchies for categorization and syntactic information so that it is capable of categorizing different words under the same concept.

In [4], Afsharchi et al. present a general method for improving communication between agents that have different ontologies. The agents teach each other concepts that might not totally agree by providing positive and negative examples for each of those concepts. Then, the agents use learning methods to learn the concepts. Conflicts are resolved by voting/elimination of examples. The method allows agents that are not sharing common ontologies to establish common grounds on concepts known only to some of them, if these common grounds are needed during cooperation. While the concepts learned by an

agent are only compromises between the views of the other agents, the method nevertheless enhances the autonomy of agents using it substantially.

Several ontology alignment approaches have been developed in the area of the Semantic Web [17]. Some of them present features such as full automation and efficiency that are particularly suitable for open MAS.

QOM (Quick Ontology Mapping) [39], and its extension Foam [40], are based on heuristically calculated similarity of the individual ontology entities, and is distinguished by an emphasis on the efficiency of alignment. QOM uses a dynamic programming approach in order to decrease the run-time complexity. A dynamic programming approach allows the classification of candidate mappings into promising and less promising pairs, and discard some of them entirely to gain efficiency. It allows for the ad-hoc mapping of large, light-weight ontologies.

OLA (OWL Lite Aligner) [46] is dedicated to the alignment of OWL-Lite ontologies. It has been designed with the idea of balancing the contribution of each component that composes an ontology. OLA converts definitions of distances, based on all the input structures, into a set of equations. The algorithm then looks for the matching between the ontologies that minimizes the overall distance between them. OLA uses all the available information (i.e. lexical, internal and external structure, extensional, and data-types) extracted from two given ontologies.

RiMOM (Risk Minimization based Ontology Mapping) [120] is a tool for ontology alignment by combining different strategies, aiming at finding the "optimal" alignment results. The approach is mainly based on the issue of strategy selection for ontology alignment. By strategy selection it means selecting strategies to align ontologies specific to different tasks and according to the characteristics of the ontologies. In the current version, RiMOM implements five strategies: edit-distance based strategy, statistical-learning based strategy, and three similarity-propagation based strategies (including concept-to-concept propagation strategy (CCP), property-to-property propagation strategy (PPP), and concept-to-property propagation strategy (CPP)).

COMA++ [41] is a customizable and generic tool for matching schemas and ontologies, expressed in SQL, XML Schema and OWL. COMA++ offers a GUI to visualize models, and manage the match process and mappings. It supports the combined use of several match algorithms, such as the utilization of shared taxonomies. Furthermore, different match strategies can be applied including various forms of reusing previously match results. This is known as fragment-based match approach, which decomposes a large match problem into smaller problems. COMA++ cannot only be used to solve match problems but also to comparatively evaluate the effectiveness of different match algorithms and strategies.

Falcon [69] is an automatic tool for aligning ontologies, which employs two distinct matchers in combination: a linguistic matching for ontologies, called LMO, and a graph-based matcher, called GMO. GMO takes the alignments generated by LMO as external input and then outputs additional

alignments. Reliable alignments are gained through LMO as well as GMO according to the concept of reliability. Reliability is obtained by observing the linguistic comparability and structural comparability of the two ontologies being compared. It thus enables technologies for finding, aligning and learning ontologies, and ultimately for capturing knowledge by an ontology-driven approach.

This conclude the presentation of the current approaches for ontology alignment. Interested readers are referred to [43] for fuller coverage of the state of art.

Most of the works mentioned above assume that the mappings can be pre-defined before the agents start interacting. Thus, they do not fully accommodate requirements for open multi-agent systems, where agents can dynamically join or leave and no prior assumptions can be made regarding the ontologies. In our case, the ontology mappings can also be provided a priori, but choices must be made *dynamically* due to the openness of the system, and the fact that ontologies and agents' preferences may change over time. Moreover, none of these approaches consider the preferences of agents seeking to align ontologies and most of them use single matching approaches, which is is not sufficient to obtain a satisfactory mapping. However, all of them could be used as a mapping engine in our architecture.

4.5.2 Meaning Negotiation in Open MAS

Meaning negotiation is a discipline that seeks to overcome the problem of enabling agents to understand each other, by allowing them to negotiate the intended meaning of words and expressions while engaged in conversation. The field of meaning negotiation is relatively new, and only a few approaches have addressed the use of argumentation or negotiation between agents with respect to ontology alignments.

[110] has proposed an ontology mapping negotiation to establish a consensus between different agents using the MAFRA alignment framework. The approach is based on the utility functions used to evaluate the confidence in a certain mapping. According to the confidence value, the mapping is accepted, rejected or negotiated. A meta- utility function is also applied to evaluate the effort made in relaxing (increasing) the confidence value so that the mapping rule might be accepted. This confidence value is further applied by each agent in the evaluation of the global agreement. Unfortunately, the approach is highly dependent on the MAFRA framework and cannot be flexibly applied in other environments.

van Diggelen and et. al [126] presents an approach for agents to agree on a common ontology in a decentralised way. The approach assumes that each agent adopts a private ontology and shares an intermediate ontology. The private ontology is used for storing and reasoning with operational knowledge, i.e. knowledge relevant to a particular problem or task at hand. The intermediate ontology is used for the communication of operational knowledge. Communication proceeds by translating from the speaker's private ontology to the intermediate ontology which the hearer translates back again into its own private

ontology. The authors show how to establish such an intermediate ontology, which is the common goal for every agent in the system. In our approach, on the other hand, the result of the negotiation is a set of relations between the terms of different ontologies. This allows for each agent to maintain control over its own ontology but still to be able to communicate with each other.

Beun and colleagues [19] present a computational framework for the detection of ontological discrepancies in multiagent systems by using feedback utterances. In particular, the authors show how ontological discrepancies can be detected during a communicative situation, and how a dialogue agent can react to these observed discrepancies. Depending on the kind of discrepancy, the agent generates a particular feedback message in order to establish alignment of its private ontology with the ontology of the sender. The approach accounts for the detection and handling of ontological discrepancies by the agents themselves, on the basis of their own subjective view on the world, without any reference to a (implicit) third ontology.

Bailin [11] presents an ontology negotiation protocol to provide semantic interoperability in MAS in an automated fashion at run-time. The ontology negotiation protocol enables agents to discover ontology conflicts and to exchange parts of their ontology. Then, through a process of incremental interpretations, clarifications, and explanations, they establish a common basis for communicating with each other. The end result of this process is that each agent will converge on a single, shared ontology. The ontology negotiation protocol uses only lexicons and synonyms to retrieve alternative forms of a concept that can be use for concept matching.

In [87], Morge et al. propose an argumentation framework for inter-agent dialogue to reach an agreement on terminology, which formalizes a debate in which the divergent representations (expressed in Description Logic) are discussed. The proposed framework is stated as being able to manage conflicts between claims, with different relevances for different audiences, in order to compute their acceptance. However, no details are given about how agents will generate such claims.

Malucelli and et al. [82] focusses on the resolution of negotiation conflicts in a B2B domain. They define a set of services for tackling the interoperatiblity problems which arise during inter-agent communication, in particular providing a service for the resolution of ontological conflicts. The similarities between concepts are regarded as bridging elements between the involved ontologies and can be used to support the inter-agent negotiation process. A mediator agent, called OSAg, is responsible for the resolution of all negotiation conflicts that occur within the MAS. Each agent has its own private ontology which remains unchanged during the entire negotiation process. The matched concepts are memorised by the OSAg and kept for future negotiation rounds. The mapping between ontologies is established by comparing, for each pair of concepts, the attributes (grouped by data type), the relation has-part and the descriptions of the concepts. The comparison includes both syntactic and semantic measurements.

4.6 Conclusions

So far, we have seen how the agents' ability to communicate is achieved through speech-act inspired languages, which determine the content of the messages and enable agents to position themselves within a particular interaction context. The actual content of the messages is expressed in an ontology. Consequently, when two autonomous and independently designed agents meet, they have the possibility of exchanging messages, but little chance of understanding each other unless they share a common content language ontology. In this chapter, we have shown that a solution to receive and understand messages expressed in an unknown ontology is by ontology alignment, which involves bringing two ontologies into mutual agreement while keeping them separate.

However, an ontology alignment is not the only way to address the problem of using different ontologies in MAS. In this chapter, we have reported the most representative work in this domain, identifying two main areas: ontology alignment and meaning negotiation.

Although Ontology alignment provides a common layer from which several ontologies could be accessed and hence could exchange information in semantically sound manners, it does not address important requirements and inherent issues for the agent community.

In the next section, we highlight those open issues that have not so far been addressed, providing a detailed summary of the requirements, and present a framework that specifically addresses them.

Part II

An Argumentation Based Approach for Communication in MAS

Chapter 5

Agents Arguing over Mappings

In this chapter we introduce and motivate a mapping-oriented argumentation framework, in which agents dynamically negotiate the meaning of the terms they use to communicate. By argumentation, agents generate and exchange different arguments, that support or reject possible ontology mappings, according to their own preferences.

We start by motivating our framework, introducing the challenges that an ontology alignment process has to address when applied to an agent's context and how this thesis addresses them (Section 5.1). Section 5.2 outlines the prerequisites of our work, including the scenarios, the scope and assumptions that define the boundaries of the system scope that we are to consider herein. Section 5.3 presents our mapping-oriented argumentation framework, introducing in Sections 5.3.1 and 5.3.2 the arguments that the agents are exchanged for accepting and rejecting the mappings, and how to reason about these arguments in a specific argumentation framework: the Value-Based Argumentation Framework. In Section 5.3.3, we define the notion of agreed and agreeable alignments for agents, and we propose a procedure to find them. An example is presented in Section 5.4 to illustrate our approach. Section 5.5 presents an argumentation ontology that we use to represent the alignments and the arguments under consideration. The chapter ends with some conclusions.

5.1 Ontology Alignment in Open MAS - Issues and Challenges

Although the idea of an ontology alignment enables the integration of disparate ontologies and supports interoperability, most of the practical approaches to alignment have not addressed important requirements and inherent issues for the agent community, and thus has achieved limited use by multi-agent systems, especially in open MAS.

In this section we investigate the challenges that need to be addressed in ontology alignment, the elements that will have to be revised when applied to an agent context and introduce how our framework is

going to overcome these challenges. This list of requirements is based on our experience when working with open and dynamic environments, such as the SW and they complement the set of requirements identified by the community so far.

5.1.1 Agent Autonomy over Ontology Alignments

A desirable feature that ontology alignment approaches should fulfill, if considered in the context of agents, is that they should conform to the requirements for supporting an agent and its properties. However, we are not aware of any efforts that directly address this. One common problem is that the majority of these approaches assume that any entities (agents, applications, etc.) that request an alignment will accept all mappings provided[1]. However, this assumption is not going to be acceptable in a multi-agents environment. Agents, has previously defined in this thesis, are autonomous entities, able to make decisions based on information from their environment, other agents or users. The concept of autonomy sets agents apart from conventional objects, which always execute any methods invoked on them. Agents, in contrast, should be able to *refuse* an action. Thus, an important requirement for ontology alignment, in a multi-agent context, is to give the possibility for agents to choose mappings suitable for them. Note that such a decision can be made on the behalf of their user (e.g., in the semantic web environment), or can depend on the environment or simply on the agent's individual views. Independently of the reasons behind such a decision, our approach takes into account the characteristics of autonomy and rationality that are characteristic of agents, by allowing the selection of those mappings that best suit the interests of each agent. On the other hand, the possibility for agents to choose the mappings externally provided, on the basis of their different interests, will give rise to conflicts and inconsistent point of views between agents over the same set of candidate mappings. In order to overcome these conflicts and reach consensual agreements, we use an argumentation-based approach.

5.1.2 Ontology Alignment is Context-Dependent

It is widely agreed that the interpretation of an ontology is *contextual* in the sense that it depends on the context in which the interpretation is carried out [51], and the same domain can be represented by several ontologies, where each ontology is described with respect to a particular context.

Consequently, interacting agents can have different perspectives upon their relevant domain ontologies for a variety of contextual reasons. This may lead to situations where the semantics of communication are not entirely successful, e.g., the concepts used in communications may be superficially similar, but are defined from the point-of-view of entirely different communities of interest.

Therefore, an ontology alignment cannot only depend on the semantics of the entities within an ontology, but it must also depend on the agents' interpretation of these entities and their intended meaning.

[1] However, these entities can still subsequently apply some interpretation and selection to these mappings before use.

As Sowa pointed out in [113]:

> "*the primary connections are not in the bits and bytes that encode the signs, but in the mind of the people who interpret them..... The ultimate source of meaning is the physical world and the agents who use sign to represent entities in the world and their intentions concerning them.*"

An alignment of ontologies should take into account not only the representation of these entities and their relation to the real world entities they are referencing (i.e. their meaning), but also their purpose and usage in the real world (i.e. the context of some community of agents). When taking the contexts into consideration, it is natural that ontology alignments will co-evolve with their communities of use. In addition, the agents' interpretation of contexts in the use and disambiguation of an ontology alignment often plays an important role.

Besana and colleagues [18], for example, suggest the idea that mappings between agents' ontologies should be contextual with respect to their interactions, providing interoperability within the scope of a specific interaction. They propose a framework in which mappings between terms may be hypothesised dynamically as the terms are encountered during interaction.

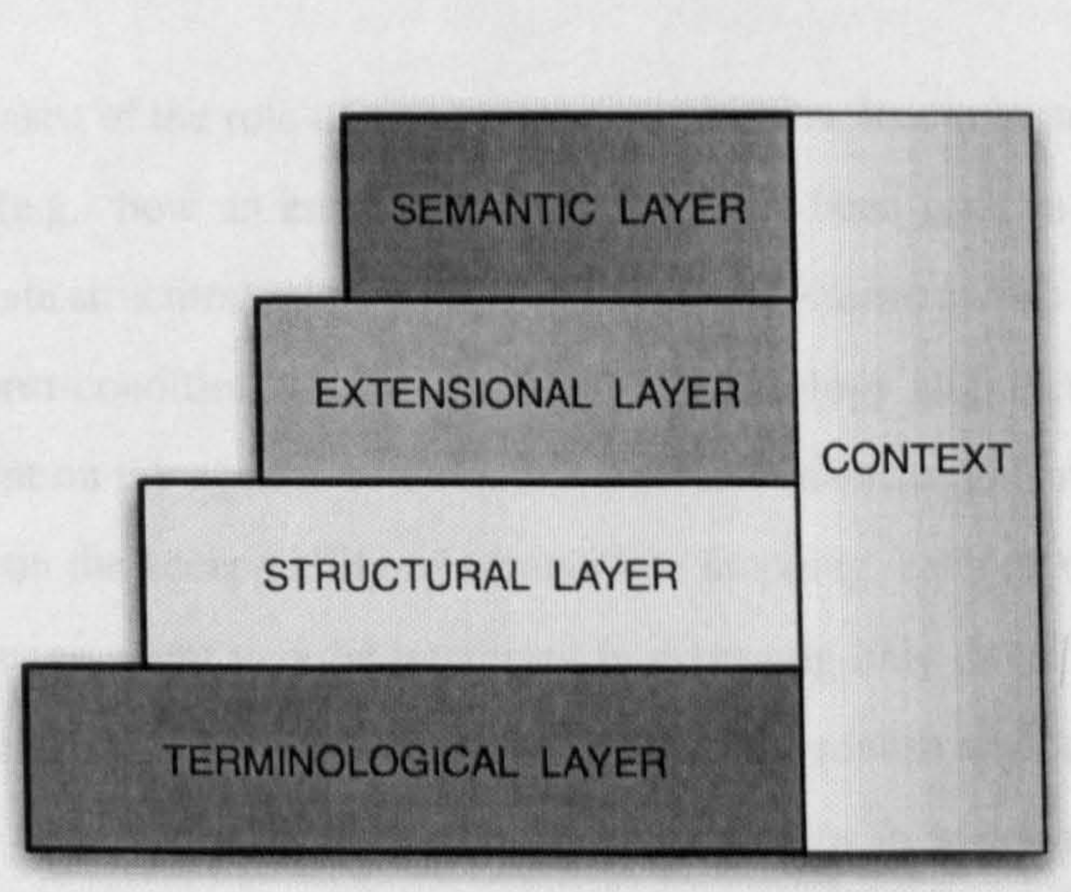

Figure 5.1: Layers of Ontology Alignment.

To this end, an alignment could be seen as consisting of the following different layers, as shown in figure 5.1:

- The *Terminological Layer* corresponds to the comparison of the labels of the entities. This layer is thus based on the textual descriptions of concepts, relations and individuals. This may use techniques such as string matching algorithms, e.g., edit distance and linguistic algorithms.

- In the second layer, the *Structural layer*, we consider the semantic relations between the entities

(*external structure*) and their *internal structure*. The external structure principally compares the position of the entities within their taxonomy: if two entities from two ontologies are similar, their neighbours might also be similar in some respects. On the other hand, the internal structure compares criteria such as the value ranges of their properties (attributes and relations), their cardinality, and their transitivity and/or symmetry.

- The *Extensional layer* compares the set of other entities that are attached to them (in general the instances of classes).
- The next layer, the *Semantic layer* compares the interpretation models of the entities.
- The *Context* applies across all layers of an ontology alignment. It considers how the entities of the ontology are used in some external context, and this implies that the interpretation of a mapping should depend on some additional external information.

Although the importance of "context" has often been emphasized within different research communities, such as cognitive psychology, philosophy and computer science, we do not here adhere to any particular interpretation and definition of this concept. In this thesis, the "context" of an ontology alignment consists of *"an environmental situation and agent state, and which gives a meaning to an alignment"*.

Thus, context can consist of the role of an agent, its capabilities, its environment as defined in [51], the application context (e.g. how an entity of an ontology has been used in the context of a given application), the immediate structural relationships of an ontology term, and so on.

In our approach, a post-condition for the usability of an ontology alignment is that it should have been conceived dependent on the agents' context. For a given context, agents might have different and conflicting perspectives on the acceptability of a candidate mapping, each of which may be rationally acceptable. For example, an agent may be interested in accepting only those mappings that have linguistic similarities, since its ontology is too structurally simple to realise any other type of mismatch. The results proposed in this thesis are, thus, relevant for scenarios in which available ontologies and alignments are used in the context of a particular application or scope.

5.1.3 The Dynamics of Ontology Alignment

In most of the approaches for ontology alignment, the generation of the mappings requires several steps. These can include the definition of an initial alignment, or the training over some examples, and these invariably involve some form of interpretation of preliminary results. This type of approach can only be effective when used to support semantic interoperation at *design time* in closed or semi-open environments, where the agents taking part in the interaction have been identified in advance, where ontology changes are controlled and thus the alignments can be established before the systems interact. However,

these approaches are not sufficient to support semantic interoperation in open MAS, where agents are free to enter and leave the system any time; and no prior assumption can be made about the ontologies to align. Assuming that the ontology alignments are already known in advance is thus not feasible, as it is impossible to foresee all the combinations of agents involved in the interactions. Moreover, ontologies often suffer frequent modifications due to changes introduced in the conceptualisation of the ontology (or in the domain).

On the other hand, aligning whole ontologies is often a lengthy and difficult process that may not be efficient at real time, since the agent interactions should be quick and many can occur at the same time.

One solution to this problem is to enable different agents to agree on the semantics of the terms used during the interoperation, and reaching this agreement can only come through some sort of negotiation process.

Agents, in our approach, have the ability to define and agree upon the semantics of the terms used at *run-time*, by reaching agreements over ontology mappings; where the ontology mappings can be externally provided at run-time or design-time.

5.1.4 Diverse Approaches to Ontology Alignment

Examining a representative sample of the software tools currently available to assist in the alignment process (see Section 4.5) shows that, while a number of promising tools are already available, more work is needed in this area. Furthermore, it is unlikely that one single tool will ever emerge that satisfactorily handles all aspects of ontology alignment: different tools provide mappings, focusing on slightly different aspects such as intuitive linguistic measures, labels, or structural information. Referring to the layer model that we have present above, an ontology alignment depends on a number of different factors, all of which should be taken into consideration when performing ontology reconciliations. Thus, an ontology alignment should not only rely on a single method, but needs to use a range of methodologies. In the architecture we have designed, mappings originate from implementations of several different approaches to ontology alignment, that may employ differing algorithms to calculate potential mappings between ontology entities. In this way, the reconciliation can cover different aspects, allowing our approach to provide a way to select them dynamically.

5.1.5 Ontology Alignment Reuse

Existing ontology alignment work has paid little attention to applying mapping results. As ontologies are understood as means to share and *reuse* declarative knowledge, ontology mappings can be treated as a potential way to make progress in this area. Thus, it is reasonable to store and reuse known alignments.

A rationale behind alignment reuse is that many ontologies to be matched are similar to already matched ontologies, especially if they are describing the same application domain. Eventually, once

an alignment has been determined, it can be saved, and further reused just like any other data. Thus, a (large) repository of mappings has the potential to increase the effectiveness of matching systems by providing yet another source of domain specific knowledge. Reuse of mappings, created by different tools and at different times, however, implies resolving, among others, challenges as the appropriateness of mappings when using them in the new applications.

The architecture envisioned in this thesis provides the infrastructure to reuse ontology mappings. This is achieved by the use of a dedicated agent that has, among the others, the functionality to store the resulting mappings. Our argumentation based approach will allow agents to validate mappings dynamically by using argumentation, in order to take into account the context of the agents and the evolution of the ontologies.

In the following, we describe in detail our approach that addresses these challenges, presenting first the assumptions underlying this work.

5.2 Key Assumptions

In the previous chapters we have presented several topics, such as agents, agent communication, ontologies and heterogeneity. Before we describe our approach, we show how these things fit together and describe our point of departure, by a characterization of the context and the assumptions in which our framework takes place.

We termed our framework as *mapping-oriented argumentation*, which is defined as

> *"a process that dynamically and automatically enables agents, situated in a open environments, with different terminologies, preferences and interests, to reach consensus on the meaning of the terminology they use to interact - by arguing."*

This gives agents the ability to understand each other sufficiently to carry out their objectives. Figure 5.2 depicts the general architecture of our framework.

Firstly, we adopt the essential assumption that a terminology of each agent is represented according to its own conceptualisation, which is explicitly specified according to its own ontology.

Since a considerable effort[2] has been made to increase the degree of standardization in ontology language, here we assume that all agents' ontologies are encoded in the same language, the standard OWL [96].

[2] www.w3.org

We also assume an open, decentralised environment. By decentralised we mean that both control and data are logically and often geographically distributed, but allows specialized services that mediate between agents with interoperability problems. Open means that the platform is free for agents to join and leave, and no standard is imposed on them. Thus, while it is reasonable to introduce a standard for ontology languages, it is impractical to promote the use of a global ontology within a system of open agents: agents will often, quite rightly, differ in the ontology they commit to. The Semantic Web is perhaps the most prominent example of such an environment and, for this reason, is the one we focus on primarily in this research. Others include peer-to-peer computing, pervasive computing and the Grid and, moreover, we believe that many of the insights we develop will equally well transfer to these domains.

"Ontology alignment" is considered as a solution to reconcile heterogeneous agents' ontologies. However, as we argued in Section 1.1, in order to enable agents to successfully communicate, agents must understand and agree on the terminology they use [140]. Therefore, the agents need to reach an agreement on these ontology alignments. Because an agent is assumed to be autonomous and social, every agent in a MAS should be able to solve its own communication problems and enable itself to socially interact when necessary. Thus, the process of reaching agreement should be as automatic as possible and should minimize involvement from human users. Moreover, for obvious efficiency reasons, the argumentation approach is applied only over the ontological terms the agents need in order to understand each other, rather than on the whole ontology.

The potential mappings are provided by a dedicated agent, called an *Ontology Alignment Service (OAS)*. Such an agent has the objective to provide specific services to a multi-agent system, such as to align ontologies, to store the mappings, to translate expressions and so on. The mappings can originate from independent mapping engines that employ differing algorithms to calculate potential mappings between the entities of different ontologies. An alignment consists of a set of mappings $m = \langle e, e', n, R \rangle$ between the two ontologies (see Section 4.3). Each mapping m includes a set of justifications G, that explain why it has been generated. Such information forms the basis for our framework to enable agents to dynamically generate arguments and supply the reasons for their mapping choices. Currently, only a few approaches for ontology alignment provide such justifications [109]. However, tools such as QOM [39] combine different similarity metrics, and these measures can be used to build the required justifications.

We also assume that each agent has a (partial or total) pre–ordering of preferences over different types of ontology mismatches (*Pref*), and a private threshold value ϵ which is compared to the degree of confidence associated with each mapping. These preferences are based on the motivations of the agent, and determine whether a mapping is accepted or rejected by that agent.

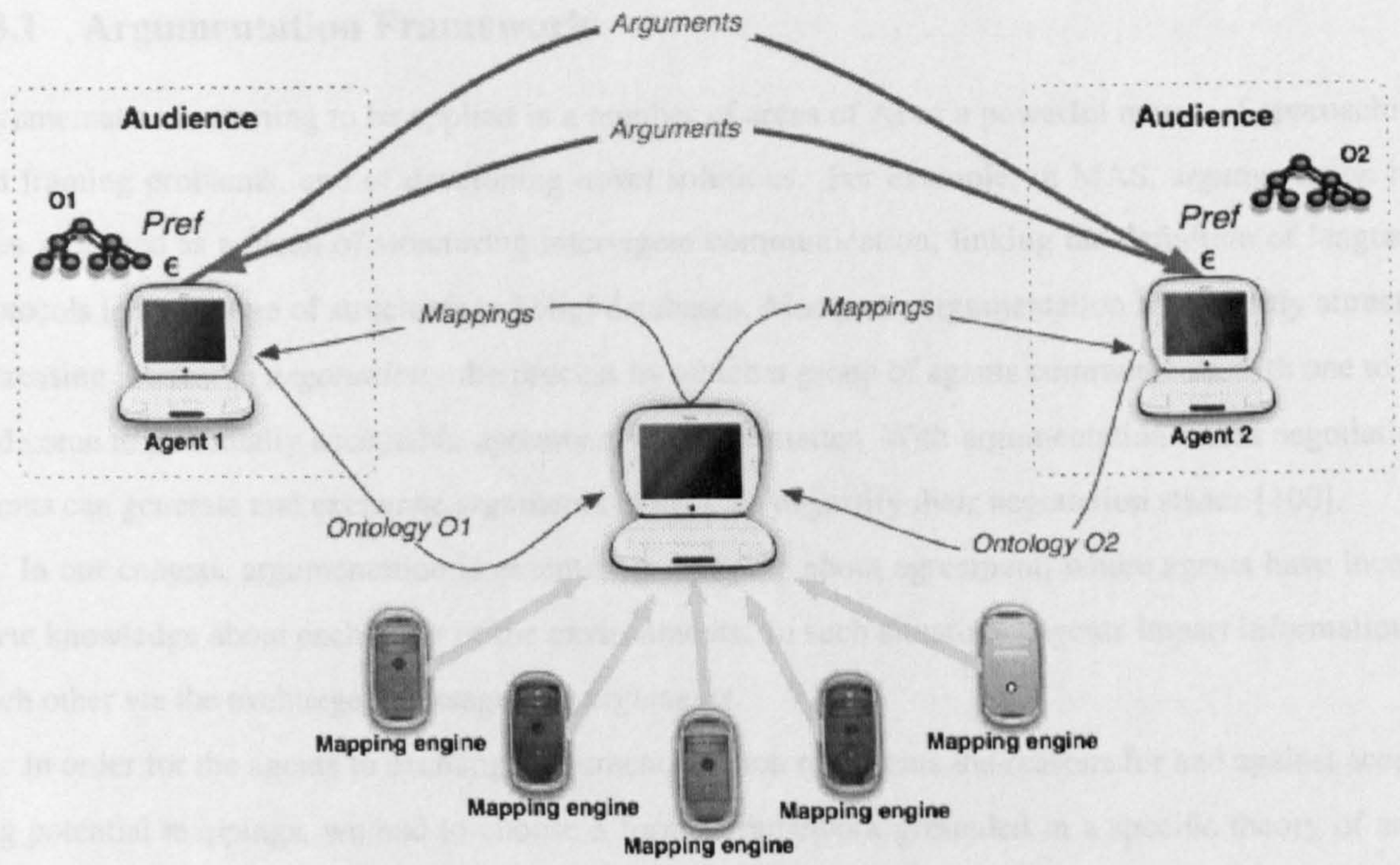

Figure 5.2: Mapping-Oriented Argumentation Architecture.

Moreover, agents are arranged in some sort of group, termed *audiences* which are characterized for a set of agents that have similar preferences. *Audiences* are not aware of one another and the coupling of the group may simply be one agent or it may be an entire society of open agents. Different audiences will consider the mappings differently – one audience may prefer to accept a mapping for some reason, one audience may prefers to reject it for other different motivations.

Finally, we assume that agents are able to locate other agents capable of performing their specific tasks, as already been explored in the research of agent discovery capability and further explored in the next chapter.

In the next section, a mapping-oriented argumentation framework for open multi-agent systems that satisfy the above assumptions, is presented.

5.3 A Mapping-oriented Argumentation Framework for Open MAS

In this section, we define a framework for achieving shared understanding among autonomous agents in open systems. We first introduce the reader to the Value-Based Argumentation framework and its definitions that are used through in this thesis. Then we discuss the exchanged arguments. Finally, we present the process of negotiating, in an attempt to reach consensus, defined in terms of agreed and agreeable alignments.

5.3.1 Argumentation Framework

Argumentation is starting to be applied in a number of areas of AI as a powerful means of approaching and framing problems, and of developing novel solutions. For example, in MAS, argumentation has been proposed as a mean of structuring inter-agent communication, linking the definition of language protocols to the design of structures in belief databases. Moreover, argumentation has recently attracted increasing interest in *negotiation* - the process by which a group of agents communicate with one to try and come to a mutually acceptable agreement on some matter. With argumentation-based negotiation, agents can generate and exchange arguments to back up or justify their negotiation stance [100].

In our context, argumentation is essential to bringing about agreement, where agents have incomplete knowledge about each other or the environments. In such situations, agents impart information to each other via the exchanged message, the *arguments*.

In order for the agents to exchange arguments, which represents the reasons for and against accepting potential mappings, we had to choose a formal framework grounded in a specific theory of argumentation. Several argumentation frameworks have been proposed in the last fews years [72, 89, 35]. The framework we have picked for our purpose is the Value-based Argument Frameworks ($VAFs$) of Bench-Capon [15], which capture the definition of *audience* and preference *values*. In a Value-based Argument Framework, the arguments can be related to underlying values and disagreements between audiences resulting from different preferences among values can be modelled. *Value* and *audience* represent important and distinct elements of reasoning that needs to be accounted in multi-agent systems. Values are used to provide motivating reasons for having given aspirations. An audience refers to an individual (or group) of agents that may have a particular value preference which can differ from that of other audiences and so we can account for differences in opinion, even in matters where facts are agreed upon.

We start with the presentation of argument systems of Dung [35], upon which the $VAFs$ rely.

Definition An Argumentation Framework (AF) is a pair $AF = \langle AR, A \rangle$, where AR is a set of arguments and $A \subset AR \times AR$ is the *attack* relationship for AF. A comprises a set of ordered pairs of distinct arguments in AR. A pair $\langle x, y \rangle$ is referred to as "x *attacks* y". We also say that a set of arguments S attacks an argument y if y is attacked by an argument in S.

An argumentation framework can be simply represented as a directed graph whose vertices are the arguments and whose edges correspond to the elements of A.

Given a set of arguments, it is necessary for the agents to consider which of them they should accept. Given an argument framework, acceptability of an argument is defined as follows:

Definition Let $\langle AR, A \rangle$ be an argumentation framework. Let R, S, subsets of AR. An argument $s \in S$ is attacked by R if there is some $r \in R$ such that $\langle r, s \rangle \in A$. An argument $x \in AR$ is *acceptable* with

respect to S if for every $y \in AR$ that attacks x there is some $z \in S$ that attacks y. S is *conflict free* if no argument in S is attacked by any other argument in S. A conflict free set S is *admissible* if every argument in S is acceptable with respect to S. S is a *preferred extension* if it is a maximal (with respect to set inclusion) admissible subset of AR.

In addition, because there may be multiple preferred extensions, we say that an argument x is *credulously accepted* if there is *some* preferred extension containing it; whereas x is *sceptically accepted* if it is a member of *every* preferred extension.

The key notion here is the *preferred extension* which represents a consistent position within AF, which is defensible against all attacks and which cannot be further extended without becoming inconsistent or open to attack.

In Dung's framework, attacks always succeed. This is reasonable when dealing with deductive arguments, but in many domains, including the one under consideration, arguments lack this coercive force: they provide reasons which may be more or less persuasive. Moreover, their persuasiveness may vary according to their audience. To handle such defeasible reasons giving arguments we need to be able to distinguish attacks from successful attacks, i.e, those which defeat the attacked argument. One approach, taken in [6], is to rank arguments individually. An alternative, which we follow here, is to use a Value Based Argumentation framework (VAF) [15], which prescribes different strengths to arguments on the basis of the values they promote and the ranking given to these values by the audience for the argument. This allows us to systematically relate strengths of arguments to their motivations, and to accommodate different audiences with different interests and preferences.

Definition A *Value-Based Argumentation Framework* (VAF) is defined as $\langle AR, A, \mathcal{V}, \eta \rangle$, where (AR, A) is an argumentation framework, $\mathcal{V}$ is a set of k *values* which represent the types of arguments and η: $AR \rightarrow \mathcal{V}$ is a mapping that associates a value $\eta(x) \in \mathcal{V}$ with each argument $x \in AR$

In section 5.3.2, the set of values $\mathcal{V}$ will be defined as the different types of ontology mismatch, which we use to define the categories of arguments and to assign to each argument one category.

The Value-Based Argumentation Framework makes use of the notion of an *audience*. Audiences as individuated by their preferences between values, since if there is agreement on the ranking of values, there will be agreement on which attacks succeed. Thus, we have potentially as many audiences as there are orderings on $\mathcal{V}$ and the set of arguments will be assessed by an audience in accordance with its preferred values. The definition of audience is presented as follows:

Definition An *audience* for a VAF is a binary relation $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V}$ whose (irreflexive) transitive closure, $\mathcal{R}^*$, is asymmetric, i.e. at most one of (v, v'), (v', v) are members of $\mathcal{R}^*$ for any distinct v, $v' \in \mathcal{V}$. We say that v_i *is preferred to* v_j *in the audience* $\mathcal{R}$, denoted $v_i \succ_{\mathcal{R}} v_j$, if $(v_i, v_j) \in \mathcal{R}^*$.

Let $\mathcal{R}$ be an audience, α is a *specific audience* (compatible with $\mathcal{R}$) if α is a *total ordering* of $\mathcal{V}$ *and* $\forall\, v,\, v' \in \mathcal{V},\ (v, v') \in \alpha \Rightarrow (v', v) \notin \mathcal{R}^*$

In this way, we take into account that different agents - represented by different audiences - can have different perspectives on the same candidate mapping. Acceptability of an argument relative to some audience is defined in the following way:

Definition Let $\langle AR, A, \mathcal{V}, \eta \rangle$ be a VAF and $\mathcal{R}$ an audience.

a. For arguments x, y in AR, x is a *successful attack* on y (or x *defeats* y) *with respect to the audience* $\mathcal{R}$ if: $(x, y) \in \mathcal{A}$ *and* it is *not* the case that $\eta(y) \succ_{\mathcal{R}} \eta(x)$.

b. An argument x is *acceptable to the subset* S with respect to an audience $\mathcal{R}$ if: for every $y \in AR$ that *successfully attacks* x with respect to $\mathcal{R}$, there is some $z \in S$ that successfully attacks y with respect to $\mathcal{R}$.

c. A subset S of AR is *conflict-free with respect to the audience* $\mathcal{R}$ if: for each $(x, y) \in S \times S$, either $(x, y) \notin \mathcal{A}$ or $\eta(y) \succ_{\mathcal{R}} \eta(x)$.

d. A subset S of AR is *admissible* with respect to the audience $\mathcal{R}$ if: S is conflict free with respect to $\mathcal{R}$ and every $x \in S$ is acceptable to S with respect to $\mathcal{R}$.

e. A subset S is a *preferred extension* for the audience $\mathcal{R}$ if it is a maximal admissible set with respect to $\mathcal{R}$.

f. A subset S is a *stable extension* for the audience $\mathcal{R}$ if S is admissible with respect to $\mathcal{R}$ and for all $y \notin S$ there is some $x \in S$ which successfully attacks y with respect to $\mathcal{R}$.

In order to determine whether the dispute is resolvable, and if it is, to determine the preferred extension with respect to a value ordering promoted by distinct audiences, [15] introduces the notion of objective and subjective acceptance as follows:

Definition Given a VAF, $\langle AR, A, \mathcal{V}, \eta \rangle$, an argument $x \in AR$ is *subjectively acceptable* if and only if, x appears in the preferred extension for some specific audiences but not all. An argument $x \in AR$ is *objectively acceptable* if and only if, x appears in the preferred extension for *every* specific audience. An argument which is neither objectively nor subjectively acceptable is said to be *indefensible*.

In [15], an algorithm is presented for computing the preferred extensions of a VAF given a value ordering. In particular, it is shown, given a value ordering, that it is possible to construct an AF equivalent to the VAF, by removing from attacks those attacks which fail because faced with a superior value. It is also shown that, in a VAF, provided it contains no cycles in a single value, the equivalent AF has an unique, non empty preferred extension.

This section concludes the presentation of Value-Based Argumentation framework here, as the account given above suffices for the purposes required in this thesis. The interested reader is referred to [14, 37] for further discussion, examples and computational complexity of a VAF.

5.3.2 Arguments about Mappings

In the last section, we have described the argumentation framework to represent the arguments, the relationships between them and how to choose a "reasonable" subset of arguments from the given set. In this section, we are concerned with the exact nature of an argument and the relationships under consideration. We are only interested on arguments about mappings, which they represent the motivations for the agents on their mapping choices.

Formally, the arguments here are defined as follows:

Definition An argument $x \in AR$ is a triple $x = \langle G, m, \sigma \rangle$ where m is a mapping $\langle e, e', n, R \rangle$; G is the grounds justifying a prima facie belief that the mapping does, or does not hold; and σ is one of $\{+, -\}$ depending on whether the argument is that m does or does not hold.

A key issue is that the interaction between arguments is based on a notion of attack; an argument x is attacked by the assertion of its negation $\neg x$, namely the *counter-argument*, defined as follows:

Definition An argument $y \in AR$ *rebuts* (or attacks) an argument $x \in AF$ if x and y are arguments for the same mapping but with different signs, e.g. if x and y are in the form $x = \langle G_1, m, + \rangle$ and $y = \langle G_2, m, - \rangle$, x counter-argues y and vice-versa.

Moreover, if an argument x supports an argument y, they form the argument $(x \rightarrow y)$ that attacks an argument $\neg y$ and is attacked by argument $\neg x$.

The arguments are clearly identified and grounded on the underlying ontology language OWL.
The grounds G justifying mappings can be extracted from the knowledge in ontologies. This knowledge includes both the extensional and intensional OWL ontology definitions. The intensional knowledge is concerned with the definitions of the concepts, the extraction and statement of their semantic properties and relations. Extensional knowledge refers to application-specific individuals relating to the concepts and roles.
Bearing in mind the type of categorizations underlying ontology matching algorithms that has been presented in Section 4.4.1, the grounds justifying mappings have been classified as following:

semantic (M)**:** the sets of models of two entities do or do not compare.

internal structural (IS)**:** two entities share more or less internal structure (e.g., the value range or cardinality of their attributes).

external structural (ES)**:** the set of relations, each of two entities have, with other entities do or do not compare.

terminological (T)**:** the names of two entities share more or less lexical features.

extensional (E)**:** the known extensions of two entities do or do not compare.

In our framework, we use the types of arguments described above as values for the VAF; hence $\mathcal{V} = \{M, IS, ES, T, E\}$. Therefore, for example, an audience may specify that terminological arguments are preferred to semantic arguments, or vice versa. Note that this may vary according to the context in which an alignment is used and to the nature of the ontologies being aligned. For example, semantic arguments are given more weight in a fully axiomatised ontology, compared to that in a lightweight ontology where there is very little reliable semantic information on which to base such arguments.

The classification presented above, is not meant to constitute an exhaustive typology of arguments, since the type of arguments must be interpreted and are effective within a particular ontology mismatch context and domain.

In order to give a precise structure to these arguments and to construct them, we summarize the reasons for the justification of candidate OWL ontology alignments, shown in Table 5.1 and Table 5.2. The tables represent an (extensible) set of argument schemes, the instantiations of which include AR. Thus it can be looked upon as ways statements that express reasons why an argument justify (or refute) a mapping. The first column of the Tables 5.1.and 5.2 represent the mapping, as it has defined in Section 4.4, the second column represent the sign σ of the argument, the third column represent the grounds justifying the arguments. The last column gives some explanation about the argument.

Thus, an external structural argument for a mapping between two concepts c and c', i.e., $m = \langle c, c', n, \equiv \rangle$ is, for example, that c and c' have mapped sibling-concepts, sub-concepts or super-concepts, formally $\exists m_i = \langle ES(c), ES(c'), n', \equiv \rangle$. On the other hand, an external structural argument against a mapping between two properties p and p', i.e., $\langle p, p', n, \equiv \rangle$ is that there are not sub-properties of p and p' mapped, i.e. $\nexists m_i = \langle ES(p), ES(p'), n', \equiv \rangle$.

An internal structural argument for a mapping between two concepts c and c', i.e., $m = \langle c, c', n, \equiv \rangle$ is, for example, that c and c' have mapped properties, i.e., $\exists m_i = \langle ES(c), ES(c'), n', \equiv \rangle$. An internal structural argument against a mapping between two properties p and p', i.e., $m = \langle p, p', n, \equiv \rangle$ is that the range of the properties p and p' are not mapped, i.e. $\nexists m_i = \langle IS(p), IS(p), n', \equiv \rangle$.

An extensional argument for a mapping between two concepts c and c', is, for example, that the instances of c and c' are mapped, i.e., $\exists m_i = \langle E(c), E(c'), n', \equiv \rangle$. An extensional argument against a mapping between two properties p and p' is that the instances of the properties p and p' are not mapped, i.e. $\nexists m_i = \langle E(p), E(p'), n', \equiv \rangle$.

A terminological argument for a mapping between two properties p and p', is, for example, that their

label share lexical features, i.e., $label(p) \approx_T label(p')$.

A terminological argument against a mapping between two properties p and p', is, for example, that their URI does not share lexical features, i.e., $URI(p) \not\approx_T URI(p')$.

Attacks between these arguments arise when we have arguments for the same mapping but with conflicting values of σ, thus yielding attacks that can be considered symmetric. Moreover, the relations in the mappings can also give rise to attacks: if relations are not deemed exclusive, an argument against inclusion is a fortiori an argument against equivalence (which is more general).

In the following, we show a simple example of arguments.

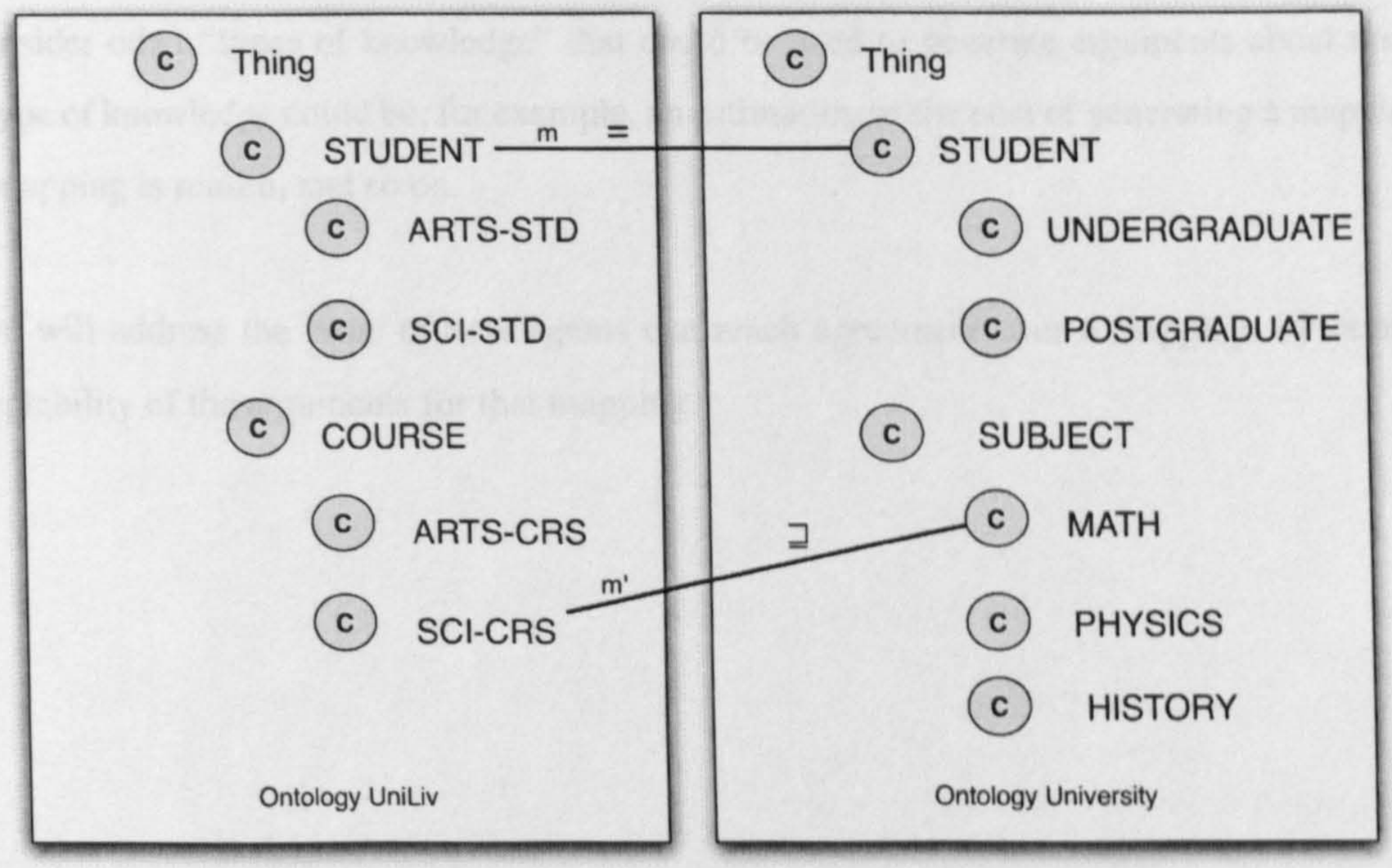

Figure 5.3: Example of Arguments.

Example Consider two different ontologies covering the domain of students, *UnivLiv* ontology and *University* ontology, shown in Figure 5.3.

The ontology *UnivLiv* includes the concepts STUDENT with subclasses ARTS-STD and SCI-STD and COURSE with subclasses ARTS-CRS and SCI-CRS.

The ontology *University* includes the concepts STUDENT with subclasses UNDERGRADUATE and POSTGRADUATE and SUBJECT with subclasses MATH, PHYSICS and HISTORY.

Assuming that two candidate mappings between the two ontologies *UnivLiv* and *University* are the following: $m = \langle$ STUDENT , STUDENT, $0.9, \equiv\rangle$ and $m' = \langle$ SCI-CRS, MATH, $0.65, \sqsupseteq\rangle$.

An argument for accepting the mapping m may be that the labels of the two concepts are identical, formally $\langle label(STUDENT) \approx_T label(STUDENT), +, m\rangle$.

An argument against the mapping m may be that no mapping is defined for some of their sub-concepts, formally $\langle \nexists m_i = \langle ES(STUDENT), ES(STUDENT), n', \equiv\rangle, -, m\rangle$

An argument for accepting the mapping m' may be that some instances of MATH that are mapped with

some of the instances of SCI-CRS, formally $\langle \exists m_i = \langle E(MATH), E(SCI-CRS), 0.6, \equiv \rangle, +, m' \rangle$.

Therefore, in $VAFs$, arguments against or in favour of a candidate mapping are seen as grounded on their type. In this way, we are able to motivate the choice between preferred extensions by reference to the type ordering of the audience concerned. Moreover, the pre-ordering of preferences *Pref* for each agent is over $\mathcal{V}$, that corresponds to the determination of an audience.

Before moving on to the next subsection, it should be stressed that the arguments considered here are computed exclusively on the basis of a set of mappings that has been externally provided by the OAS, and no other information is used. However, a branch of work that could be addressed in future would be to consider other "types of knowledge" that could be used to generate arguments about mappings. Such a type of knowledge could be, for example, an estimation of the cost of generating a mapping, how often a mapping is reused, and so on.

Next, we will address the issue of how agents can reach agreement over a mapping, by establishing the acceptability of the arguments for that mapping.

Mapping	σ	Grounds	Comment
External structural arguments			
$\langle c, c', n, \equiv \rangle$	+	$\exists m_i = \langle ES(c), ES(c'), n', \equiv \rangle$	c and c' have mapped neighbours ES (i.e., super-concepts, sibling-concepts, sub-concepts).
$\langle c, c', n, \sqsubseteq \rangle$	+	$\exists m_i = \langle ES(c), ES(c'), n', \sqsubseteq \rangle$	(some or all) Neighbours ES (i.e., super-concepts, sibling-concepts, sub-concepts) of c are mapped in those of c'.
$\langle c, c', n, \equiv \rangle$	-	$\nexists m_i = \langle ES(c), ES(c'), n', \equiv \rangle$	No neighbours ES of c and c' are mapped.
$\langle c, c', n, \sqsubseteq \rangle$	-	$\nexists m_i = \langle ES(c), ES(c'), n', \sqsubseteq \rangle$	No neighbours ES of c are mapped to those of c'.
$\langle c, c', n, \sqsubseteq \rangle$	-	$\exists m_i = \langle ES(c), ES(c'), n', \sqsupseteq \rangle$	(some or all) Neighbours ES of c' are mapped to those of c.
$\langle p, p', n, \equiv \rangle$	+	$\exists m_i = \langle ES(p), ES(p'), n', \equiv \rangle$	p and p' have mapped neighbours ES (i.e., super-properties, sibling-properties, sub-properties).
$\langle p, p', n, \sqsubseteq \rangle$	+	$\exists m_i = \langle ES(p), ES(p'), n', \sqsubseteq \rangle$	(some or all) Neighbours ES (i.e., super-properties, sibling-properties, sub-properties) of p are mapped in those of p'
$\langle p, p', n, \equiv \rangle$	-	$\nexists m_i = \langle ES(p), ES(p'), n', \equiv \rangle$	No neighbours ES of p and p' are mapped
$\langle p, p', n, \sqsubseteq \rangle$	-	$\nexists m_i = \langle ES(p), ES(p'), n', \sqsubseteq \rangle$	No neighbours ES of p are mapped to those of p'
$\langle p, p', n, \sqsubseteq \rangle$	-	$\exists m_i = \langle ES(p), ES(p'), n', \sqsupseteq \rangle$	(some or all) Neighbours ES of p' are mapped to those of p
Internal structural arguments			
$\langle c, c', n, \equiv \rangle$	+	$\exists m_i = \langle IS(c'), IS(c), n', \equiv \rangle$	The concepts c c' have mapped properties.
$\langle c, c', n, \sqsubseteq \rangle$	+	$\exists m_i = \langle IS(c), IS(c'), n', \sqsubseteq \rangle$	(some or all) Properties of concept c are mapped to those of concept c'
$\langle c, c', n, \equiv \rangle$	-	$\nexists m_i = \langle IS(c'), IS(c), n', \equiv \rangle$	No properties in c and c' are mapped
$\langle c, c', n, \sqsubseteq \rangle$	-	$\nexists m_i = \langle IS(c), IS(c'), n', \sqsubseteq \rangle$	No properties of c are mapped to those of c'
$\langle c, c', n, \sqsubseteq \rangle$	-	$\exists m_i = \langle IS(c), IS(c), n', \sqsupseteq \rangle$	(some or all) Properties of c' are mapped to those of c
$\langle p, p', n, \equiv \rangle$	+	$\exists m_i = \langle IS(p), IS(p), n', \equiv \rangle$	The range and/or the domain of the property p is mapped with those of p'
$\langle p, p', n, \sqsubseteq \rangle$	+	$\exists m_i = \langle IS(p), IS(p), n', \sqsubseteq \rangle$	The range and/or the domain of the property p is mapped with those of p'
$\langle p, p', n, \equiv \rangle$	-	$\nexists m_i = \langle IS(p), IS(p), n', \equiv \rangle$	The range and/or the domain of the properties p and p' are not mapped.
$\langle p, p', n, \sqsubseteq \rangle$	-	$\nexists m_i = \langle IS(p), IS(p), n', \sqsubseteq \rangle$	The range and/or the domain of the properties p and p' are not mapped.
Extensional arguments			
$\langle c, c', n, \equiv \rangle$	+	$\exists m_i = \langle E(c), E(c'), n', \equiv \rangle$	(some or all) instances E of the concepts c and c' are mapped
$\langle c, c', n, \sqsubseteq \rangle$	+	$\exists m_i = \langle E(c), E(c'), n', \sqsubseteq \rangle$	(some or all) instances E of concepts c are mapped to those of c'

Table 5.1: Argument Scheme for OWL Ontological Alignments (continued in Table 5.2.)

Mapping	σ	Grounds	Comment
Extensional arguments			
$\langle c, c', n, \equiv \rangle$	-	$\not\exists m_i = \langle E(c), E(c'), n', \equiv \rangle$	No instances E of concepts c and c' are mapped
$\langle c, c', n, \sqsubseteq \rangle$	-	$\not\exists m_i = \langle E(c), E(c'), n', \equiv \rangle$	No instances E of concepts c are mapped to those of c'
$\langle c, c', n, \sqsubseteq \rangle$	-	$\exists m_i = \langle E(c), E(c'), n', \sqsupseteq \rangle$	(some or all) instances of concepts c' are mapped to those of c
$\langle p, p', n, \equiv \rangle$	+	$\exists m_i = \langle E(p), E(p'), n', \equiv \rangle$	(some or all) instances E of properties p and p' are mapped
$\langle p, p', n, \sqsubseteq \rangle$	+	$\exists m_i = \langle E(p), E(p'), n', \sqsubseteq \rangle$	(some or all) instances E of properties p are mapped to those of p'
$\langle p, p', n, \equiv \rangle$	-	$\not\exists m_i = \langle E(p), E(p'), n', \equiv \rangle$	No instances E of properties p and p' are mapped
$\langle p, p', n, \sqsubseteq \rangle$	-	$\not\exists m_i = \langle E(p), E(p'), n', \equiv \rangle$	No instances E of properties p are mapped to those of p'
$\langle p, p', n, \sqsubseteq \rangle$	-	$\exists m_i = \langle E(p), E(p'), n', \sqsupseteq \rangle$	(some or all) instances of properties p' are mapped to those of p
Terminological arguments			
$\langle c, c', n, \equiv \rangle$ $\langle c, c', n, \sqsubseteq \rangle$	+	$label(c) \approx_T label(c')$	Concepts' labels share lexical features (e.g., synonyms and lexical variants)
$\langle c, c', n, \equiv \rangle$ $\langle c, c', n, \sqsubseteq \rangle$	-	$label(c) \not\approx_T label(c')$	Concepts' labels do not share lexical features (e.g., homonyms)
$\langle c, c', n, \equiv \rangle$ $\langle c, c', n, \sqsubseteq \rangle$	+	$URI(c) \approx_T URI(c')$	Concepts' URIs share lexical features
$\langle c, c', n, \equiv \rangle$ $\langle c, c', n, \sqsubseteq \rangle$	-	$URI(c) \not\approx_T URI(c')$	Concepts' URIs do not share lexical features
$\langle p, p', n, \equiv \rangle$ $\langle p, p', n, \sqsubseteq \rangle$	+	$label(p) \approx_T label(p')$	Properties' labels share lexical features (e.g., synonyms and lexical variants)
$\langle p, p', n, \equiv \rangle$ $\langle p, p', n, \sqsubseteq \rangle$	-	$label(p) \not\approx_T label(p')$	Properties' labels do not share lexical features (e.g., homonyms)
$\langle p, p', n, \equiv \rangle$ $\langle p, p', n, \sqsubseteq \rangle$	+	$URI(p) \approx_T URI(p')$	Properties' URIs share lexical features
$\langle p, p', n, \equiv \rangle$ $\langle p, p', n, \sqsubseteq \rangle$	-	$URI(p) \not\approx_T URI(p')$	Properties' URIs do not share lexical features
$\langle i, i', n, \equiv \rangle$ $\langle i, i', n, \sqsubseteq \rangle$	+	$label(i) \approx_T label(i')$	Individuals' labels share lexical features (e.g., synonyms and lexical variants)
$\langle i, i', n, \equiv \rangle$ $\langle i, i', n, \sqsubseteq \rangle$	-	$label(i) \not\approx_T label(i')$	Individuals' labels do not share lexical features (e.g., homonyms)
$\langle i, i', n, \equiv \rangle$ $\langle i, i', n, \sqsubseteq \rangle$	+	$URI(i) \approx_T URI(i')$	Individuals' URIs share lexical features
$\langle i, i', n, \equiv \rangle$ $\langle i, i', n, \sqsubseteq \rangle$	-	$URI(i) \not\approx_T URI(i')$	Individuals' URIs do not share lexical features

Table 5.2: (continued) Argument Scheme for OWL Ontological Alignments.

5.3.3 Agreed and Agreeable Alignments

The next step is to evaluate the arguments to determine the acceptability of the mappings. However, before this, we first discuss some particular issues associated with the nature of arguments and the VAF.

In $VAFs$ there is always a unique non-empty preferred extension with respect to a specific audience, provided the AF does not contain any cycles in a single argument type [15]. However, an agent may have multiple preferred extensions either because no preference between two values in a cycle has been expressed, or because a cycle in a single value exists. The first may be eliminated by committing to a specific audience, but the second cannot be eliminated in this way. In our domain, where many attacks are symmetric, two cycles are frequent and in general an audience may have multiple preferred extensions. Thus, given a set of arguments justifying mappings organised into an argumentation framework, an agent is able to determine which mappings are acceptable by computing the preferred extensions with respect to its preferences. If there are multiple preferred extensions, the agent must commit to the arguments present in all preferred extensions, but has some freedom of choice with respect to those in some but not all of them. This partitions arguments into three sets: *desired arguments*, present in all preferred extensions, *optional arguments*, present in some but not all, and *rejected arguments*, present in none. If we have two agents belonging to different audiences, these sets may differ. Doutre et al. [34] describe a means by which agents may negotiate a joint preferred extension on the basis of their partitioned arguments so as to maximise the number of desired arguments included, whilst identifying which optional arguments need to be included to support them.

Based on the above considerations, we thus define an *agreed alignment* and an *agreeable alignment* as follows:

Definition An *agreed alignment* is the set of mappings supported by those arguments which are in every preferred extension of every agent.

An *agreeable alignment* extends the agreed alignment with those mappings supported by arguments which are in some preferred extension of every agent.

Moreover, a mapping m is *supported* by an argument x if x in on the form $\langle G, m, + \rangle$.

A mapping m that is not included in the *agreed alignment* and *agreeable alignment* will be *rejected*.

Whilst the mappings included in the agreed alignments can be considered valid and consensual for all agents, the agreeable alignments have a uncertain background, due to the different alternative positions that each agent can take. However, given our context of agent communication, we seek to accept as many candidate mappings as possible. We therefore take into consideration both set of alignments - agreed and agreeable. Next, we show how the agents can achieve them.

Instantiating Argumentation Frameworks

In order to reach agent consensus about ontology alignments, first we have to build the argumentation frameworks and evaluate them to find which arguments are agreed and agreeable and which are rejected. There are four main steps in applying our argumentation approach:

1. Each agent individually constructs an argumentation framework for each candidate mapping, by considering the repertoire of argument schemes available to it, and instantiating these schemes with respect to its interests. Each argument either supports or rejects the conclusion that the mapping is valid. Having established the set of arguments, the agent then determines the attacks between them by considering their mappings and signs, and the other factors discussed above. This step produces several VAFs for reasoning about the candidate mappings.

2. Each agent considers its individual frameworks with all the argument sets of all the other agents and then extends the attack relations by computing the attacks between the arguments present in its framework with the other arguments.

3. Then, for each VAF, the agents determine which arguments are undefeated by attacks from other arguments. The algorithm 2 can be employed for computing the preferred extensions of a VAF given a value ordering. The global view is considered by taking the union of these preferred extensions for each audience.

4. Finally, each agent considers which arguments are in every preferred extension of every audience. The mappings that have only favourable arguments are included in the agreed alignments, and the mappings that have only arguments against are rejected. For mappings whose acceptability cannot be established, agents extend the search space to consider those arguments which are in some preferred extension of every audience. The mappings supported by those arguments are part of the set of agreeable alignments.

The Algorithm 1 shows how to find such agreed and agreeable alignments.

Step 3 introduces the idea of an argumentation framework for a group of agents (or audiences) built from their individual argumentation frameworks. This amounts to making precise the set of arguments of a group of agents and the global attack relation for that group. When we merge n Value-Based Argumentation Frameworks, i.e., $VAF_1 = \langle AR_1, A_1, \mathcal{V}, \eta \rangle, \ldots VAF_n = \langle AR_n, A_n, \mathcal{V}, \eta \rangle$ where each index i corresponds to a specific agent, we define an argumentation framework which reflects how arguments interact for the whole group of agents. The sets of arguments considered acceptable for the group of agents can be defined as follows:

Definition An acceptability relation, denoted by Acc, for a group of agents $Ag_1 \ldots Ag_n$, with respectively n Value-Based Argumentation Frameworks, $VAF_1, \ldots VAF_n$, is a total function from $2^{\bigcup A_i}$ to $\{true, false\}$ which associates each subset S of $\bigcup A_i$ with *true* if S is an acceptable set for $VAF_1, \ldots VAF_n$ and with *false* otherwise. An acceptable set corresponds to the preferred extensions.

A merged Value-Based Argumentation Framework VAF is then simply the union of n Value-Based Argumentation Framework $VAF_1, \ldots VAF_n$, i.e. , $VAF = \bigcup_{i=1}^{n} \langle AR_i, A_i, \mathcal{V}, \eta^* \rangle$ defined by $VAF = \langle \bigcup_{i=1}^{n} AR_i, \bigcup_{i=1}^{n} A_i, \mathcal{V}, \eta^* : \bigcup_{i=1}^{n} AR_i, \rightarrow \mathcal{V} \rangle$.

Thus, given a merged Value-Based Argumentation Framework VAF, the agreed alignments is the set of mappings, which support arguments S are such that $Acc(S) = true$. The preferred extensions of

Algorithm 1 Find agreed and agreeable alignments

Require: a set of agents $Ag_1, \ldots, Ag_n \in MAS$ grouped in a set of audiences $\mathcal{R}_1, \ldots, \mathcal{R}_s$ with $s \leq n$, a set of candidate mappings $m_1, \ldots, m_r$ provided by $OAS \in MAS$;
Ensure: Agreed alignments AG and agreeable alignments AG_{ext}

```
1: AG:=∅
2: AG_ext:=∅
3: Generation of Arguments (see algorithm 4).
4: for all agent Ag_i do
5:    build VAFs,
6: end for
7: for all audience R_j do
8:    for all VAF_i do
9:       compute the preferred extensions for R_j
10:   end for
11: end for
12: compute the agreed Arguments
13: for all x ∈ agreed Arguments do
14:   if x is ⟨G, m, +⟩ then
15:      AG := AG ∪ {m}
16:   else
17:      reject the mapping m
18:   end if
19: end for
20: if ∃ m ∈ M such that m is neither in AG or rejected then
21:   compute the agreeable Arguments
22:   for all x ∈ agreeable Arguments do
23:      if x is ⟨G, m, +⟩ then
24:         AG_ext := AG_ext ∪ {m}
25:      else
26:         reject the mapping m
27:      end if
28:   end for
29: end if
```

a given VAF, which contains r symmetric attacks, can be achieved computing the preferred extension of all possible combination of frameworks obtained from VAF, removing the symmetric attacks. The number of these frameworks is thus exponential to the number of symmetric attacks. For a given audience we are able to construct an AF equivalent to each VAF, by removing from attacks those attacks which fail because faced with a superior value [15]. Bench-Capon in [15] shows that the step for the computation of the preferred extension for an AF is linear to the number of attacks in it. The method is to select all unattacked arguments and include them in the preferred extension. Next remove all arguments attacked by those included so far. Either no arguments remain, or there are some new unattacked arguments. Include these and repeat until no arguments remain. The algorithms 2 and 3 show how to compute the preferred extensions of a VAF, containing symmetric attacks. The algorithm 3 is based on

the algorithm presented in [15].

Algorithm 2 Compute Preferred Extension of a VAF

Require: a value-based argumentation framework VAF, which contains r symmetric attacks, and a set of audiences $\mathcal{R}_1, \ldots, \mathcal{R}_s$.
Ensure: Preferred extensions $\{P_1, \ldots, P_t\}$
1: $VAF_1, \ldots, VAF_t$ are the combinations of VAF achieved removal of the attacks symmetric, with $t = 2^r$.
2: **for all** audience $\mathcal{R}_j$ **do**
3: **for all** VAF_i **do**
4: construct an AF_i equivalent to the VAF_i by removing those attacks which fail because faced with a superior value.
5: **end for**
6: **end for**
7: **for all** $AF_i = \langle AR_i, A_i \rangle$ **do**
8: P_i=*Compute Preferred Extension of* $AF_i = \langle AR_i, A_i \rangle$
9: **end for**

Algorithm 3 Compute Preferred Extension of a AF

Require: an argumentation framework AF
Ensure: Preferred extensions S
1: $S := \{s \in AR_i : \forall y,$ *not defeats*(y, s)
2: $R := \{r \in AR_i : \exists s \in S$ for which defeats (s, r)
3: **if** $R = \emptyset$ **then**
4: return S and Halt
5: **end if**
6: $AR' = AR_i \setminus (S \cup R)$
7: $A' = A_i \setminus (S \times R) \cup (R \times AF) \cup (AF \times R))$
8: Return $S \cup AF' = \langle AR, A \rangle$

The dialogue between agents can thus consist of the exchange of arguments sets, from which agents can individually compute acceptable mappings. If necessary and desirable, these can then be reconciled into a mutually acceptable position through a process of negotiation, as suggested in [34] which defines a dialogue process for evaluating the status of arguments in a VAF, and shows how this process can be used to identify mutually acceptable arguments. In constructing a position, an ordering of values best able to satisfy the joint interests of the agents concerned is determined.

The above technique considers sets of mappings and complete argumentation frameworks. If instead the problem is to determine the acceptability of a single mapping it may be more efficient to proceed by means of a dialectical exchange, in which a mapping is proposed, challenged and defended. Particular dialogue games have been proposed based on Dung's Argumentation Frameworks, e.g. [36], and on VAFs [14].

The next section shows that our framework can allow agents to improve shared understanding through an illustrative example.

5.4 An Illustrative Example

To illustrate the ideas presented in this chaper, we consider the agent Ag_1 which wants to communicate with the agent Ag_2. Ag_1 and Ag_2 use two independent but overlapping ontologies. The first agent, Ag_1 uses the bibliographic ontology[3] from the University of Toronto, based on bibTeX; whereas the second agent, Ag_2, uses the General University Ontology[4] from the French company Mondeca[5]. For space reasons, we will only consider a subset of these ontologies, shown in Figure 5.4, where the first and second ontologies are represented by O_1 and O_2 respectively.

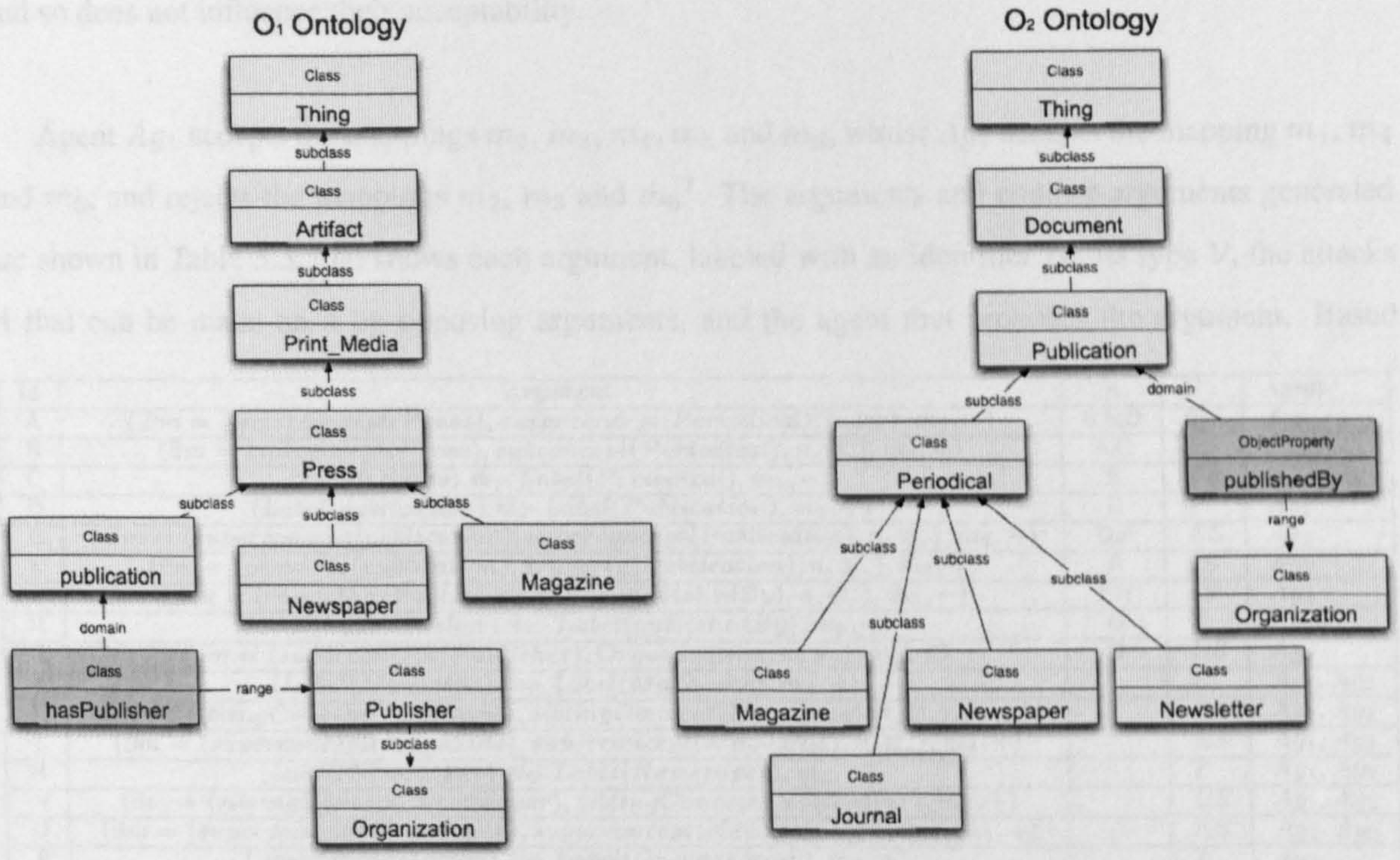

Figure 5.4: Excerpts of O_1 and O_2 Ontologies.

We assume that the only terms initially used by the agent Ag_1 are O_1: $Press$ and O_1: $Publication$. Ag_2 cannot understand the agent Ag_1 until the ontologies are aligned. Thus, the agents invoke the ontology alignment service OAS,which provides the following set of possible mappings, originating from different mapping engines that may employ differing algorithms[6]:

m_1=$\langle O_1$: $Press, O_2$: $Periodical, n, =\rangle$; m_1 states an equivalence mapping with confidence n between the concept $Press$ in the ontology O_1 and the concept $Periodical$ in the ontology O_2.

m_2=$\langle O_1$: $publication, O_2$: $Publication, n, =\rangle$;

m_3=$\langle O_1$: $hasPublisher, O_2$: $publishedBy, n, =\rangle$;

[3]http://www.cs.toronto.edu/semanticweb/maponto/ontologies/BibTex.owl

[4]http://www.mondeca.com/owl/moses/univ.owl

[5]Note that ontology O_2 has been slightly modified for the purposes of this example.

[6]Note that we have listed all of the mappings that are taking part in the argumentation.

m_4=⟨O_1: $Magazine$, O_2: $Magazine$, n, =⟩;

m_5=⟨O_1: $Newspaper$, O_2: $Newspaper$, n, =⟩;

m_6=⟨O_1: $Organization$, O_2: $Organization$, n, =⟩.

Assume now that Ag_1 is agreed with the audience $\mathcal{R}_1$, which prefers terminology to external structure, ($T \succ_{\mathcal{R}_1} ES$). Ag_2 is agreed with the audience $\mathcal{R}_2$, which prefers external structure to terminology ($ES \succ_{\mathcal{R}_2} T$). The pre-ordering of preference *Pref* corresponds to each agent's audience. Moreover, all the above candidate mappings have a degree of confidence n that is above the threshold of each agent, and so does not influence their acceptability.

Agent Ag_1 accepts the mappings m_2, m_3, m_4, m_5 and m_6, whilst Ag_2 accepts the mapping m_1, m_4 and m_5, and rejects the mappings m_2, m_3 and m_6[7]. The arguments and counter-arguments generated are shown in Table 5.3, that shows each argument, labeled with an identifier Id, its type $\mathcal{V}$, the attacks A that can be made on it by opposing arguments, and the agent that proposes the argument. Based

Id	Argument	A	$\mathcal{V}$	Agent
A	$\langle \nexists m = \langle superconcept(Press), superconcept(Periodical), n, \equiv, \rangle, m_1, - \rangle$	B,L,O	ES	Ag_1
B	$\langle \exists m = \langle subconcept(Press), subconcept(Periodical), n, \equiv, \rangle, m_1, + \rangle$	A,C	ES	Ag_2
C	$\langle Label(Press) \not\approx_T Label(Periodical), m_1, - \rangle$	B	T	Ag_1
D	$\langle Label(publication) \approx_T Label(Publication), m_2, + \rangle$	E	T	Ag_1
E	$\langle \nexists m = \langle superconcept(publication), superconcept(Publication), n, \equiv, \rangle, m_2, - \rangle$	D,F	ES	Ag_2
F	$\langle \exists m = \langle property(publication), property(Publication), n, \equiv, \rangle, m_2, + \rangle$	E	IS	Ag_1
G	$\langle \nexists m = \langle range(hasPublisher), range(publishedBy), n, \equiv, \rangle, m_3, - \rangle$	F,H	IS	Ag_2
H	$\langle Label(hasPublisher) \approx_T Label(publishedBy), m_3, + \rangle$	G	T	Ag_1
I	$\langle \exists m = \langle superconcept(Publisher), Organization, n, \equiv, \rangle, m_6, + \rangle$	G	ES	Ag_1
J	$\langle Label(Magazine) \approx_T Label(Magazine), m_4, + \rangle$		T	Ag_1, Ag_2
K	$\langle \exists m = \langle siblingConcept(Magazine), siblingConcept(Magazine), n, \equiv, \rangle, m_4, + \rangle$		ES	Ag_1, Ag_2
L	$\langle \exists m = \langle superconcept(Magazine), superconcept(Magazine), n, \equiv, \rangle, m_4, + \rangle$		ES	Ag_1, Ag_2
M	$\langle Label(Newspaper) \approx_T Label(Newspaper), m_5, + \rangle$		T	Ag_1, Ag_2
N	$\langle \exists m = \langle siblingConcept(Newspaper), siblingConcept(Newspaper), m_5, + \rangle$		ES	Ag_1, Ag_2
O	$\langle \exists m = \langle superconcept(Newspaper), superconcept(Newspaper), n, \equiv, \rangle, m_5, + \rangle$		ES	Ag_1, Ag_2
P	$\langle Label(Organization) \approx_T Label(Organization), m_6, + \rangle$		T	Ag_1

Table 5.3: Arguments For and Against the Mappings m_1, m_2, m_3, m_4, m_5 and m_6.

upon these arguments and the attacks, we can construct the argumentation frameworks which bring the arguments together so that they can be evaluated. These are shown in Figure 5.5, where nodes represent arguments (labelled with their Id) with the respective type value $\mathcal{V}$. The arcs represent the attacks A, whereas the direction of the arcs represents the direction of the attack.

By instantiating the general VAF according to their own preferences, Ag_1 and Ag_2 obtain an argumentation framework, that for sake of simply we have represented as two VAFs, (a) and (b). In the argumentation framework (a), we have two arguments against m_1, and one for it:

- A is against the mapping m_1, since none of the super-concepts of O_1: $Press$ are mapped to any super-concept of O_2: $Periodical$.

- B argues for m_1 because two sub-concepts of O_1: $Press$, (O_1: $Magazine$ and O_1: $Newspaper$),

[7]Next chapter shows how each agent can evaluate a mapping and generate arguments.

are mapped to two sub-concepts of O_2: *Periodical*, (O_2: *Magazine* and O_2: *Newspaper*), as established by m_4 and m_5.

- C argues against m_1, because *Press* and *Periodical* do not have any lexical similarity.

Moreover, we have six arguments supporting the mappings m_4, m_5 and m_6. K, L and M justify the mapping m_4, since, respectively, the labels of O_1: *Magazine* and O_2: *Magazine* are lexically similar; their siblings are mapped, as established by m_5 and their super-concepts; O_1: *Press* and O_2: *Periodical* are mapped by m_1. There is a similar situation for the arguments M, N and O. Clearly, argument A attacks the arguments L and O.

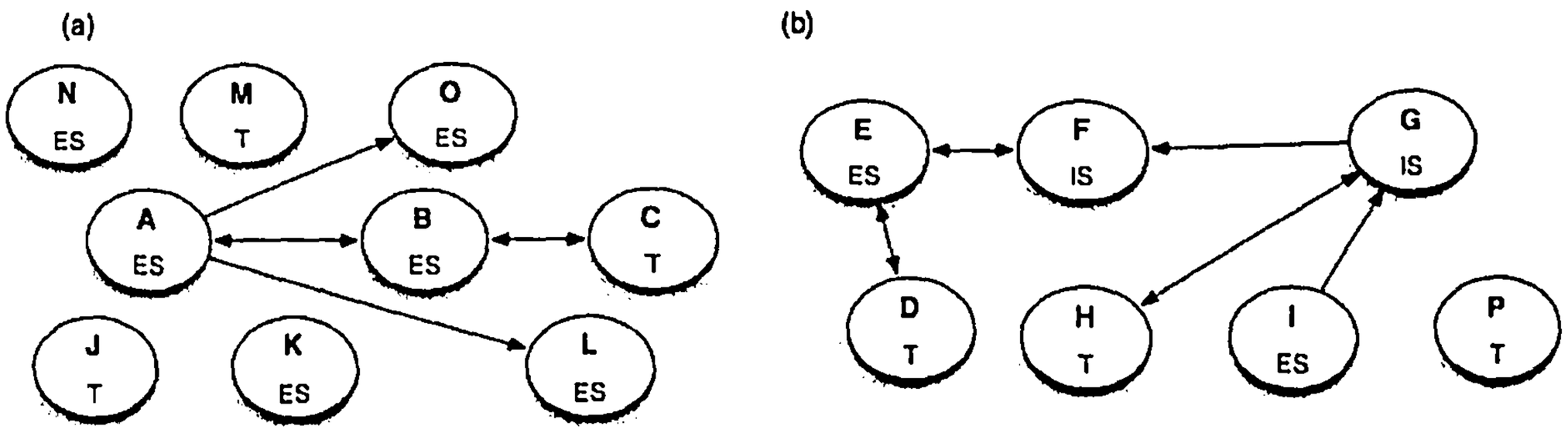

Figure 5.5: Value-Based Argumentation Frameworks.

In the second argumentation framework (b) we relate the following arguments: D justifies the mapping m_2, since the labels of O_1: *publication* and O_2: *Publication* are lexically similar. Their super-concepts, however, are not mapped (argument E). Argument F is based on the fact that O_1: *publication* and O_2: *Publication* have mapped properties, O_1: *hasPublisher* and O_2: *publishedBy*, as defined in m_3. F is then attacked by G, which states that the range of these properties, respectively O_1: *Publisher* and O_2: *Organization*, are not mapped. This is in turn counter-attacked by the arguments H and I. The argument H states that the mapping m_3 is correct, since O_1: *hasPublisher* and O_2: *publishedBy* are lexically similar. The argument I attacks the justification of G stating that the ranges of these properties are similar, since a super-concept of O_1: *Publisher*, (O_1: *Organization*), is already mapped to O_2: *Organization*. The argument P states that O_1: *Organization* and O_2: *Organization* are mapped since their labels are lexically similar.

The above analysis gives different, but sometimes overlapping reasons to argue for and against several candidate mappings. Given the two audiences, $\mathcal{R}_1$ and $\mathcal{R}_2$, the preferred extensions for the union of the argumentation frameworks (a) and (b) are shown in Table 5.4. Therefore, the arguments that are accepted by both audiences are $\{I, H, J, K, M, N, P\}$. Arguments A, C, D, E, and F are, however, all potentially acceptable, since both audiences can choose to accept them, as they appear in some preferred extension for each audience. This means that the mapping m_1 will be rejected (since B is unacceptable to $\mathcal{R}_1$), while the mappings m_3, m_4, m_5 and m_6 will be all accepted (they are all

Preferred Extensions	Audience
$\{A, C, J, K, M, N, D, F, I, H, P\}$	$\mathcal{R}_1$
$\{A, C, J, K, M, N, D, F, I, H, P\}$, $\{B, O, L, J, K, M, N, D, F, I, H, P\}$ $\{A, C, J, K, M, N, E, I, H, P\}$, $\{B, O, L, J, K, M, N, E, I, H, P\}$	$\mathcal{R}_2$

Table 5.4: Preferred Extensions.

accepted by $\mathcal{R}_1$ and all acceptable to $\mathcal{R}_2$). m_2 will be acceptable too, because the arguments supporting it are in some preferred extension for these audiences, as defined in section 5.3.3. The *agreed alignment* is then m_3, m_4, m_5 and m_6, while the *agreeable alignment* adds m_2. Following this argumentation process, the two agents, Ag_1 and Ag_2, can now communicate using the five negotiated mappings in order to translate messages from each other.

Interestingly, in this scenario, should an agent wish to reject the mappings m_2 and m_3, it can achieve this by considering a new audience $\mathcal{R}_3$, in which internal structure is valued more than external structure, which is valued more than terminology ($IS \succ_{\mathcal{R}_3} ES \succ_{\mathcal{R}_3} T$). In this case, the preferred extension from framework (b) is $\{E, G, I, P\}$, since the new preference allows G to defeat H and resist I. G will also defeat F leaving E available to defeat D. This clearly shows how the acceptability of an argument crucially depends on the audience to which it is addressed.

5.5 An Ontology for the Semantics of Arguments over Mappings

Different formal models have been proposed for structured queries on the arguments and it has been shown that they enhance traceability of design decision, help in conflict resolution and enhance reusability [101]. An ontology is obviously such model. Ontologies are themselves formal models, and recently they have been also used to formalize arguments, such as the Argument Interchange Format (AIF), used for data exchange between Argumentation tools or communication in Multi-Agent Systems (MAS) [26], and the approaches in [112, 22].

Our framework utilizes an argumentation ontology to make the representation of the arguments, mappings and the argumentation process, explicit, machine readable and sharable; agents willing to participate to a argumentation process will commit to the shared ontology, which will model all the arguments exchanged, and the mapping under consideration.

The advantages of using ontologies to represent arguments and mappings are mainly to facilitate argument interchange between agents across and within multi-agent systems and facilitate sharing knowledge about an arbitrary number of mapping engines; but also to support inference over the arguments as well as their relationships. Moreover, since we are applying a dynamic approach to the choice of mappings, the communication implied by our approach requires an agreed common vocabulary for argumentation, with a precise semantics, and hence we need an ontological approach.

The ontology models a number of relevant concepts, each of them highlights a different aspect of

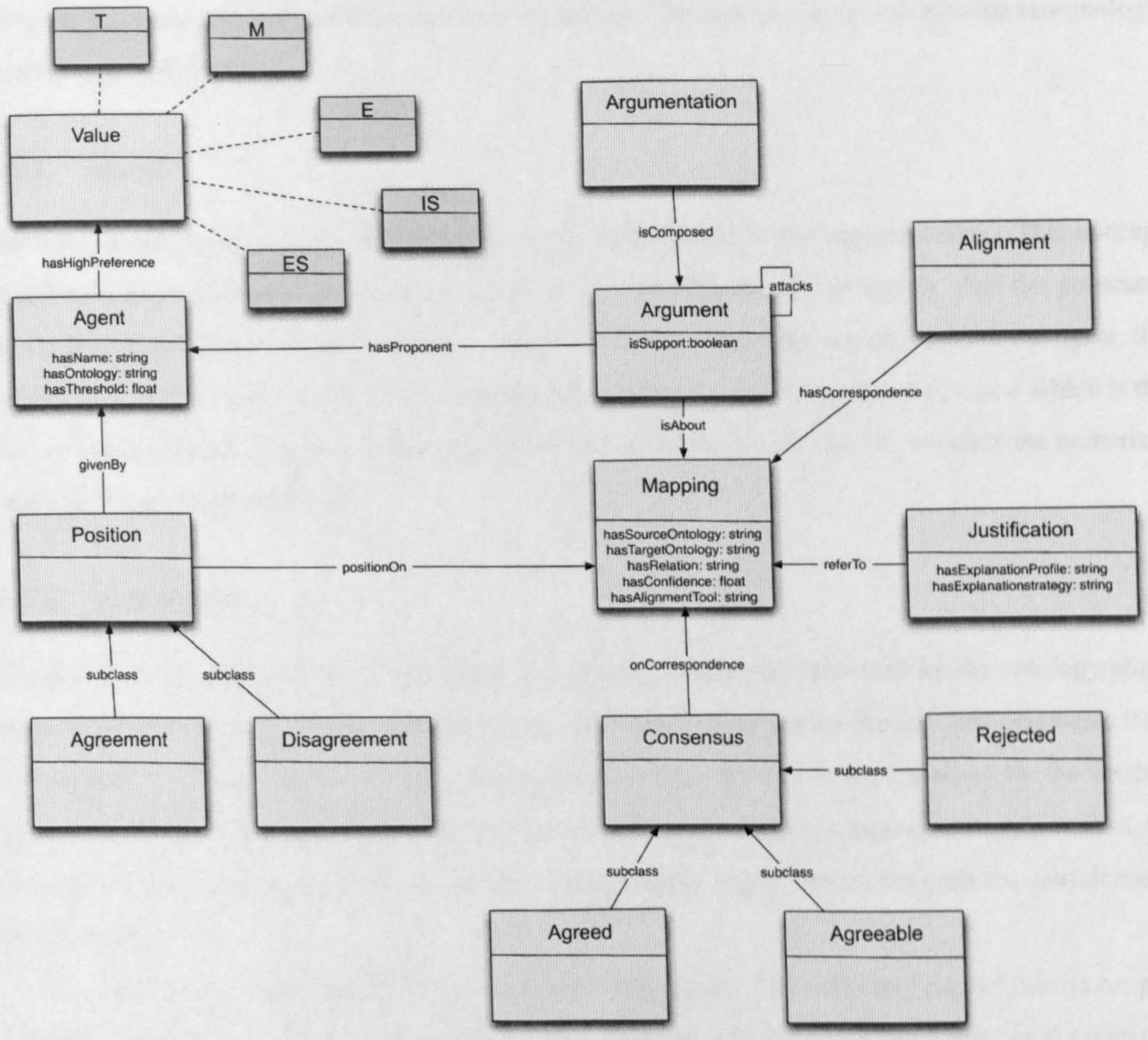

Figure 5.6: Argumentation Ontology

alignments and arguments, which are briefly presented as follows:

- Knowledge concerning the alignments and the mappings;
- Knowledge concerning the justification, i.e. what are the concepts that describe the justifications;
- Knowledge concerning the arguments and relations between arguments;
- Knowledge concerning a single agent (be it human or electronic) or an organisation of agents which participate in an argumentation process.

In the sub-sections that follow, we give an overview of the ontology: the key concepts, the slots associated with these concepts, the relationships between these concepts, and axioms. In the interests of comprehensibility, we do not present all the components of the ontology. Also note that our presentation is informal: we aim to give an overview of the ontology, rather than present all the low-level technical details. Figure 5.6 depicts an overview of the argumentation ontology, in terms of the main classes, the

sub-class relationships between them and their instances. The ontology is in line with the terminology used through in this thesis.

5.5.1 Agents

Our starting concept is `Agent`, which is the entity participating to the argumentation. This concept describes a single agent (be it human or electronic) or an organisation of agents. For the purposes, agents have the following slots: `hasName`, which is a string representating the name of the agent, the `hasOntology`, which is a URI of his committed ontology; the `hasHighPreference` which is the `Value` representing the highest preference of the agent, and the `Threshold`, which is the numerical values of the personal threshold.

5.5.2 Alignment

The next concept is `Alignment`. An alignment is a set of mappings provided by the ontology alignment service. A `Mapping` is the topic of discussion in the argumentation process, and originates from independent mapping engines. A mapping engine has been defined in the ontology by the concept `AlignmentTool`. A mapping consists of a source ontology's entities, a target ontology's entities, the semantic relation holding between the ontology entities and a degree, which suggests the confidence of that mapping.

Each mapping is associated with a number of justifications. The concept `Justification` provides two types of knowledge: knowledge about the description of the justification (the `explanation profile`) and knowledge about how an explanation is derived (the `explanation strategy`).

Let us briefly consider slots and properties of the concept `Mapping`. A `Mapping` has the following slots:

- `hasSourceOntology`: which is the URI representing the source ontology.
- `hasTargetOntology`: which is the URI representing the target ontology.
- `hasConfidence`: either a float or null, with a float indicating the confidence value of that mapping; null indicates that this information is not known.
- `hasRelation`: which represent the semantic relation holding between the ontological entities; the values are currently restricted to: `equivalent`, `disjoint`, `subsumed` and `subsume`. Note that the relations do not necessarily belong to the ontology languages. As such, they do not have to be interpreted by the ontology semantics.
- `derivedFrom` which refers to the `AlignmentTool` that has been provided that mapping.

- `typeAlignment`: this slot is the `Value` that refers to the type of alignment.
- `hasJustification` is the `Justification` that explains why a such mapping has been or has not been generated.

The concept `Justification` has the following slots:

- `hasExplanationProfile`: is a string indicating the reason why that mapping has been provided or why not.
- `hasExplanationStrategy`: is a string indicating how the mapping is derived.
- `referTo`: is the `Mapping` that such justification refers to.

The concept `Value` represents the type of mappings. `Value` is one of `M` , `IS`, `ES`, `T`, `E` (see Section 5.3.2), which are the instances of that concept.

5.5.3 Argument

The concept `Argument` represent the `Position` expressed by the participants agents over a candidate mapping. An argument takes part in an `Argumentation` process, and it is proposed and received by an `Agent`. The concept `Argument` has the following slots:

- `isSupported`: is a Boolean value, indicating whether the argument is supporting a mapping or is against (i.e. a counter-argument); a value of *false* here would indicate that the arguments is against; a value of *true* here would indicate that the arguments is for.
- `isAbout`: is the `Mapping` that the arguments is about.
- `hasProponent`: is the `Agent` that proposes the argument.
- `hasOpponent`: is the `Agent` that receives the argument.
- `attacks`: is the `Argument` that it attacks.

Within an arguments thread the agent can state an `Position`, that clarifies its point of view on a candidate mapping, by explicitly declaring its `Agreement` or `Disgreement`. When an argumentation process has been terminated, a `Consensus` has been reached. A consensus is associated with a `Mapping` and ranges from `agreed`, `agreeable` and `rejected`.

The reader is refer to the appendix B for an OWL version of this ontology.

5.6 Conclusions

In this chapter we have outlined a framework that provides a novel way for agents, with different ontologies, to agree upon an ontology alignment. Rather than considering that all agents will accept indifferently all ontology mappings that have been externally provided as an off-line, design-time process, we have argued that in an open multi-agent system the acceptability of such mappings should be made as a dynamic, run-time process. This allows agents to come to a mutual understanding about the terms they used to communicate and to tailor the meaning of these terms to the context in which they find themselves. This is obtained by application of Argumentation Theory.

In our framework, the acceptability or rejection of candidate mappings are based on the ontological knowledge and the agent's preferences. This gives agents the ability to understand each other sufficiently to carry out their objectives. Argumentation is based on the exchange of arguments, against or in favour of a mapping, that interact with each other using an attack relation. Each argument instantiates an argumentation schema, and utilises domain knowledge, extracted from extensional and intensional ontology definitions. When the full set of arguments and counter-arguments has been produced, the agents consider which of them should be accepted. As we have seen, the acceptability of an argument depends on the ranking - represented by a particular preference ordering on the type of arguments.

We have further presented an argumentation ontology to make the representation of the arguments, mappings and the argumentation process, explicit, machine readable and sharable for the agents and for different mapping engines.

Having specified the framework, the following chapter provides an overview of the MAS architecture we have envisaged for our approach, the structure of the agents and a protocol for evaluating the acceptability of a potential mapping.

Chapter 6

Mapping-oriented Argumentation - Architecture and Protocol

Bearing in mind the global picture of what multi-agent systems are and where they reside in the family of distributed AI, in this chapter we have a closer look at the multi-agent environment for our mapping-oriented argumentation approach.

First, in Section 6.1, we introduce in detail a multi-agent system-based architecture and then, in the same section, we focus on the communication mechanism. Argumentative agents and their capabilities for argument generation are presented in Section 6.2. Section 6.3 describes an argumentation protocol that the agents adhere to when they negotiate. We end this chapter with some conclusions.

6.1 A MAS Architecture for the Argumentation Based Approach

In this section, we describe the multi-agent system architecture that supports inter-agent argumentation over ontology mappings, with the emphasis on the agent communication infrastructure. Our open MAS architecture is shown in Figure 6.1.

In the architecture we envisage, some agents can provide intelligent support and advanced services to users, some can provide complex problem solving capabilities with respect to a given application domain and some are mediators between applications. Our architecture is completely open and dynamic, allowing agents to connect themselves to the system or to leave it and to register/unregister new services at run-time.

Having considered a general decentralised and dynamic architecture, where both control and data are logically and often geographically distributed, we are faced with the problems of agents come from

different designers or different vendors, whose ontology, protocols or mechanisms are not often foreseen. Therefore, agents may be heterogeneous on different aspects. In our architecture, agents are heterogeneous on some or both the following components:

- *Ontologies.* Agents with different ontologies that view the world differently.
- *Preference.* Different agents may have different preferences over the same set of candidate mappings.

Whilst, agents are homogeneous on the following components[1]:

- *Communication language.* Agents require a common language that facilitates communication between them. Communication languages such as FIPA ACL and KQML, which are similar in syntax and semantics, are both suitable candidates. However, it is worth noting that while both languages offer the benefits of being more or less standard agent communication languages, they fail to capture all utterances needed in an argumentation interaction. Therefore, in this thesis we consider only a small number of locutions to express the desire to enter or leave an argumentation interaction, to provide an explicit critique to a proposed mapping or to request or propose an argument for or against. These locutions are generic enough to be used on their own, or incorporated into other interaction protocols or ACLs (including FIPA ACL)[2].
- *Argumentation mechanism of an agent.* Although, agents may have different roles, agents are may homogenous in how they are specified, and how they reason about interaction. In particular, agents share the same mechanism to evaluate mappings, generate arguments and update their mental state accordingly.
- *Protocol.* The dialogues between agents adhere to the same communication and argumentation protocol. Indeed, agent does not only commit to a common agent communication language, but must also agree to use a particular protocol when it negotiates.

The argumentation mechanism of an agent and the argumentation protocol are described, respectively, in Sections 6.2 and 6.3.

An agent can communicate with another in order, for example, to request a specific service. The agents are aware of each other's existence (see Section 6.1.1), but have no knowledge about the other agents, for example, about the services that the other may offer. This situation would arise when the agents are able to utilise a registry service, such as the FIPA Directory Facilitator, but are not able to understand the services offered due to the fact they utilise different ontologies. Therefore, the agents

[1]Although we are considering heterogeneous agents, we assume they share some common features. In fact, communication between *completely* heterogeneous agents is paradoxical as the idea of interaction itself presumes some kind of organization. However, the assumptions we have made are still realistic and enable the agents to reach a shared understanding.

[2]Interested readers are referred to [84] for a more specific discussion of the problems associated with FIPA ACLs semantics concerning agent argumentation protocols.

need to reach agreement on an ontology alignment so that they may determine whether the services offered match that required or whether an agent is able to answer to a query.

The ontology alignment centers around the design of a mediating component - one agent that isolates and processes the knowledge needed for configuring different agents' ontologies to communicate together in a particular application. The middle component, known as an ontology alignment service (*OAS*), provides and stores candidate mappings between ontology entities. These mappings originate from independent mapping engines that employ differing algorithms to calculate potential mappings between the entities of different ontologies. Note that the *OAS* does not store every possible pairwise mapping between entities, but applies a threshold to the submitted mappings in order to store only those mappings plausably supported by the ontological knowledge. The assumption of using different mapping engines does not imply any loss of generality, since these are now becoming increasingly available[3].

It is worth noting that mediators must be considered ideally as third-parties, whose the main goal is to approximate different viewpoints, avoiding decisions that could possibly privilege one of the involved parties. *OAS* also monitors all the communication between the agents.

In order to enable the agents to come to agreement on a suitable alignment for the query or service requested, without requiring a complete alignment between the ontologies, the requesting agent specifies which of the entities from its own ontology are involved in the message request, and only seeks to generate an alignment with the ontology of the other agent with regard to these entities. The pattern of communication between the requesting agent, providing agent and *OAS* in the specific case of requesting a service is shown in table 6.1.

1	$Agent_1$ wants a service X and knows which components of its own ontology O_1 are involved in requesting X.
2	$Agent_1$ knows $Agent_2$ exists, but nothing else.
3	$Agent_1$ requests $Agent_2$'s ontology O_2.
4	$Agent_1$ sends $Agent_2$ the terms from O_1 involved in X.
5	Both agents get the candidate mappings from *OAS*.
6	Both agents exchange their arguments sets.
7	Both agents instantiate their argumentation frameworks.
8	Both agents then calculate their preferred extensions.
9	Both agents then determine the agreed alignment, by exchanging these preferred extensions.
10	$Agent_1$ then sends its service request X to $Agent_2$.
11	$Agent_2$ compares X with its service descriptions - $S2_1, S2_2$, etc. using the agreed alignment, and sends any matching service descriptions to $Agent_1$.
12	$Agent_1$ examines these service descriptions using the alignment to confirm that they match the required service X.
13	If a matching service is confirmed, $Agent_1$ sends the service request to $Agent_2$.

Table 6.1: Communication Path between Agents.

[3]See http://oaci.ontologymatching.org/

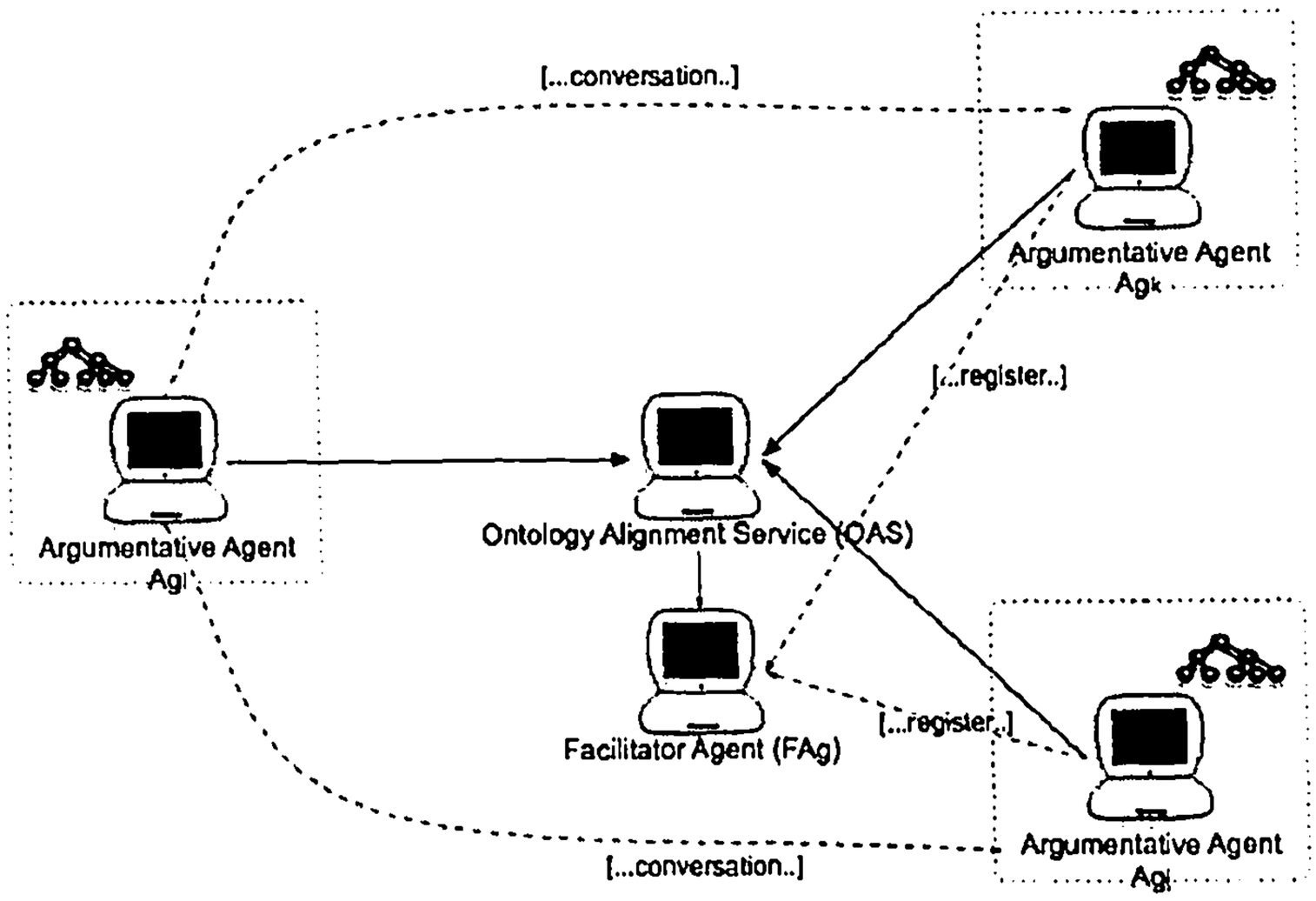

Figure 6.1: Open MAS Architecture.

Our architecture mainly include three types of agents: the agents that wish to communicate with each other and able to engage in "argumentation" (*Argumentative Agents*), the ontology alignment service agent (OAS) and the facilitator agent (FAg). OAS is involved in all the process for facilitating the communications and the argumentation between the agents. The facilitator agent is able to locate agents, services, and data sources over the network. In particular, the facilitator agent offers a yellow-pages directory service[4] to other agents, in which agents can advertise themselves and their functionalities. Its key responsibility is therefore to provide directory services (e.g. a listing of services and resources available at the artefact). The structure of an argumentative agent is described in Section 6.2.

The multi-agent system architecture outlined above is supported by the agent communication infrastructure, explained in more detail in the next subsection.

6.1.1 Communication Mechanism

This section gives an overview on the communication mechanism envisioned in this thesis.
First, each agent is able to communicate with other agents by sending and receiving messages and the communication between them is not always successful. The messages are instances of the following structure:

$$(id_m, S, R, M)$$

where id_m , S, R and M denote respectively a message identifier, its sender, its receiver and the message itself as of set of ontological entities. The sender and the receiver are identifiers of participating agents. The message can be either the description of a particular good/product/service that an agent required or

[4]http://www.fipa.org/

simply a communicative assertion (e.g. a query). The structure of a message, obviously, depends on the agent communication language employed. An example of a FIPA message is given below.

```
(inform
:sender I
:receiver J
:content  "weather(today,raining)"
:language OWL DL
:ontology weatherOntology)
```

The first line in the message is the performative, which specifies the illocutionary force, in this case *inform*. The other lines in a FIPA message specify the sender and the receiver, the content and the language that is used to describe the content and the ontology that underlies it.

We now turn to the acquaintance problem, i.e. how agents become aware of the presence of other agents in the system. In general, agents may be thought to have some a-priori acquaintance information, but in an open society of software agents the most interesting and useful interactions cannot be predefined but emerge at runtime. In order to allow this possibility, the most common approach is the implementation of facilitators or middle agents that may implement a yellow and white pages services, providing:

- basic name-address translation: a middle agent providing this service (also called Agent Name Server) receives a message from any other agent which has to declare its name and network address. In this way the middle agent is able to supply the address of an agent to another one that known its name.

- information on agent capabilities (services): this is a more advanced service that requires a preliminary indication of services offered by agents (and a dynamic update of this information, in case of dynamism in agent capabilities) and allows the expression of queries requiring the names and addresses of agents capable of providing a specific service. Of course, services and the attributes adopted for their description could be included in an ontology.

Note that we are assuming that agents must have a predefined information on how to reach a middle agent.

Another possibility is that an agent wants to know who is able to perform a specific service, and it will query the middle agent asking for agents that notified that they are capable of supplying this service. In this case the middle agent replies that an agent is able to perform that task and thus another conversation will start.

Figure 6.2 shows the communication protocol applied for the messages exchanged between the agents. For example, an agent Ag_1 is looking for same agent with some specific capabilities or services.

Ag_1 sends its request to the facilitator agent FAg. The Facilitator Agent sends an announcement to the agent Ag_2 asking for that request.

The agent Ag_2 that has received the message, may respond in three different ways:

- Ag_2 may respond with another message with the content *"unknown"*. This may happen because an agent is not able to understand the message since it uses a different ontology. If the answer is *"unknown"*, the ontology alignment service agent, who is monitoring all the messages, understands that an agent may satisfy the message but he may not know the meaning of some terms used in the message.

- Ag_2 may respond with a message with the content *"unsatisfiable"*. This may happen because an agent does not have the requested product/service/good description or capabilities.

- If Ag_2 understands the message and the description and it may formulates proposals directly with the agent Ag_1.

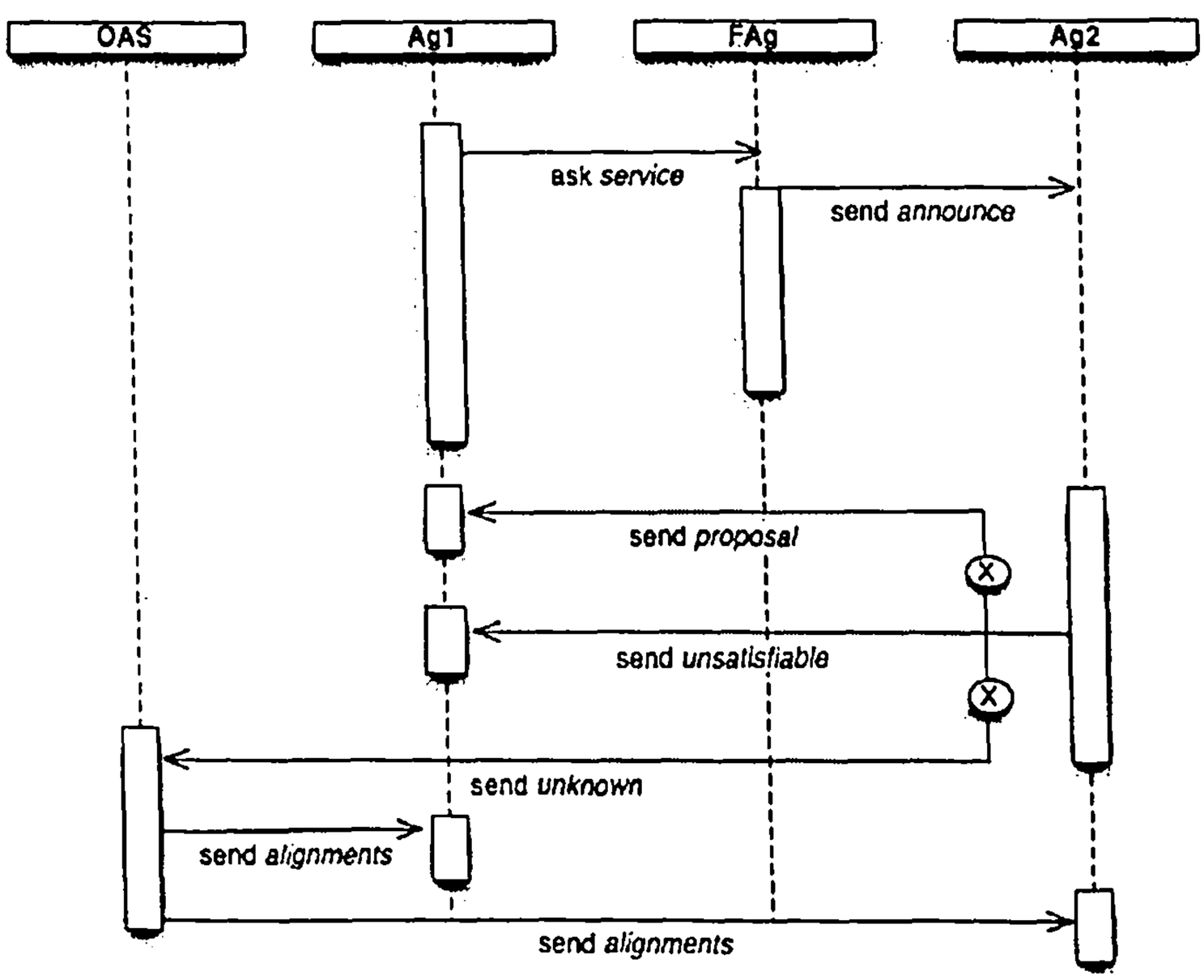

Figure 6.2: Ontology Alignment Protocol.

When an agent sends a message whose content is *"unknown"*, the OAS starts the alignment process, sending the potential mappings to the agents and the argumentation approach takes place. Note that it is still possible that the direct communication fails. This could happen, for example, if the OAS is not able to provides any mappings between the terms used in the message. However, this thesis does not deal with this problem and it may be a subject of future investigation.

6.2 Argumentative Agent

In this section we examine the main characteristics of the agent architecture employed in order to exchange the arguments that have been presented in the last chapter, and the way such arguments are generated and evaluated by the agents.

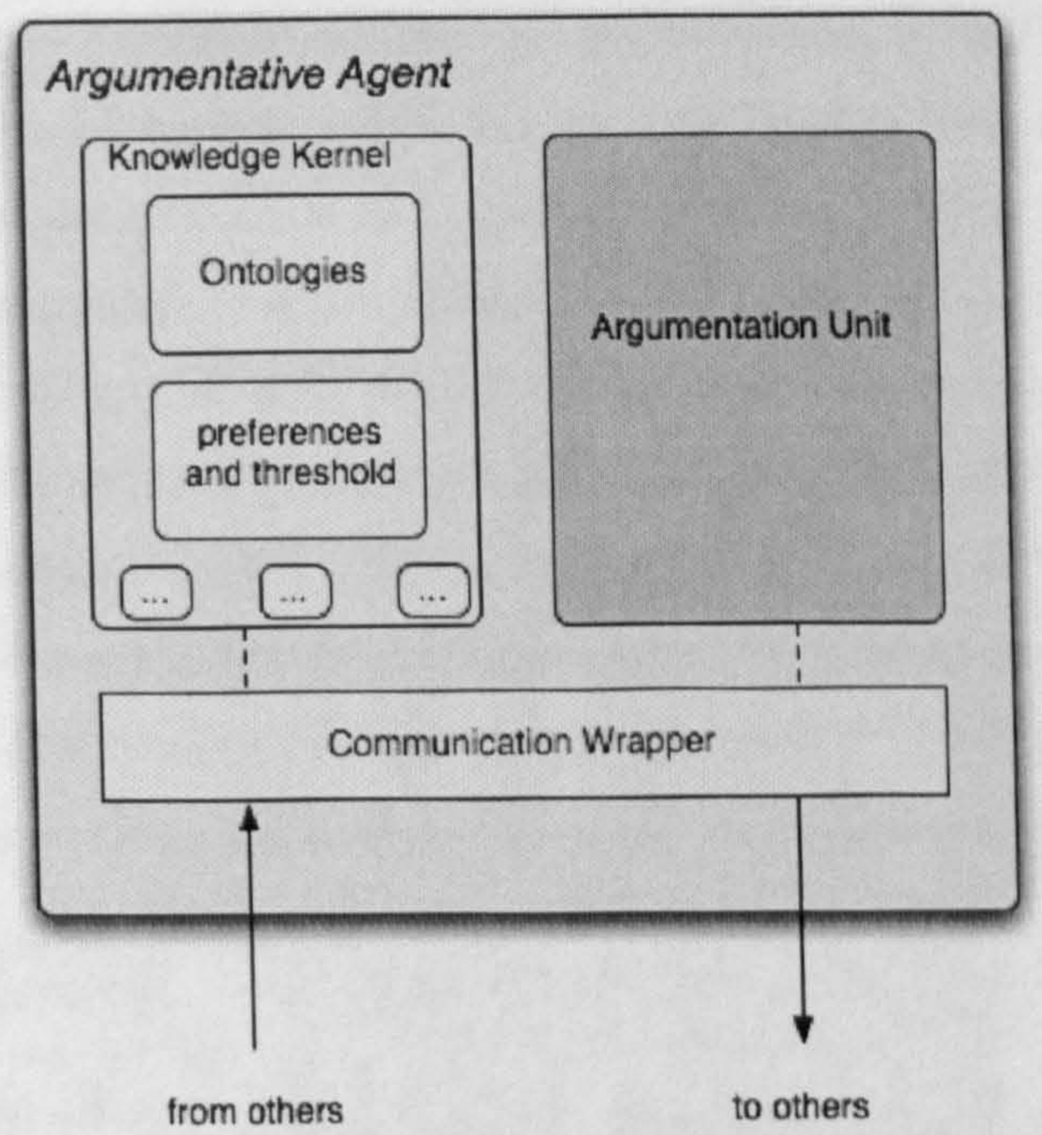

Figure 6.3: Argumentative Agent Structure.

The structure of an argumentative agent is shown in Figure 6.3. An argumentative agent mainly consists of an *Argumentation Unit* and a *Knowledge Kernel*. An argumentative agent communicates with other agents through a *Communication Wrapper*. The *Knowledge Kernel* includes, among others elements, an ontology (or more than one), the preferences over different types of ontology mismatches and a private threshold, which is compared to the degree of confidence associated with each mapping. The preferences and threshold are based on the motivations of the agent, and determine whether a mapping is accepted or rejected. They form part of the mental attitude of an agent and guide the agent's decision, i.e. agent's goal. The *Argumentation Unit* is the core component that allows agents to engage in an argumentation interaction.

Every agent Ag_i is autonomous (can serve different users or providers and fulfill different goals) and has access to its own ontology:

Definition An agent Ag_i is characterised by a 4-tuple $\langle O_i, VAF_i, Pref_i, \varepsilon_i \rangle$ where O_i is the OWL ontology, representing the agent's knowledge; $VAF_i = \langle AR_i, A_i, \mathcal{V}, \eta_i \rangle$ is the Valued-based Argumentation Framework; $Pref_i$ is the private pre-ordering of preferences over $\mathcal{V}$ and ε_i is the private threshold value.

A set of agents $A = \{Ag_1, \ldots, Ag_n\}$ along with the OAS, forms a multi-agent system (*MAS*).

The set of arguments shared by all agents are not necessarily disjoint, and are called *common arguments* AR_c: $AR_c \subseteq \bigcap_{x \in AR_i} AR_i \in VAF_i$. We denote with ARg the set of all arguments available to the MAS with $ARg \subseteq AR_c$. The values $\mathcal{V} = \{M, IS, ES, T, E\}$ are common and shared by all agents.

In order for an agent to be capable of engaging in argumentation, it needs the following capabilities/components: a *Mapping Evaluator* and an *Argument Generator*. Note that, often, the *argument generation* may take place in conjunction with *argument selection*, which is concerned with selecting the *best* argument from the point of view of an agent. In this thesis, all the arguments that have been generated are then all exchanged, thus we are not concerned with any selection of them. A *Mapping Evaluator* encompasses the ability of the agent to assess a mapping presented by another, which strictly depends on its preferences and threshold. This is the fundamental component that allows negotiators' positions to change. An *Argument Generator* is the component that allows the agents to generate the set of possible arguments, either to support or reject a mapping, which is based on the mapping evaluation. The Argument Generator and Mapping Evaluator forms the Argumentation unit, shown in Figure 6.4.

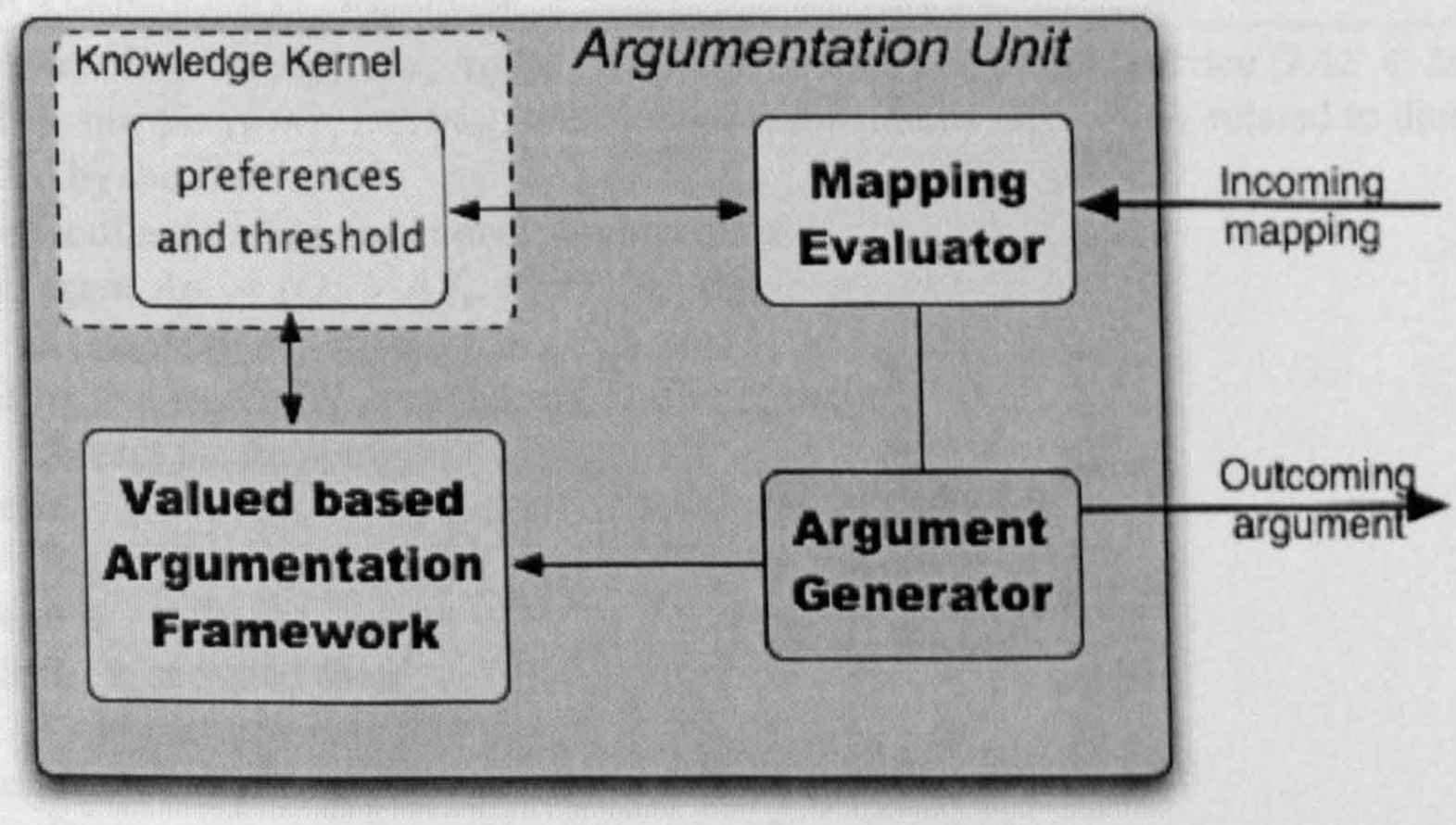

Figure 6.4: Argumentation Unit.

As previously mentioned, the preferences and threshold selected by an agent depend on its context and situation. A major feature of this context is the agent's ontology, and the structural features thereof, such as the depth of the subclass hierarchy and branching factor, ratio of properties to concepts, etc. An agent can then determine its preferences and threshold based on the characteristics of its ontology. For example, selecting a preference for terminological mapping if the ontology is lacking in structure, or preferring an extensional mapping if its ontology is rich in instances. It is worth pointing out that we deliberately do not present to the reader a mechanism that agents (or users) can use to rank their own preferences, based for example on some ontology feature metrics. Indeed, we strongly believe that such

a choice can only be made with regard to the personal judgment of the agent (or user) and cannot be restricted by certain conditions or parameters. However, the analysis of the components of the ontology can be aligned with approaches to ontology evaluation, for instance in [32, 142], and can be formalized in terms of their feature metrics if one wishes.

Given an agent $Ag_i = \langle O_i, VAF_i, Pref_i, \varepsilon_i \rangle$, and a candidate mapping m with a set of justifications G, the agent Ag_i first evaluates the acceptability of m. This is achieved by the *Mapping Evaluator* component, which evaluates the acceptability of a mapping in the following way:

Definition A mapping m is accepted by an agent Ag_i if there exist justifications G for m that correspond to the highest preference $Pref_i$ (with respect to the pre-ordering), and if the degree of confidence in m is greater than its private threshold ε_i. In the other cases, the agent Ag_i rejects the mapping m.

Subsequently, the response is sent directly to the *Argument Generator*. If the mapping m is accepted, then the agent generates a set of arguments in favour: $x = \langle G, m, + \rangle$, by instantiating the argumentation schema, presented in Section 5.3. Otherwise it generates arguments against: $x = \langle G, m, - \rangle$ (see algorithm 4).

Algorithm 4 Generation of Arguments

Require: a set of agents $Ag_1, \ldots, Ag_n \in MAS$, an ontology alignment service $OAS \in MAS$, a set of candidate mappings $m_1, \ldots, m_m$, and a set of justifications $G_i, \ldots, G_t$ related to these mappings, provided by the OAS.
Ensure: a set of arguments and counter-arguments x.

```
for all agent Ag_i = ⟨O_i, VAF_i, Pref_i, ε_i⟩ do
    for all mapping m_j = ⟨e_j, e'_j, n_j, R_j⟩ do
        if n_j ≥ ε_i and ∃ G_k such that G_k = Pref_i then
            Accept the mapping m_j
        else
            Reject the mapping m_j
        end if
        if m_j is accepted then
            Generate arguments for m_j : x = ⟨G_k, m_j, +⟩
        else
            Generate arguments against m_j : x = ⟨G_k, m_j, −⟩
        end if
    end for
end for
```

6.3 Argumentation Protocol

An agent does not only commit to its own ontology and a common agent communication language, but must also agree to use a particular protocol when it negotiates. In fact, for a negotiation to be completed successfully, all parties (i.e. agents) must clearly understand the rules of engagement (i.e., negotiation protocol).

In this section we introduce an argumentation protocol which can be used to evaluate the acceptability of a mapping in our framework. The idea behind the argumentation protocol is to allow n agents ($n \geq 2$) to argue about the acceptability of a potential mapping m, arriving at a joint solution that is based on their preferences and the information they exchange during argumentation. The agents also interact with the OAS agent, which provides the mappings and helps to generate the arguments.

There are a number of argumentation protocols, such as the protocol of Artikis et al. [7] or the Fatio Argumentation Protocol in [84], that have recently proposed. Many of these protocols provides locutions to question and contest information, and rules to discourage disruptive behaviour. The argumentation protocol proposed in this thesis is based upon a revisitation of the negotiation protocol, suggested in [5], where the authors propose a general protocol to govern the high-level behaviour of interacting agents and specify the legal moves in the dialogue.

Our argumentation protocol is formally a tuple $\langle Mapping, Agents, Acts, Replies, Move, Dialogue, Result \rangle$ such that:

Mapping: is a candidate mapping m proposed by OAS, subject to evaluation between the agents.

Agents: is the set of agents taking part in the dialogue: $Agents = \{Ag_1, \ldots, Ag_n, OAS\} \subseteq MAS$.

Acts: is the set of possible argumentation speech acts: $Acts = \{propose, support, contest, withdraw\}$.

Replies: $Acts \rightarrow Acts$ is a mapping that associates to each speech act its possible replies:

- $Replies(propose) = \{support, contest, withdraw\}$
- $Replies(support) = \{support, contest, withdraw\}$
- $Replies(contest) = \{support, contest, withdraw\}$
- $Replies(withdraw) = \emptyset$.

Move: is defined as $Move_i = Act_i(S_i, H_i, c)$, where $S_i \in Agents$ is the agent which makes the move; $H_i \subseteq Agents$ is the set of agents to which the move is addressed and Act_i is the speech act applied to a content c, with $c \in ARg \cup Mapping$. We denote with M the set of all moves that can be built from $Agents$ and $Acts$.

We now define the moves in more details as follows:

- $propose(S_i, H_i, m)$ - denotes that an agent S_i sends a proposal to the agent H_i to evaluate the candidate mapping m. In doing so, S_i creates a dialectical obligation within the dialogue for the agent H_i to provide arguments for or against that mapping.

- $support(S_i, H_i, a \vdash^{+} m)$ - denotes that an agent S_i is able to provide an supported argument $a \in ARg$ for the mapping m to the agent H_i. Thus, the agent S_i seeks to accept that mapping.
- $contest(S_i, H_i, a \vdash^{-} m)$ - denotes that an agent S_i is able to provide an counter-argument $a \in ARg$ for the mapping m to the agent H_i. Thus, the agent S_i seeks to reject that mapping.
- $withdraw(S_i, H_i, a \nvdash^{m})$ - denote that an agent S_i is not able to provide any arguments $a \in ARg$ for the mapping m to the agent H_i. Thus, the agent S_i will exit from the dialogue.

Dialogue: is a finite non-empty sequence of moves $Dialogue = \{M_0, \ldots, M_p\}$ such that:

- $Move_0 = propose(S_0, H_0, c_0)$ with $S_0 = OAS$, $H_0 = \{Ag_1, \ldots, Ag_n\}$ and $c_0 = m$;
- $\nexists i, j \leq p$ such that $Move_i = Move_j$

The first condition says that the dialogue start by the agent OAS presenting the potential mapping $m \in Mapping$. The second condition prevents repeating the same move. This is necessary for avoiding possible loops.

Result: $Mapping \rightarrow \{agreed, agreeable, rejected\}$ returns the result of the acceptability of the mapping m.

Note that M, the set of all moves, is finite since the number of *Agents* and *Acts* are assumed to be finite.

The protocol can be represented as a state transition diagram that gives the various legal states that an agent may be in during the argumentation, and thus the legal transitions between states that an agent is allowed to perform. The state transition diagram of our protocol for two agents is shown in Figure 6.5.

We further assume, in accordance with prior work in agent communication [80, 58], that a Commitment Store CS is associated with each agent, which stores, in a manner which all agents may read, the commitments made by that agent in the course of a dialogue. In particular, we suppose that each agent's CS will have the following parts: $CS.M$ will contain all the mappings proposed during the argumentation, and $CS.Arg$ will contain the set of arguments presented by the agent. In order to avoid agents repeating what they have already uttered during previous runs of the protocol, agents must keep a record of their previous commitments. Thus all the CS are empty only at the first run of the protocol.

Having presented the syntax of argumentation protocol, we next describe how it is works.

The process begins (*State 0*) when the agent OAS makes a proposal to the agents $Ag_1, \ldots, Ag_n$, denoted by $propose(OAS, \{Ag_1, \ldots, Ag_n\}, m)$, where m is the mapping being proposed. Once the mapping has been sent (*State 1*), each agent Ag_i evaluates the acceptability of the mapping m. The agent Ag_i can accept the mapping m, generating an argument to support m, $x_i = \langle G_i, m, + \rangle \in AR$, denoted

by $support(Ag_i, \{Ag_1, \ldots, Ag_n\}, x_i \vdash^+ m)$ (*State 2*). The Ag_i can reject the mapping m, generating a counter-argument $x_i = \langle G_i, m, - \rangle \in AR$, denoted by $contest(Ag_i, \{Ag_1, \ldots, Ag_n\}, x_i \vdash^- m)$ (*State 3*). If the agent does not have any arguments or counter-arguments to propose, then it can withdraw by $withdraw(Ag_i, m)$ and the dialogue terminates (*State 4*).

Moreover, each agent Ag_i checks whether it agrees with the set of mappings $m_1, \ldots, m_k$ used in the arguments that are exchanged. If they have arguments or counter-arguments to present, they start a new dialogue to evaluate each of those mappings by $propose(Ag_i, \{Ag_1, \ldots, Ag_n\}, m_j), \forall j \in \{1, \ldots, k\}$. This protocol iterates until each mapping involved in the initial dialogue on m has been evaluated and when all agents withdraw (*termination of a dialogue*). When the dialogue terminates, each agent builds the argumentation framework, considering the arguments contained in $CS.Arg$ and computing the attacks. Then they check the acceptability of the mapping by computing its preferred extensions. The agents will then exchange their preferred extensions in order to compute if the mapping m is $agreed$, $agreeable$ or $rejected$.

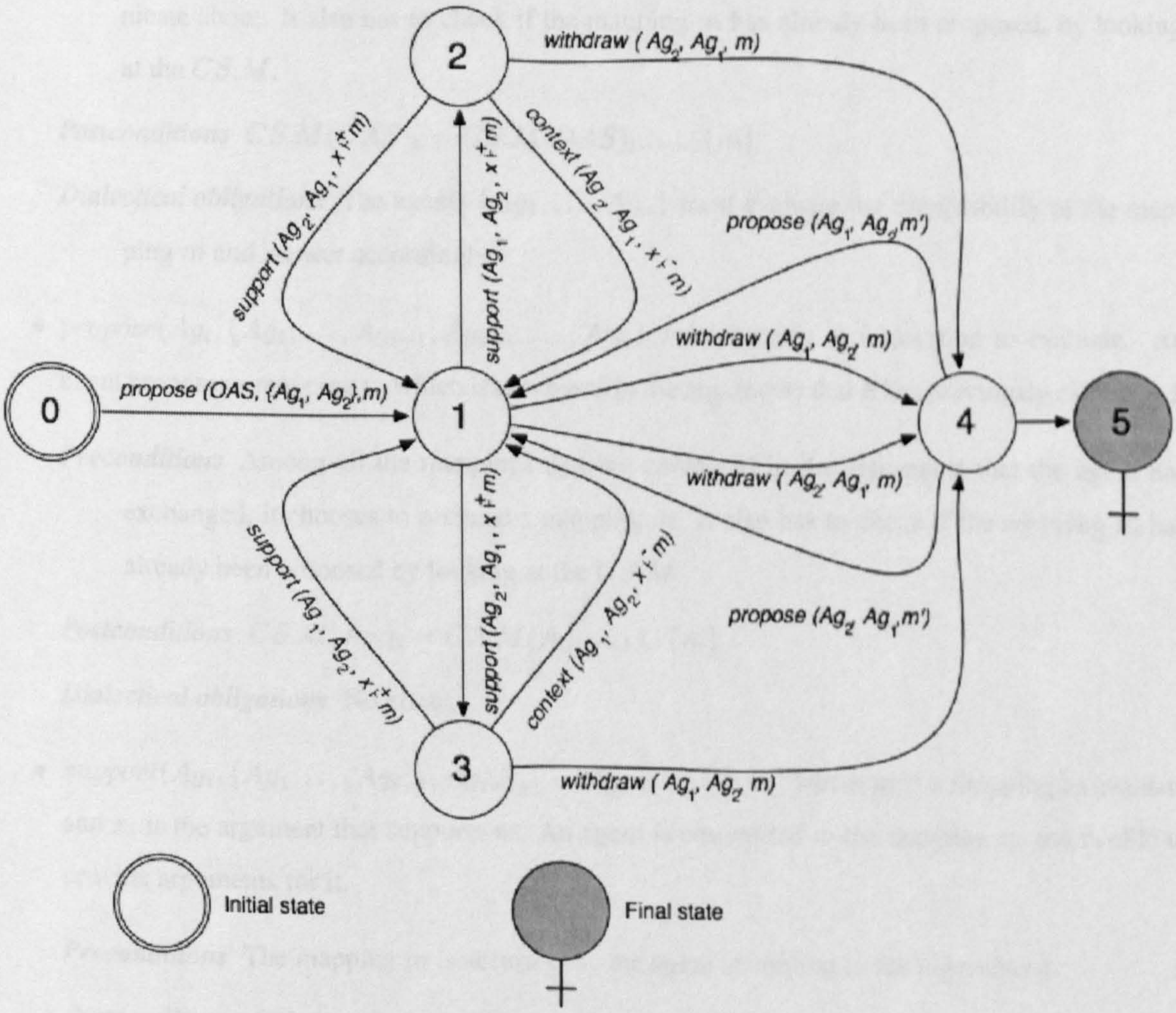

Figure 6.5: Argumentation Protocol for Two Agents.

The termination of the dialogue is also ensured by having a finite number of candidate mappings,

plus a time-out to handle undecidable arguments, and it *terminates successfully* either with $Result(m) = agreed$, $Result(m) = agreeable$ or $Result(m) = rejected$.

6.3.1 Conditions on the Argumentation Acts

In what follows, we briefly specify the rules governing which move an individual agent can utter as: a set of pre-conditions that should hold before the dialogue move can be used; their effects on the commitment stores as post-conditions and the dialectical obligations[5]. We also show how a move updates the agents' commitment stores.

- $propose(OAS, \{Ag_1, \ldots, Ag_n\}, m)$ where m is a mapping to evaluate. This is the basic move in negotiation.

  Preconditions Among all the mappings that an OAS may provide between the ontologies, it chooses to propose the mapping m over the two terms that the agents are trying to communicate about. It also has to check if the mapping m has already been proposed, by looking at the $CS.M$.

  Postconditions $CS.M(OAS)_t = CS.M(OAS)_{t-1} \cup \{m\}$

  Dialectical obligations The agents $\{Ag_1, \ldots, Ag_n\}$ must evaluate the acceptability of the mapping m and answer accordingly.

- $propose(Ag_i, \{Ag_1, \ldots, Ag_{i-1}, Ag_{i+1}, \ldots, Ag_n\}, m)$ where m is a mapping to evaluate. An agent proposes a mapping m which is contained in the arguments that it has previously exchanged.

  Preconditions Among all the mappings that are contained in the arguments that the agent has exchanged, it chooses to propose a mapping m. It also has to check if the mapping m has already been proposed by looking at the $CS.M$.

  Postconditions $CS.M(Ag_i)_t = CS.M(Ag_i)_{t-1} \cup \{m\}$

  Dialectical obligations No effect.

- $support(Ag_i, \{Ag_1, \ldots, Ag_{i-1}, Ag_{i+1}, \ldots, Ag_n\}, x_1 \vdash^+ m)$ where m is a mapping to evaluate and x_i is the argument that supports m. An agent is committed to the mapping m and is able to provide arguments for it.

  Preconditions The mapping m is accepted by the agent according to the algorithm 4.

  Postconditions $CS.Arg(Ag_i)_t = CS.Arg(Ag_i)_{t-1} \cup \{x_i\}$

  Dialectical obligations No effect.

[5]The way we have specified the argumentation with semi-formal semantics shares limitations with similar approaches (e.g. FIPA ACL [1]). More elaborate semantics can be explored in the future.

- $contest(Ag_i, \{Ag_1, \ldots, Ag_{i-1}, Ag_{i+1}, \ldots, Ag_n\}, x_i \vdash^{-} m)$ where m is a mapping to evaluate and x_i is the arguments that rebuts m. An agent is committed to reject the mapping m and it is able to provide arguments against it.

  Preconditions The mapping m is rejected by the agent according to the algorithm 4.

  Postconditions $CS.Arg(Ag_i)_t = CS.Arg(Ag_i)_{t-1} \cup \{x_i\}$

  Dialectical obligations No effect.

- $withdraw(Ag_i, m)$ where m is a mapping to evaluate. An agent has no arguments to present.

  Preconditions The agent no longer has arguments to present.

  Dialectical obligations No effect.

Using our protocol, any argumentation between the n agents ends, either with $Result(m) = agreed$, $Result(m) = agreeable$, or $Result(m) = rejected$.

Example We give a brief example of a dialogue conducted under the argumentation protocol, between two agents labelled Ag_1 and Ag_2. These agents committed to the two heterogeneous ontologies, the *UnivLiv* ontology and *University*, that were presented in the previous chapter (see Section 5.3).

We assume that the OAS has provided the mapping $m = \langle$ STUDENT , STUDENT, 0.9, $\equiv\rangle$.

Assume now that Ag_1 prefers terminology to external structure, $(T \succ_{\mathcal{R}_1} ES)$ and Ag_2 prefers external structure to internal structure $(ES \succ_{\mathcal{R}_2} T)$.

Thus, the agent Ag_1 accepts the mapping m, whilst Ag_2 rejects it.

The arguments shared by the agents Ag_1 and Ag_2 are the following:

R : The labels of the two concepts, $STUDENT, STUDENT$ are identical, i.e.,

$\langle label(STUDENT) \approx_T label(STUDENT), +, m\rangle$

P : There are some instances of STUDENT that are mapped with some of the instances of STUDENT,

i.e., $\langle \exists m_i = \langle E(STUDENT), E(STUDENT), 0.6, \supseteq\rangle, +, m\rangle$

Q : No mappings are defined between their sub-concepts, i.e.,

$\langle \nexists m_i = \langle ES(STUDENT), ES(STUDENT), n', \equiv\rangle, -, m\rangle$

S : No mappings are defined between their super-concepts, i.e.,

$\langle \nexists m_i = \langle ES(STUDENT), ES(STUDENT), n', \equiv\rangle, -, m\rangle$

In the dialogue, the moves are numbered in sequence.

1. $propose(OAS, \{Ag_1, Ag_2\}, m)$

   The ontology alignment service OAS propose the candidate mapping m, seeking for the agents Ag_1 and Ag_2 to evaluate it.

2. $support(Ag_1, Ag_2, R \vdash^+ m)$

   The agent Ag_1 accepts the mapping and provides an argument for.

3. $contest(Ag_2, Ag_1, Q \vdash^- m)$

   The agent Ag_2 rejects the mapping and provides an argument against.

4. $support(Ag_1, Ag_2, P \vdash^+ m)$

   The agent Ag_1 provides another argument for.

5. $contest(Ag_2, Ag_1, S \vdash^- m)$

   The agent Ag_2 provides another argument against.

6. $withdraw(Ag_1, Ag_2)$

   The agent Ag_1 is not able to provide anymore argument for.

7. $withdraw(Ag_1, Ag_2)$

   The agent Ag_2 is not able to provide anymore argument against and the dialogue terminates. Then, the correspondent $VAFs$ can be built by the agents in order to determine the status of the mapping m by computing the preferred extensions.

Although this is only a very simple example, it illustrates the use of the argumentation protocol for our framework, with agents proposing, providing and receiving reasons for accept or reject the mappings proposed in a dialogue.

6.4 Conclusions

The primary contribution of this chapter has been to define a multi-agent environment that employed all the concepts presented until now. We have shown its main components: an MAS architecture, the communication infrastructure, the argumentative agents and the argumentation protocol. The architecture centers around the design of a mediating component, OAS, as agent that isolates and processes the knowledge needed for configuring different agents' ontologies to communicate together in a particular application. We have briefly described the communication mechanism used by the agents and how they deal with the acquaintance problem, i.e. how they become aware of the presence of other agents in the system. Then, we have examined the main characteristics of the agent architecture used to exchange the arguments for and against potential mappings, and the mechanism used to generate these arguments. Finally, we have presented a protocol that the agents adhere to in order to evaluate potential mappings, and exchange arguments. The argumentation protocol has deliberately designed to be concise and generic enough to be used on its own, or incorporated into other interaction protocols or ACLs.

Part III

Evaluation and Application

Chapter 7

Implementation and Evaluation

The goal of this chapter is to give evidence for the practical applicability of the mapping-oriented argumentation framework presented in this thesis.

After having proposed an argumentation framework and an architecture for selecting mappings that represented the agents' preferences, we now present our prototype implementation of many of the ideas of this thesis, its evaluation and its application in a specific scenario: the lifecycle for Ontology development process.

The prototype is being developed and applied on the Knowledge Web project[1], an EU-funded Network of Excellence aiming at enabling a wide scale dissemination of Semantic Web technologies in industrial and educational context, as well as an effective research collaboration among leading research parties in this community across Europe. The implementation makes use of the Alignment API[2], which is a Java-based API for generating and maintaining alignments and is developed by INRIA[3].

The evaluation and the related experiments have been carried out by using the Ontology Alignment Evaluation Initiative (OAEI) test suite[4]. OAEI provides a systematic benchmark test suite to compare alignment systems and algorithms on the same basis.

This chapter also provides an application of how the instantiation of the mapping-oriented framework can be also used to find agreements between evolving ontology in the DINO ontology lifecycle framework. The DINO ontology lifecycle framework has been developed within the EU Knowledge Web to support the collaborative ontology development process, in dynamic and data-intensive domains.

This chapter is organized as follows. Section 7.1 introduces the prototype and its main components. This section focused on specification of functionality, rather than technical details. Section 7.2 presents the result of an empirical evaluation of the behavior of the prototype which we have used to perform

[1] *http://knowledgeweb.semanticweb.org/*
[2] *alignapi.gforge.inria.fr/*
[3] *www.inria.fr/*
[4] *http://oaei.ontologymatching.org/*

a quantitative and qualitative validation of our framework. It describes the testing methodology and summarizes the results. Section 7.3 discusses and evaluates the applicability of our approach in the web service composition and discovery domain. Section 7.4 introduces how the proposed framework has been applied for the development of ontologies. This chapter concludes with a short discussion of our results.

7.1 An Implementation for the Mapping-oriented Argumentation Framework

This section gives a brief introduction to the prototype developed. The prototype acts as a proof of concept of the practical applicability of the mapping-oriented argumentation framework, implementing many of the core ideas elaborated in this thesis.

The current implementation uses two ontologies as input and accepts ontologies written in the OWL ontology language and RDF.

The prototype is fully implemented in JAVA using JADE agent development environment[5]. JADE is an open-source middleware for the development of distributed multi-agent applications based on a peer-to-peer communication architecture. JADE is Java-based and compliant with the FIPA specification. It provides libraries for agent communication and interaction, based on FIPA standards. It also provides tools for agent lifecycle management, inspection of exchanged messages and debugging.

The implemented system has been used for the evaluation, and the results of this endeavor are presented later in this chapter (see Section 7.2). It is to be noted that a full implementation (including the extent of stability, scalability or efficiency that would be desired in a piece of commercial software) has not been possible within the time frame of this research. Rather, a minimal system has been produced which is sufficient to demonstrate the framework. Our intention is to show the fundamental plausibility of the proposed argumentation approach and its value to the shared understanding task, without attempting to address every possible issue.

The prototype is composed of three main developed parts:

- **Alignment module** is responsible to provide the infrastructure for aligning the agents' ontologies.
- **Argumentation module** is the core component responsible to represent the arguments, the relationships between these arguments and to compute their acceptability.
- **Agents** that wish to communicate with each other and thus must align their ontologies prior to meaningful communication. The agents applied our approach in order to achieve consensual

[5]JADE Project. http://jade.cselt.it/

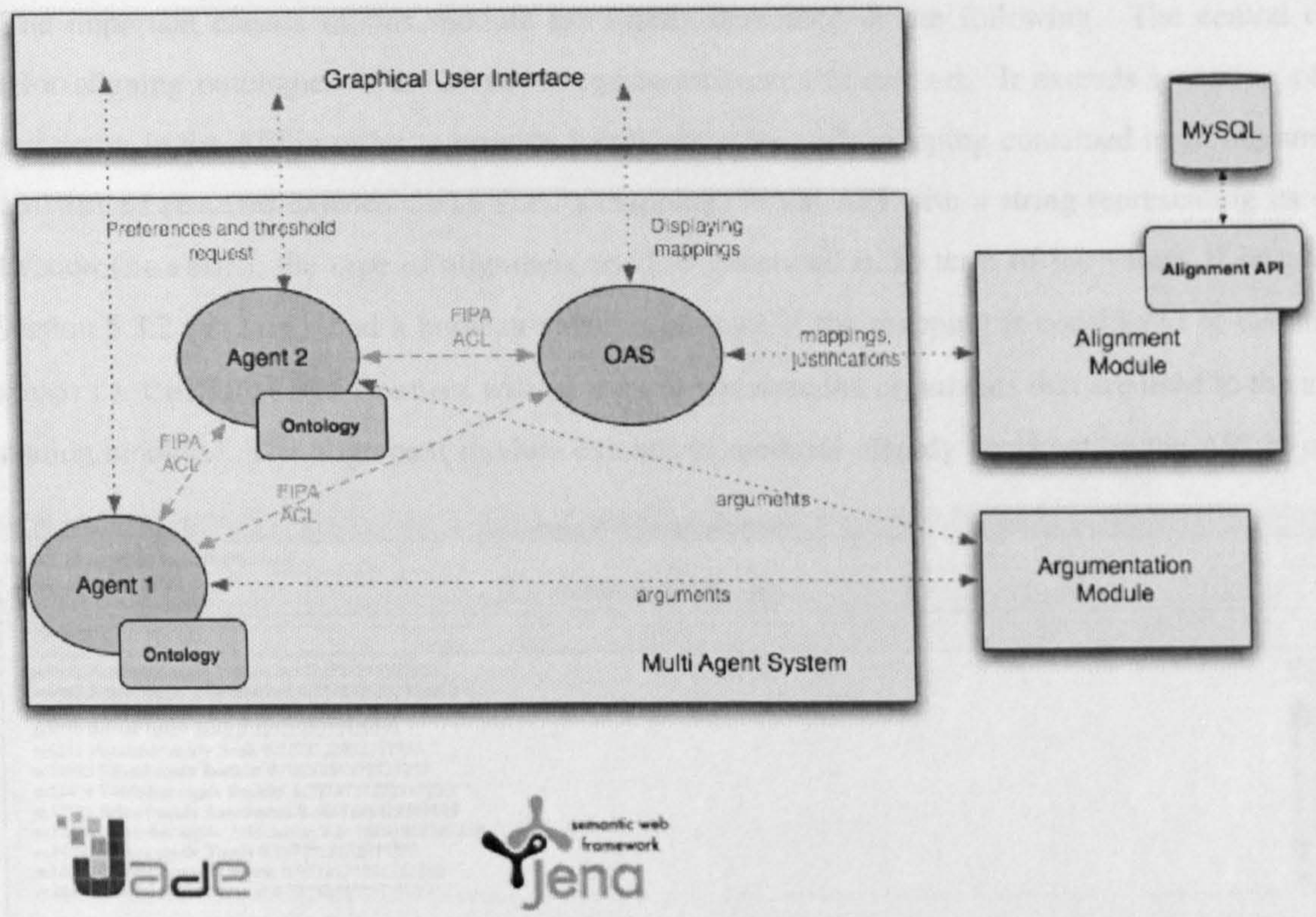

Figure 7.1: Mapping-Oriented Argumentation Prototype.

mappings between their respective ontologies. We have also considered a dedicated agent, OAS, to offer services, such as to align agents' ontologies.

Figure 7.1 illustrates the overall system architecture of the prototype and how these components interact with each other.

The remainder of this section gives an overview of the three components.

7.1.1 Alignment Module

The alignment module is responsible to provide the infrastructure for aligning the agents' ontologies.

Prior to starting any argumentation process, the agents are provided an overview of the mappings generated by the Ontology Alignment Service agent (OAS) - as depicted in Figure 7.2. During the alignment process the tool keeps track of the mappings currently being generated and their justifications[6]. The mappings and the justifications will form the bases for the generation of arguments and counter-arguments. A screenshot of a sample of arguments is depicted in Figure 7.3.

The alignment module extends the Alignment API in order to provide the justifications for the mappings provided. The alignment module must also collects evidence for those potential mappings that are dismissed[7].

[6]Note that the mappings can be permanently stored in a MySQL database if required.
[7]This knowledge is necessary for the generation of counter-arguments

The important classes of this module are briefly described in the following. The central concept for aligning ontologies is the class `AlignmentJustificated`. It extends to notion of an `Alignment` in the API in order to provide justification for each mapping contained in an alignment. `CellJustificated` extends `Cell` (i.e., a mapping) in the API with a string representing its own motivation (`reason`), the type of alignment used to generated it, in term of the values $\mathcal{V}$ described in Section 5.3.2 (`value`), and a boolean value to indicate if the mapping is considered or dismissed (`support`). `CellJustificated` will be used to generate the arguments that are used in the argumentation module. The alignment module extends to methods already provided by the API in order

Figure 7.2: Displaying Mappings.

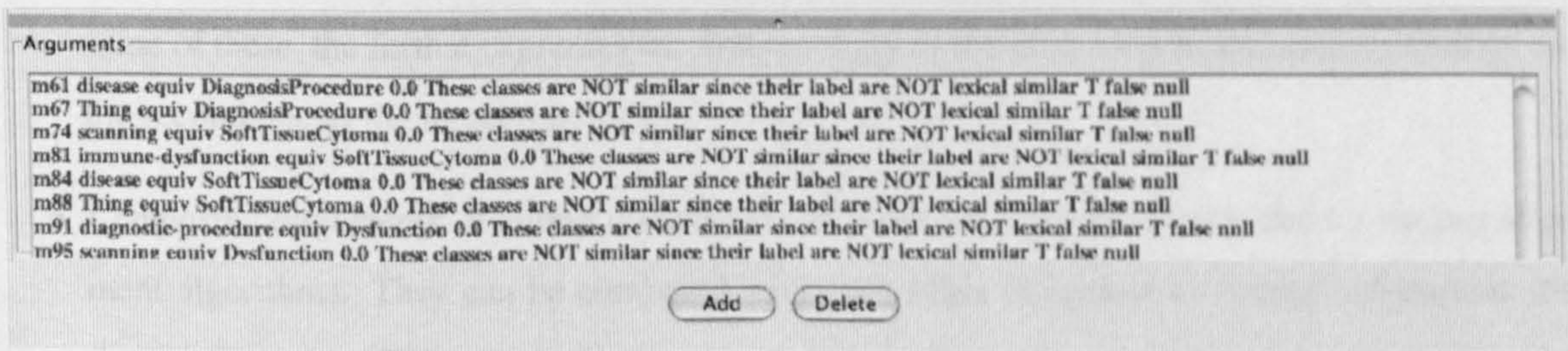

Figure 7.3: Displaying Arguments.

to compare not only name similarity, but also the external and internal structure of the ontologies. The algorithms used generate mappings by the calculation of a similarity metric between ontology entities as an adaptation of the SRMetric in [119].

Next, we present the Alignment API, upon which the alignment module relies.

7.1.2 Alignment API

The Alignment API has been proposed by INRIA, France's national research institute for computer science, with the aim of manipulating a standard alignment format for sharing among mapping engine

systems. The alignment format is not solely tied to triggering alignment algorithms but can help to achieve other goals such as to transform the alignment into some translation programme or articulation axioms, thresholding mapping in an alignment on some criterion, and comparing alignment results. The design of the Alignment API follows that of the OWL API in separating the various concerns that are involved in the manipulation and implementation of the API.

Aspects of Functionality

The Alignment API itself is a Java description of tools for accessing the common format. It defines four main interfaces (`Alignment`, `Cell`, `Relation` and `Evaluator`) and proposes the following services:

- *Storing, finding, and sharing alignments.* The API is able to collect hand-made or automatically created alignments in libraries that can be used for linking two particular ontologies;
- Piping alignment algorithms (improving an existing alignment);
- Manipulating alignments;
- *Generating processing output.* The obtained alignment can, not only be generated in the RDF serialisation form of the Alignment format, but also in other formats. Currently, the available formats are RDF, HTML, OWL axioms (expressing *subsumption*, *equivalence* and *exclusivity relations*), XSLT stylesheet, C-OWL mapping, SWRL rules and SKOS mapping document. For some of these, the format expresses the first ontology in the alignment as the source ontology and the second one as the target ontology.
- *Comparing alignments.* It allows comparison between the alignments provided by various alignment algorithms. They can be compared with each other or against a "correct" alignment. For that purpose, the API proposes the `Evaluator` interface.

The API also provides the ability to compose matching algorithms and manipulate alignments through programming. The API can be used for producing transformations, rules or bridge axioms independently from the algorithm that produced the alignment. The alignments are indexed by ontology pairs and by surrogates allowing fast retrieved. To one surrogate corresponds only one alignment while for an ontology pair, there can be several such alignments. There is no constraint that the alignments are computed online or off-line (i.e., they are stored in the alignment store) or that they are processed by hand or automatically. However, this kind of information can be stored together with the alignment in order for the client to be able to discriminate among them. The main principle of the Alignment API is that it can always be extended. In particular, it is possible to add new matching algorithms and mediator

generators that will be accessible through the API.
So far, alignments contain information about:

- the kind of alignment it is (1:1 or n:m for instance);
- the algorithm that provided it (or if it has been provided by hand);
- the language level used in the alignment;
- the confidence in each mapping

For our specific purpose, an alignment has been extended to provide the justifications of the mappings that that alignment includes.

Alignment Server

The API provides the use of an Alignment Server which is able to store and retrieve alignment resources. The server (see Figure 7.4) ensures the persistence of the alignments through the storage of these in a relational database (MySQL database). The access to the API is achieved through a protocol presented in [44]. Plug-ins allow the remote invocation of the alignment server and, currently, three plug-ins are available for the server :

- HTTP/HTML plug-in for interacting through a browser;
- JADE/FIPA ACL for interacting with agents;
- HTTP/SOAP plug-in for interacting as a web service.

The server is available either at design-time or at run-time through the web service access of the server (or any other available plug-in). The components of the Alignment Server as well as the connected clients can be distributed in different machines. Several servers can share the same databases.
In our prototype, we are using JADE for the remote invocation of the alignment server.

7.1.3 Argumentation Module

The argumentation module is the core component to represent the agents' arguments, the relationships between these arguments and to evaluate their acceptability.
Since an argumentation framework can be always represented as a directed graph in which the arguments are vertices and edges represent attacks between arguments, we have implemented a Value-Based Argumentation framework as an RDF graph, using Jena. Jena[8] is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, including a rule-based inference engine.

[8] jena.sourceforge.net/

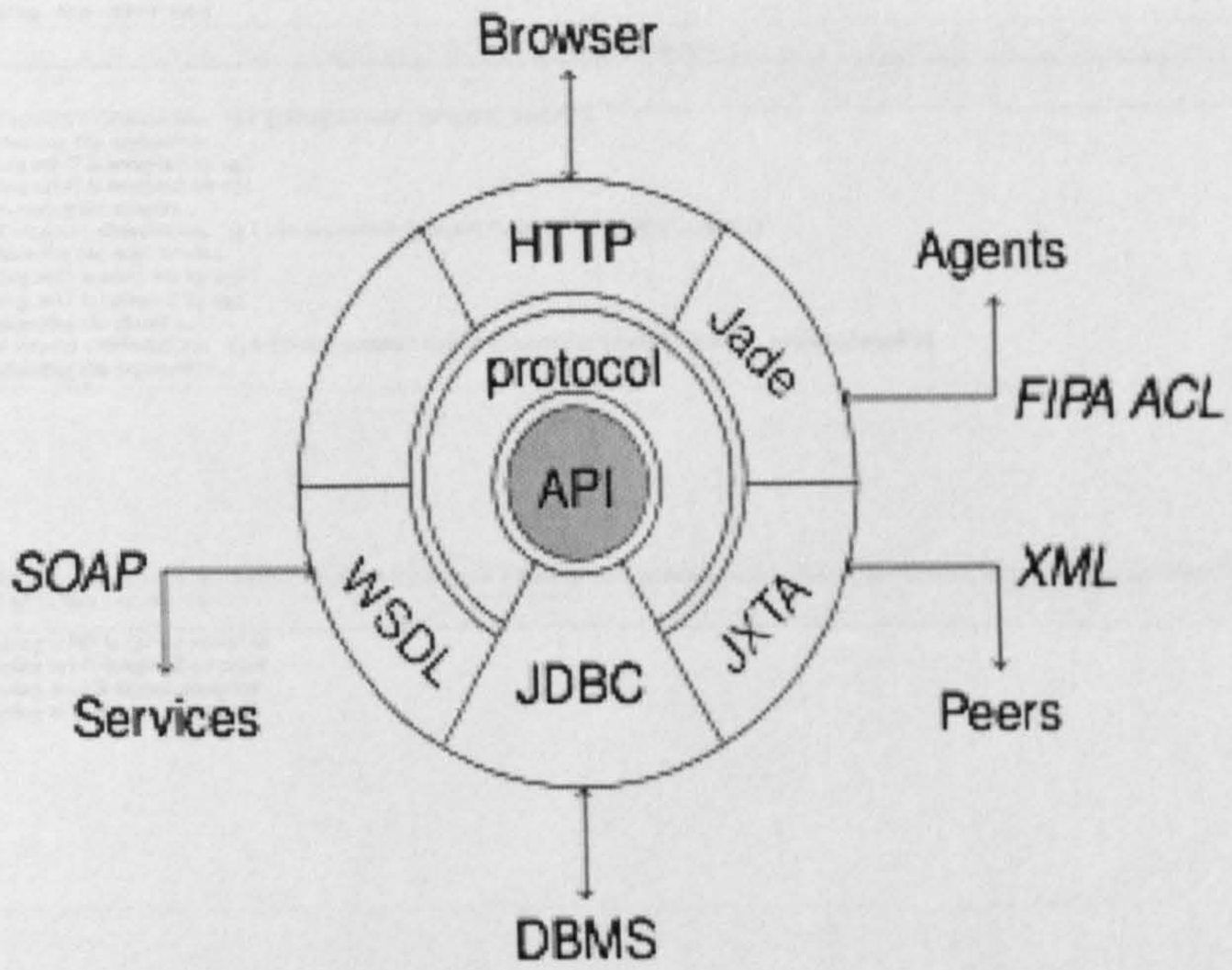

Figure 7.4: Alignment Server.

The main important classes of the Argumentation module are briefly described in the following.

An `ValuedArgumentationFramework` is an instance of a Jena RDF Model. An RDF Model is a set of RDF statements, which are defined as a triple consisting of a predicate, a subject and an object. The predicate is a RDF property that represents the *attack relation*. The subject and object are RDF Resources representing the arguments. `ValuedArgument` is a `Resource` (URI nodes) and the attack relations between these arguments are provided by `Statements` which link them. A `ValuedArgument` possess different attributes such as the assertion (the mapping `Cells`), the ground `Justification`, the `sigma` (a boolean value, indicating whether the argument is supporting a mapping or is against), an `identifier` and its `value`. The attack relation can be inferred from the nodes they connected by looking at the boolean values in `sigma`.

The `ValuedArgumentationFramework` class implements algorithms to compute the preferred extensions and thus to determinate if the mappings are agreed, agreeable or rejected (see Figure 7.5).

7.1.4 Agents

Two main agents have been implemented in our prototype: the agents that are using our approach in order to achieve consensual mappings between their ontologies; and the ontology alignment service (OAS) to provide several services to them, e.g., align ontologies. Both type of agents have been developed using JADE. The agents are instances of an user defined Java class that extend the base class `Agent`

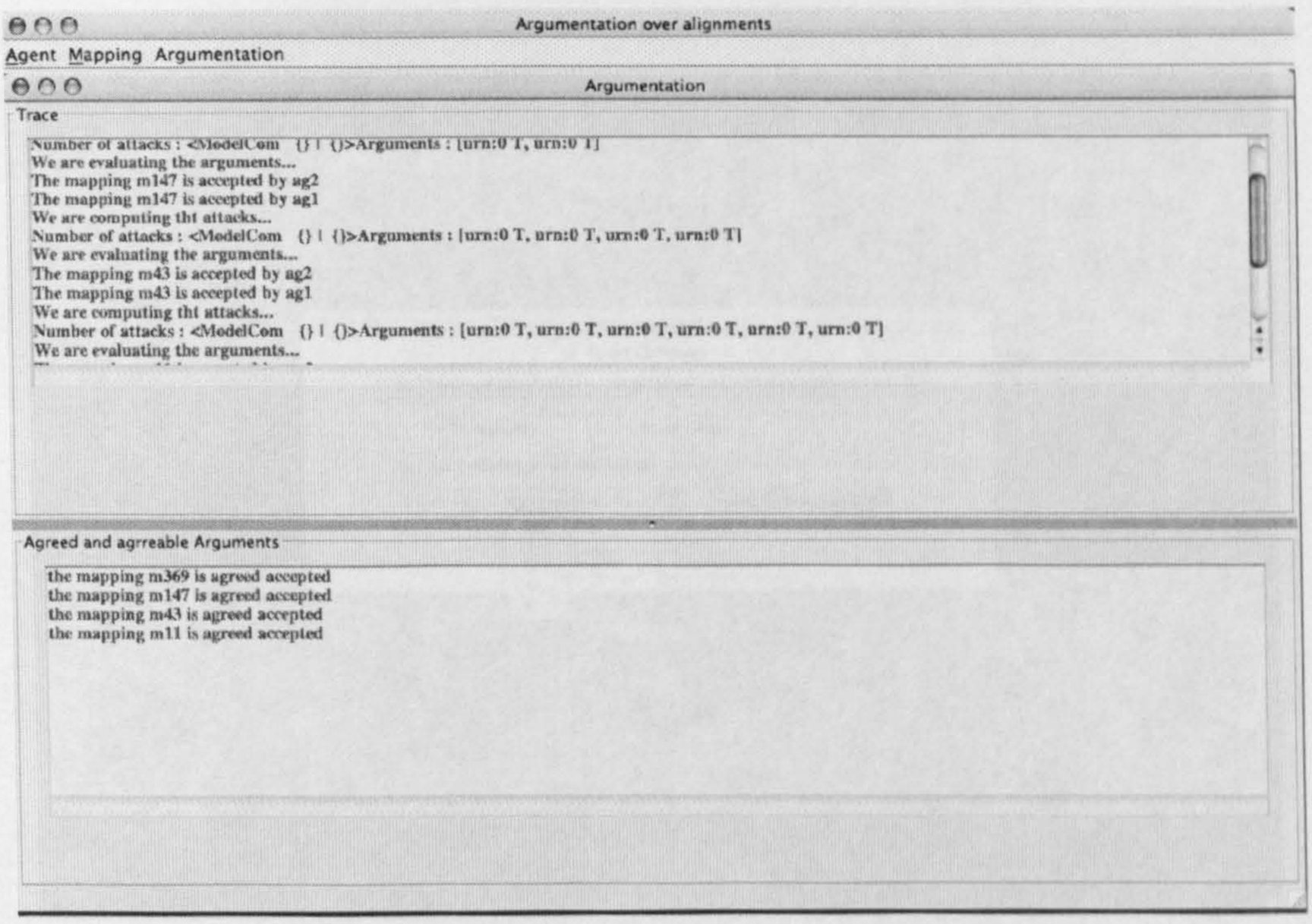

Figure 7.5: Displaying Agreed Mappings.

and their actions of agents have been implemented by `Behaviour` classes. Complex actions based on specific protocols are implemented by several methods invoked at specific states of the protocol.

Each agent mainly consists of four components: (a) an ontology which contains the domain knowledge (Knowledge Base); (b) a communication module which handles incoming and outcoming messages (JADE platform); (c) the preferences and threshold which represent the motivations of that agent to accept or reject a mapping; and (d) an argumentation module (see above) for reasoning with arguments over mappings. The preferences, the threshold and the ontology of an agent are externally chosen by an user. Thus, an agent acts on behalf of a user via a Graphical User Interface (GUI) (see Figure 7.6)

OAS is a JADE agent that respond to an alignment request from the agents. OAS invokes the Alignment server to execute alignments until the process completes. OAS can compute the alignments on the fly (run-time) or is able to store and retrieve alignments (design-time). OAS is able to perform a number of alignment tasks and offer them to the other agents or services. These tasks are summarised in Table 7.1. Most of these services correspond to what is provided by any implementation of the Alignment API.

We now turn to the empirical evaluation of our framework.

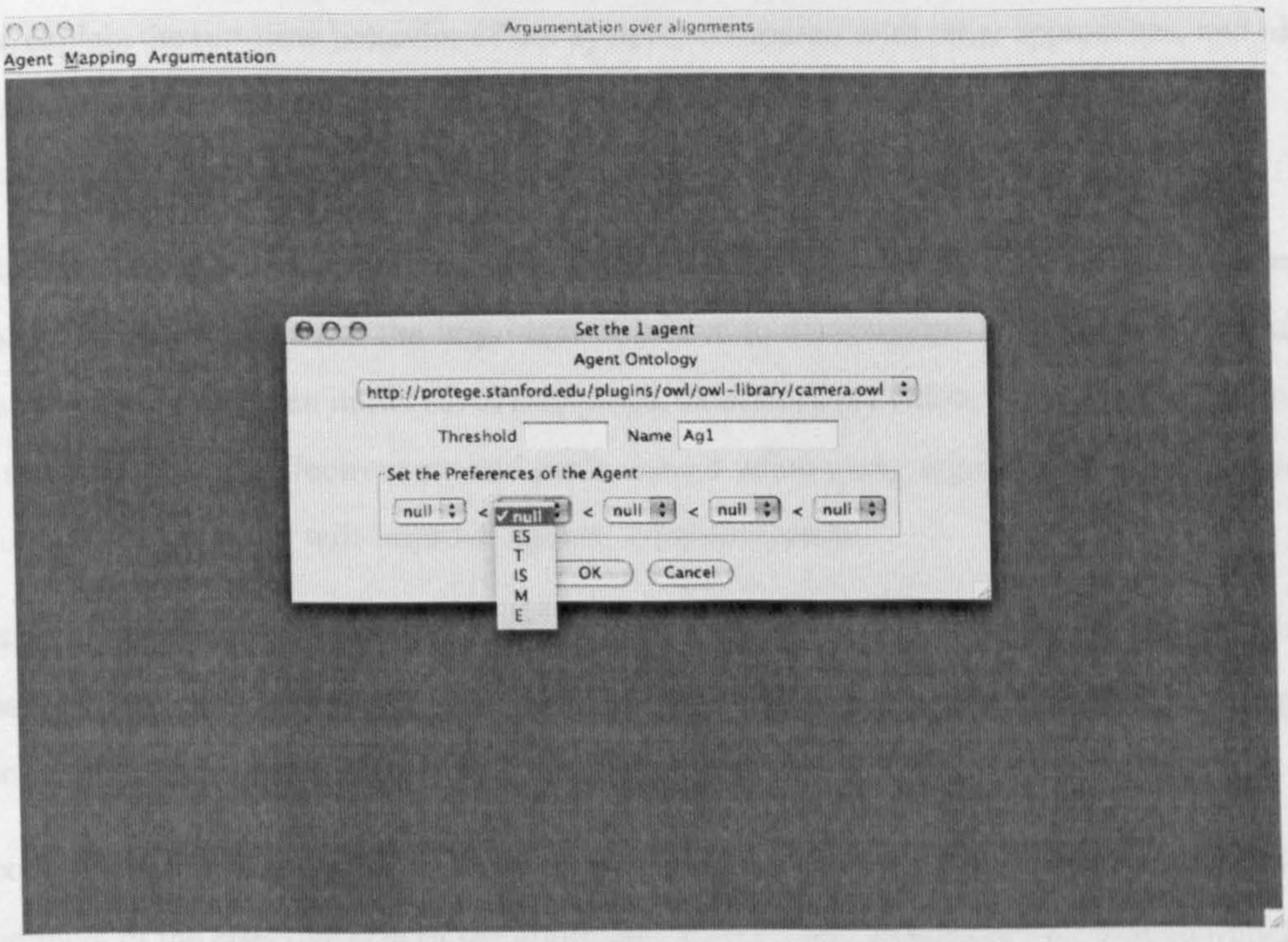

Figure 7.6: GUI to Set Preferences, Ontologies and Threshold of an Agent.

Service	Syntax
Finding a similar ontology	$O' \Leftarrow Match(O, T)$
Align two ontologies	$A' \Leftarrow Align(O, O', A, P)$
Thresholding	$A' \Leftarrow Threshold(A, V)$
Generating code	$P \Leftarrow Render(A, language)$
Translating a message	$m' \Leftarrow Translate(m, A)$
Storing alignment	$n \Leftarrow Store(A, O, O')$
Suppressing alignment	$Delete(n)$
Finding (stored) alignments	$n \Leftarrow Find(O, O')$
Retrieving alignment:	$\langle O, O', A \rangle \Leftarrow Retrieve(n)$

Table 7.1: Servives Provided by the Ontology Alignment Service.

7.2 Empirical Evaluation

In order to gauge the effectiveness and efficiency of the proposed approach, we have conducted several experiments. To this end, we considered pairs of argumentative agents that respect the protocol described in Section 6.3 and that use the reasoning mechanism, described in Section 6.2, to generate and evaluate the arguments explained in Section 5.3.2.

Each of the experiments addresses various dimensions of our approach with respect to a set of identified performance metrics, such as the matching accuracy, the processing time and costs for the argumentation.

The task of such an evaluation is to validate whether or not our approach generates consensual

mappings, explain the run-time behavior of our system, compared with other approaches, and ultimately to validate and optimize our algorithm.

The experiments are briefly described below:

- **Experiments on the impact of the argumentation approach over a set of mappings.** The first set of experiments has the important objective to demonstrate how our approach affects the matching accuracy of an initial set of mappings, externally provided. Thus, these experiments help to show the relative effectiveness and performance when using argumentation and, consequently, shows if our approach will improve agents' communication.

- **Experiments demonstrating the effects of varying the preferences.** By varying the preferences of an agent, the matching accuracy can be either increased or decreased. These experiments measure the quality of the alignments reached by varying this factor.

- **Experiments on processing costs for argumentation.** These experiments consists of the measurement of the effective cost of the argumentation process, in term of the computational time and resources utilised in the process.

- **Comparison of our argumentation approach wrt. current alignment tools.** These experiments compare our argumentation approach with a number of other ontology alignment tools representative of the state-of-the-art.

In this empirical study, we are attempting to confirm our hypothesis that the selection of mappings that conform to the preferences and interests of the agents, does not only reflect the characteristics of agent autonomy and rationality, but could also improve the accuracy of the alignments and thus may lead to a better shared understanding between the agents. Note that we also expect that the determination of an agent's preferences plays an important role for such results. For this purpose, our approach has been mainly evaluated in terms of precision and recall.

The remainder of this section presents the setting and evaluation results, with respect to the above dimensions.

7.2.1 Experimental Setup

The experiments were executed on an Apple PowerBook G4, with 1GHz PowerPC G4 processor and 768 MB DDR SDRAM. The operating system was Mac OS X version 10.4.7, and Java (JDK version 1.5) was used as the programming language for creating the software and for the experiments.

For the evaluation, we use the Ontology Alignment Evaluation Initiative (OAEI) test suite as test ontologies[9]. The Ontology Alignment Evaluation Initiative is a coordinated international initiative that

[9]http://oaei.ontologymatching.org/

organizes the evaluation of the increasing number of ontology mapping engine systems. The main goal of the OAEI is to be able to compare systems and algorithms on the same basis and to allow anyone to draw conclusions about the best aligning strategies. OAEI provides a systematic (and consensual) benchmark test suite, with pairs of ontologies to align – one reference ontology for a bibliographic domain to be compared with other ontologies. The reference ontology contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals. Most of the other ontologies originate from the reference ontology by making some changes. These changes are focusing the characterization of the behavior of the tools rather than having them compete on real-life problems. The test suite also includes the expected (human-based) results. The ontologies are described in OWL-DL and serialized in RDF/XML format.

The experiments involved two agents Ag_1 and Ag_2 – Ag_1 having the reference ontology in each test and Ag_2 having the respective test ontology. Ag_1 and Ag_2 conduct negotiations with each other by exchanging a set of arguments over each single potential mapping. For all experiments, their threshold has been set to zero, and so will not influence the process.

The evaluation sets we used are the following:

simple tests: The reference ontology is compared with itself, with another irrelevant ontology (the wine ontology used in the OWL primer[10]) or the same ontology restricted or generalized to OWL-Lite. Tests 101, 102, 103, 104.

systematic tests: The reference ontology is compared with modified ones. These modifications involved discarding some features, such as names, comments, hierarchy, instances, relations, restrictions, etc. It aims at evaluating how an algorithm behaves when some information is lacking. Tests 201, 202, 204, 205, 206, 221, 222, 223, 224, 225, 228, 230.

complex tests: The reference ontology is compared with four real-life ontologies for bibliographic references found on the web and left unchanged. Tests 301, 302, 303, 304.

Table 7.2 shows what has been retracted from the reference ontology for each test. We refer the reader to the OAEI web site for a full description of these tests.

To decide whether a correct set of agreed ontology mappings is obtained, we used standard information retrieval metrics [104] to assess the results of our tests: *Recall*, *Precision* and *F-measure*. *Precision* measures the ratio between the number of correct mappings and the number of all mappings found. It estimates the reliability of the automatic procedure for the match prediction relative to the human based procedure. *Recall* measures the ratio between the number of correct mappings and the total number of correct mappings that should be found. It specifies the share of real matches that are found. *F-measure*

[10] http://www.w3.org/TR/owl-guide/wine.rdf

Test	Type test
101	Reference alignment
102	Irrelevant ontology: wine ontology
103	Language generalization
104	Language restriction
201	No names
202	No names, no comment
204	Naming conventions
205	Synonyms
206	Translation
221	No specialisation
222	Flatenned hierarchy
223	Expanded hierarchy
224	No instance
225	No restrictions
228	No properties
230	Flattened classes
301	Real: BibTeX/MIT
302	Real: BibTeX/UMBC
303	Real: Karlsruhe
304	Real: INRIA

Table 7.2: Test Cases.

combines the measures of precision and recall as single measure:

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

The preferences $Pref_1$ and $Pref_2$, respectively for the agents Ag_1 and Ag_2, have been chosen on the basis of the ontological information (see Table 7.3).

Test	$Pref_1$	$Pref_2$
101,102,103,104,204,224,225,228,230	$T \succ ES$	$T \succ ES$
201,202	$T \succ ES$	$ES \succ IS \succ T$
205,206,223	$T \succ ES$	$ES \succ T$
221,301,302	$T \succ ES$	$T \succ IS$
222,304	$T \succ ES$	$T \succ IS \succ ES$
303	$T \succ ES$	$T \succ ES \succ IS$

Table 7.3: Ag_1 and Ag_2's Preferences for each Test.

As previously mentioned, our argumentation approach can be applied solely over those ontological terms that the agents need in order to understand each other, rather than over the entire ontologies. However, since we want to evaluate the overall performance of our framework, we will measure precision and recall of the alignments between all of the terms in the ontologies, rather then considering only few terms in a particular message.

In the following sections we describe each of the experiments and their results.

7.2.2 Experiments and Results

Experiments on the Impact of the Argumentation Approach over a Set of Mappings

In these experiments we determined how well our approach would perform in improving shared understanding between two agents, simply evaluating the resulting alignments with and without argumentation. When not arguing, both agents will accept all the ontology mappings that have been externally provided. Precision and recall, and the corresponding F-measure, have been calculated for this set of mappings, namely *Precision without-argumentation*, *Recall without-argumentation* and *F-measure without-argumentation*. These measures represent the matching accuracy over all mappings provided, without considering the agents' preferences and any other constraints.

With argumentation, each agent, taking into account its preferences (see Table 7.3), accepted or rejected each of the ontology mappings. After that both agents exchange the arguments generated, and the argumentation frameworks will return the mappings that are mutually consensual for both of them. Precision, recall, and F-measure, were then calculated for this set of agreed mappings. They have been denoted as *Precision with-argumentation*, *Recall with-argumentation* and *F-measure with-argumentation*. These measures represent the matching accuracy over a set of agreed mappings, that are achieved by both agents using our approach. The *F-measure without-argumentation* and the *F-measure with-argumentation* are shown in Figure 7.7.

Next, we measured the *F-measure with-argumentation* for each agent and contrast it with the *F-measure without-argumentation*. The *F-measure with-argumentation* for each agent will represent the matching accuracy over the mappings that are achieved using our approach and that are only agreeable by that agent. Figures 7.8 7.9 show the *F-measure without-argumentation* and the *F-measure with-argumentation*, respectively, for Ag_1 and Ag_2. Note that the *F-measure without-argumentation* is the same for each agent, since any constraints or preferences have been considered.
A summary of the resulting measurements is shown in Figure 7.10.

The results from Figure 7.7 show that the *F-measure with-argumentation* is greater than the *F-measure without-argumentation* throughout, with exceptions for the tests 201, 202, 205 and 206. For tests 201 and 202, our approach is not able to perform well due to a lack of candidate mappings that our system provided to argue over. In these test ontologies the concept labels have been replaced by random strings and consequently the ontology knowledge is not enough to generate mappings. For tests 205 and 206, the argumentation process is not able to produce any agreement because the information in the two ontologies causes the agents to select directly opposing preferences, which leads to an inability to reach agreement on many of the mappings.

Similar situation from the results of Figures 7.8 and 7.9 : the *F-measure with-argumentation* for

each of the two agents is mostly greater than the *F-measure without-argumentation*.
Figure 7.10 shows that the *F-measure with-argumentation* for both agents is quite similar to the *F-measure with-argumentation* for each agent. Thus, the process of reaching agreement between Ag_1 and Ag_2 does not decrease the resulting performance.

We also compared the *F-measure with-argumentation* with the F-measure calculated over those alignments acceptable to each agent when considering only its own preferences, before any argumentation takes place (*F-measure pre-argumentation*). The results for each of the agents, Ag_1 and Ag_2, are shown in Figure 7.11. When considering such results we observe a slight increase in F-measure when using argumentation compared to the pre-argumentation F-measure and thus the accuracy of the negotiated alignment is generally better than any of the acceptable alignments for an individual agent, when considering only its own preferences.

These overall results demonstrate that our approach not only allows the selection of those mappings that best suit the interests of each agent, but, in most cases, improves mapping accuracy in terms of *F-measure*. Generally speaking the improvement of the mapping accuracy will also improve the quality of the communication between the agents and thus their understanding.

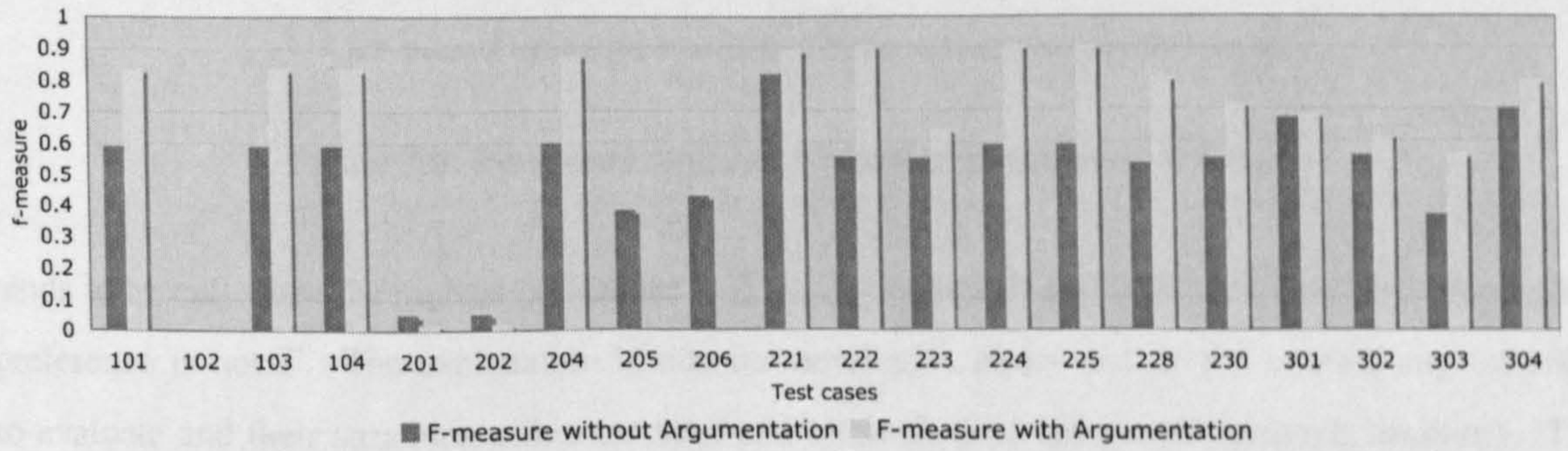

Figure 7.7: F-measure with and without argumentation.

Experiments Demonstrating the Effects of Varying the Preferences

We have argued that agents' preferences play an important role in the quality of the mappings reached and this, in turn, contributes to the effectiveness of the overall approach. In order to confirm this, we performed some experiments. These experiments are intended to determine the effects of varying the agents' preferences for the specific test case 301. The choice of the preferences were made randomly and then the corresponding results were observed. This is depicted in Figure 7.12 showing the effects of preferences over the accuracy of the mappings. If we observe the graph, there appears to be an interesting trend where the F-measure increases if the highest preference of the agent Ag_2 is T (i.e. terminological), and where it decreases for the other preference settings. In particular, the F-measure

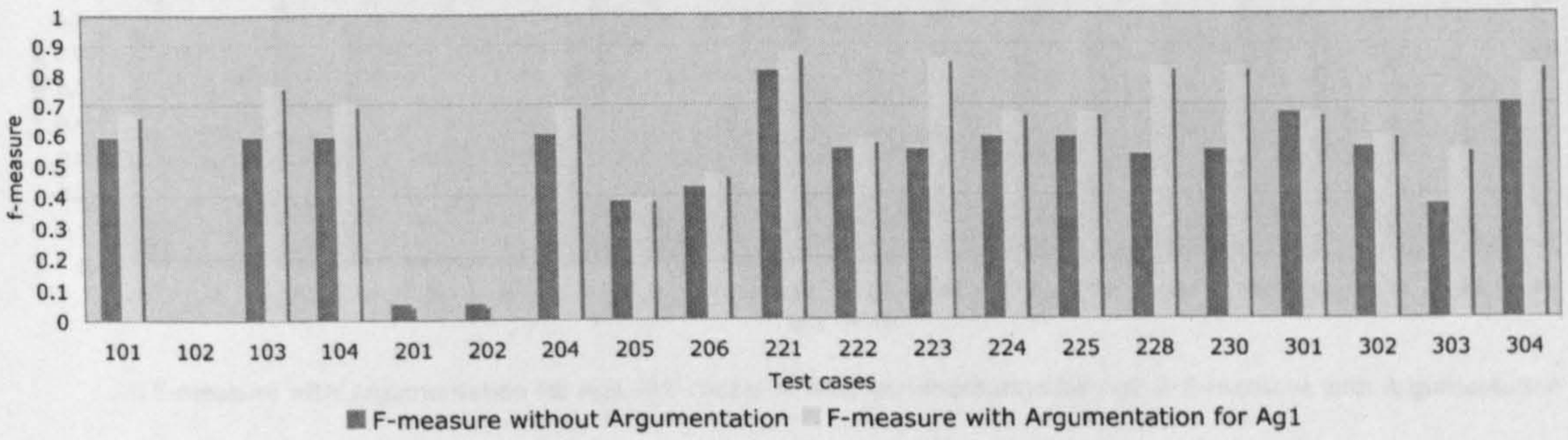

Figure 7.8: F-measure with and without argumentation for Ag_1.

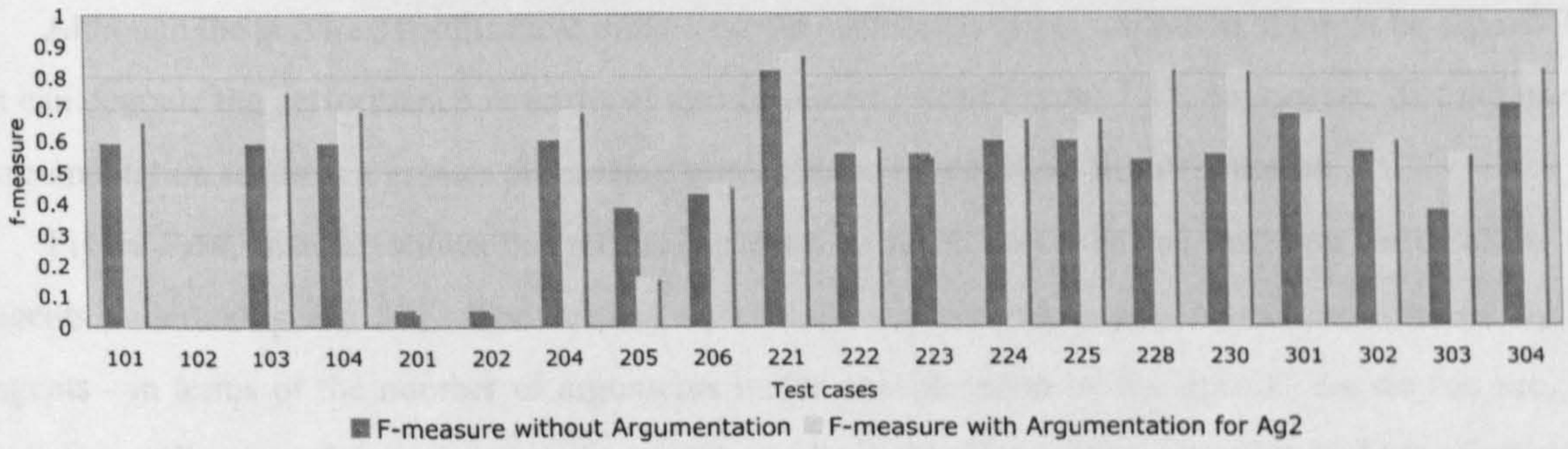

Figure 7.9: F-measure with and without argumentation for Ag_2.

tends to be null when the highest preference is E, i.e. extensional, and it remains low when the highest preference is not T. The explanation is that the ontologies under test do not contain any instances to evaluate and their structure, either external and internal, does not provide enough similarity. This confirms our expectations that the quality of the agreed mappings strictly depends on the preference settings and the ontology. Thus the choice of the preferences should not be decide randomly but it should depend on the agents' context and situation and, in particular, on the ontology and its features.

Experiments on the Processing Costs for Argumentation

Figure 7.13 shows the processing time required for the agents to consider the ontology mappings for the test case 301 in both interactions – *with* and *without* argumentation. The processing time with the argumentation includes the time to evaluate the mappings, generate the arguments and the time to reach agreements, whereas the processing time without argumentation includes only the time for the agents to extract the mappings.

The authors acknowledge that these processing time measurements may not be useful for external comparison since we are not taking into account any specific agents' task; however, they are useful in

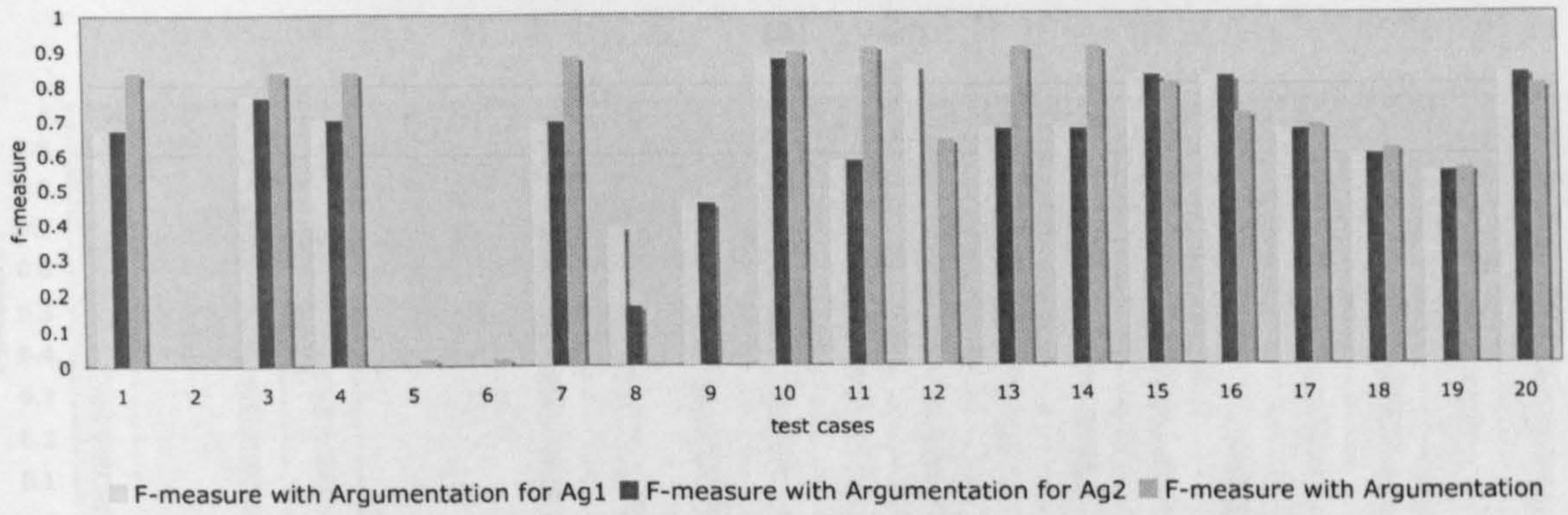

Figure 7.10: Comparison between F-measure with argumentation for both agents and for each single one.

this thesis to quantify the relative overhead incurred by our argumentation process.

Although the previous results have illustrated the usefulness of our approach, it might be argued that it can degrade the performance in terms of speed. Indeed, from Figure 7.13, we can see that the use of argumentation requires a greater processing time relative to not using any negotiation.

Figure 7.14, instead, shows the processing costs to reach a consensual mapping, incurred by the agents undertaking test 301. The vertical axis (`Resources`) represents resource utilization by the agents - in terms of the number of arguments under consideration by the agents. As we can see, the resource utilization of our argumentative agents can be divided into three main phases. In the first phase the resource usage increases in an approximately linear manner as the arguments are generated. In the second phase, the resource usage remains level as the arguments are evaluated by each of the agents. In the third phase, utilization of resources gradually decreases because the number of arguments being processed decreases as agreements are reached on individual mappings.

Figure 7.15 shows instead the communication costs to reach a consensual mapping, incurred by the agents undertaking test 301. The communication costs are estimated as the total number of messages exchanged by the agents for that test. As we can see, initially, there are no messages because the agents are internally evaluating the arguments. After, there is a upward trend as the time increases to a peak, since the agents are now ready to exchange the arguments. After that, the number of messages decrease as the time increase until there are no messages.

It is evident that the time of an argumentation approach is greater than traditional approaches to ontology alignment, due to the additional time and costs of the argumentation process itself. However, we believe that this is outweighed by the benefits of the argumentation process, in terms of obtaining better alignments - from an individual agent's point of view (which in essence is part of the argumentation framework)[11].

[11]However, it is worth pointing out that a real comparison with traditional approaches to ontology alignment, in term of communication cost and processing cost was not possible because these information was not available or difficult to derive.

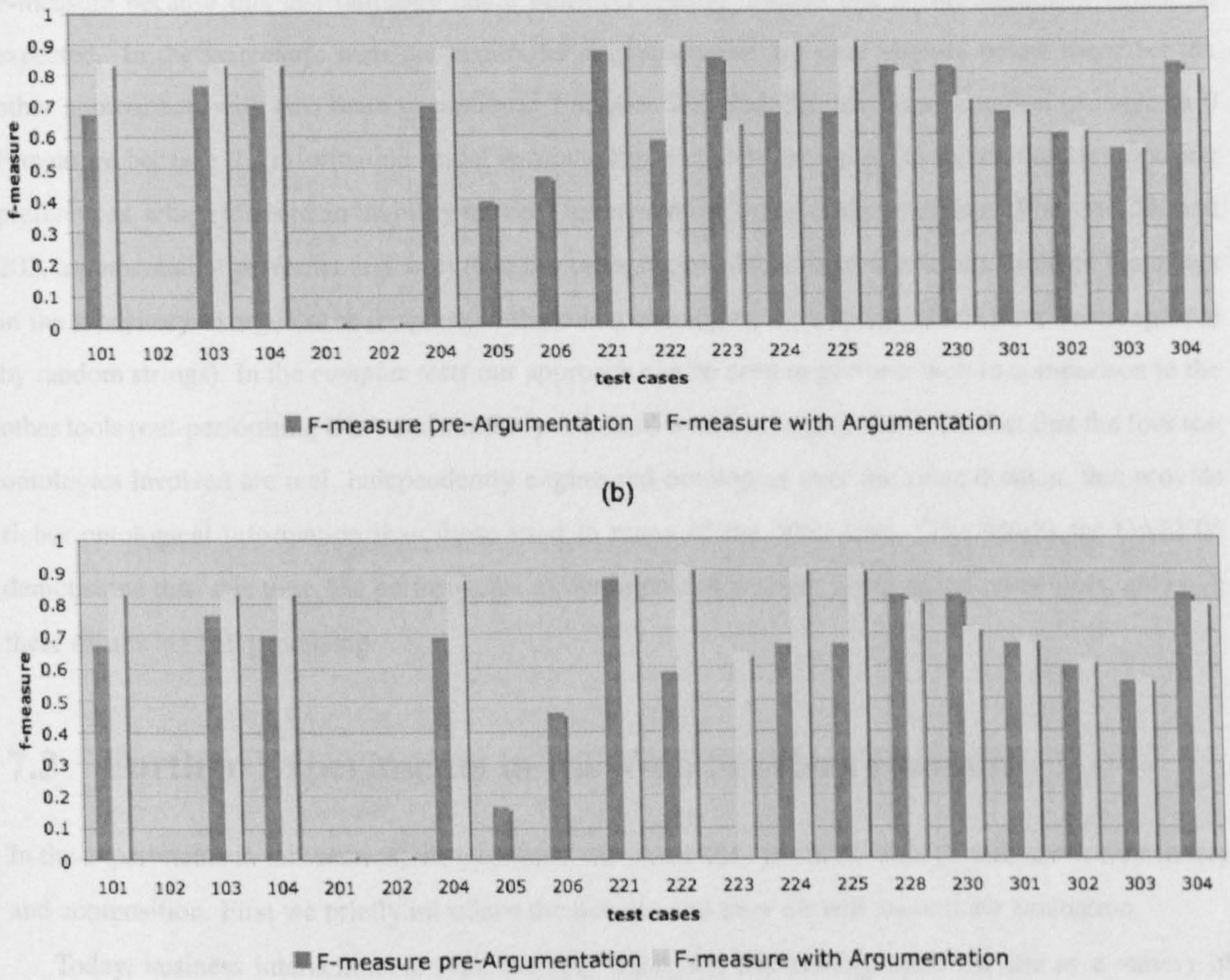

Figure 7.11: (a) F-measures for Ag_1. (b) F-measures for Ag_2.

Comparison of the Argumentation Approach wrt. Current Alignment Tools

In further experiments, we have compared our argumentation approach with a number of other ontology alignment tools representative of the state-of-the-art. These tools have been selected from the best three performers in OAEI'05 and OAEI'06. The three best performers in OAEI'05 are: Foam [40], Falcon OA [69] and OWL-Lite Alignment (OLA) [46]. The three best performers in OAEI'06 are COMA [41], RiMOM [120] and again Falcon OA. We compared the *post-argumentation F-measure*, that we previously measured, with the F-measure of each of these tools, measured over the same test cases[12]. The results are shown in Figures 7.16 and 7.17. Although all these approaches cannot be fully compared with ours, the results for OAEI'05 demonstrate that, in the majority of the test cases, our argumentation approach produces an information recall F-measure that is in a similar range to those produced by the other tools. Examination of the results in the different test groups shows that for the

[12]The official results have been published in http://oaei.ontologymatching.org/2005/results/ and http://oaei.ontologymatching.org/2006/results/

simple tests all the approaches produce comparable results, with, for example, test 102 showing a 0 F-measure because this test ontology has a non-overlapping domain and so no alignment would be expected. In the *systematic tests* the results for argumentation are only slightly below those for the other approaches, with two main exceptions. For tests 205 and 206 the argumentation produces a 0 F-measure because the information in the two ontologies causes the agents to select directly opposing preferences, which leads to an inability to reach agreement on many of the mappings. For tests 201 and 202, argumentation performs less well than the other approaches due to a lack of candidate mappings in the repository to argue over (because in these test ontologies the concept labels have been replaced by random strings). In the *complex tests* our approach can be seen to perform well in comparison to the other tools (out-performing OLA and similarly to Foam) which is largely due to the fact that the four test ontologies involved are real, independently engineered ontologies over the same domain, that provide richer ontological information than those used in many of the other tests. The results for OAEI'06 demonstrate that, this time, the performance of our approach is slight lower of the other tools, although these results are still promising.

7.3 Further Experiments in the Web Services Domain

In the experiments in this section, we evaluated our framework in the domain of web service discovery and composition. First we briefly introduce the domain and then we will present the evaluation.

Today, business interactions are increasingly triggering and relying upon the use of a variety of electronic and information technologies. In this context, *Web services* are conceived as an essential component for promoting interoperability of business processes and *software agents* as a key enabling technology for such processes to be dynamically discovered, combined and executed.

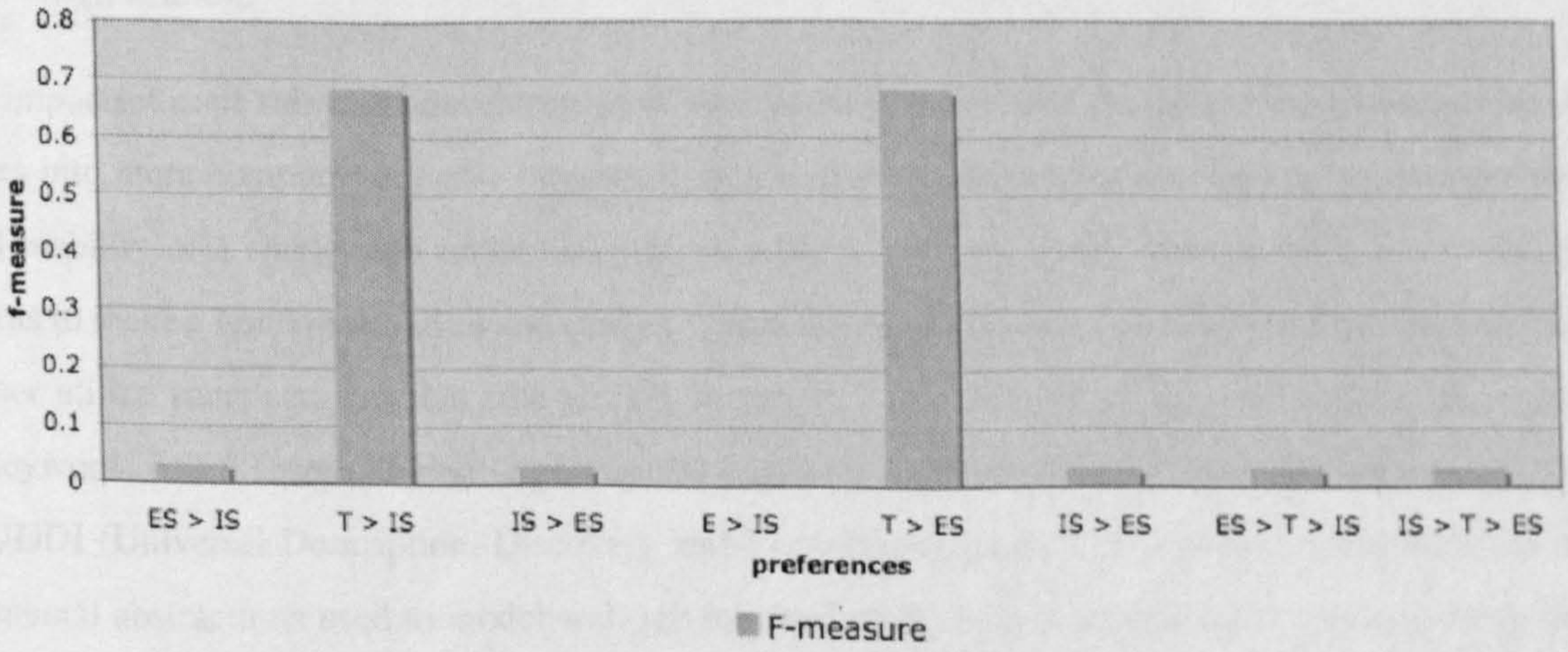

Figure 7.12: Plot demonstrating the effects of varying the preferences for the test case 301.

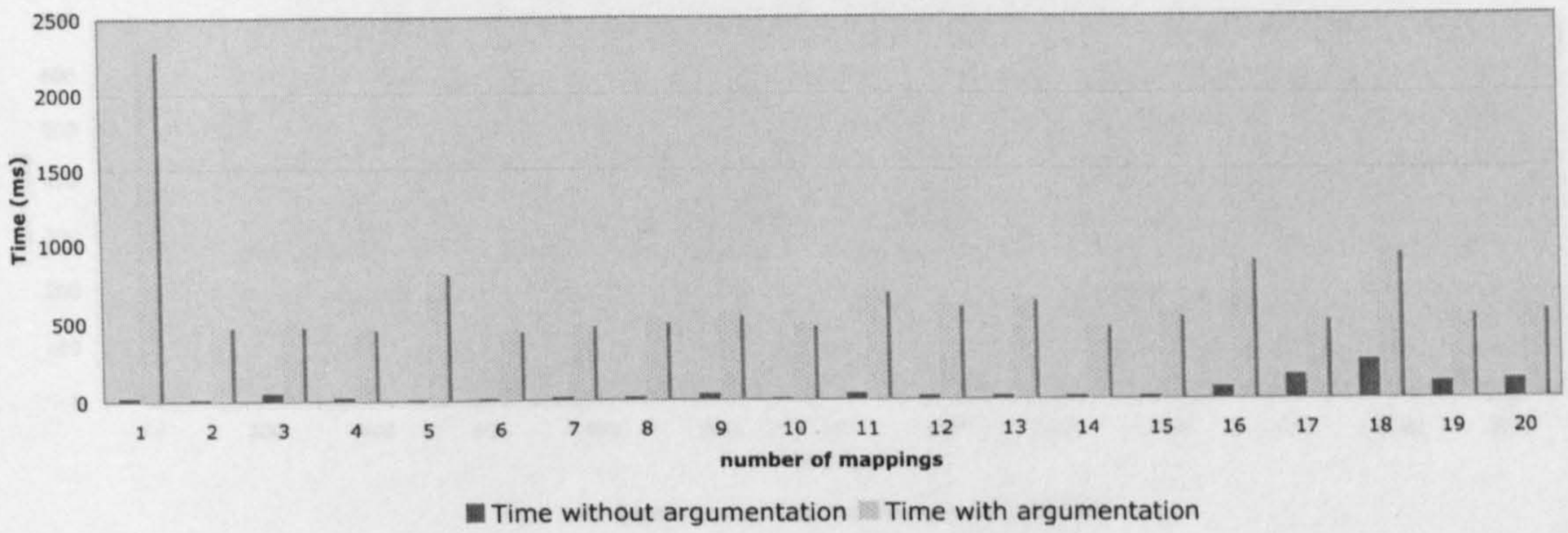

Figure 7.13: Processing time with and without argumentation.

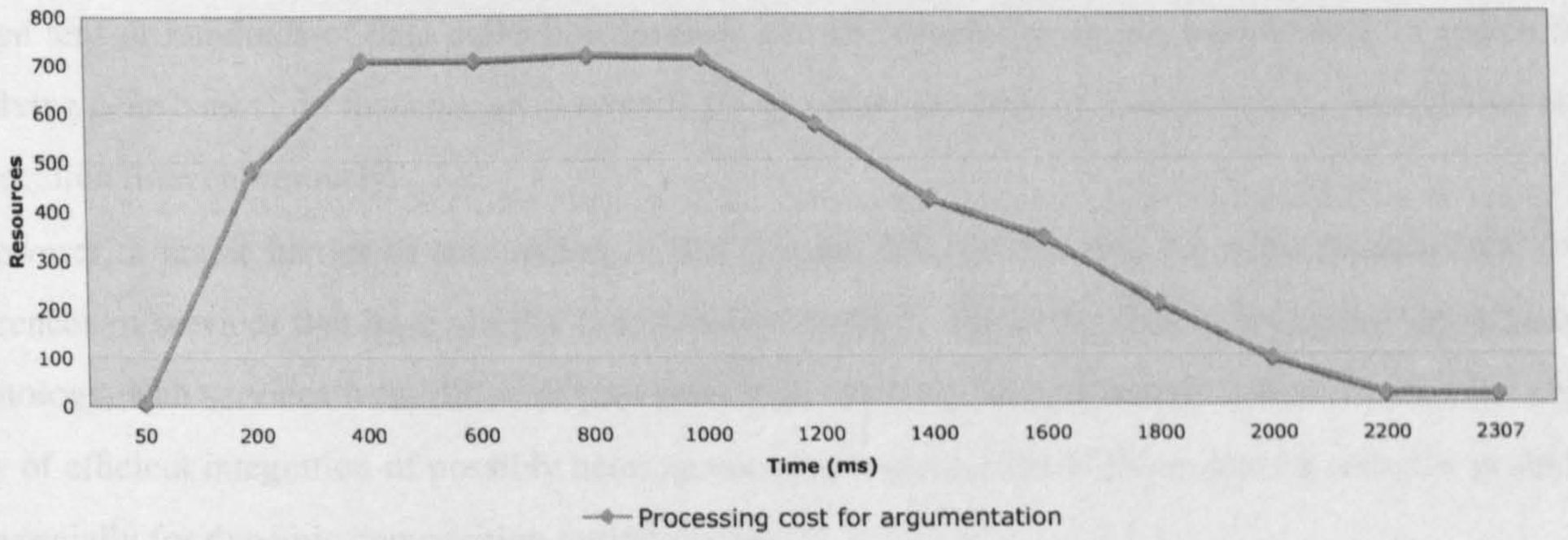

Figure 7.14: Processing cost for the argumentation over a single mapping.

As clearly expressed in the Web Services Architecture specification of the W3C[13]

> *"... software agents are the running programs that drive Web services - both to implement them and to access them as computational resources that act on behalf of a person or organization. "*

An important issue related to development of Web services is the need for composing existing web services into more complex services. In general, this is a result of complex and increasing user demands and inability of a single web service to achieve a users goals by itself. For example, a traveler who wants to make a hotel reservation and find an italian restaurant less than three miles from the hotel may either utilize some services that s/he already knows or try to find the services by looking them up in a keyword- based search engine (e.g., expedia.com or google.com) or in a web service registry (e.g., a UDDI (Universal Description, Discovery and Integration registry). The problem lies with the fundamental abstractions used to model web services and methods to compose these services using these abstractions. In more complex examples of scientific data exploration through service compositions,

[13]http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

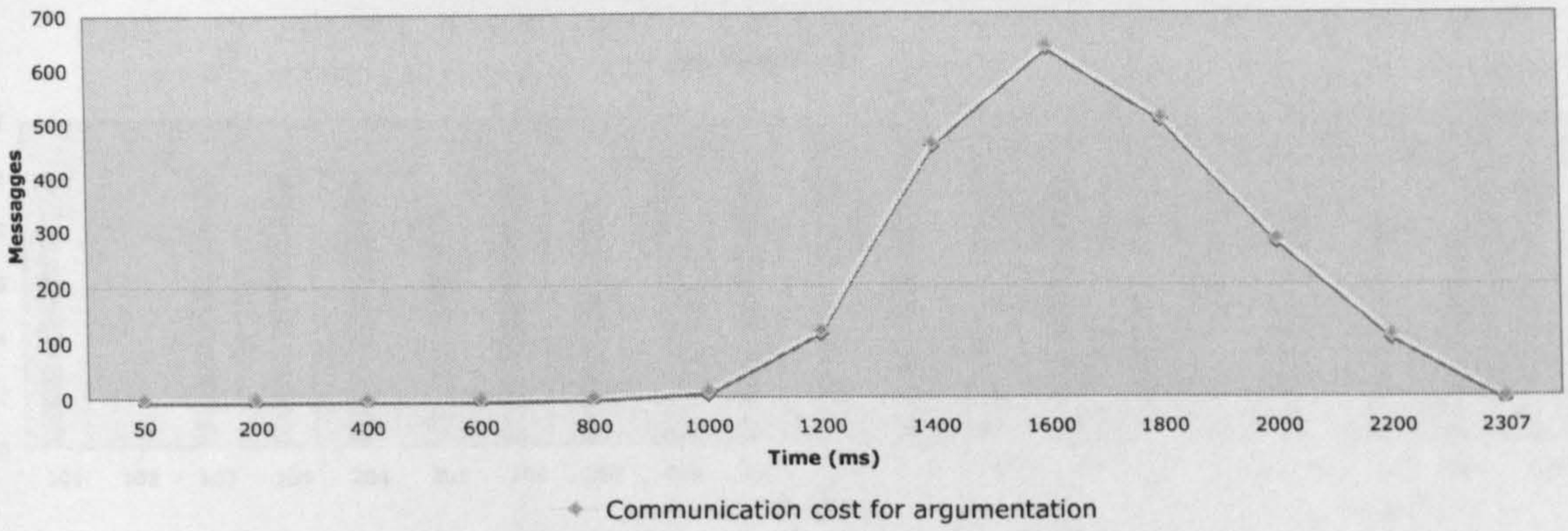

Figure 7.15: Communication cost for the argumentation over a single mapping.

even tens or hundreds of data collection services can be involved in a composition (e.g., a search involving gene banks). In that case an *automatic composition* can help in reducing query formulation and execution time enormously.

However, a major barrier to automation in this domain is understanding the commonalities and differences of services that have similar functionality. Indeed, due to the lack of an agreed-upon global ontology, web services from different providers typically have heterogeneous semantics. So, the ability of efficient integration of possibly heterogeneous services on the Web becomes a complex problem (especially for dynamic composition during runtime).

Agents that automatically reconcile ontologies, and thereby understand and integrate the information from different sources, would greatly facilitate Web service-based application interoperability.

A common solution is to consider middle agents that have ontological representation of the messages that their web services receive and transmit. Then, an external cross-organizational workflow/supply chain management agent for service composition has the specific task to relate these ontologies when composing multiple services, where often these ontologies may be different. Our approach can thus be applied to relate these ontologies in a way that take into account the preferences of these agents. This will allow agents to communicate more effectively about similar services. In addition, service composition agents will become better prepared for the introduction of future similar services by containing a more complete understanding of the domain.

Our approach can also be applied in web service discovery. Imagine the case when an agent is searching for a particular web service, and its ontology may not match the ontologies of other agents listing their web services. These agents need to be able to align consensually their ontologies to each other in order to find matches between their needed services and those advertised by another agent.

These scenarios are just some examples of the wide range of possible way to apply our approach. The next chapter demonstrates an alternative application in the context of ontology engineering for MAS.

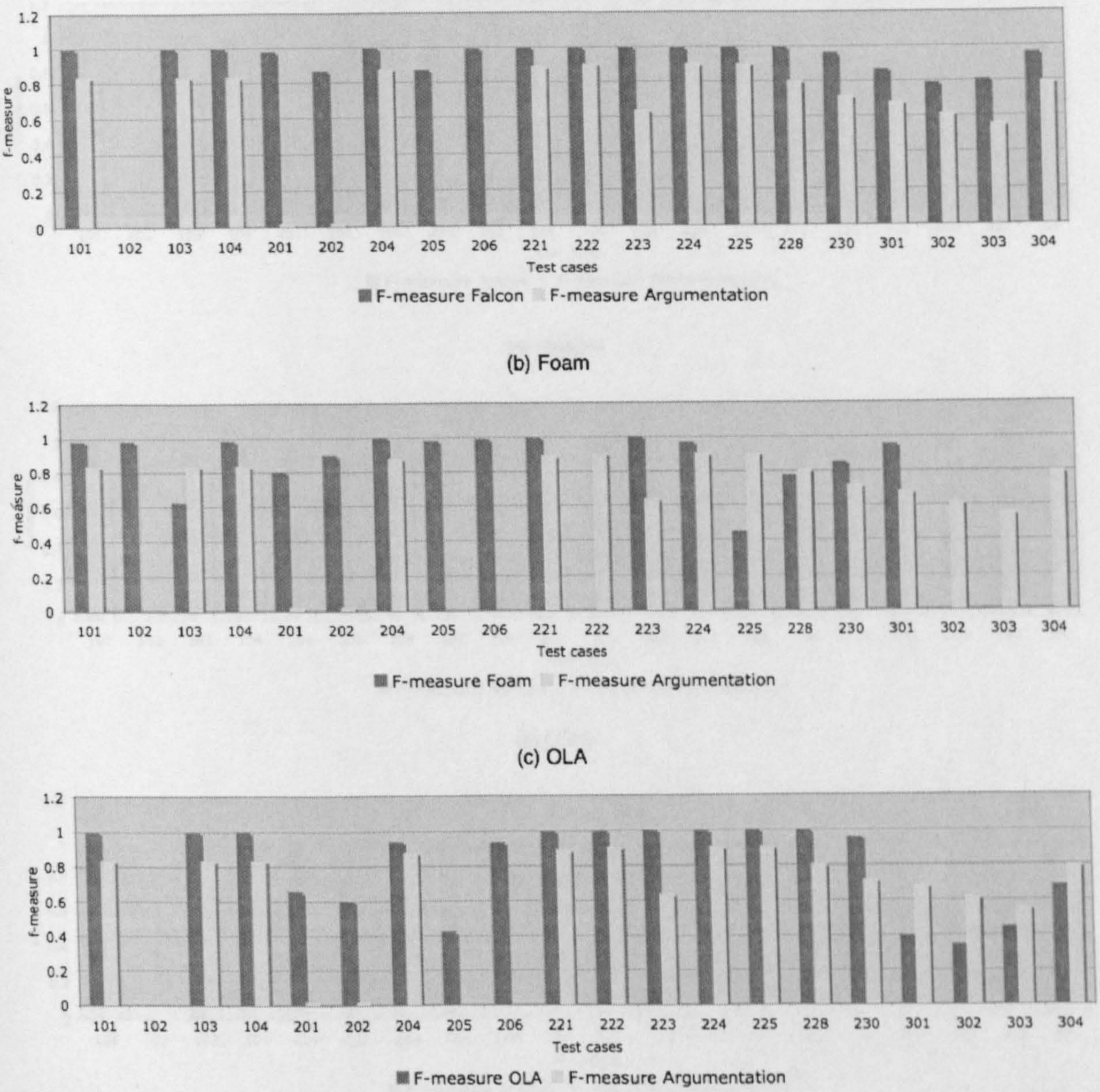

Figure 7.16: Plots showings the comparison of our argumentation approach wrt. current alignment tools (OAEI'05).

The experiments performed here mainly aimed to evaluate the ontology alignment of similar services in terms of precision and recall. Specifically, we considered set of hypothetical but diverse ontologies (15 in total) from a service-oriented travel domain. The ontologies represent the input messages of two Web services for ticket purchasing. Using these web services creates the potential for airlines to offer tickets purchasing services to other airlines. An example of two of these ontologies are shown in Figure 7.19.

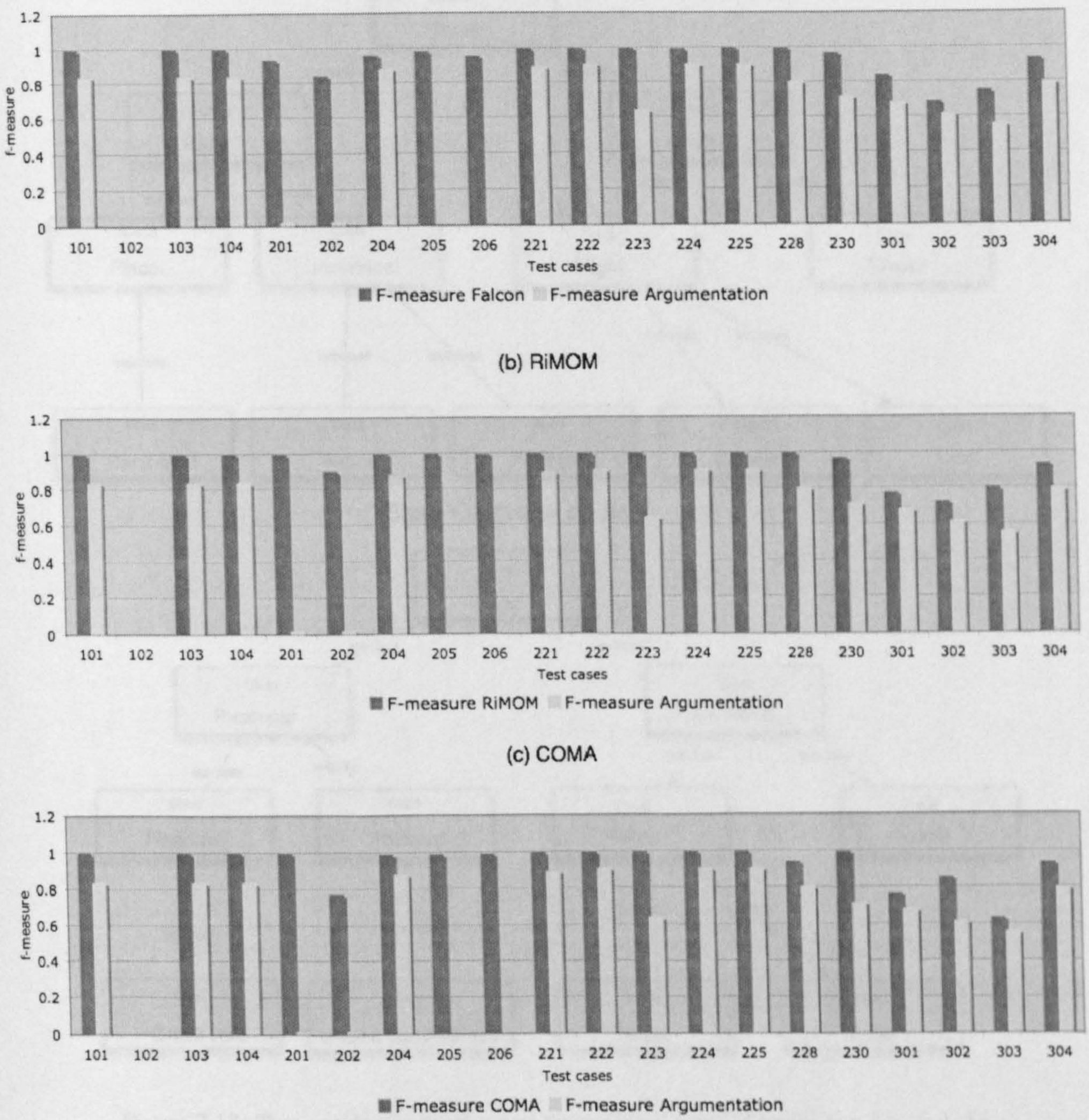

Figure 7.17: Plots showings the comparison of our argumentation approach wrt. current alignment tools (OAEI'06).

The precision and recall are measured over the set of agreed mappings achieved by the agents applying our approach. Our experiments simulate having a set of agents, each of which has a local ontology and is willing to communicate with the other agents. They try to reconcile their local ontologies to find consensus mappings. Similarly to previous experiments, the preferences for the agents have been chosen on the basis of the ontological information.

To decide whether a consensus set of mappings is obtained, we asked two ontology experts to carry out a manual mapping and we compared their results with ours. Precision and recall measurements are

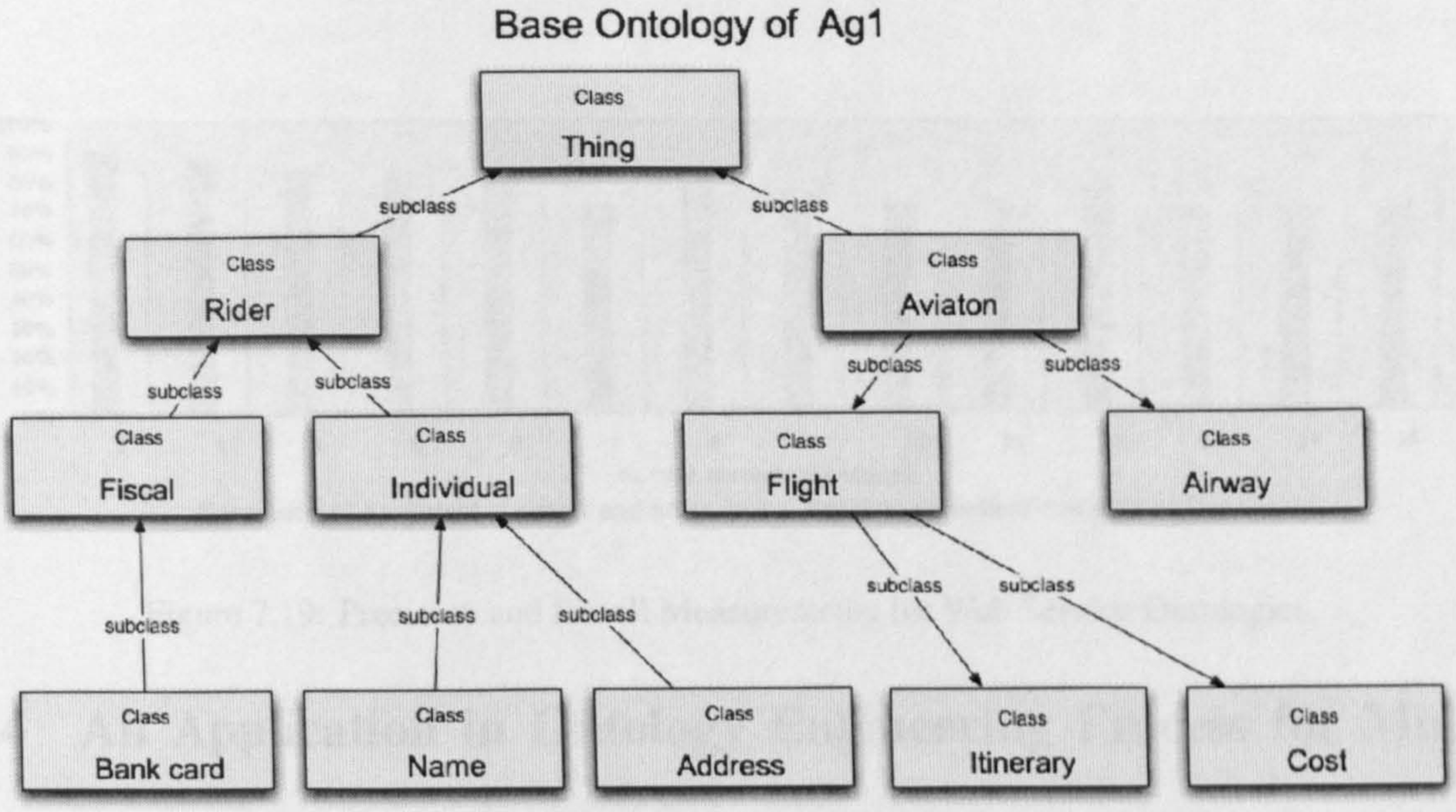

Figure 7.18: Two ontologies from a service-oriented travel domain for Ag_1 and Ag_2.

applied in the evaluation during the process of arguing the potential mappings of the ontologies at a time. The evaluation result is shown in Figure 7.19. The figure shows that the precision is almost constant for the first six ontologies, range from 90% to 83.1%, and then decreases very slowly with respect to the number of ontologies. There is a similar situation for recall, which range from 82% to 51%. The average precision and recall are, respectively, 72.3% and 62.8%, thus reflecting a promising result.

This experimentation demonstrates the promise of exploiting argumentation over ontological mapping in the domain of automated service composition. However, the experiment are too small to draw strong conclusions in this specific domain. A more comprehensive analysis, in a realistic web services environment, will need to be addressed in future work.

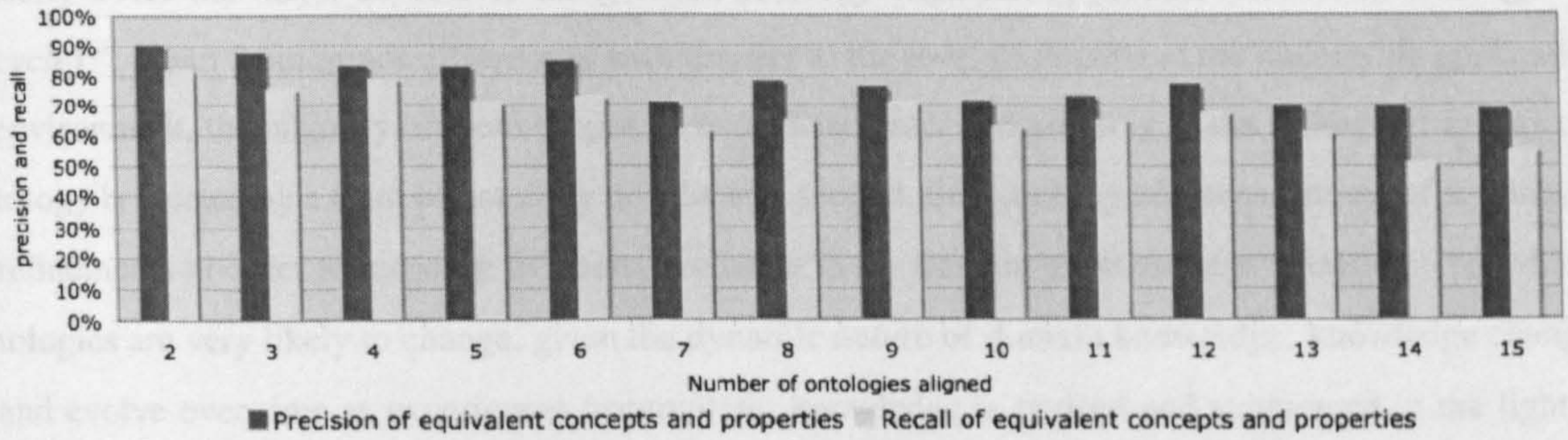

Figure 7.19: Precision and Recall Measurements for Web Service Ontologies.

7.4 An Application in Ontology Engineering Process for Multi-Agent Systems

This section briefly presents an application of how the instantiation of the mapping-oriented framework, developed in the previous chapters, has been used in the context of ontological engineering.
At first we introduce ontology engineering for MAS and the DINO lifecycle framework. Further on, we briefly present how we apply our approach in the context of the DINO framework and a simple illustrative example of concrete usage of the framework.

7.4.1 A Brief Introduction to Ontology Engineering for MAS

An ontology forms the basis for specifying the meaning of terms used by communicating agents. Therefore, an adequate methodology and technological support for building, maintaining and reusing ontologies must form one of the main stages in the development of a multi-agent system.
Due to the challenging and resource-intensive nature of the ontology building process special attention has been paid to automatic methods to assist this process, such as those based on the linguistics-based acquisition of domain-specific knowledge from document corpora [21]. The sum of these initiatives are referred to as *ontology engineering*.
As pointed out in [111], the engineering of an ontology is a

> *"social, evolving process which involves a geographically dispersed community of individuals, with different knowledge and expertise. Depending on the methodology applied to develop and maintain the ontology, its lifecycle may consist of a specific series of stages, in which the engineering team decides how to model the domain of interest in order to best suit the needs of the ontology users and the requirements of the application setting."*

Several engineering methodologies were elaborated in the last decades [115, 112, 78] and they typically cover the major aspects of the general ontology engineering process (known as *ontology life cycle*)[14]. Apart from minor differences with respect to the level of detail and the underlying application environment, the majority of them propose a two-phase process consisting of the following: first an ontology is created by a team of ontology developers; second, the ontology becomes subject of continuous refinements in order to response to a certain change in the domain or its conceptualization. Indeed, ontologies are very likely to change, given the dynamic nature of domain knowledge: knowledge changes and evolve over time as experiences accumulate; knowledge is revised and augmented in the light of deeper understanding; new knowledge is created while at the same time other knowledge passes into obsolescence. This is especially true in scientific domains, e.g., biomedicine, where newly discovered knowledge, which was previously unknown, classified or otherwise unavailable, may become accessible and have to be integrated.

Although ontology-engineering tools have matured over the last decade, they only marginally addressed the integration of new knowledge in these large and highly dynamic domains. Given this background, there is a clear demand for ontology engineering methodologies to allow agents/humans to collaboratively extract, build, refine, and integrate ontologies in a dynamic and data-intensive environments.

This is the goal of the DINO lifecycle framework. The DINO framework has been developed within the EU IST 6th frameworks Network of Excellence Knowledge Web to provide a solution for dealing with dynamics in large, scale, based on properly developed connection of ontology learning and dynamic collaborative development. The main key aspects of DINO are:

1. the ability to process new knowledge (resources) automatically whenever it appears, especially when it is impossible or inappropriate for humans to integrate it;

2. the ability to automatically integrate new knowledge with an existing ontology that has been manually and collaboratively designed (here denoted as "master ontology");

3. the ability to resolve logical inconsistencies between the new and existing knowledge;

4. the ability to automatically sort the new knowledge according to user-defined preferences and present it to them in a very simple way, thus further alleviating human efforts in the task of final incorporation of the knowledge.

In particular, the integration of evolving ontologies requires considerable customization in form of matching, merging, mapping or alignment. Different versions of an ontology need be compared and integrated into a common structure in order to cope with the heterogeneity, abundance and distributed

[14]The reader is referred to [116] for a state of the art in Ontology Engineering Methodologies.

nature of the data in a feasible manner. Ontology alignment will find commonalities among a set of ontologies and the affiliated instance data. Once similarities have been found, a merging process will generate a unified target ontology, obtained by aggregating concepts from the source ontologies which have been assigned a feasible similarity ranking.

However, a simple alignment between these ontologies is not enouth. In such settings, the resulting common ontologies need to adequately capture relevant commonalities and differences in meaning. These ontologies cannot easily be merged, as they represent strong individual interests and entrenched work practices of the various participants. This means that such value-laden ontologies can only be defined in a careful and gradual process of negotiation. In order for ontology to be truly a shared conceptualization of knowledge of a domain, support is needed for the team to cooperate and reach the appropriate degree of consensus on ontology mappings. In DINO, we are applying our argumentation approach to support domain experts to reach consensus through the negotiation of the mappings. To our knowledge, there has been no earlier attempt to use automated negotiation techniques in this specific context.

Although, the DINO framework is able automatically to deal with large amounts of changing data, the final incorporation of new knowledge still needs to be decided by the human users. This it will allow possible errors and inappropriate findings of the automatic techniques to be repaired. The key to success and applicability is to let machines do most of the tedious and time-consuming work and provide people with concise and simple suggestions on ontology extension.

7.4.2 Using the Mapping-oriented Argumentation framework into DINO

DINO stands as abbreviation of some of the key elements of the ontology lifecycle scenario: *Dynamics*, *INtegration*, *Ontology* and *Data* and *INtensive*. All these features express the primary aim of the DINO efforts – *to make the knowledge efficiently and reasonably manageable in data-intensive and dynamic domains.*

Figure 7.20 below depicts the phases of the DINO ontology life-cycle: *creation* (comprises both manual and automatic ontology development and update approaches), *versioning*, *evaluation* and *negotiation* (comprises ontology alignment and merging as well as negotiation among different possible alignments). The four main phases are indicated by the boxes annotated with their respective names. The *creation* phase of the ontology lifecycle has two major parts as it consists of the automatic ontology extraction (ontology learning) and community-driven manual (collaborative) development. Ontologies or their instances in time are represented by trees, with arrows expressing various kinds of information flow. The A boxes present actors (agents, institutions, companies, research teams etc.) involved in ontology development, where A_1 is zoomed-in in order to show the lifecycle's components in more detail.

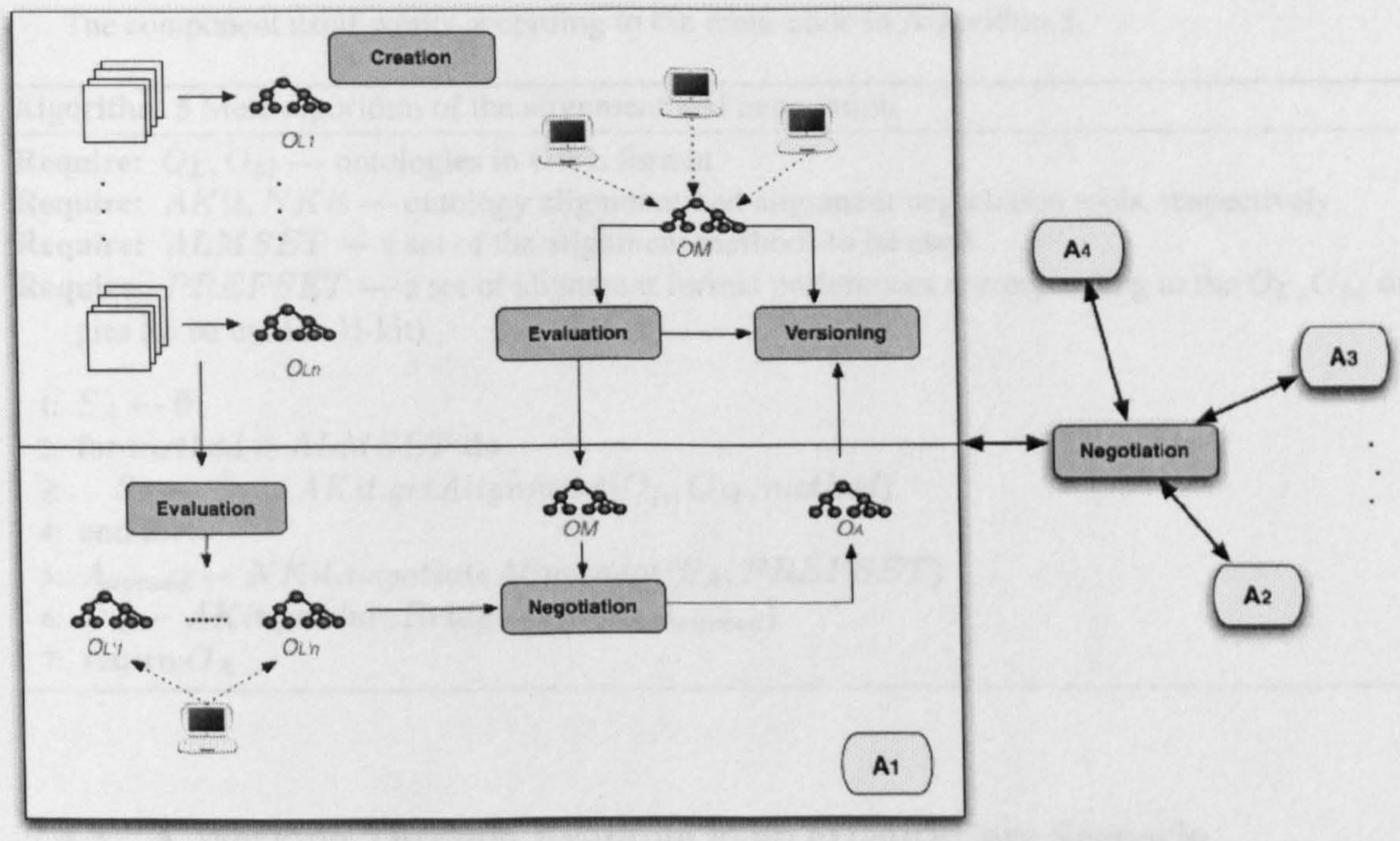

Figure 7.20: DINO Architecture.

The general dynamics of the DINO lifecycle goes as follows. Community experts build a specific domain ontology (the *Community* part of the *Creation* component). They use means for continuous ontology *evaluation* and *versioning* to maintain high quality and manage changes during the development process. If the amount of data suitable for knowledge extraction is too large to be managed by the community, an *ontology learning* process takes place. The results of the ontology learning are *evaluated* and partially (only the results with quality above a certain threshold are taken into account) integrated into the reference community ontology. The reference community ontology (the "master ontology") is typically more precise but relatively small. The integration is based on the alignment and subsequent merging of the learned and master ontologies, performed by the *negotiation* component. The *negotiation* component implements all claims proposed in the previous chapters on this thesis and can also be useful when interchanging or sharing the knowledge with other independent actors in the field.

Thus, the *negotiation* component uses argumentation-based negotiation that uses preferences over the types of mappings in order to choose the mappings that will be used to finally merge the ontologies. The preferences mainly depend on the characteristics of the ontology.

The *negotiation* component thus interfaces two tools – one for the ontology alignment discovery and one for the negotiation of agreed alignments, denoted here as *AKit* and *NKit*, respectively. For the former, the ontology alignment API has been used. For the negotiation, the mapping-oriented argumentation framework has been applied. The wrapper will then produce a merged ontology between the learned and master ontology. Thus merged ontology is used in consequent factual merging and refinement in the Ontology Reasoning/Management wrapper (see Appendix D for details).

The component itself works according to the meta-code in Algorithm 5.

Algorithm 5 Meta-algorithm of the alignment and negotiation

Require: O_L, O_M — ontologies in OWL format
Require: $AKit, NKit$ — ontology alignment and alignment negotiation tools, respectively
Require: $ALMSET$ — a set of the alignment methods to be used
Require: $PREFSET$ — a set of alignment formal preferences corresponding to the O_L, O_M ontologies (to be used in H-kit)

1: $S_A \leftarrow \emptyset$
2: **for** $method \in ALMSET$ **do**
3: $\quad S_A \leftarrow S_A \cup AKit.getAlignment(O_L, O_M, method)$
4: **end for**
5: $A_{agreed} \leftarrow NKit.negotiateAlignment(S_A, PREFSET)$
6: $O_A \leftarrow AKit.produceBridgeAxioms(A_{agreed})$
7: **return** O_A

7.4.3 A Working Through Example in an eHealthCare Scenario

In order to better illustrate the way the DINO framework can be used to enhance ontology development processes, we provide a complete example of its capabilities in terms of an eHealth case study. The goal of the example is to outline our experiences in applying the idea presented in this thesis in the context of DINO framework.

In this example, a set of ontology engineers (or agents, experts, etc.) are confronted with the task of building an medical ontology.

We have chosen a eHealth scenario for several reasons. First, the vast amount of data available in laboratory databases, together with the growing volume of electronically available clinical information call for automated (or at least semi-automated) methods for high-quality indexing, annotation, and cross-referencing through discovery of patterns and relationships. Thus, there is a need for the integration of data derived from divergent sources of the sort which ontology can provid. Second, the nature of this domain is highly dynamic and means to process and integrate new knowledge are fundamental.

In particular, in our scenario, we assume that a medicine institution needs to develop an ontology covering the basic concepts in clinical practice and research in the specific domain of Genomics and Proteomics Research. This is a typical example that show the needs to bridge the research and clinical practice and how to integrate specific knowledge, e.g. in the Gene Ontology (GO)[15] or UMLS[16] medical controlled dictionaries. These ontology can be used, for example, to model a multi-agent system for community healthcare management. We assume that an ontology O_M has been developed from a collaborative process but it needs to be extended by new findings in research (e.g. when new treatment or diagnosis methods are developed and published). The related information can be found in several

[15] http://www.geneontology.org/
[16] umlsinfo.nlm.nih.gov

documents, such as research papers, industry white-papers and so on. The Ontology Learning wrapper is able to generate an ontology O_L from these resources. Figure 7.21 presents a sample text fragment with the respective learned OWL O_L ontology (we omit the namespace for simplicity).

```
...while cerebellar astrocytoma is usually
discovered by means of CT... using a diagnostic
procedure of scanning... GVHD, an immune
dysfunction... GVHD, a disease being a type of
dysfunction...

. . .
<owl:ObjectProperty rdf:ID="discovered-by"/>
<owl:Thing rdf:ID="CT"/>
<owl:Thing rdf:ID="cerebellar-astrocytoma">
<discovered-by rdf:resource="#CT"/>
</owl:Thing>
<owl:Class rdf:ID="diagnostic-procedure"/>
<owl:Class rdf:ID="immune-dysfunction"/>
<owl:Class rdf:ID="dysfunction"/>
<owl:Class rdf:ID="scanning">
<rdfs:subClassOf rdf:resource="#diagnostic-procedure"/>
</owl:Class>
<immune-dysfunction rdf:ID="GVHD"/>
<owl:Class rdf:ID="disease">
<rdfs:subClassOf rdf:resource="#dysfunction"/>
</owl:Class>
. . .
```

Figure 7.21: A Text Sample and the Learned Ontology.

Excerpts of O_M ontology is shown in Figure 7.22.

The master and learned ontologies are then aligned and the set of possible mappings are the following[17]:

$m56$=$\langle Learned$: $scanned, Master$: $DiagnosisProcedure, 0.725, =\rangle$;

$m161$=$\langle Learned$: $Disease, Master$: $Disease, 1, =\rangle$;

$m110$=$\langle Learned$: $immune-dysfunction, Master$: $Dysfunction, 0.88, =\rangle$;

$m109$=$\langle Learned$: $dysfunction, Master$: $Dysfunction, 1, =\rangle$;

$m53$=$\langle Learned$: $diagnostic-procedure, Master$: $DiagnosisProcedure, 0.72, =\rangle$;

We have two audiences, which preferences have been chosen on the features of the two ontologies: $\mathcal{R}_1$

[17] We only consider a subset of these ontologies.

```
...
<owl:ObjectProperty rdf:ID="InstrumentalProperty"/>
<owl:ObjectProperty rdf:ID="DiscoveredUsing">
<rdfs:subPropertyOf rdf:resource="#InstrumentalProperty"/>
<rdfs:range rdf:resource="#Manifestation"/>
<rdfs:domain rdf:resource="#DiagnosisProcedure"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Manifestation"/>
<owl:Class rdf:ID="Procedure"/>
<owl:Class rdf:ID="DiagnosisProcedure"/>
<rdfs:subClassOf rdf:resource="#Procedure"/>
</owl:Class>
<owl:Class rdf:ID="SoftTissueCytoma"/>
<owl:Class rdf:ID="AstroCytoma"/>
<rdfs:subClassOf rdf:resource="#SoftTissueCytoma"/>
</owl:Class>
<owl:Class rdf:ID="Disease"/>
<owl:Class rdf:ID="Dysfunction"/>
<rdfs:subClassOf rdf:resource="#Disease"/>
</owl:Class>
...
```

Figure 7.22: A Master Ontology Sample.

Table 7.4: Except of Arguments and Counter-Arguments

Argument	$\mathcal{V}$
$\langle \exists m = \langle superclass(scanning), superclass(DiagnosisProcedure), \equiv, \rangle, m56, + \rangle$	ES
$\langle Label(disease) \approx_T Label(Disease), m161, + \rangle$	T
$\langle Label(immune-dysfunction) \approx_T Label(Dysfunction), m110, + \rangle$	T
$\langle Label(dysfunction) \approx_T Label(Dysfunction), m109, + \rangle$	T
$\langle Label(diagnostic-procedure) \approx_T Label(DiagnosisProcedure), m53, + \rangle$	T
$\langle \exists m = \langle subclass(disease), subclass(Disease), \equiv, \rangle, m161, - \rangle$	ES
$\langle \exists m = \langle subclass(immune-dysfunction), subclass(Dysfunction), \equiv, \rangle, m110, - \rangle$	ES
$\langle \exists m = \langle subclass(dysfunction), subclass(Dysfunction), \equiv, \rangle, m109, - \rangle$	ES
$\langle \exists m = \langle superclass(disease), superclass(Disease), \equiv, \rangle, m161, - \rangle$	ES
$\langle \exists m = \langle siblingclass(disease), siblingrclass(Disease), \equiv, \rangle, m161, - \rangle$	ES

with respect to O_L, which prefers terminology to internal structural, $(T \succ_{\mathcal{R}_1} IS)$[18] and $\mathcal{R}_2$, with respect to O_M, which prefers external structure to terminology to internal structural $(ES \succ_{\mathcal{R}_2} T_s ucc_{\mathcal{R}_2} IS)$[19]. A sample of arguments and counter-arguments generated are shown in Table 7.4, that shows each argument with its type $\mathcal{V}$. A screenshot of the mappings and arguments is illustrated in Figure 7.23. Based upon all arguments and how these arguments attacks each other, the argumentation frameworks for each participant can be built. The argumentation frameworks bring the arguments together so that they can be evaluated. The evaluation is carried out by computing the preferred extensions for the resulting argumentation frameworks, respect the audiences. The negotiated mappings achieved at the end

[18] $\mathcal{R}_1$ because O_L contains fews properties and is lacking in structure.
[19] $\mathcal{R}_2$ because O_M has a rich hierarchy of concepts and their inter-relationships but only fews properties.

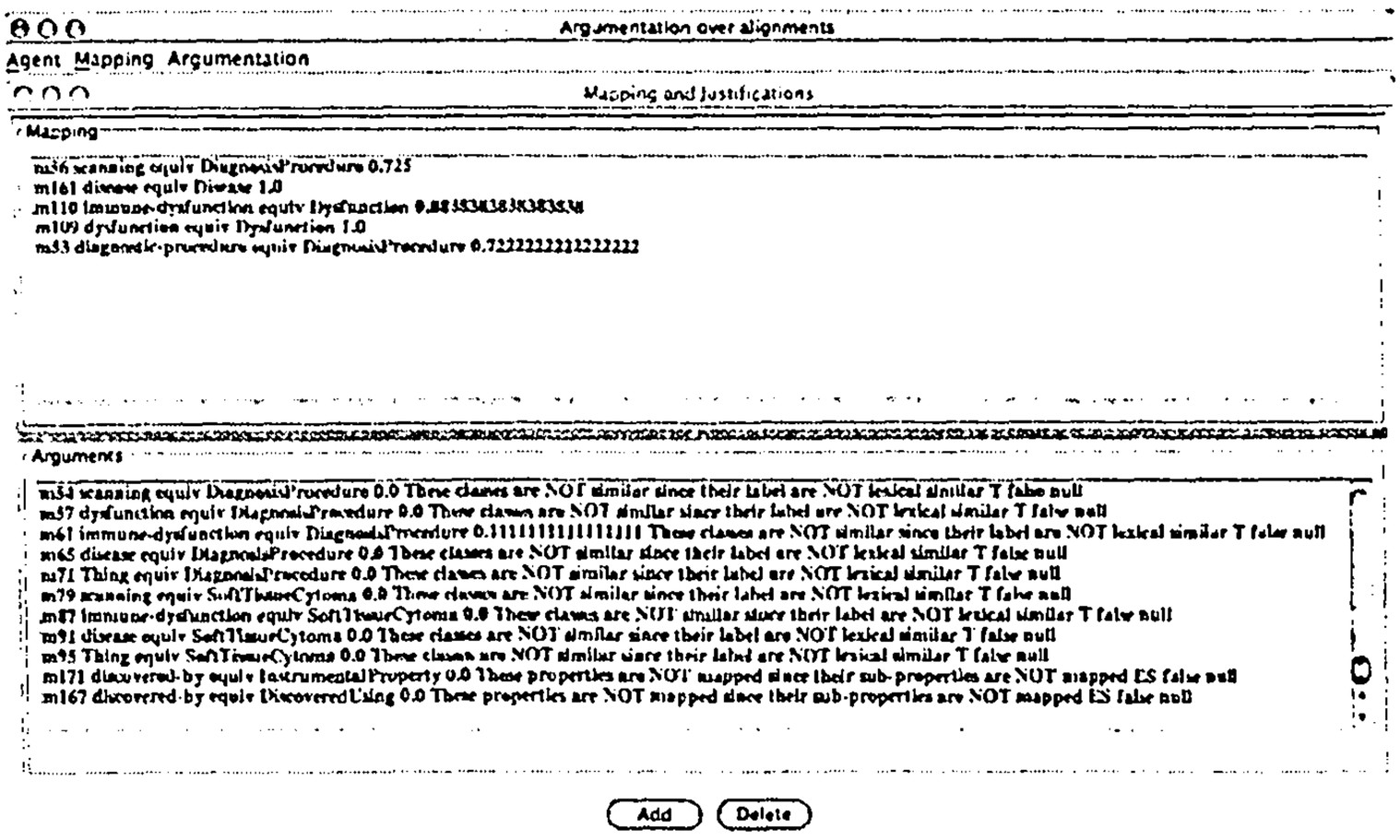

Figure 7.23: Candidate Mappings and Arguments.

are shown in Figure 7.24. These mapping are now used to merge the learned and master ontologies.

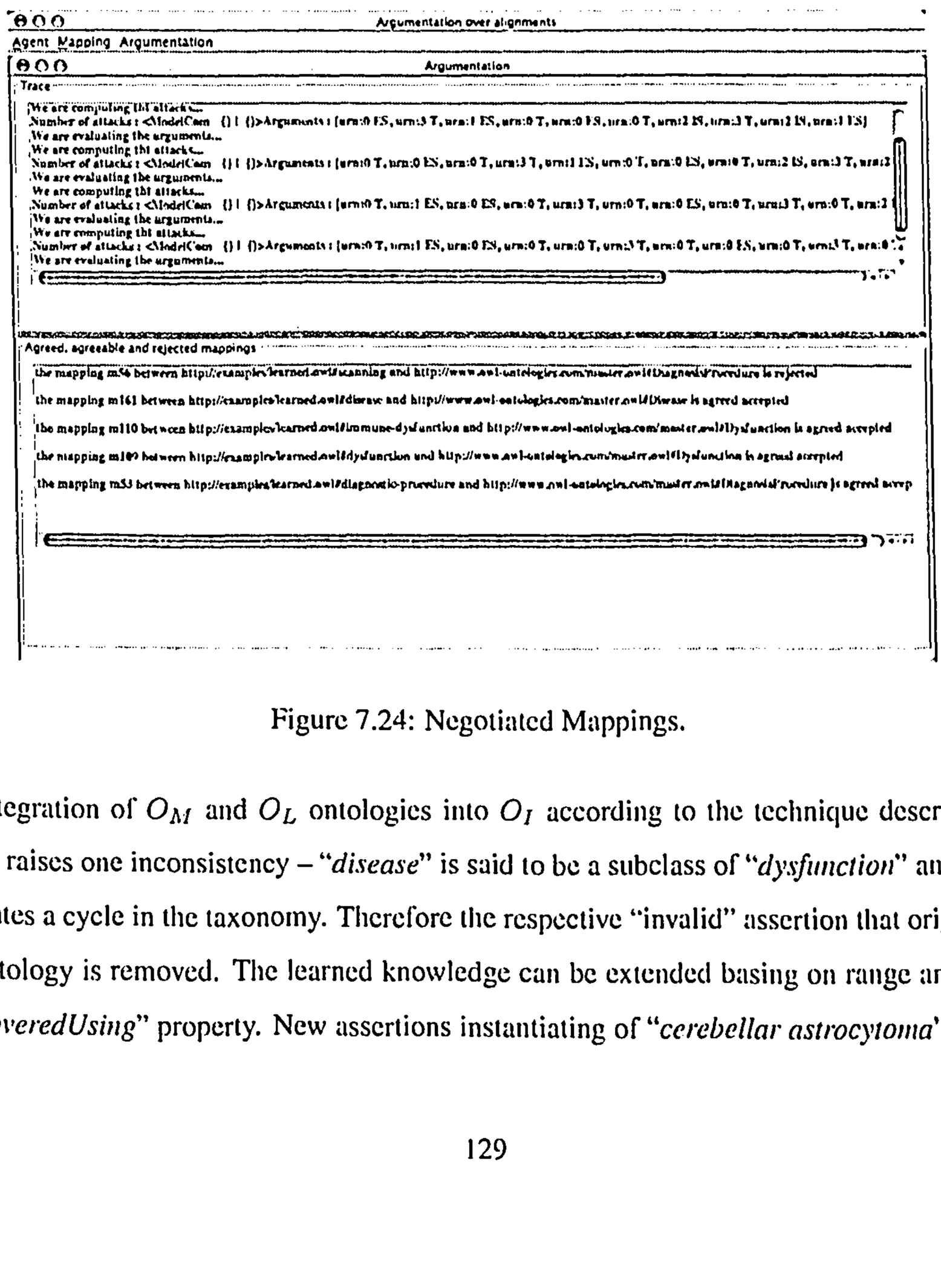

Figure 7.24: Negotiated Mappings.

The integration of O_M and O_L ontologies into O_I according to the technique described in Section D.1.3, raises one inconsistency – "*disease*" is said to be a subclass of "*dysfunction*" and vice versa, which creates a cycle in the taxonomy. Therefore the respective "invalid" assertion that originated from the O_L ontology is removed. The learned knowledge can be extended basing on range and domain of the "*DiscoveredUsing*" property. New assertions instantiating of "*cerebellar astrocytoma*" (instance of

"*Manifestation*") and "*CT*" (instance of "*DiagnosisProcedure*") can now be inferred.

The triples (with O_L equivalent labels replaced by those from O_M) from the O_I merge are produced, together with respective suggestions based on the differences between O_I and O_M. The sorted triples and their transformations into natural language statements are shown in Table 7.5[20].

<AstroCytoma rdf:ID="cerebellar-astrocytoma"/>	CEREBELLAR ASTROCYTOMA is a *new istance* of ASTROCYTOMA.
<Manifestation rdf:ID="cerebellar-astrocytoma"/>	CEREBELLAR ASTROCYTOMA is a *new istance* of MANIFESTATION.
<DiagnosisProcedure rdf:ID="CT"/>	CT is a *new istance* of DIAGNOSIS PROCEDURE.
<immune-dysfunction rdf:ID="GVHD"/>	GVHD is a *new istance* of IMMUNE DYSFUNCTION.
<owl:Class rdf:ID="scanning"> <rdfs:subClassOf rdf:resource= "#DiagnosisProcedure"/> </owl:Class>	A *new class* SCANNING is a *sub-class* of DIAGNOSIS PROCEDURE.
<owl:Thing rdf:ID="cerebellar-astrocytoma"> <DiscoveredUsing rdf:resource="#CT"/> </owl:Thing>	CEREBELLAR ASTROCYTOMA is DISCOVERED USING CT.
<owl:Class rdf:ID="immune-dysfunction"> <rdfs:subClassOf rdf:resource="#Dysfunction"/> </owl:Class>	A *new class* IMMUNE DYSFUNCTION is a *sub-class* of DYSFUNCTION.

Table 7.5: Extension Triples and the respective NL Suggestions

Note that the DINO framework can also be used if we just need to align and possibly extend the ontologies by another institution's knowledge base – the only difference is that we do not perform the ontology learning and also omit retractions in the integration process. This can be applied in the critical task of inter-mediation of medicine information.

7.5 Conclusions

This chapter has introduced the implementation, evaluation and applicability of the mapping-oriented argumentation framework presented in this thesis. The prototype, realized under the EU Knowledge Web project, implements many of the core ideas elaborated in this thesis. It provides methodological and technological support to find and and argue over a set of mappings. The way the implementation has been tested and the results of this endeavor are also presented and discussed in this chapter.

The preliminary results of these experiments are promising and suggest that our argumentation approach can be beneficial and provides an effective solution to the problem of distributed agents to dynamically align their heterogeneous ontologies. In particular, we have shown that the consideration of different agents' preferences, which we believe is fundamental to reflect the autonomous and rational features of agents, can also improve the quality of the agreed mappings. The explanation is that our approach collects and integrates evidence for ontology mappings, in the form of arguments. This allows

[20]The respective processes described in Sections D.1.5 and D.1.6. The relevance values are assumed to be 0 in this example without any exact preferences defined. The reader can refer to [92] for an example with relevance values computed according to certain provided user-preferences.

the agents to browse a space of candidate mappings in order to select, via the definition of acceptability of arguments, the adequate candidates that reflect their personal preferences. We also demonstrated that such preferences are the key element of the approach, which effect the quality of the mappings reached. This subsequently reinforces our assumption that the agents should rationally decide their preferences principally on the basis on the ontological information.

One of the main problems encountered with our approach is that the argumentation can be somewhat computationally expensive. This is not entirely suitable for our target domain, where efficient run-time performance is required for agent communication. However, the cost of argumentation, in terms of time and costs taken, are, in general, fairly negligible when compared to the overall time taken for the agent communication process. An interesting topic for future research would be to investigate the cases when it is worth while to apply argumentation, in terms of resources and time employed. Due to time and resource limitations, evaluation in this line has not been possible so far.

In order to evaluate the performance of our approach, we have measured the quality of the determined mappings using precision and recall. The goal of using these metrics is to check if, given two ontologies, the alignments reached through argumentation is the same as the alignment that a human expert would generate - thus a mapping is *correct* is if a human judge agrees with it. They provide a simple mechanism for comparing different systems in a consistent and repeatable way. Using these metrics we can see that, despite our approach being created to produce mappings that take into account the different agents' points of view, our approach performs well in comparison with current ontology alignment tools.

However, such use of metrics has the drawback that they do not take into account the agent's context. An alignment may deviate from the expected human result but it may be correct for a specific agent and conversely another alignment may be very close to it but be incorrect for the agent. The reason for this is that if an alignment is correct from the human viewpoint it does not mean that it should be correct for an agent. It will be helpful to formally define when a mapping is correct for an agent and to evaluate our approach over it.

The experiments conclude with some evaluation in the domain of web service. Because there is no agreed-upon global ontology, web services supplied by different providers typically have individual and unique semantics, described by independently developed ontologies. The seamless connection of these distributed Web services for business-to-business applications depends heavily on reconciling disparate semantics, possibly by integrating the ontologies. We have discussed and evaluated how our approach can aim to reconcile ontologies by reaching agreements over candidate mappings. However, these experiments were too small to draw strong conclusions and a more comprehensive analysis, with larger data groups and other experiments in this domain, have to be addressed in future.

This chapter concludes with a potential application of the mapping-oriented argumentation framework to the specific field of ontology engineering. Towards this aim, we have briefly presented the DINO ontology life-cycle framework, in which we have applied our approach.

The domains which to DINO could be applied, contain many features we believe to be typical for open heterogeneous environments and therefore provide realistic scenarios to test the ideas introduced in the preceding chapters.

By applying our approach to the community experts and reaching the appropriate degree of consensus on ontology mappings, we have shown that all claims we made in the previous chapter can be applied to the integration of learned and collaboratively developed knowledge and overcome ontology problems.

To our knowledge, this work constitutes the first application of automated negotiation technology to the problem of ontology evolution. In particular, it is the first attempt at using argumentation techniques to support the integration process.

Part IV

Synopsis

Chapter 8

Conclusions and Future Work

"What we call the beginning is often the end.
And to make an end is to make a beginning.
The end is where we start from. "
- T. S. Eliot.

This chapter provides a summary of the research carried out and presents a number of avenues for future research. Initially, in Section 8.1 we review the work that has been carried out, we discuss the implications of our work and enumerate the key research achievements. The chapter concludes with a number of future research areas which were identified throughout this research, and which address some of the main limitations of the work presented (Section 8.2).

8.1 Review of Contributions

As stated in the introductory chapter, the aim of this thesis was *to develop an approach towards supporting ontology-based shared understanding in open multi-agent systems.*

In pursuing our goal, we focus on producing an argumentation-based approach over ontology mappings to overcome the problem of a lack of communication between heterogeneous agents, in a way that conforms to the characteristics of agents and their enviroment. The work we have carried out provides a concrete instantiation of the meaning negotiation process that we would like agents to achieve, which it is believed may provide the basis to "shared understanding".

This research aim addresses the more specific research objectives set out in Section 1.2. Below we summarise the work that has been carried out, our contributions and state how they address our research goals.

We have first reviewed the theoretical foundation of this research. An obvious point of departure was therefore be an introduction to the fundamental principles behind Agents, Multi-Agent Systems and Agent Communication in Chaper 2. We have defined the notion of an agent and multi-agent systems and emphasized the fact that an agent should be able to interact with its environment and with other agents either to solve common problems or either simply reach its goal. We have presented open MAS as an important class of those systems, where agents can freely join and leave at any time and where the agents are owned by various stakeholders with different aims and objectives. This discussion led to the recognition that the only thing that the agents of an open system have in common is their ability to *communicate*. Using its communicative abilities, an agent can work together cooperatively to accomplish its goal.

In Chapter 3, ontologies have been advocated as key components in inter-agent communication, by providing the definitions of the vocabularies used by agents to describe the world. It was shown that an agent can use such a vocabulary to express its beliefs and actions, and so communicate about them. However, a careful study on the nature of ontologies and open MAS has revealed that specifying an ontology alone is generally not sufficient for this purpose. Autonomous agents in open systems may be heterogeneous and may view their world differently.

Therefore, Chapter 4 has examined the resulting semantic interoperability problems, presenting *ontology alignment* as one of the prerequisite to enable interoperability between agents that use different ontologies. We have analysed the current approaches (ontology alignment and meaning negotiation) to semantic interoperability, and we have reached the conclusion that they have presented weaknesses, especially when applied in an open environment, which was one of the initial requirements of this thesis. In defining this analysis, *our first contribution has been to identify a set of requirements that the mechanisms for shared understanding between agents should fulfill in open MAS.* In particular, we have seen that a major requirement for achieving shared understanding in open MAS is that an alignment should conform with the characteristics of agents. Since agents are autonomous and self-interested, they should freely choose these mappings and such choice should be made dynamically (at run time), due to the fact that the agents have no prior knowledge of either the existence or constraints of other agents.
This leads us to an approach to dynamically negotiate the meaning of the terms they use to communicate.

Chapter 5 then have introduced our argumentation based approach over ontology mappings. We have shown how agents can generate and exchange different arguments, that support or reject possible ontology mappings. Each agent can then decide, according to its preferences, whether to accept or refuse a mapping. We have clearly identified the set of potential arguments, which were grounded on the underlying ontology language OWL. In order to give a precise structure to these arguments, we have summarized the reasons for the justification of candidate OWL ontology alignment in an (extensible) set of argument schemes. The Value-based Argumentation framework of Bench-Capon has been used

to express agents preferences between the categories of arguments.

In Chapter 6, we have presented in detail the multi-agent environment for our mapping-oriented argumentation framework. We have discussed in details the architecture of our MAS, the argumentative agents and their capabilities for argument generation, and a protocol for evaluating the acceptability of a mapping.

Our approach represents a first attempt to extend the notion of *reaching agreement* through automated argumentation-based negotiation over ontology mappings between heterogeneous agents. Up to this moment ontology alignment processes have concentrated their attention on improving their performance and accuracy but very few have addressed the problem of finding dynamic agreements over the concepts in the messages exchanged between agents. Our approach allows agents to dynamically choose the best vocabulary to use for an interaction, without having to agree on a fixed, predefined vocabulary.

At the same time, we have also presented a mechanism to express agents' preferences over the types of mappings they use when aligning the ontologies, in order to reflect the behaviour and characteristics of the notion of *agency*. Thus, our account is intended *to give a more effective and applicable approach to shared understanding that reflects the characteristics of autonomy and rationality of agents and tailored to the requirements of an open environment.*

The core theoretical considerations of our research have been prototypically implemented within the Knowledge Web project. The prototype, presented in Chapter 7, acted as a proof of concept of the practical applicability of the mapping-oriented argumentation framework, implementing many of the core ideas elaborated in this thesis. The prototype was fully implemented in JAVA by using JADE agent development environment.

Our research has been mainly evaluated by using the Ontology Alignment Evaluation Initiative (OAEI) test suite as test ontologies. The task of such an evaluation has validated whether or not our approach generates consensual mappings, explained the run-time behavior of our system, and compared with other approaches. To decide whether a correct set of agreed ontology mappings is obtained, we have used standard information retrieval metrics to assess the results of the tests: Recall, Precision and F-measure. The results of the overall procedure are promising and suggested that our argumentation approach can be beneficial and an effective solution to the problem of distributed agents to dynamically align their heterogeneous ontologies. In particular, we have demonstrated that the consideration of different agents points of view, which we believe is fundamental to reflect the autonomous and rational features of agents, can also improve the quality of the agreed mappings. The mapping-oriented argumentation framework has been applied in the context of DINO framework, developed within the EU Knowledge Web to support the collaborative ontology development process, in dynamic and data-intensive domains. In DINO, we have applied our argumentation approach to supports domain experts

to reach consensus through the negotiation of the ontology mappings between existing and new ontologies, and thus used to generate new ontologies.

The last contribution has thus been the implementation a proof of concept of an argumentation-based mapping framework in order to demonstrate its applicability and effectiveness in inter-agent communication and show its benefits on the field of ontology engineering for MAS.

8.2 Future Work

During the course of this work, a number of areas of interest came to our attention which we were not able to further develop or study due to time constraints. In this section, we summarise the areas which we consider to be worthy of future research and outline a possible path for the future development of some ideas discussed in this thesis.

As we discussed in the previous sections, the evaluation of our argumentation based approach was not aimed at any particular scenario and has only proven useful in the domain of ontology engineering and marginally in the domain of web services. Therefore, we encourage the application of the results of this work in the context of different domains. In particular, an important domain would be Information Retrieval (IR), which is concerned with the problem of how agents should find a (ranked) set of documents that are relevant for a specific information need of a user. Using the mapping-oriented argumentation approach proposed in this thesis to deal with the differences in the vocabularies used by the user and the selected information resources could benefit the field in several ways. Principally, it would be beneficial to select only those data sources available that reflect the users preferences and context, which seems desirable in light of the vast amount of information available in electronic format.

A second line of future research would be to investigate how to apply our argumentation framework to argue about the entire alignment, and not only the individual candidate mappings. Indeed, the arguments that we presented in this thesis are statements about single mappings - mappings between single entities in two ontologies. However, the state of the art also proposes mechanisms for measuring the global similarity between the two entire ontologies, see [81] for instance. Therefore, new arguments about whole alignments are needed to be developed when a global similarity measure between two ontologies is applied.

Another challenge emerging from this research would be to further develop the structure of the justifications. In this thesis, we have introduced the concept of *justification* of an ontology mapping as the meaningful explanation of why that mapping has been generated. This explanation includes information concerning where a mapping came from, how it was derived (or retrieved) and so on. However, in this thesis, its structure has not be formally defined. A formal description of such justifications, using

Description Logics for example, can be used as a deductive proof trace of the provenience of a mapping, and thus the generation of the arguments can be, for example, formally infered by a DL reasoner. In this way, the satisfiability of the justifications can be completely verified. Moreover, use of DL will provide a consensual representation of the justifications, that enables them to be portable and distributed among mapping engines.

Another area to address in future work would be to explore a quantitative analysis of the *economic knowledge* discovered by the alignments in addition to the *technical knowledge*, which can be useful to revise the applicability of the argumentation-based approach. A thorough and in-depth empirical and economic analysis of how the mapping engines perform the mapping tasks and the quality of the alignment produced is needed to decide whether to apply our approach, reuse the alignments, and so on. Relevant questions that need to be answered in this analysis include the following:

- What are the scale and complexity of the mapping tasks?
- What is the quality of the expected results from a mapping task?
- What kind of time and resource constraints are there in a mapping task?
- What is the economic effort of the mapping tasks?

Answering the above questions would help us in identifying a *"means to estimate and control the overall costs and utilities of alignment approaches"*. A cost model specific to alignment strategies can be essential to calculate the amount of effort that they will invest during an alignment, and consequently, for the choice and use of those approaches. Thus, in contrast to current ontology alignment procedures, the choice of alignment can based on three clearly identified elements: (i) the estimation of the efforts invested during its generation; (ii) the estimation in terms of utility of the results; and (iii) the personal point of view of each agent, which is private and specific to each agent.

Although, we are satisfied that the implementation of our framework, presented in Chapter 7, meets our objective of supporting the generation and exchange of arguments concerning ontology alignments, it fails to simulate two major aspects that are typical of MAS and which will become even more important if the intention is to allow autonomous agents to interoperate in real open environments: (i) the implemented agents are not fully autonomous (their preferences and threshold have been chosen by a user GUI) (ii) these agents do not have any specific role or task. An obvious next step in our work is to evaluate the appropriateness of our argumentation approach with fully autonomous agents that carry out specific tasks, in a real open environment.

Part V

Appendicies

Appendix A

OWL – Web Ontology Language

This appendix provides a summary of the OWL Lite and OWL DL language features. The content of this appendix is based on the W3C Recommendation[1].

A.1 OWL Lite Synopsis

Construct	Explanation
`Class`	A class defines a group of individuals that belong together because they share some properties.
`rdfs:subClassOf`	Class hierarchies may be created by making one or more statements that a class is a subclass of another class. For example, the class `Person` is a subclass of the class `Mammal`.
`rdf:Property`	Properties can be used to state relationships between individuals or from individuals to data values. An example of properties is `hasChild`.
`rdfs:subPropertyOf`	Property hierarchies may be created by making one or more statements that a property is a subproperty of one or more other properties.
`rdfs:domain`	A domain of a property limits the individuals to which the property can be applied.
`rdfs:range`	The range of a property limits the individuals that the property may have as its value.
`Individual`	Individuals are instances of classes, and properties used to relate one individual to another.
`equivalentClass`	Two classes may be stated to be equivalent.
`equivalentProperty`	Two properties may be stated to be equivalent.
`sameAs`	Two individuals may be stated to be the same.
`differentFrom`	An individual may be stated to be different from other individuals.
`AllDifferent`	A number of individuals may be stated to be mutually distinct.
`inverseOf`	One property may be stated to be the inverse of another property.

[1] http://www.w3.org/TR/2004/REC-owl-features-20040210/

Construct	Explanation
`TransitiveProperty`	Properties may be stated to be transitive. For example, `ancestor` is a transitive property.
`SymmetricProperty`	Properties may be stated to be symmetric.
`FunctionalProperty`	Properties may be stated to have a unique value. For example, `hasPrimaryEmployer` is FunctionalProperty.
`InverseFunctionalProperty`	Properties may be stated to be inverse functional.
`allValuesFrom`	The restriction `allValuesFrom` is stated on a property with respect to a class.
`someValuesFrom`	The restriction `someValuesFrom` is stated on a property with respect to a class.
`minCardinality`	Cardinality is stated on a property with respect to a particular class. For example, the class `Parent` has a minimum cardinality of 1 on the `hasOffspring` property.
`maxCardinality`	Cardinality is stated on a property with respect to a particular class.
`cardinality`	Cardinality is provided as a convenience when it is useful to state that a property on a class has both `minCardinality` 0 and `maxCardinality` 0 or both `minCardinality` 1 and `maxCardinality` 1.
`intersectionOf`	OWL Lite allows intersections of named classes and restrictions. For example, the class `EmployedPerson` can be described as the intersection of `Person` and `EmployedThings`.

A.2 OWL DL Synopsis

Construct	Explanation
`oneOf`	Classes can be described by enumeration of the individuals that make up the class.
`hasValue`	A property can be required to have a certain individual as a value.
`disjointWith`	Classes may be stated to be disjoint from each other.
`unionOf`, `intersectionOf` `complementOf`	OWL DL allow arbitrary Boolean combinations of classes and restrictions.

Appendix B

Argumentation Ontology

In this appendix we show the OWL version of the Argumentation Ontology (see Chapter 5), that models arguments for and against potential mappings between heterogeneous agents' ontologies.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/AgentArgueAlignment.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
xml:base="http://www.owl-ontologies.com/AgentArgueAlignment.owl">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="Alignment">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001/
      XMLSchema#int">1</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="hasmapping"/>
      </owl:onProperty>
    </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf rdf:resource=
"http://www.w3.org/2002/07/owl#Thing"/>
```

```
</owl:Class>
<owl:Class rdf:ID="mapping"/>
<owl:Class rdf:ID="Value"/>
<owl:Class rdf:ID="Position"/>
<owl:Class rdf:ID="Disagreement">
  <rdfs:subClassOf rdf:resource="#Position"/>
</owl:Class>
<owl:Class rdf:ID="Justification"/>
<owl:Class rdf:ID="Argumentation">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:minCardinality rdf:datatype="http://www.w3.org/2001
      /XMLSchema#int">1</owl:minCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="isComposed"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource=
  "http://www.w3.org/2002/07/owl#Thing"/>
</owl:Class>
<owl:Class rdf:ID="Agreed">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Consensus"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Agreeable">
  <rdfs:subClassOf rdf:resource="#Consensus"/>
</owl:Class>
<owl:Class rdf:ID="Argument"/>
<owl:Class rdf:ID="Rejected">
  <rdfs:subClassOf rdf:resource="#Consensus"/>
</owl:Class>
<owl:Class rdf:ID="Agent"/>
<owl:Class rdf:ID="Agreement">
```

```
  <rdfs:subClassOf rdf:resource="#Position"/>
</owl:Class>
<owl:ObjectProperty rdf:about="#isComposed">
  <rdfs:range rdf:resource="#Argument"/>
  <rdfs:domain rdf:resource="#Argumentation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasValue">
  <rdfs:range rdf:resource="#Value"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasmapping">
  <rdfs:domain rdf:resource="#Alignment"/>
  <rdfs:range rdf:resource="#mapping"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="attacks">
  <rdfs:domain rdf:resource="#Argument"/>
  <rdfs:range rdf:resource="#Argument"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="onmapping">
  <rdfs:domain rdf:resource="#Consensus"/>
  <rdfs:range rdf:resource="#mapping"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasOpponent">
  <rdfs:range rdf:resource="#Agent"/>
  <rdfs:domain rdf:resource="#Argument"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="positionOn">
  <rdfs:range rdf:resource="#mapping"/>
  <rdfs:domain rdf:resource="#Position"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasProponent">
  <rdfs:domain rdf:resource="#Argument"/>
  <rdfs:range rdf:resource="#Agent"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasJustification">
  <rdfs:range rdf:resource="#Justification"/>
```

```
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="souceTool">
  <rdfs:range rdf:resource="http://www.w3.org/2001/
  XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasConfidence">
  <rdfs:range rdf:resource="http://www.w3.org/2001/
  XMLSchema#float"/>
  <rdfs:domain rdf:resource="#mapping"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasOntology">
  <rdfs:domain rdf:resource="#Agent"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
  XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasRelation">
  <rdfs:range>
<owl:DataRange>
  <owl:oneOf rdf:parseType="Resource">
    <rdf:rest rdf:parseType="Resource">
      <rdf:rest rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/
            XMLSchema#string"
            >disjoint</rdf:first>
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/
            22-rdf-syntax-ns#nil"/>
          </rdf:rest>
          <rdf:first rdf:datatype="
  http://www.w3.org/2001/XMLSchema#string"
  >subsume</rdf:first>
</rdf:rest>
<rdf:first rdf:datatype="
http://www.w3.org/2001/XMLSchema#string"
>subsumed</rdf:first>
```

```
      </rdf:rest>
      <rdf:first rdf:datatype=
      "http://www.w3.org/2001/XMLSchema#string"
      >equivalent</rdf:first>
    </owl:oneOf>
  </owl:DataRange>
  </rdfs:range>
  <rdfs:domain rdf:resource="#mapping"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isSupport">
  <rdfs:domain rdf:resource="#Argument"/>
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:rest rdf:parseType="Resource">
          <rdf:first rdf:datatype="http://www.w3.org/2001/
          XMLSchema#boolean" >false</rdf:first>
          <rdf:rest rdf:resource="http://www.w3.org/1999/02/
          22-rdf-syntax-ns#nil"/>
        </rdf:rest>
        <rdf:first rdf:datatype=
        http://www.w3.org/2001/XMLSchema#boolean"
        >true</rdf:first>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasTargetOntology">
  <rdfs:range rdf:resource=
  "http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#mapping"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasSourceOntology">
  <rdfs:range rdf:resource=
  "http://www.w3.org/2001/XMLSchema#string"/>
```

```
  <rdfs:domain rdf:resource="#mapping"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasExplanationStrategy">
  <rdfs:domain rdf:resource="#Justification"/>
  <rdfs:range rdf:resource=
  "http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="hasAlignmentTool">
  <rdfs:domain rdf:resource="#mapping"/>
  <rdfs:range rdf:resource=
  "http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:FunctionalProperty rdf:ID="referTo">
  <rdf:type rdf:resource="
  http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Justification"/>
  <rdfs:range rdf:resource="#mapping"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasExplanationProfile">
  <rdfs:range rdf:resource="
  http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="
  http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Justification"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasThreshold">
  <rdf:type rdf:resource="
  http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="
  http://www.w3.org/2001/XMLSchema#float"/>
  <rdfs:domain rdf:resource="#Agent"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasHighPreference">
  <rdfs:range rdf:resource="#Value"/>
  <rdfs:domain rdf:resource="#Agent"/>
```

```
  <rdf:type rdf:resource="
  http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="isAbout">
  <rdfs:domain rdf:resource="#Argument"/>
  <rdf:type rdf:resource="
  http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#mapping"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="hasName">
  <rdf:type rdf:resource=
  "http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Agent"/>
  <rdfs:range rdf:resource="
  http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="givenBy">
  <rdf:type rdf:resource="
  http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:range rdf:resource="#Agent"/>
  <rdfs:domain rdf:resource="#Position"/>
</owl:FunctionalProperty>
<Value rdf:ID="M"/>
<Value rdf:ID="IS"/>
<owl:DataRange>
  <owl:oneOf rdf:parseType="Resource">
    <rdf:rest rdf:parseType="Resource">
      <rdf:rest rdf:parseType="Resource">
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/
  22-rdf-syntax-ns#nil"/>
  <rdf:first rdf:datatype="
  http://www.w3.org/2001/XMLSchema#string"
  >subsumption</rdf:first>
</rdf:rest>
<rdf:first rdf:datatype="
```

```
    http://www.w3.org/2001/XMLSchema#string"
    >disjoint</rdf:first>
  </rdf:rest>
  <rdf:first rdf:datatype="
  http://www.w3.org/2001/XMLSchema#string"
  >equivalence</rdf:first>
      </owl:oneOf>
    </owl:DataRange>
    <Value rdf:ID="ES"/>
    <Value rdf:ID="E"/>
    <Value rdf:ID="T"/>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)
http://protege.stanford.edu -->
```

Appendix C

An Overview on Argumentation in MAS

This appendix briefly introduces some key notions in argumentation theory and its applicability in multi-agent negotiation and communication. The content of the appendix is mainly based on [2, 100].

C.1 Argumentation Theory

The term "argumentation" has been introduced to computer science as a mean to give concepts and insights that help to understand what is going on when people argue and that yield us instruments to evaluate and help improve their practice.

With interests in field such as philosophy, linguistics, and psychology, argumentation theory provides techniques and results that have found a wide range of applications in both theoretical and practical branches of artificial intelligence and computer science and recently it has been gaining increasing interest in the multi-agent systems research community. In particular, argumentation has made solid contributions to the theory and practice of multi-agent dialogues and negotiation.

According to van Eemeren and colleagues [127], argumentation can be defined as :

> *"... a verbal and social activity of reason aimed at increasing (or decreasing) the acceptability of a controversial standpoint for the listener or reader, by putting forward a constellation of propositions intended to justify (or refute) the standpoint before a rational judge."*

Argumentation can thus be seen as the principled interaction of different, potentially conflicting arguments, in order to arrive to a consistent conclusion.

An argument, according to Walton [132]

"is a social and verbal means of trying to resolve, or at least to contend with, a conflict or difference that has arisen or exists between two (or more) parties. An argument necessarily involves a claim that is advanced by at least one of the parties. In an asymmetrical case, one party puts forward a claim, and the other party questions it. In a symmetrical case, each party has a claim that clashes with the other partys claim. The claim is very often an opinion, or claim that a view is right, but it need not be. In a negotiation argument, the claim could be to goods or to financial assets."

Argumentation gives means for allowing an agent to reconcile conflicting information within itself, for reconciling its informational state with new perceptions from the environment, and for reconciling conflicting information between multiple agents through communication. It is for these reasons that argumentation has begun to receive great interest in the multi-agent systems community mainly as a vehicle for facilitating "rational interaction" (i.e., interaction which involves the giving and receiving of reasons). This is because argumentation provides tools for designing, implementing and analysing sophisticated forms of interaction among rational agents.

In particular, argumentation has made solid contributions to the practice of multi-agent dialogues. Application domains include: legal disputes, business negotiation, labour disputes, team formation, scientific inquiry, deliberative democracy, risk analysis, scheduling, and logistics. It provides a framework for ensuring that interaction respects certain principles (e.g. consistency of each participants statements).

Moreover, argumentation theory has recently attracted increasing interest in *negotiation* - the process by which a group of agents communicate with one to try and come to a mutually acceptable agreement on some matter. With argumentation-based negotiation, agents can generate and exchange arguments to back up or justify their negotiation stance.

In the following we give a brief overview of the applicability of argumentation in agent communication and agent negotiation.

C.2 Argumentation for Agent Communication

Argumentation theory has been an inspiration for studying and formalising various aspects of agent communication. In particular, it has been a major inspiration for exploring different types of dialogues in MAS. Walton and Krabbe [134], for example, have identified a number of distinct dialogue types used in human communication, that has proved to be influential in the study of argumentation theory and its application to agent systems. All these types are characterised by their preconditions (in terms of participants beliefs) and the outcome that participants seek from the dialogue. This dialogue typology significantly affects how one should interpret, analyze or evaluate an argument. The classification is summarised below:

- *Information Seeking*: One participant seeks an answer to some question from another participant. The first participant believes that the second may have such answer.

- *Persuasion*: Two (or more) participants have conflicting beliefs. One participants seeks to change another participants belief.

- *Inquiry*: A number of participants collaborate to reach an answer to some open question, that is, a question for which no one participant knows the answer.

- *Deliberation*: A number of participants seek to decide on a course of action.

- *Negotiation*: A number of participants, with conflicting interests and a need to cooperate, attempt to reach agreement over the division of some scarce resources.

In the formal specification of different types of dialogues, two main argumentation-theoretic concepts were adopted by the MAS community: *dialogue games* and *argument schemes*. Dialogue-games are interactions between two or more players, where each player makes a move by making some utterance in a common communication language, and according to some pre-defined rules known as a "dialogue game protocol". Such moves are exchanged by players until the dialogue terminates, according to some "termination rules". Example of dialogue-games are [16, 61].

Another main inspiration from argumentation theory in MAS is the notion of an argumentation scheme. Argumentation scheme are schemes that capture stereotypical (deductive or non-deductive) patterns of reasoning found in everyday discourse. For example, Walton [133] specifies 25 argumentation schemes for common types of presumptive reasoning. The most useful aspect of argumentation schemes is that they each have an associated set of critical questions. These critical questions help identify various arguments that can be presented in relation to a claim based on the given scheme. They provide a guidance for the generation of arguments and counter-arguments in specific situations. Argumentation schemes also helps reduce the computational cost of argument generation, since only certain types of propositions need to be established. Example of argumentation schemas are the *Toulmins Argument Schema* [121] and the argument scheme for practical reasoning proposed by Atkinson et al. in [8]

.

C.3 Argumentation-Based Negotiation

Argumentation- Based Negotiation (ABN) has been proposed by Rahwan et al. [100] in an attempt to aid resolution of conflicts through argumentation and to enable agents to negotiate more efficiently. ABN involves the use of additional constructs in offers exchanged so as to make these offers more attractive to an opponent and therefore reach an agreement faster. These constructs aim to provide additional information about the agents properties, resources, or attributes or about the offer made, that can reduce

the uncertainty about their action set and preferences (without revealing their exact preferences). In so doing, this information reduces the time to find an agreement by allowing the agents to search a small number of issues they value most.

Thus, argumentation-based approaches to negotiation allows agents to exchange additional information, and to *argue* about their beliefs and other mental attitudes during the negotiation process. In the context of negotiation, Jennings et al. [68] view an argument as a piece of information that may allow an agent to (a) justify its negotiation stance; or (b) influence another agents negotiation stance. Thus, in addition to accepting or rejecting a proposal, an agent can offer a critique of it. By understanding why its counterpart cannot accept a particular deal, an agent may be in a better position to make an alternative offer that has a higher chance of being acceptable. Arguments can also allow an agent to influence another agents preferences by providing the latter with new information. This may make it possible to change the other agents preferences, or its perception of the negotiation space itself.

In [100], the authors discuss, in detail, the essential elements of argumentation-based negotiation frameworks and the agents that operate within these frameworks. They provide the background motivation for the need for such an approach by discussing the shortcomings of existing approaches to negotiation, including game-theoretic and heuristic-based approaches. They also give full details of a conceptual model of argumentation-based negotiation, which involve external elements (namely, the communication and domain languages, the negotiation protocol, and the information stores) and agent-internal elements (namely, the ability to evaluate, generate, and select proposals and arguments). The intuition behind argumentation-based negotiation is that agents may increase the likelihood and quality of an agreement by exchanging arguments which influence each others states. Additionally, it is stipulated that the exchange of arguments is sometimes essential to this process when various assumptions about agents rationality do not suffice. In making use of argumentation within a negotiation setting the designers of these models are aiding the conflict resolution process by enabling participating agents to exchange arguments in an attempt to persuade their counterparts to cooperate more with them.

This concludes our overview on argumentation theory in MAS. The interested reader is referred to [2, 100] for a full account of argumentation and ABN.

Appendix D

DINO – The Dynamic Ontology Lifecycle Scenario

D.1 Dynamic Integration of New Learned Knowledge in the DINO Framework

In this appendix we give details on the integration of changing knowledge in data-intensive domains, within the DINO Framework.

Its integration scheme (see Figure D.1) details the usage of generic lifecycle's components – mainly the *negotiation* and *versioning* – in the process of integrating learned ontologies into the collaboratively developed ones. It includes the following components: the Ontology Learning Wrapper, Ontology Collaborative Development Portal, Ontology Alignment/Negotiation Wrapper, Ontology Reasoning/Management Wrapper, Triple Sorter, Ontology Difference Wrapper and Natural Language (NL) Suggestions Generator. Each of these components, except the Ontology Alignment/Negotiation Wrapper (see Section 7.4), are briefly described in the following sections.

D.1.1 Ontology Learning Wrapper

In this phase, ontology construction relies on machine learning techniques to extract concepts and ontological relations from structured and unstructured resources (white papers, documents, publications, etc.). In the DINO framework, this component is realised using the Text2Onto [27], a framework for ontology learning from textual resources. Text2Onto interfaces indirectly within the collaborative ontology development portal based on MarcOnt Portal architecture (see Section D.1.2). Configuration of the learning algorithms is set using a special user interface in the portal. The settings is used for

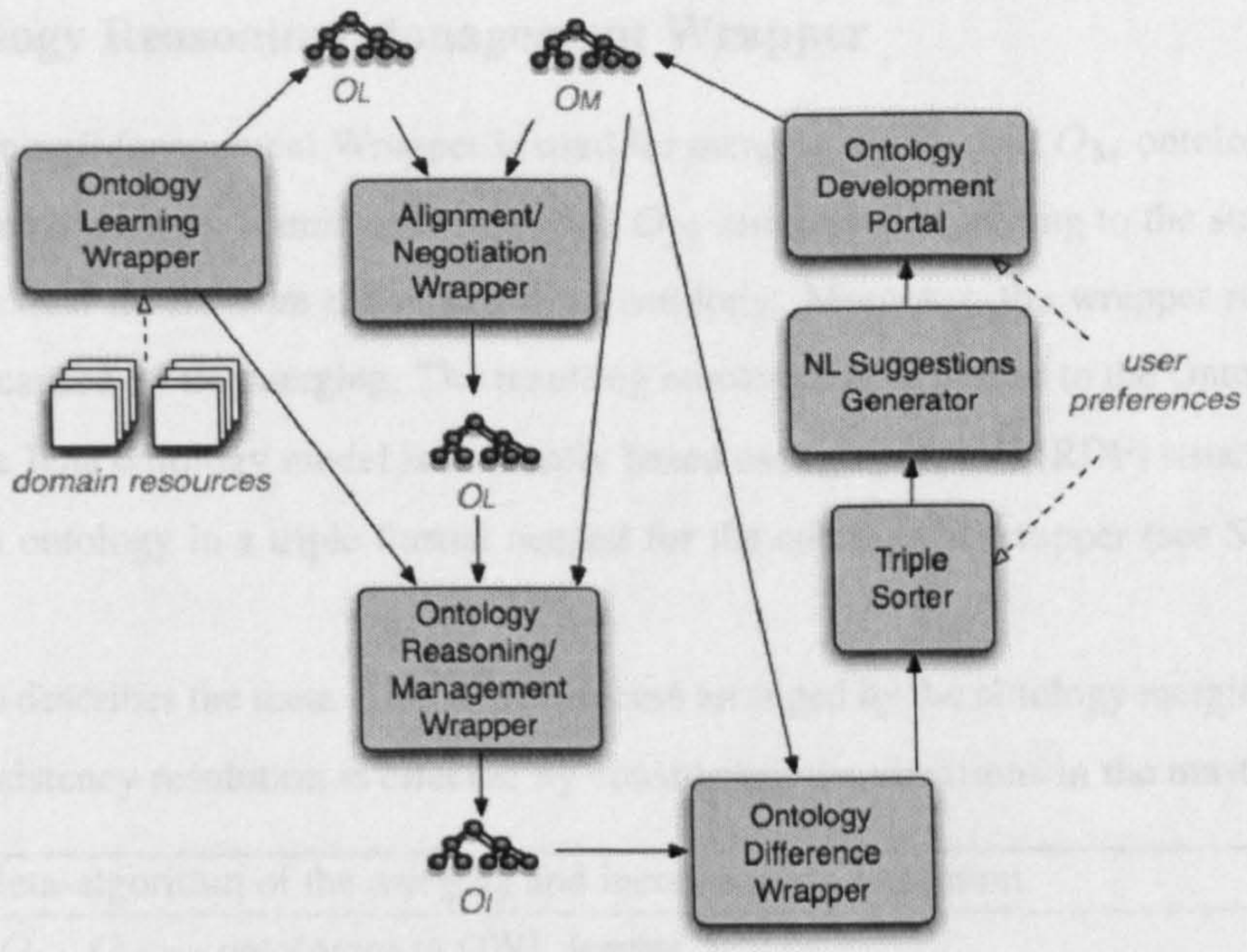

Figure D.1: Dynamic Integration Scheme.

batch processing of the new resources fed to the ontology learning component. The results of one round of ontology learning – the O_L ontology in Figure D.1– are optionally evaluated or refined using the Text2Onto confidence values and then passed to the Alignment/Negotiation Wrapper (see Section **??**).

D.1.2 Ontology Collaborative Development Portal

The whole integration as well as the DINO framework is based on the MarcOnt Portal architecture [73] for collaborative ontology development. MarcOnt Portal is a part of a broader initiative aimed mainly at utilisation of various digital library development and maintenance efforts[1].

MarcOnt Portal offers domain-independent means for efficient distributed and collaborative ontology development. It supports features like ontology editing, ontology versioning, voting on ontology changes and evaluation of these votes. The ontology versioning is supported by the SemVersion system [128]. The key elements of the DINO framework, that represent the various parts of the lifecycle, have being implemented into the portal's core, with access provided by respective new parts of portal's user and administrative interfaces.

The ontology being developed using the portal's collaborative interfaces is the master reference ontology, O_M. The ontology O_M, in Figure D.1 will be integrated with the O_L ontology resulting from the learning process. The final suggestions (see Section D.1.6) will form a base for a next version of the O_M ontology submitted after the integration.

[1] See `http://www.marcont.org` for more details on the whole MarcOnt Initiative.

D.1.3 Ontology Reasoning/Management Wrapper

Ontology Reasoning/Management Wrapper is used for merging the O_L and O_M ontologies. The wrapper uses Jena Ontology API. It merges the O_L and O_M ontologies according to the statements in O_A, preferring the lexical labels from the master O_M ontology. Moreover, the wrapper resolves possible inconsistencies caused by the merging. The resulting ontology O_I is passed to the Ontology Difference Wrapper. As the Jena ontology model is internally based on a graph/triple (RDF) structure, it allows to easily export an ontology in a triple format needed for the consequent wrapper (see Section D.1.4 for details).

Algorithm 6 describes the meta-code of the process arranged by the ontology merging and reasoning wrapper. Inconsistency resolution is effected by considering the assertions in the master O_M ontology

Algorithm 6 Meta-algorithm of the merging and inconsistency resolution

Require: O_L, O_M, O_A — ontologies in OWL format
Require: $getEq()$ — function selecting all assertions of type *owl:equivalentClass*, *owl:sameAs*, *owl:equivalentProperty*
Require: $getRM()$ — function returning wrapper combining a generic ontology manager and (incomplete OWL Full) reasoner bound to the given ontology

1: $O_{tmp} \leftarrow copy(O_L)$
2: $O_I \leftarrow copy(O_M)$
3: $R_M \leftarrow getRM(O_M)$
4: $R_{tmp} \leftarrow getRM(O_{tmp})$
5: $R_L \leftarrow getRM(O_L)$
6: $R_A \leftarrow getRM(O_A)$
7: $R_I \leftarrow getRM(O_I)$
8: $equivalencies \leftarrow \{owl:equivalentClass, owl:sameAs, owl:equivalentProperty\}$
9: $UNIFIED \leftarrow \emptyset$
10: **for** $id \in getEq(O_A)$ **do**
11: $\quad R_{tmp}.replaceLabels(id.O_L, id.O_M)$
12: $\quad UNIFIED \leftarrow UNIFIED \cup id.O_M$
13: **end for**
14: $R_{ref} \leftarrow copy(R_{tmp})$
15: **for** $eq \in R_{tmp}.getAxiomsWithLabels(UNIFIED)$ **do**
16: $\quad R_{tmp}.retractAxioms(eq)$
17: $\quad R_I.addAxioms(eq)$
18: **end for**
19: $R_A.removeAxiomsOfType(equivalencies)$
20: $R_I.addAxioms(R_{tmp}.getAllAxioms())$
21: $R_I.addAxioms(R_A.getAllAxioms())$
22: $R_I.resolveInconsistencies(R_{ref})$
23: $R_I.augmentStructure()$
24: **return** O_I

to be more relavant. Therefore the R_{ref} structure (see line 14 of the Algorithm 6), with the axioms of learned ontology, has been queried in the resolution process.

The wrapper handles the following inconsistencies:

- **sub-class** hierarchy **cycles**: these are resolved by cutting the cycle by removing an *owl:subClassOf* statement present in R_{ref};

- **disjointness-subsumption** conflicts: if classes are said to be disjoint and a sub-class relationship holds between them at the same time, the conflicting assertion indicated by R_{ref} is removed;

- **disjointness-instantiation** conflicts: if an individual is said to be an instance of classes that are disjoint, the assertion indicated by R_{ref} is removed.

When there are several removal candidate axioms involved in one inconsistency, the axioms are sorted according to the frequency of the respective subject/object labels (indicated by the underlying triple-based representation of the ontology) in the R_{ref} reference structure. The axioms with least overall frequency are removed until the inconsistency is resolved. The conflicting assertions which originate from the O_M master ontology are kept for the users.

The function *augmentStructure*() attempts to complete the structure of learned axioms using the more precise and complex knowledge in the O_M master ontology. Currently, augmentation of *owl:subClassOf* and instantiation relations using *rdfs:domain* and *rdfs:range* assertions in property definitions from O_M ontology is taken into account. Note that it is possible to include even the "equal" labels from the learned ontology, by removing the renaming and subtractions in lines 10-16 and 19 on the algorithm 6 and include the respective equality statements from O_A into O_I, together with respective axioms from O_L. The decision depends on users – whether they want to prefer the labels from master ontology or not (e.g. when looking for possible unknown synonyms of important terms from O_M in domain resources; this could be useful for example in the medicine domain when performing the task of identifying different names for the same drugs and/or proteins).

D.1.4 Ontology Difference Wrapper

An extension of a master ontology O_M by elements contained in the merged and refined ontology O_I corresponds to the differences between them. The differences are discovered by means of the SemVersion library [128], which is interfaced within this wrapper. In particular, the possible extensions are equal to the additions O_I brings into O_M. The extensions are computed from the triple-based representation of O_I and O_M ontologies, and then passed to the triple sorter.

D.1.5 Triple Sorter

The addition triples passed to this component form a base to the eventual suggestions for the domain experts. However, the number of additions can often be quite large, so an ordering that takes a relevance measure of possible suggestions into account is also needed. Thus, the suggestions with low relevance level when presenting the final set to the users are eliminated.

This is achieved by the implementation of a method based on string subsumption and Levenshtein distance [76]. These two measures are used within relevance computation by comparing the predicate, subject and object lexical labels of a triple to two sets (S_p, S_n) of words, provided by users. The S_p and S_n sets contain preferred and unwanted words respectively, concerning the lexical level of optimal extensions. The general structure of the sorting function is given in Algorithm 7.

Algorithm 7 Meta-algorithm of relevance-based triple sorting

Require: $TRIPLES$ — list of triples
Require: $PREF = \{S_p, S_n\}$ — user preferences

1: $HASH = \{\}$
2: **for** $T \in TRIPLES$ **do**
3: $\quad HASH[getScore(T, S_p, S_n)] \leftarrow T$
4: **end for**
5: **return** $sort(HASH)$

The $getScore()$ function is given by the formula:

$$getScore(T, S_p, S_n) = rel(T, S_p) - rel(T, S_n),$$

where $rel(T, S)$ is a function measuring the relevance of the triple T with respect to the words in the set S. The higher the value, the more relevant the triple is. The function[2] naturally measures the "closeness" of the P, S, O labels to the set of terms in S_w. The value of 1 is achieved when the label is a direct substring of or equal to any word in S_w or vice versa. When the Levenshtein distance between the label and a word in S_w is lower than or equal to the defined threshold t, the relevance decreases from 1 by a value proportional to the fraction of the distance and t. If this is not the case (i.e. the label's distance is greater than t for each word in S_W), a similar principle is applied for possible word-parts of the label and the relevance is further proportionally decreased (the minimal possible value being 0).

D.1.6 Natural Language Suggestions Generator

The DINO framework is supposed to be used primarily by users who are not experts in ontology engineering. Although the MarcOnt Portal [73] already offers a very simple ontology editing interface, further development has been done for helping the user in ontology augmentation by the learned knowledge. Therefore the suggestions are produced in the form of very simple natural language statements. These are obtained directly from the sorted triples passed to this component, using a minor modification of the generation process described in CLIE [118]. The suggestions are still bound to the underlying triples, therefore the user can easily add the respective OWL axioms into the new version of the O_M master ontology without dealing with the OWL syntax itself.

[2]For more information about the relevance function and complexity analysis (which is in feasible class of $O(m \log(m))$ with respect to the number of triples), the reader is referred to [92].

The description of the system found in this appendix partly occurs in [93, 95, 94] with is co-authored by Vit Novacek.

Bibliography

[1] FIPA Communicative Act Library Specification. Document status: standard. Foundation for Intelligent Physical Agents. http://www.cselt.stet.it/fipa, 2002.

[2] *Special issue on Argumentation in Multi-Agent Systems. Journal on Autonomous Agents and Multi-Agent Systems*, volume 11, pages 115–206. Springer Netherlands, 2005.

[3] K. Aberer and et al. Emergent semantics principles and issues. In *Proceedings of the International Conference of Database Systems for Advanced Applications, (DASFAA 04)*, pages 25–38, 2004.

[4] M. Afsharchi, B. H. Far, and J. Denzinger. Ontology-guided learning to improve communication between groups of agents. In *Proceedings of the international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pages 923–930. ACM, New York, NY, USA, 2006.

[5] L. Amgoud, S. Belabbes, and H. Prade. Towards a formal framework for the search of a consensus between autonomous agents. In *Proceedings of the fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 4049 of *Lecture Notes in Computer Science*, pages 264–278. Springer Berlin / Heidelberg, 2005.

[6] L. Amgoud and C. Cayrol. On the Acceptability of Arguments in Preference-Based Argumentation. In I. G. Cooper and S. Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 1–7. Morgan Kaufmann, 1998.

[7] A. Artikis, M. Sergot, and J. Pitt. An executable specification of an argumentation protocol. In *Proceedings of the 9th International Conference on Artificial intelligence and Law*, pages 1–11, 2003.

[8] K. Atkinson, T. Bench-Capon, and P. McBurney. Computational representation of practical argument. *Knowledge, Rationality and Action, a special section of Synthese*, 152(2):157–206, 2006.

[9] F. Baader and W. Nutt. Basic Description Logics. *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95, 1993.

[10] K. Baclawski and T. Niu. *Ontologies for Bioinformatics*. The MIT Press, October 2005.

[11] S. C. Bailin and W. Truszkowski. Ontology Negotiation: How Agents Can Really Get to Know Each Other. In *Proceedings of the WRAC 2002*, pages 320–334, 2002.

[12] J. R. Bayardo and et al. Infosleuth: Agent-based semantic integration of information in open and dynamic environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 195–206. ACM, New York, NY, USA, 1997.

[13] R. Belecheanua and et al. Commercial Applications of Agents: Lessons, Experiences and Challenges. In *Industrial Track, Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2006)*, pages 1549–1555, May 2006.

[14] T. J. M. Bench-Capon. Agreeing to Differ: Modelling Persuasive Dialogue Between Parties Without a Consensus About Values. *Informal Logic*, 22:231–245, 2002.

[15] T. J. M. Bench-Capon. Persuasion in Practical Argument using Value-based Argumentation Frameworks. *In Journal of Logic and Computation*, 13(3):429–448, 2003.

[16] J. Bentahar, B. Moulin, and B. Chaib-draa. A persuasion dialogue game based on commitments and arguments. In *First International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2004)*, pages 148–164, July 2004.

[17] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284:35–43, May 2001.

[18] P. Besana and D. Robertson. Contexts in dynamic ontology mapping. In *Proceeding of the AAAI 05 Workshop on Context and Ontology: Theory, Practice and Applications.*, 2005.

[19] R. Beun, R. M. van Eijk, and H. Prust. Ontological Feedback in Multiagent Systems. In *Proceedings of the 3rd international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 110–117. ACM Press, 2004.

[20] P. Borst. *Construction of Engineering ontologies for knowledge sharing and reuse.* PhD thesis, Centre for Telematica and Information Technology, University of Twente, 1997.

[21] C. Brewster, F. Ciravegna, and Y. Wilks. Background and Foreground Knowledge in Dynamic Ontology Construction: Viewing Text as Knowledge Maintenance. In *Proceedings of the Semantic Web Workshop, SIGIR, Toronto, Canada*, 2003.

[22] J. Burge and D. Brown. Rationale Support for Maintenance of Large Scale Systems. In *Proceedings of the Workshop on Evolution of Large-Scale Industrial Software Applications (ELISA), ICSM '03, Amsterdam, NL*, 2003.

[23] T. Bylander and B. Chandrasekaran. Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies*, 26(2):231–243, 1987.

[24] A. Campbell and S. Shapiro. Ontological mediation: An overview. In *Proceedings of the IJCAI Workshop on Basic Ontological Issues for Knowledge Sharing, Montreal, QC, Canada*, 1995.

[25] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, and J. P. Rice. Open Knowledge Base Connectivity 2.0. Technical report, KSL-98-06, Knowledge Systems Laboratory, Stanford, 1997.

[26] Chesñevar and et al. Towards an argument interchange format. *The Knowledge Engineering Review*, 21(4):293–316, 2006.

[27] P. Cimiano and J. Völker. Text2Onto - a Framework for Ontology Learning and Data-Driven Change Discovery. In Springer-Verlag, editor, *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 3513, pages 227–238, 2005.

[28] R. Conte, C. Castelfranchi, and F. Dignum. Autonomous Norm-acceptance. In M. J. Muller and A. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, pages 319–334, 1999.

[29] O. Corcho. *A Layered Declarative Approach to Ontology Translation with Knowledge Preservation*, volume 116. IOS Press, 2005.

[30] O. Corcho, A. Gomez-Perez, A. Leger, C. Rey, and F. Toumani. An ontology-based mediation architecture for e-commerce applications. In *Proceedings of the International IIS at the IIPWM 03 Conference*, pages 477–48, 2003.

[31] K. Decker, K. Sycara, and M. Williamson. Matchmaking and Brokering. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, 1996.

[32] K. Dellschaft and S. Staab. On How to Perform a Gold Standard Based Evaluation of Ontology Learning. In *Proceeding on the International Semantic Web Conference*, volume 4273, pages 228–241. Springer Berlin / Heidelberg, 2006.

[33] I. Dickinson and M. Wooldridge. Towards Practical Reasoning Agents for the Semantic Web. In *Proceedings of the Second International joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 827 – 834, 2003.

[34] S. Doutre, T. Bench-Capon, and P. E. Dunne. Determining Preferences through Argumentation. In *Proceedings of AI*IA'05*, volume 3673 of *Lecture Notes in Computer Science*, pages 98–109, 2005.

[35] P. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-person Games. *In Artificial Intelligence*, 77(2):321–358, 1995.

[36] P. Dunne and T. J. M. Bench-Capon. Two Party Immediate Response Disputes: Properties and Efficiency. *Artificial Intelligence*, 149:221–250, 2003.

[37] P. E. Dunne and T. Bench-Capon. Complexity in Value-based Argument Systems. In *Proceeding of the European Conference on Logics in Artificial Intelligence (JELIA 2004)*, pages 360—371, 2004.

[38] E. H. Durfee and V. R. Lesser. Negotiating Task Decomposition and Allocation Using Partial Global Planning. *Distributed Artificial Intelligence*, 2, 1989.

[39] M. Ehrig and S. Staab. QOM - Quick Ontology Mapping. In *Proceedings of the Third International Semantic Web Conference*, pages 683–697, 2004.

[40] M. Ehrig and Y. Sure. FOAM -Framework for Ontology Alignment and Mapping - Results of the Ontology Alignment Initiative. In *Proceedings of the Workshop on Integrating Ontologies*, 2005.

[41] D. Engmann and S. Massmann. Instance Matching with COMA++. In *Proceeding on the Workshop of Model Management und Metadaten-Verwaltung*, pages 28–37, 2007.

[42] J. Euzenat. Alignment infrastructure for ontology mediation and other applications. In M. In Hepp, A. Polleres, F. van Harmelen, and M. Genesereth, editors, *In Proceedings of the First International Workshop on Mediation in semantic web services*, pages 81–95, 2005.

[43] J. Euzenat and et al. D2.2.3: State of the art on ontology alignment. Knowedge Web Deliverable. Technical report, Institut National de Recherche en Informatique et en Automatique (INRIA), 2004.

[44] J. Euzenat, L. Laera, V. Tamma, and A. Viollet. D2.3.7: Negotiation/argumentation techniques among agents complying to different ontologies. knowledge web deliverable. Technical report, Institut National de Recherche en Informatique et en Automatique (INRIA), 200o.

[45] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, 2007.

[46] J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In *Proceedings of the workshop on Semantic Integration at ISWC*, pages 33–38, 2003.

[47] D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.

[48] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an agent communication language. In A. Press, editor, *Proceedings of the Third International Conference on Information and Knowledge Management*, pages 456–463, 1994.

[49] S. Franklin and A. Graesser. Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. In *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, volume 1193 of *Lecture Notes in Computer Science*, pages 21–35. Springer Berlin / Heidelberg, 1996.

[50] M. R. Genesereth and S. P. Katchpe. Software agents. *Communications of the ACM*, 37(7):48–53, 1997.

[51] F. Giunchiglia. Contextual reasoning. Technical report, IRST, Istituto per la Ricerca Scientifica e Tecnologica, 1992.

[52] G. A. Grimnes, S. Chalmers, P. Edwards, and A. Preece. Granitenights - a multi-agent visit scheduler utilising semantic web technology. In *Proceeding of the Seventh International Workshop on Cooperative Information Agents*, pages 137–151, 2003.

[53] T. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer Academic Publishers, 1993.

[54] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[55] N. Guarino. Understanding, Building, and Using Ontologies. *International Journal of Human-Computer Studies*, 46:293–310, 1997.

[56] N. Guarino. Formal ontologies and information systems. In I. N. Guarino, editor, *Proceedings of the International Conference on Formal Ontology in Information Systems*, 1998.

[57] V. Haarslev and R. Möller. RACER System Description. In Springer-Verlag, editor, *Proceeding of International Joint Conference on Automated Reasoning, IJCAR'2001*, pages 701–705, 2001.

[58] C. L. Hamblin. *Fallacies*. Methuen, 1970.

[59] P. Hayes. Resource Description Framework (RDF) semantics. W3C Recommendation, 2004.

[60] C. Hewitt and P. de Jong. Analyzing the Roles of Descriptions and Actions in Open Systems. In *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pages 162–167, 1983.

[61] J. Hulstijn. *Dialogue models for enquiry and transaction.* PhD thesis, Universiteit Twente, Enschede, The Netherlands, 2001.

[62] C. K. I. Foster and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *The International Journal of High Performance Computing Applications*, 15(3):200–222, 2001.

[63] R. Ichise, H. Takeda, and S. Honiden. Rule Induction for Concept Hierarchy Alignment. In *Proceedings of the Workshop on Ontology Learning at the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 26—29, 2001.

[64] R. V. J. Heflin and J. Dale. Requirements for a web ontology language. In *Technical report, World Wide Web Consortium (W3C)*, 2002.

[65] N. J. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[66] N. R. Jennings. Specification and Implementation of a Belief-Desire-Joint-Intention Architecture for Collaborative Problem Solving. *International Journal of Intelligent and Cooperative Information Systems*, 2(3):289–318, 1993.

[67] N. R. Jennings. On Agent-Based Software Engineering. *Artificial Intelligence Journal*, 117(2):277–296, 2000.

[68] N. R. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Research and Development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[69] N. Jian, W. Hu, G. Cheng, and Y. Qu. Falcon-AO: Aligning Ontologies with Falcon. In *Proceedings of the of K-CAP Workshop on Integrating Ontologies*, pages 87–93, 2005.

[70] M. Kifer, G. Lausen, and J. We. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of ACM*, 42:741–843, 1995.

[71] M. Klein. Combining and Relating Ontologies: an Analysis of Problems and Solutions. In A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, and M. Uschol, editors, *Proceedings of the IJCAI'01 workshop on Ontologies and Information Sharing*, pages 53–62, 2001.

[72] P. Krause, S. Ambler, M. Elvang-Goeransson, and J. Fox. A Logic of Argumentation for Reasoning under Uncertainty. *Computational Intelligence*, 11(1):113–131, 1995.

[73] S. Kruk, J. Breslin, and S. Decker. MarcOnt initiative. Technical report, Lion Deliverable 3.01, DERI, Galway, 2005.

[74] Y. Labrou, T. Finin, and Y. Peng. Agent Communication Languages: the current landscape. *IEEE Intelligent Systems*, 2(14):45–52, 1999.

[75] H.-J. Lebbink. *Dialogue and Decision Games for Information Exchanging Agents*. PhD thesis, Universiteit Utrecht, Utrecht, The Netherlands, 2006.

[76] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics Control Theory*, 10:707–710, 1966.

[77] G. Levi, A. Vagliengo, and A. Goy. Talea: An Ontology-based Framework for e-Business Applications Development. In *Proceeding of the Second Italian Semantic Web Workshop on Semantic Web Applications and Perspectives (SWAP 2005), Trento, Italy*, pages 489–498. Springer London, 2005.

[78] M. Lopez. Overview of methodologies for building ontologies. In *Proceedings of the IJCAI Workshop on Ontologies and Problem-Solving Methods*, 1999.

[79] M. Luck, P. McBurney, and C. Preist. Manifesto for Agent Technology: Towards Next Generation Computing. *Autonomous Agents and Multi-Agent Systems*, 9(3):203–252, November 2004.

[80] J. MacKenzie. Question-begging in non-cumulative systems. *Journal of philosophical logic*, 8:117–133, 1979.

[81] A. Maedche and S. Staab. Measuring Similarity between Ontologies. In *Proceeding of the International Conference on Knowledge Engineering and Management (EKAW'02)*, volume 2473, pages 251 – 263, 2002.

[82] A. Malucelli, D. Palzer, and E. Oliveira. Ontology-based Services to help solving the heterogeneity problem in e-commerce negotiations. *Journal of Electronic Commerce Research and Applications - Special Issue Electronic data engineering: the next frontier in e-commerce*, 5(3):29–43, 2006.

[83] D. Martin, H. Oohama, D. Moran, and A. Cheyer. Information Brokering in an Agent Architecture. In *Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*. The Practical Application Company Ltd, 1997.

[84] P. McBurney and S. Parsons. Locutions for argumentation in agent interaction protocols. In *Proceedings of International Workshop on Agent Communication, New-York (NY US)*, pages 209–225, 2004.

[85] D. L. McGuinness. Ontologies come of age. In *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press*, 2002.

[86] G. Miller. WordNet: a Lexical Database for English. *Communications of the ACM*, 38(11), 1995.

[87] M. Morge, Y. Secq, and J.-C. Routier. Formal framework for inter-agents dialogue to reach an agreement about a representation. In *Proceedings of the Workshop on Computational Models of Natural Argument (CMNA 2006)*, 2006.

[88] R. Neches and et al. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, 1991.

[89] S. H. Nielsen and S. Parsons. A generalization of Dung's abstract framework for argumentation. In *Proceedings of the Third International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2006)*, 2006.

[90] H. P. Nii. Blackboard systems. In A. Bar, P. Cohen, and E. Feigenbaum, editors, *The Handbook of Artificial Intelligence IV*, pages 1–82, 1989.

[91] M. Nodine and A. Unruh. Facilitating Open Communication in Agent Systems: The Infosleuth Infrastructure. In *Proceedings of Fourth International Workshop on Agent Theories, Architectures and Languages (ATAL'97)*, volume 1365, pages 281–295. Springer-Verlag, 1997.

[92] N. Novacek, M. Dabrowski, S. Ryszard Kruk, and S. Handschuh. Extending community ontology using automatically generated suggestions. In *Proceedings of FLAIRS 2007, AAAI Press Key West, Florida, USA*, 2007.

[93] V. Novacek, L. Laera, and H. Handschuh. Semi-automatic integration of learned ontologies into a collaborative framework. In *Proceedings of the Internation Workshop on Ontology Dynamic at ESWC 2007, Innsbruck, Austria*, 2007.

[94] V. Novacek, L. Laera, and S. Handschuh. Aiding the data integration in medicinal settings by means of semantic technologies. In *Proceedings of the International Workshop on Making Semantics Work for Business at ESTC 2007, Vienna, Austria*, 2007.

[95] V. Novacek, L. Laera, and S. Handschuh. Dynamic integration of medical ontologies in large scale. In *Proceedings of the WWW First International Workshop on Health Care and Life Sciences Data Integration for the Semantic Web. Banff, Alberta, Canada*, 2007.

[96] P. Patel-Schneider, P. Hayes, and I. Horrocks. Web ontology language OWL. Abstract Syntax and Semantics. W3C Recommendation, 2004.

[97] P. Patel-Schneider, I. Horrocks, and F. van Harmelen. Reviewing the design of DAML-OIL: An Ontology Language for the Semantic Web. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'02)*, pages 792–797, 2002.

[98] J. G. Quenum and at al. Dynamic protocol selection in open and heterogeneous systems. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2006 Main Conference Proceedings) (IAT'06)*, pages 333–341, 2006.

[99] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[100] I. Rahwan, S. D. Ramchurn, N. J. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *The Knowledge Engineering Review*, 18:342–375, 2003.

[101] B. Ramesh and V. Dhar. Supporting system development by capturing deliberations during requirements engineering. *IEEE Transaction on Software Engineering*, 18:489–510, 1992.

[102] A. S. Rao and M. P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, 91.

[103] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing special issue on Peer-to-Peer Networking*, 6(1):50–57, 2002.

[104] T. Sakai. On the reliability of information retrieval metrics based on graded relevance. *Information Processing and Management*, 43(2):531–548, 2007.

[105] J. Sampson. Ontology alignment in agent systems: Current and future challenges. In *International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, pages 168–173, 2005.

[106] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 2001.

[107] J. R. Searle. *Speech Acts. An Essay in the Philosophy of Language.* Cambridge, 1969.

[108] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.

[109] P. Shvaiko, F. Giunchiglia, P. Pinheiro da Silva, and D. L. McGuinness. Web explanations for semantic heterogeneity discovery. In *Proceeding of the European Conference on Semantic Web*, pages 303–317, 2005.

[110] N. Silva, P. Maio, and J. Rocha. An approach to ontology mapping negotiation. In *Proceedings of the Workshop on Integrating Ontologies*, 2005.

[111] E. Simperl, C. Tempich, H. Pinto, and M. Luczak. Argumentation-based ontology engineering. *Special Issue on Argumentation Technology. Journal of IEEE Intelligent Systems. To appear*, 2007.

[112] H. Sofia Pinto, S. Staab, and C. Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *Proceeding of the European Conference on Artificial Intelligence*, pages 393–397, 2004.

[113] J. F. Sowa. Ontology, Metadata, and Semiotics. In B. Ganter and G. W. Mineau, editors, *ICCS'2000*, pages 55–81, 2000.

[114] R. Studer, V. Benjamins, and D. Fensel. Knowledge Engineering, Principles and Methods. *Data and Knowledge Engineering*, 25(1–2):161–197, 1998.

[115] Y. Sure. A Tool-supported Methodology for Ontology-based Knowledge Management. *Ontology and Modeling of Real Estate Transactions in European Juristictions*, 372:155–166, 2002.

[116] Y. Sure and C. Tempich. SEKT Deliverable D7.1.1.c. State of the Art in Ontology Engineering Methodologies. Technical report, Institute AIFB, University of Karlsruh, 2004.

[117] K. P. Sycara. MultiAgent Systems. *AI Magazine*, 19(2):79–92, 1998.

[118] V. Tablan, T. Polajnar, H. Cunningham, and K. Bontcheva. User–friendly ontology authoring using a controlled language. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 06)*, 2006.

[119] V. Tamma, I. Blacoe, B. Lithgow Smith, and M. Wooldridge. Introducing Autonomic Behaviour in Semantic Web Agents. In *Proceedings of the Fourth International Semantic Web Conference (ISWC 2005), Galway, Ireland, November*, volume 3729 of *Lecture Notes in Computer Science*, pages 653–667, 2005.

[120] J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang. Using bayesian decision for ontology mapping. *Journal of Web Semantic*, 4(4):243–262, 2006.

[121] S. Toulmin. *The Uses of Argument*. Cambridge University Press, Cambridge, UK, 1958.

[122] M. Uschold. Where is the Semantics in the Semantic Web? In *Proceedings of Workshop on Ontologies in Multi Agent Systems at the 5th Conference on Autonomous Agents*, May 2001.

[123] M. Uschold and M. Gruninger. Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[124] M. Uschold and R. Jasper. A Framework for Understanding and Classifying Ontology Applications. In *Proceedings of the IJCAI99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden*, volume 18. SRDG Publications, 1999.

[125] J. van Diggelen. *Achieving Semantic Interoperability in Multi-agent Systems: A Dialogue-based Approach*. PhD thesis, Institute of Information and Computing sciences at the University of Utrecht in the Netherlands, 2007.

[126] J. van Diggelen, R. Beun, F. Dignum, R. van Eijk, and J.-J. Meyer. A Decentralized Approach for establishing a Shared Communication Vocabulary. In *Proceedings of the AAMAS Internation Workshop on Agent Mediated Knowledge Management (AMKN)*, 2005.

[127] F. H. van Eemeren, R. F. Grootendorst, and F. S. Henkemans. *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Applications*. Lawrence Erlbaum Associates: Hillsdale NJ, USA, 1996.

[128] M. Völkel and T. Groza. SemVersion: RDF-based Ontology Versioning System. In *Proceedings of the IADIS International Conference WWW/Internet (ICWI 2006)*, volume 1, pages 195–202, 2006.

[129] P. Visser, D. Jones, T. Bench-Capon, and M. Shave. Assessing Heterogeneity by Classifying Ontology Mismatches. In N. Guarino, editor, *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS'98)*, pages 148–162, 1998.

[130] P. Visser and V. Tamma. An Experiment with Ontology-Based Agent Clustering. In *Proceedings on the IJCAI-99 Workshop on Ontologies and Problem-SolvingMethods: Lessons Learned and Future Trends, Stockholm, Sweden*, 1999.

[131] H. Wache and et al. Ontology-Based Integration of Information - A Survey of Existing Approaches. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop: Ontologies and Information Sharing, Seattle, USA*, 2001.

[132] D. N. Walton. What is reasoning? what is argument? *The Journal of Philosophy*, 87:399–419, 1990.

[133] D. N. Walton. *Argumentation Schemes for Presumptive Reasoning*. Erlbaum, Mahwah NJ, USA, 1996.

[134] D. N. Walton and E. C. W. Krabbe. *Commitment in Dialogue: Basic Conceptions of Interpersonal Reasoning*. State University of New York Press, Albany, NY, 1995.

[135] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *Computer Magazine of the Computer Group New of the IEEE Group Society*, 25(3):38 – 49, 1992.

[136] G. Wiederhold. Intelligent Integration of Information. In *Proceedings of ACM SIGMOD Conference of Management of Data*, pages 434–437. ACM New York, NY, USA, 1993.

[137] G. Wiederhold. An algebra for ontology composition. In *Proceedings of 1994 Monterey Workshop on Formal Methods, U.S. Naval Postgraduate School, Monterey CA.*, pages 56–61, 1994.

[138] A. B. Williams and Z. Ren. Agents teaching agents to share meaning. In *Proceedings of the fifth International Conference on Autonomous agents*, pages 465–472. ACM New York, NY, USA, 2001.

[139] M. Wooldridge. Verifying that agents implement a communication language. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), Orlando, FL, United States*, pages 52 – 57, 1999.

[140] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, February 2002.

[141] M. Wooldridge and N. R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[142] Z. Yand, D. Zhang, and C. Ye. Evaluation metrics for ontology complexity and evolution analysis. In *Proceeding of the IEEE International Conference on e-Business Engineering (ICEBE'06)*, pages 162–170, 2006.

[143] F. Yergeau and et. al. Extensible Markup Language (XML) 1.0. W3C Recommendation, 2004.

[144] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Organizational Abstractions for the Analysis and Design of Multi-agent Systems. In *Proceeding on the First International Workshop on Agent-Oriented Software Engineering (AOSE), Limerick, Ireland*, volume 1957 of *Lecture Notes in Computer Science*, pages 235–251. Springer Berlin / Heidelberg, 2000.

[145] A. Zimmermann, M. Krötzsch, J. Euzenat, and P. Hitzler. Formalizing ontology alignment and its operations with category theory. In B. Bennett and C. Fellbaum, editors, *Proceedings of the Fourth International Conference on Formal Ontology in Information Systems (FOIS 2006)*, volume 150, pages 277–288. IOS Press, 2006.