Ontology Modularization: Principles and Practice

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

October 2009

Abstract

Technological advances have provided us with the capability to build large intelligent systems capable of using knowledge, which relies on being able to represent the knowledge in a way that machines can process and interpret. This is achieved by using ontologies; that is logical theories that capture the knowledge of a domain. It is widely accepted that ontology development is a non-trivial task and can be expedited through the reuse of existing ontologies. However, it is likely that the developer would only require a part of the original ontology; obtaining this part is the purpose of ontology modularization.

In this thesis a graph traversal based technique for performing ontology module extraction is presented. We present an extensive evaluation of the various ontology modularization techniques in the literature; including a proposal for an entropy inspired measure. A task-based evaluation is included, which demonstrates that traversal based ontology module extraction techniques have comparable performance to the logical based techniques.

Agents, autonomous software components, use ontologies in complex systems; with each agent having its own, possibly different, ontology. In such systems agents need to communicate and successful communication relies on the agents ability to reach an agreement on the terms they will use to communicate. Ontology modularization allows the agents to agree on only those terms relevant to the purpose of the communication. Thus, this thesis presents a novel application of ontology modularization as a space reduction mechanism for the dynamic selection of ontology alignments in multi-agent systems. The evaluation of this novel application shows that ontology modularization can reduce the search space without adversely affecting the quality of the agreed ontology alignment.

To my family

Contents

List of Figures

List of Tables

Acknowledgements

Whilst one name appears on the front of this work it would not have been possible without the help of so many people.

My deepest gratitude must go to Dr. Valentina Tamma for giving me the chance, opportunity and freedom to explore my ideas, even if some of them were a little crazy! Your help, guidance and friendship has proved invaluable.

I also thoroughly thank all past and present members of the Semantic Web Lab, but I especially thank: Ian for knowing the answers to all my questions; Lori for always putting a smile on my face; Luigi for providing me with some solid advice and many memorable nights out; and last, but not least, Ignazio for always being dependable, reliable and honest.

My thanks, also, go to thank Prof. Frank Wolter and Dr. Terry Payne for critiquing my work. Your feedback has made me a better researcher and this thesis better.

I must thank the EPSRC for funding my studentship and the Department for awarding it to me; it truely wouldn't have been possible without their money.

Finally, I am indebted to my whole family who are always their for me and keeping my feet firmly on the ground. I must particulalry thank my parents for always encouraging me to grow and learn, both emotionally and intellectually. It is also essential that I thank my grandparents for showing me that strong principles and hard work take you far.

A special thank you must go to Elisa; your support, understanding and love has been immense.

Part I

Background and Context

Chapter 1

Introduction

'For we are all ants on beaches of knowledge.' - Edward A. Feigenbaum

Summary The aim of this thesis is to outline the principles underlying the the process of ontology modularization; the process of identifying subsets of an ontology. From this it is possible to consider how to use ontology modularization in practice. This thesis applies ontology modularization to the problem of ontology reuse and to the novel problem of using ontology modularization as a space reduction mechanism for the dynamic selection of ontological alignments in multi-agent systems.

This chapter presents the background and motivation for this thesis, and it provides the research aims and contributions before moving on to give an outline of the thesis structure.

1.1 Background & Motivation

The last 60 years have seen a rapid growth in the use and exploitation of computer systems, in academia, industry and through the growth of the Internet wider society too. This has changed the way people perceive and access knowledge. How often, when one needs a question answering, is the Internet the first port of call?

This development in computer technology coincided with the birth of Artificial Intelligence. Ever since Alan Turing proposed his test [136], the 'Turing Test', people have attempted to build intelligent machines or, at least, machines that appear intelligent. The Turing Test requires a machine to fool a human observer into believing that it is human, based not on appearance but on its ability in a conversational task.

This has motivated researchers to consider many ways of giving a machine intelligence. In this direction Artificial Intelligence has two important results relevant to this thesis: the physical symbol system hypothesis (PSSH) [99] and the knowledge principle (KP) [86]. The physical symbol system hypothesis states that "A physical symbol system has the necessary and sufficient means for general intelligent action"; this implies that intelligence is some kind of symbolic manipulation and as such a machine can be

intelligent because a symbol system is sufficient for intelligence. The knowledge principle states that "if a program is to perform a complex task well then it must know about the world in which it operates"; the implication being that knowledge is needed to be intelligent.

Searle's Chinese Room [114] is a famous retort to the physical symbol system hypothesis. The Chinese Room is a thought experiment where there is a room that receives a question as input, in Chinese, a rule book for answering all the questions, and a human that can match the input to the rules and generate an answer, in Chinese. Searle argues that because the human in the room does not understand Chinese then the Room is only a simulation of intelligence. Levesque [88] presents a riposte to the Chinese Room by considering, a simpler room, the Summation Room. This is similar to the Chinese Room but involves the task of adding up twenty, ten-digit numbers. Levesque shows that the rule book required even for this simple task is not feasible (in terms of the number of entries required in the rule book) and, thus, undermines Searle's argument.

Why is all this relevant today? In today's society large computer systems are the norm and they are expected to perform ever more complex tasks, and with this the expectation that computer systems need to be more 'intellignet'. Indeed, interacting systems are now the norm in the everyday computing world, even trivial systems contain sub-systems that need to interact [146]. An obvious example being the Internet, where countless computers interact to perform a myriad of tasks; but even your television interacts with a remote control!

We have all experienced the Internet and its vast swathe of content. However, "the Internet is one big ocean of unedited data, without any pretense of completeness" [126]. It is hard to find what you need, for example a query for "jaguar" will give results about both cars and cats. So, with the Internet there is the need to move toward a more 'intelligent' Web; a web that 'knows' what things are. This requires a shift from data to knowledge. For example, the shift from "red light" to "red light" from a "traffic light" allows us to discern that we should probably stop. We need to 'know' what a page is about; in other words we should follow the knowledge principle; if we want to be able to separate the "jaguar" cars from the "jaguar" cats then the program running the query needs to be able to make the distinction.

In Computer Science this shift from data to knowledge came with a new perspective, the *knowledge level* [98]. The knowledge level provides us with a level of abstraction above implementation concerns; allowing us to specify what something knows without concerning ourselves with the mechanics.

One way to encode this knowledge is to use ontologies (see Section 2.1), these provide us with a logical theory which gives an explicit, partial account of a conceptualization [64]. They provide a vocabulary of terms and express relations that hold between

them. Ontologies have been successfully employed in order to solve problems, such as interoperability and heterogeneity, deriving from the management of shared, distributed knowledge, and the efficient integration of information across applications [38].

Much of this success depends on the ability to share and reuse existing ontologies [54]. However, as some ontologies[1] are sizable, such as SNOMED CT[2] and the NCI Thesaurus[3], then there is a motivation to reduce them to more manageable chunks. For example, if one is building an ontology and wishes to reuse only part of an existing ontology. Even more so when considering that ontology construction is deemed to be a time consuming and labour intensive task [33]. Ontology modularization allows one to identify subsets of an ontology, with these subsets being termed ontology modules.

Ontology modularization can be used for efficiency gains in a diverse range of tasks, such as reasoning, query answering, reuse (see Chapter 5), etc. The common theme for all tasks being that why should one use something which is larger than necessary for the task. This thesis presents a technique for ontology module extraction (see Section 3.4.1) based on graph traversal.

Technological advancement has brought with it ever more possibilities to exploit the knowledge level perspective. The latest effort being the Semantic Web [134] which aims to add a layer of meaning to the World Wide Web and is gaining mainstream attention [4, 121, 144].

The Semantic Web is one example of an open, distributed environment as few constraints are placed on the participants (agents). The participants can enter and leave the environment at will and have their own separate internal models of the world. Some constraints are, however, required, for example the adoption of a standard ontology language to encode knowledge in order to prevent knowledge systems from being "isolated monoliths" [57]; without this the agents would be unable to exchange knowledge.

However, even with the same ontology language agents are likely to have mismatches that need to be reconciled, a fundamental problem to be overcome for agent communication [137]. These mismatches occur when the agents internal models model the same thing in different ways. For example, one agent could model `Transport` as `Car` and `Train`, and another agent could model `Transport` as `AirBased` and `LandBased`. Thus, the agents require some way to reconcile their differences.

Typically this reconciliation is achieved through the use of ontology alignments, which provide a mapping from entities in one ontology to entities in another ontology. Unfortunately, however, the techniques for ontology alignment generation either take a long time or are user-led making them unsuitable for the type of environment being considered here. That is the agents must be capable of doing everything for themselves

[1] For the purposes of this discussion no distinction is made between lightweight ontologies, such as thesauri, and more heavyweight ontologies; all are termed ontology here.

[2] Systematized Nomenclature of Medicine-Clinical Terms - `http://www.ihtsdo.org/snomed-ct/`

[3] National Cancer Institute - `http://ncit.nci.nih.gov/`

without intervention.

However, it is viable to assume that these alignments exist somewhere in the environment; the alignments could be stored, for example, in a repository that the agents can access to retrieve the relevant ones. The problem is now that there are likely to be many alignments available to the agents. Thus, the two agents now need some way to agree upon a solution based on the existence of multiple solutions for the reconciliation of their internal models.

Argumentation, a systematic process of reasoning, allows the agents to arrive at a mutually acceptable solution. Agents can put forth arguments as to why one ontology mapping should or should not be accepted; allowing both of their views to be considered. However, the argumentation process is computationally complex. Thus, we shall again apply the knowledge principle.

Agents perform tasks, that is they are trying to actively achieve something. It is likely that only some of the terms (concepts) in their ontology are relevant to the task at hand, after all, why should they agree upon things that are irrelevant? Agents have bounded resources so they should not want to waste them arguing over unimportant concepts. Thus, we can apply ontology modularization to produce a module that is relevant and appropriate for the task (see Chapter 6). The agents are able to select the subset of their ontology that is relevant to the task and only argue over the concepts in these subsets. So, now the agents are able to reach a mutually agreeable solution on only those concepts that are relevant for their task, which as a consequence reduces the cost of reaching an agreement.

1.2 Research Aims & Contributions

This section summarises the aims and objectives of this thesis and aims to characterise the contributions made to the state of the art.

The research aim of this thesis is to investigate what principles underlie ontology modularization in such a way that ontology modularization can be used in practice. This can be summarised by the following two research questions:

1. How can part of an ontology be reused instead of the whole?

2. How can the ontology modules, obtained as a result of ontology modularization, be used in practice?

In order to answer these questions three main research directions arose.

1. **Ontology Module Extraction.** This led to the development of a traversal based ontology module extraction method that aimed to extract a module about a single concept that could be refined by an Ontology Engineer.

2. **Module Evaluation.** From the need to effectively evaluate the disparate ontology modularization techniques a further research direction arose in the area of ontology evaluation. Despite the significant bodies of work in the areas of ontology modularization and ontology evaluation, few efforts had directly considered the problem of evaluating ontology modules. This resulted in the development of an entropy inspired metric for evaluating ontology modules. All this work was carried out with the consideration of applying ontology modularization to ontology reuse.

3. **Novel Application.** A novel application of ontology modularization in the area of interoperability in distributed and open systems. Specifically, various ontology modularization techniques were investigated for their suitability as a space reduction mechanism for the dynamic selection of ontology alignments in multi-agent systems.

1.3 Thesis Structure

This thesis presents the principles of ontology modularization and how it can be used in practice; these meet the research aims described above. The thesis is divided into five parts, which are further divided into seven chapters and the appendices. Part I presents the background and context of the research relevant to the contributions of this thesis. Part II describes the principles of both ontology for artificial intelligence and ontology modularization. Part III details an evaluation of the different ontology modularization techniques. Then Part IV shows how ontology modularization can be used in practice in two areas: ontology reuse and the dynamic selection of ontology alignments. Finally, Part V outlines the main results of the thesis and discusses some possibilities for future work.

A more detailed description of the structure of the thesis is as follows:

Chapter 1. Defines the motivation and background for this thesis, as well as detailing the research aims and contributions.

Chapter 2. Introduces the fundamental principles of ontology in artificial intelligence. Giving an overview of the differing definitions of ontology in the literature, as well as detailing different representation formalisms. A detailed overview of Description Logics is provided as it is the chosen ontology representation for this thesis. This allows the reader to follow the principles of ontology modularization detailed in the next chapter.

Chapter 3. Addresses the problem of ontology modularization. The existing techniques for ontology modularization are reviewed, highlighting the principles un-

derlying them. Possible definitions of a common framework for ontology modularization are also presented and discussed.

Chapter 4. Evaluates some of the different ontology modularization techniques presented in the previous chapter. Before this the literature regarding ontology evaluation is reviewed. Then the evaluation is conducted along two dimensions: metric based and task-based. The metric based evaluation applies the previously presented metrics to assess the performance of the different ontology modularization techniques. Lastly, the task-based evaluation is presented to compare the performance of the different ontology modularization techniques with respect to three tasks relevant to query answering.

Chapter 5. Applies ontology modularization to the problem of ontology reuse. Existing Ontology Engineering methodologies are presented, noting how they include steps for ontology reuse. Then two methodologies are detailed that allow an Ontology Engineer to reuse an ontology module instead of the whole ontology.

Chapter 6. Applies ontology modularization to the dynamic selection of ontology alignments in multi-agent systems, showing how ontology modularization is used as a space-reduction mechanism. Then the notions of agent and multi-agent systems are given; along with a discussion of the problem of semantic heterogeneity and how ontology alignments overcome this. Next argumentation, specifically the value-based argumentation framework, is introduced and it is shown how this can be used by agents to reach a mutually acceptable alignment. This solution suffers high-complexity so ontology modularization is applied as a space reduction mechanism; this could result in information loss and two solutions are offered. Lastly, an evaluation is presented that shows that ontology modularization successfully reduces the space for the argumentation process without negatively affecting the quality of the agreed alignment.

Chapter 7. Presents some conclusions and identifies some areas of future work based on some open issues of the work presented in this thesis.

This thesis contains some content that has previously been published; detailed as follows:

- Doran, P., Tamma, V., Palmisano, I. and Payne, T. *Ontology Modularization as a space reduction mechanism for the dynamic selection of ontological alignments in MAS.* IEEE Transactions on Knowledge and Data Engineering (TKDE). (In Submission).

- Palmisano, I., Tamma, V., Payne, T. R., Doran, P. *Task Oriented Evaluation of Module Extraction Techniques* In: The Eighth International Semantic Web Conference (ISWC'09) October 25th-29th 2009 Washington, D.C., USA

- Doran, P., Tamma, V., Payne, T. R., Palmisano, I. *An entropy inspired measure for evaluating ontology modularization.* In: 5th International Conference on Knowledge Capture (KCAP'09). September 1st-4th, 2009. Redondo Beach, California, USA.

- Doran, P., Tamma, V., Payne, T. R., Palmisano, I. *Dynamic selection of ontological alignments: a space reduction mechanism.* In: Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09). July 11th-17th, 2009. Pasadena, California, USA.

- Doran, P., Tamma, V., Payne, T. R., Palmisano, I. *Applying Ontology Modularization to Argumentation over Ontology Correspondences in MAS.* In: The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS-09). May 10th-15th, 2009. Budapest, Hungary.

- P. Doran, V. Tamma, I. Palmisano, L. Iannone. *Evaluating Ontology Modules using an Entropy Inspired Metric.* Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2008). 9-12th December 2008. Sydney, Australia.

- I. Palmisano, V. Tamma, L. Iannone, T. Payne, P. Doran. *Dynamic Change Evaluation for Ontology Evolution in the Semantic Web.* Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2008). 9-12th December 2008 Sydney, Australia.

- Doran, P., Tamma, V., Palmisano, I., Payne, T., Iannone, L. *Evaluating Ontology Modules Using an Entropy Inspired Metric.* ULCS Technical Reports (ULCS-08-017).

- Palmisano, I., Tamma, V., Iannone, L., Payne, T., Doran, P. *Dynamic Ontology Evolution in Open Environments.* ULCS Technical Reports (ULCS-08-012).

- Doran, P., Palmisano, I., Tamma, V. *SOMET: algorithm and Tool for SPARQL Based Ontology Module Extraction.* International Workshop on Ontologies: Reasoning and Modularity (WORM-08), ESWC 2008. June 2, 2008. Tenerife, Spain.

- Doran, P., Tamma, V., Iannone, L. *Ontology Module Extraction for Ontology Reuse: An Ontology Engineering Perspective.* Proceedings of the 2007 ACM CIKM International Conference on Information and Knowledge Management. November 6-9, 2007. Lisbon, Portugal.

- d'Aquin, M., Doran, P., Motta, E., Tamma, V. *Towards a Parametric Ontology Modularization Framework Based on Graph Transformation.* Workshop: Inter-

national Workshop on Modular Ontologies (WoMo), K-CAP 2007. October 28, 2007. Whistler, British Columbia, Canada.

- P.Doran. *Ontology Reuse via Ontology Modularisation.* In Proceedings of KnowledgeWeb PhD Symposium 2006 (KWEPSY2006). 17th June 2006. Budva, Montenegro.

Part II

Principles

Chapter 2

Ontology

'Human beings, who are almost unique in having the ability to learn from the experience of others, are also remarkable for their apparent disinclination to do so. - Douglas Adams

Summary This chapter introduces the fundamental principles of ontology for artificial intelligence. First, the various definitions of ontology are considered before moving on to the components and types of ontologies. From here an overview of various ways to represent ontologies are considered. This leads to Description Logics being introduced as the formalism for representing ontologies in this thesis, with consideration given to the open world assumption and reasoning in Description Logics. Lastly, an example ontology is presented which will be used at various places throughout the thesis.

2.1 Ontologies

Ontology is the 'branch of metaphysics concerned with the nature or essence of being or existence' [31]. The differing philosophical views of ontology are varied, for example those of Leibniz and Newton [15]. Newton postulated a reductionist view whereby the noise of our experience is reduced to that which is considered necessary; Newton's Laws of Motion being a perfect example. Conversely, Leibniz postulated a constructionist view whereby our experience shapes our Ontology, the many 'parts' of our experience form the 'whole'. For example, over many interactions (parts) with an internet search engines it will adjust its output (the whole) to fit us. Discussions surrounding these philosophical differences would be out of context in this thesis, see Guarino and Giaretta [64] for a discussion on these philosophical issues; thus the focus shall be on the definition of ontology within the context of Artificial Intelligence.

Guarino and Giaretta [64] provide a useful distinction. They state that 'Ontology' denotes the philosophical discipline concerned with the nature of being and that 'ontology' denotes its use in knowledge base systems. This thesis is concerned with 'ontology', not 'Ontology'.

2.1.1 What is an 'ontology'?

It does seem somewhat ironic that it is necessary to provide a definition for a word that supposedly describes what exists; but there are disparate views within literature that are worthy of consideration. One definition frequently used in the literature, arguably the *de facto* definition, is the one provided by Gruber [59] which is that an ontology is a "specification of a shared conceptualization". The majority of the definitions [10, 6, 64, 61, 129] can be seen as refinements on Gruber's. Let us first consider these alternative definitions:

Guarino and Giaretta. "A logical theory which gives an explicit, partial account of a conceptualization." [64]

Here the ontology is characterised as only providing a partial account of the intended conceptualization. The assumption being that it is not possible to construct an ontology that completely expresses the intended conceptualization.

Bernaras, Laresgoiti and Corera. "The ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base." [6]

This definition reflects the role that the ontology will play, but still it suggests that an ontology provides the description of a particular viewpoint about the knowledge base.

Borst. "An ontology is a formal specification of a shared conceptualization." [10]

This definition introduces the notion of a shared conceptualization, this being that an ontology reflects the common understanding of the modelled domain. One would expect this consensus to be achieved by a community of users.

Guarino. "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." [61]

This takes into account not only the conceptualization but also the language used to describe it and the commitments that come with this. It also suggests that this language should be formal. The ontological commitment should be made explicit when applying the ontology, this should facilitate its accessibility, maintainability and integrity. This is analogous to including comments in your programming code.

Studer, Benjamins and Fensel. "An ontology is a formal, explicit specification of a shared conceptualization." [129]

This extends Borst's definition by adding the constraint that the ontology should be explicit meaning that the type of concepts used, and how they are to be used are explicitly defined.

The above definitions demonstrate that the refinements to Gruber's definition occur along two dimensions, specification and conceptualization. Thus two more questions need to be posed:

1. What is a specification?
2. What is a conceptualization?

What is a Specification?

A specification, especially in Computer Science, tends to be a formal description of how something could be constructed to meet certain criteria. Indeed, this notion of specification concurs with Gruber's [58] whereby a specification can be the formal specification of a program. For example, using Backus-Naur Form (BNF) to express context-free grammars as used by some programming languages. With ontology languages it is usual to specify both the syntax and semantics.

Borst [10] states that the specification of an ontology should be formal, agreeing with the general notion above. Studer, Benjamins and Fensel [129] go one step further and suggest that the specification should also be explicit; in the sense that all concepts be explicitly defined. Whilst explicit definitions are of use they could possibly introduce unwanted arbitrary concept labels into the ontology. For example, in Description Logics (see Section 2.3) not every restriction requires a concept label, so whilst the restriction is formal and unambiguous it is an implicit part of some other explicit concept definition.

What is a Conceptualization?

Once the language of the ontology has been specified it needs to be put to use; this requires conceptualization. Conceptualization involves grounding the symbols of the ontology language in a domain. For example, the propositional logic formula

$$A(x) \rightarrow B(x)$$

is abstract and can be grounded into a domain as

$$Man(x) \rightarrow Mortal(x)$$

The conceptualization alone is of limited use unless the interpretation of the symbols used to conceptualize can be shared. This requires an ontological commitment; an agreement to use a vocabulary in a coherent and consistent manner. An ontological commitment is an agreement on the meaning of the vocabulary used to share knowledge;

a mapping between a concept and its chosen intended meaning. An agent is said to commit to an ontology if its knowledge conforms to the ontology with respect to the semantics and interpretations of the symbols.

Figure 2.1 shows this notion. The conceptualization (**C**) allows the models of some language (**M(L)**) to be constrained to a subset of intended models ($\mathbf{I_K(L)}$) due to a commitment (**K**) to a specific conceptualization. Thus, the conceptualization identifies the objects (both abstract and real) that exist in some world and the relationships that exist between them.

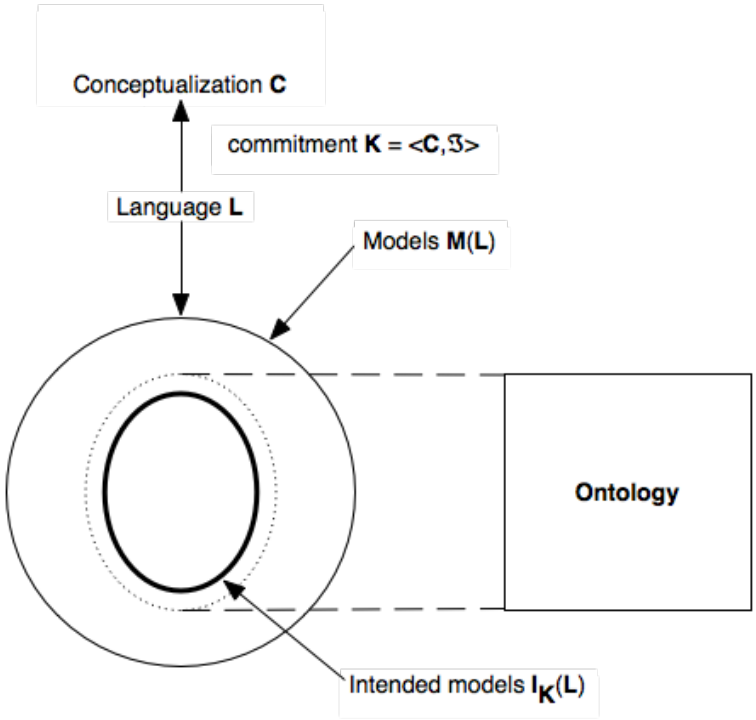

Figure 2.1: The intended models of a logical language reflect its commitment to a conceptualization. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating this set of intended models. [61]

Bernaras, Laresgoiti and Corera [6] says that the contents of a knowledge base should be a conceptualization of its knowledge. A major drawback of early expert systems was that the knowledge base contained both domain knowledge and knowledge on how to use the domain knowledge (rules). These were highly interconnected making it difficult to replicate success in one domain to another domain. For example, Mycin [16] was a rule-based expert system for blood analysis. Mycin was able to diagnose blood infections, for example menigitis, and then propose appropriate medication; this was its domain knowledge. If this knowledge had been properly conceptualized then it should have been possible to separate out the domain knowledge from the application knowledge, but the rules used for the inference encoded this knowledge directly and it proved difficult to separate them.

Studer, Benjamins and Fensel [129] and Borst [10] argue that the conceptualization should be shared. This notion of shared conceptualization is important as it implies that an ontology can be applied across a variety of applications if the conceptualization of the domain is (at least partially) shared.

Verdict

Despite these contrasting views a definition of ontology needs to be adopted for this thesis. Guarino and Giaretta [64] define an ontology as:

> "A logical theory which gives an explicit, partial account of a conceptualization."

This definition is useful to adopt for our purposes as it captures that there is an expectation that an ontology has a formal underpinning (in this thesis it is Description Logics) and that there is no assumption, or expectation, that an ontology is complete. Assuming that an ontology is complete would mean, according to Figure 2.1, that the set of ontology models was equivalent to the set of intended models. However, the ontology would only be complete with respect to the given conceptualization and it is possible that the same domain can be conceptualized differently according to a number viewpoints. This fits nicely with the Open World Assumption (see Section 2.3.1) made by Description Logics which assumes that no agent has complete knowledge.

2.1.2 Components of Ontologies

Ontologies formalise the knowledge in a domain by means of a set of components: concepts, relations, functions, axioms and instances [58, 50]. The aim here is to introduce and clarify this terminology.

- **Concepts (C)** A concept represents the abstractions used to describe objects in the world. It is described by a term (a symbol), an extension and an intension.
- **Relations (R)** The set of relationships defined over the set of concepts (C), such that each $r \in R$ is an ordered n-ple $r = (C_1 \times C_2 \times \ldots \times C_n)$
- **Functions (F)** The set of functions defined over the set of concepts that return a concept, such that each $f \in F : (C_1 \times C_2 \times \ldots \times C_{n-1} \mapsto C_n)$
- **Axioms (A)** A set of assertions, taken to be true, that constrain the meaning of concepts, relations and functions. They can also provide provision for correctness checking and inference.
- **Instances (I)** The set of individuals whereby an individual is an object of the world. Subsets of I can correspond to the extension of a concept, in this case those individuals are said to be *instances* of that concept.

Depending on the ontology language chosen and the scope of the ontology it is possible that only a subset of the above will be used.

2.1.3 Types of Ontologies

There are numerous types of ontology and these can be broadly split along two dimensions: level of generality and the type of knowledge being modelled. From Guarino [61] and van Heijst, Schreiber and Wielinga [139] it is possible to identify the following classifications of ontologies:

- **Upper-level/Generic** Describe very general concepts or 'common-sense' knowledge, such as space, time, etc., which are independent from a particular problem or domain. They express conceptualizations that are specific not to a domain, but apply across multiple domains. For example, the IEEE Standard Upper Ontology Working Group (SUO WG)[1] is trying to define an upper ontology containing concepts not unique to a domain so that its concepts can be used when creating domain ontologies.

- **Domain** Model specific domains, such as medicine, academia, etc., which means that they are not independent from a particular domain. The concepts in domain ontologies can usually be seen as specialisations of concepts defined in an upper level ontology [54]. For example, `Location` in a domain ontology is a specialization of `Spatial Point` that could be defined in an upper ontology.

- **Task** Describe generic or domain-specific activities, such as diagnosis or selling. They provide a vocabulary of terms associated with a task that may or may not be in the same domain.

- **Application** Describe concepts depending both on a particular domain and particular task. They are often specialisations of domain, task and generic ontologies, corresponding to the roles played by domain entities when they perform certain activities. For example, an application ontology could be created for a travel agent which covers the different destinations, etc., and the tasks a travel agent needs to carry out, booking tickets, etc.

- **Representation** Describe the conceptualizations that underlie knowledge representation formalisms [29]. They provide no claims about the world, but just a representational [63]. For example, the *Frame Ontology* used by Ontolingua [58].

[1] `http://suo.ieee.org/`

Along with this both Uschold and Gruninger [138] and McGuinness [94] suggest that ontologies can be classified along a dimension of formality, from highly informal to highly formal. Uschold and Gruninger [138] provide the following classification:

- **Highly-informal** The ontology is expressed in natural language, thus, suffering from the inherent ambiguity of natural language.
- **Semi-informal** The ontology is expressed in a restricted and structured form of natural language. This achieves improvement in clarity and reduces the possibility for ambiguity. For example, Hart, Johnson and Dolbear [67] present a controlled natural language that can be translated into the Description Logic (DL) equivalent to OWL (see Section 2.3); so the sentence *"Car is a kind of Transport"* is equivalent to the DL axiom

$$\texttt{Car} \sqsubseteq \texttt{Transport}$$

- **Semi-formal** The ontology is expressed in a formally defined artificial language. For example the Ontolingua [58] language for describing ontologies.
- **Highly-formal** The ontology is expressed in a language whose terms have a formal semantics. Section 2.2 provides some examples of this kind.

Whereas, McGuinness [94] provides an *"ontological continuum"* of different types of models, but this can broadly by split into the following classification:

- **Informal Models** This includes, in increasing formality, glossaries, thesauri and informal taxonomies (for example, Wikipedia's classification system).
- **Formal Models** This begins at formal taxonomies and by adding further semantics, such as value restrictions and disjointness we arrive at a point near to Description Logics (see Section 2.3).

Informal models would include WordNet [95] and the ACM Classification[2] and formal models, as they are expressed in description logic, would include many of the ontologies available for use on the Semantic Web, such as Friend of a Friend (FOAF)[3] which models descriptions of people and the links between them; Semantically-Interlinked Online Communities(SIOC)[4] [14] whose model facilitates the integration of online communities.; and GoodRelations[5] [70] which models descriptions of goods, and the terms and conditions of items and services offered on the Web.

[2] `http://www.acm.org/about/class/`
[3] `http://www.foaf-project.org`
[4] `http://sioc-project.org/`
[5] `http://purl.org/goodrelations/`

2.2 Ways to Represent an Ontology

There are numerous formalisms to represent ontologies: first-order logic, frames, semantic nets and description logics to name a few.

2.2.1 First-order Logic

First-order logic (FOL) [72, 145] is a formal deductive system that allows predicates and quantification along with the declarative propositions of propositional logic. The benefits of FOL for ontologies are its freedom in predicate choice and its use of variables, which allows us to easily capture the "all men are mortal" example with the following:

$$\forall x(Man(x) \rightarrow Mortal(x))$$

Unfortunately FOL is undecidable although semidecidable [7][6]. This means that for FOL we can have sound (the answers obtained are correct) but not complete (we might not always get all the answers) reasoning.

The Knowledge Interchange Format (KIF) [49] is based on first-order logic and was intended to represent the content of messages that were exchanged between two agents. Its syntax has a LISP-like structure, so the example above represented in KIF would be:

$$(\texttt{forall}\ (?x\ \texttt{Man})\ (=>\ (?x\ \texttt{Mortal})))$$

2.2.2 Frames

Frames [96] are data structures that can be used for knowledge representation. A frame allows for a typical situation to be captured, connecting these frames together allows an 'idea' to be captured. The reasoning capabilities of frames are usually restricted to inheritance. The representation of the "all men are mortal" example in Frames requires to know that Frames represent a typical situation. For example:

`Man` :

`isMortal` : `True`

This says that `Man` stereotypically has the property `isMortal` set to `True`, but it would be possible to create a subclass of `Man`, say `ImmortalMan`, where this property is to `False`.

KL-ONE [12] is an early example of a frame based language attempting to overcome the lack of formal semantics in semantic networks. KL-ONE allowed for subclass and superclass relations among its frames. Gruber [58] proposed to model ontologies using frames and first order logics using the modelling primitives stated in Section 2.1.2; that is concepts, functions, roles, axioms and instances. This resulted in Ontolingua [58]

[6] A theory is decidable if and only if both it and its complement are semidecidable

which was a frame based language that allowed ontologies to be translated from one ontology language to another. FLogic [80] and OKBC [18] are examples of other frame languages. OIL [44] was also a frame based language, this developed into DAML+OIL which later became OWL (see below).

2.2.3 Conceptual Graphs

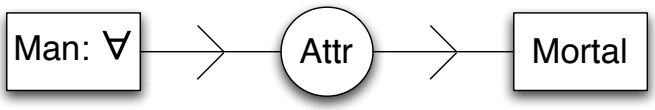

Figure 2.2: An example of a conceptual graph representing the "all men are mortal" example.

Conceptual Graphs (CGs) [122] combine the graphical notation of semantic networks with the algebraic notation of predicate logic defined by Pierce [107]; as such they have both a graphical and linear notation. CGs allow *conceptual relations* to be stated between concepts. The reasoning capabilities of CGs are essentially a variant on those offered by first-order logic (see [123] for a more detailed account). The "all men are mortal" example in the linear notation is:

$$[\texttt{Man} : \forall] - > (Attr) - > [\texttt{Mortal}]$$

This says that all `Man` has an attribute of `Mortal`; the graphical notation for this example is shown in Figure 2.2.

2.2.4 Description Logic

Description Logics are intended to be decidable subsets of FOL. The ontology language that is used throughout the rest of this thesis is Description Logics. Description Logics are the underlying model defining the W3C Web Ontology Language (OWL) [106, 74] and is the language used in the context of the Semantic Web. Section 2.3 presents the definition of Description Logic. The "all men are mortal" example in Description Logics is represented by the following axiom:

$$\texttt{Man} \sqsubseteq \texttt{Mortal}$$

That is the set of things `Man` is a subset of all things `Mortal`.

2.3 Description Logic (DL)

Description logics can be used to specify concept definitions within a domain, a terminological specification, in a structured and well formed manner. Description logics are intended to be decidable subsets of first-order logic(FOL) meaning that sound and

Name	Syntax	Semantics
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
atomic concept	A	$A^{\mathcal{I}} (\subseteq \Delta^{\mathcal{I}})$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y\ (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
existential restriction	$\exists R.\top$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x,y) \in R^{\mathcal{I}}\}$

Table 2.1: Basic (atomic) semantics

Name	Syntax	Semantics
atomic negation	$\neg A, A \in N_C$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
full negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
concept intersection	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
concept union	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
full existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
at most restriction	$\leq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\}\rvert \leq n\}$
at least restriction	$\geq nR$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\}\rvert \geq n\}$
qualified at most restriction	$\leq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \in \Delta \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\rvert \leq n\}$
qualified at least restriction	$\geq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}\rvert \geq n\}$
one-of	$\{x_1, x_2, \ldots, x_n\}$	$\{x \in \Delta^{\mathcal{I}} \mid x = x_i, 1 \leq i \leq n\}$
has value	$\exists R.\{x\}$	$\{y \in \Delta^{\mathcal{I}} \mid (y,x) \in R^{\mathcal{I}}\}$
inverse of	R^-	$\{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (y,x) \in R^{\mathcal{I}}\}$

Table 2.2: Constructors semantics

complete reasoning is possible. They rose from the need to provide frame based systems with a formal semantics, aided by Hayes [69] who demonstrated that frames could be given a formal semantics by using subsets first-order logic. This allowed reasoning to be done without the need for first-order logic theorem provers [3]. The combination of formal semantics and practical reasoning has led Description Logics to become the *de facto* ontology language; and it is the ontology language we adopt for the rest of this thesis.

A description logic knowledge base (KB) consists of two components, the TBox($\mathcal{T}$), containing intensional knowledge, and the ABox($\mathcal{A}$), containing extensional knowledge [3]. The TBox defines the terminology (vocabulary) and the ABox contains assertions about named individuals in terms of the TBox.

The terminology comprises of concepts, denoting sets of individuals, and roles, which denote binary relations between the individuals. Complex descriptions can be assigned a name in the TBox allowing Knowledge Engineers to extend beyond atomic concepts and roles, for example `Mother` $\equiv$ `Woman` $\sqcap \exists$`hasChild.Person`.

Each DL system is distinguished by their language for building descriptions (description language). Description languages are assumed to have two kinds of symbols, atomic concepts (denoted by A and B) and atomic roles (denoted by R). The semantics

of atomic concepts and roles can be seen in Table 2.1. Atomic concepts and atomic roles are the basic building blocks that are used in concept and role constructors to build complex descriptions. The letters C and D denote arbitrary more complex descriptions. The semantics of these constructors can be seen in Table 2.2.

As can be seen from Table 2.1 and 2.2 the formal semantics of a concept is considered in terms of *interpretations*, $\mathcal{I}$, that are a non-empty set $\Delta^{\mathcal{I}}$, this is the domain of interpretation, and an interpretation function. The interpretation function assigns for every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, this is then extended, as shown in Table 2.1 and 2.2, to cover concept description.

Let us consider an example of how interpretations work; assume we have the following TBox that consists of two axioms from the ontology presented in Section 2.5:

$$\texttt{Academic} \sqsubseteq \texttt{Person}$$

$$\texttt{Student} \sqsubseteq \texttt{Person}$$

This has the following semantics in terms of interpretations:

$$\texttt{Academic}^{\mathcal{I}} \subseteq \texttt{Person}^{\mathcal{I}}$$

$$\texttt{Student}^{\mathcal{I}} \subseteq \texttt{Person}^{\mathcal{I}}$$

This situation can be represented using a Venn diagram, as shown in Figure 2.3. This clearly shows that the set of individuals represented by `Academic` and `Student` are subsets of `Person`.

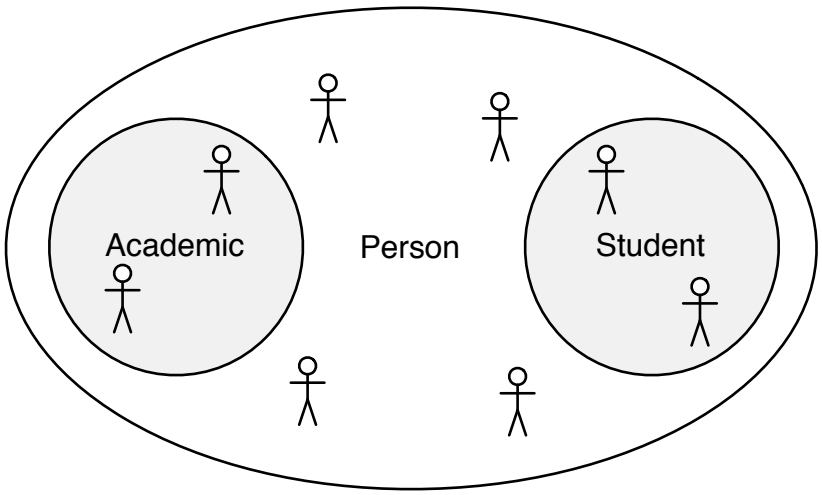

Figure 2.3: A Venn diagram representing description logic interpretations

By taking different DL operators we can construct different description logics. For example, the description logic $\mathcal{ALC}$ ($\mathcal{AL} + \mathcal{C}$) gives us:

DL Name	Constructors
$\mathcal{ALN}$	$\sqcap, \forall R.C, \exists R.\top, \geq R, \leq R$
$\mathcal{ALC}$	$\sqcap, \sqcup, \neg C, \forall R.C, \exists R.C$
$\mathcal{SHOIN}$	$\mathcal{ALN} \cup \mathcal{ALC}, R^{-}$, role hierarchies, role transitivity, role symmetry, (inverse) functional properties

Table 2.3: Description Logics constructor subsets

- $\mathcal{AL}$ = Atomic negation, concept intersection, universal restrictions and limited existential quantification.
- $\mathcal{C}$ = Negation of arbitrary concepts.

The differences between some of the different description languages are shown in Table 2.3.

Lastly, the W3C[7] have defined the Web Ontology Language (OWL) [106]. OWL comes in three flavours: OWL Lite, OWL DL and OWL Full. OWL DL is equivalent to $\mathcal{SHOIN}(\mathcal{D})$;meaning we have the following operators:

- $\mathcal{S} = \mathcal{ALC}$ with transitive roles. Allowing one to create transitive properties. For example, if `John isWith Paul` and `Paul isWith Ringo` and `isWith` is transitive then `John isWith Ringo`.
- $\mathcal{H}$ = Role hierarchy. Allowing one to create subproperties of properties. For example, `hasAuthor` has the subproperty `hasCoAuthor`.
- $\mathcal{O}$ = Nominals. Allowing one to state that a class is restricted to a given set of individuals. For example, `BeatlesMember` is *one of* the set {`John, Paul, Ringo` and `George`}, where this set is a set of instance names.
- $\mathcal{I}$ = Inverse properties. Allowing one to state an inverse property of a property. For example, `isAuthor` is the inverse of `hasAuthor`
- $\mathcal{N}$ = Cardinality restrictions. Allowing one to restrict the number of times an individual can have a certain property. For example, a `Person` can have *at most one* `dateOfBirth`.
- $\mathcal{D}$ = Datatypes. Allowing one to assign a datatype to a property. For example, the range of `name` must be a string.

There are two other variants of OWL, these are:

[7] `http://www.w3.org`

- *OWL Lite.* Is a syntactic subset of OWL DL and has expressivity $\mathcal{SHIF}(\mathcal{D})$ [75]. This results in the following: prohibits unions and complements, does not allow individuals to occur in the description of a class, and limits cardinalities to 0 or 1. This results in more efficient reasoning than OWL DL, but less expressive power.

- *OWL Full.* Contains OWL DL, but has more expressive power than Description Logics. OWL Full was designed to be fully upwardly compatible with RDF and RDFS [3] (see Section 2.4). For example, it is possible to represent a resource simultaneously as a set of individuals and an individual. This kind of construct means reasoning in OWL Full is undecidable [3].

OWL provides a standardised syntax, based on XML, for representing description logic ontologies. This facilitates the sharing of ontology specifications as it allows humans and computers to successfully give the syntax of the ontology specification the appropriate semantics.

There is a proposed W3C recommendation for OWL 2[8]. OWL 2 is a compatible extension to OWL providing some new features, such as providing extra restrictions for datatypes and qualified cardinality restrictions.

2.3.1 Open World Assumption

DLs adopt the open world assumption which informally means that no single knowledge base has complete knowledge and therefore cannot assume a closed world. A closed world is one where any statement that is not known to be true is assumed to be false, the notion being that everything is known. The adoption of the open world assumption affects what kind of inferences can be made. Furthermore, it is a fair assumption to make when considering open and distributed environments (see Section 6.2.3), such as the Semantic Web, where it would be unwise to assume an agent knew everything.

Let us consider the following example to highlight this important distinction. Assume we have the following statement.

`Statement: 'Turing' is a 'Machine'`

Now if we ask the following question.

`Question: 'John' is a 'Machine'?`

The answer from a closed and open world would be as follow.

`Closed World: No`
`Open World: Unknown`

[8] `http://www.w3.org/TR/owl2-profiles/`

So, the closed world system is able to state that `'John'` is not a `'Machine'` because nowhere does it state that he is. However, this absence of information in the open world means that we could equally derive that both `'John'` is a `'Machine'` and that `'John'` is not a `'Machine'`, and so we cannot definitively answer the question.

This raises an interesting problem for Knowledge Engineers who choose to use DLs because it means that one cannot fully specify what can be said about a concept; it is only possible to constrain the individuals that will be entailed to be an instance of the concept.

2.3.2 Reasoning with DLs

The following reasoning services[9] are supported by DLs [3]:

TBox

- *Concept Satisfiability.* Can the concept definition admit instances? C is satisfiable iff there is some model $\mathcal{I}$ of $\mathcal{T}$ such that

$$C^{\mathcal{I}} \neq \emptyset$$

- *Concept Subsumption.* Is one concept subsumed by another concept? $\mathcal{T} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. Concept equivalence and concept disjointness can be reduced to concept subsumption as follows:
  - $\mathcal{T} \models C \equiv D \leftrightarrow \mathcal{T} \models C \sqsubseteq D, D \sqsubseteq C$
  - $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset \leftrightarrow \mathcal{T} \models (C \sqcap D) \sqsubseteq \bot$

ABox

- *Consistency.* Is the ABox satisfiable with respect to some TBox? $\mathcal{A}$ is consistent iff there exists some model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$
- *Instance Checking.* Is an individual an instance of a concept? a is an instance of C iff for every model $\mathcal{I}$ of $\mathcal{T}$ and $\mathcal{A}$

$$a^{\mathcal{I}} \in C^{\mathcal{I}}$$

- *Realization.* For all individual in the ABox compute their most specific concept names with respect to some TBox such that $\mathcal{A} \models C(a)$ and C is minimal with respect to the subsumption ordering.

There are now several software implementations for carrying out these reasoning tasks, such as Pellet [120], KAON [143] and FACT++ [135].

[9] All of these services can be reduced to satisfiability, see [3]

2.3.3 When is a DL TBox an Ontology?

Having a collection of DL axioms does not mean that you have an ontology; an ontology is more than a collection of axioms; it is the result of some form of conceptualization (see Section 2.1.1). Thus, a TBox is an ontology when it actually represents a conceptualization specified by a Knowledge Engineer. For example, consider the following axioms: $A \sqsubseteq B \sqcup C$ and $Politician \sqsubseteq Left \sqcup Right$. Under the semantics of DL these two axioms are equivalent; the semantics of DL place no semantics on the label of a concept, it is entirely possible that two concepts with different labels can be inferred to be the same concept. However, the intended meaning of the second axiom is the result of some process of conceptualization and the resulting ontological commitment. That is the second axiom contains labels that are intended to convey meaning beyond the semantics of DL and relate to the domain being modelled by the axiom.

2.4 Representing a TBox as a Graph

The Web Ontology Language (OWL) [106] is a vocabulary extension of the Resource Description Framework (RDF) [68]. RDF provides a language for describing resources on the web. For example, one could have a resource describing a person that has properties such as name, age, etc. RDF has 3 syntactic representations, one of which is a graph. Thus, every OWL ontology has an RDF graph representation. Therefore, by extension every DL TBox that falls within the expressivity of OWL has a graph representation.

Being able to represent an ontology as a graph will be important in the upcoming chapters, particularly Section 3.4.1 where the ontology module extraction techniques require the graphical representation of the ontology in order to perform a conditional traversal upon it. As such, an explanation and example will be provided below to show how these transformations in representation are achieved[10].

Let us consider the following TBox:

$$\texttt{Academic} \sqsubseteq \texttt{Person}$$

$$\texttt{Student} \sqsubseteq \texttt{Person}$$

The above two axioms represent a DL ontology that falls within the expressivity of OWL, as such we can encode these axioms in a valid OWL/XML file as follows:

```
<owl:Class rdf:about="#Person">
</owl:Class>
<owl:Class rdf:about="#Academic">
```

[10]The full mapping of OWL to RDF Graphs can be found at `http://www.w3.org/TR/owl-semantics/mapping.html`

```
    <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:about="#Student">
    <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
```

The above is a fragment of a valid OWL file so we know that it has an RDF graph representation. Thus, to obtain the RDF graph we just need to produces the RDF triples for the above fragment. An RDF triple is of the form <subject, predicate, object> whereby the subject and object are nodes in the graph, and the predicate is the directed labelled edge connecting the subject and object. Therefore, we have the following RDF triples:

`#Student rdf:type owl:Class.`

`#Person rdf:type owl:Class.`

`#Academic rdf:type owl:Class.`

`owl:Thing rdf:type owl:Class.`

`#Academic rdfs:subClassOf #Person.`

`#Student rdfs:subClassOf #Person.`

The first four state that the classes we defined in the original OWL are RDF resources that are of the type `owl:Class`. The last two express the subclass relations between the classes. The above is a straightforward example, but the translation becomes, perhaps, a little less obvious with other forms of axiom. For example, consider the following axiom:

$$\texttt{PhDThesis} \sqsubseteq \exists \texttt{hasAuthor}.\texttt{PhDStudent}$$

This would be expressed in OWL/XML as follows:

```
<owl:Class rdf:about="#PhDStudent">
</owl:Class>

<owl:ObjectProperty rdf:about="#hasAuthor"/>

<owl:Class rdf:about="#PhDThesis">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasAuthor"/>
```

```
            <owl:someValuesFrom rdf:resource="#PhDStudent"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Note that the existential restriction is represented as an `owl:Restriction` which contains the property and class that the restriction applies to. This fragment can be trasnformed into the following RDF graph:

```
#PhDStudent rdf:type owl:Class.

#PhDThesis rdf:type owl:Class.

#hasAuthor rdf:type owl:ObjectProperty.

#PhDThesis rdfs:subClassOf _:blank.

_:blank rdf:type owl:Restriction.

_:blank owl:someValuesFrom #PhDStudent.

_:blank owl:onProperty #hasAuthor.
```

Here the first three triples state that `PhDStudent` and `PhDThesis` are classes, plus that `hasAuthor` is an object property. The final four triples represent the existential restriction. They make use of a blank node (`_:blank`). The blank node allows the various components of the restriction to be represented in the graph. For example, in the above `PhDThesis` is a subclass of the blank node `_:blank` and `_:blank` is defined as a restriction on property `hasAuthor` with some values from `PhDStudent`.

2.5 An Example Ontology

The example ontology presented here is an ontology about theses; passing reference will be made throughout this thesis to this example ontology. Appendix B provides the full Axiomatization along with the OWL file, with TURTLE serialisation. A graphical representation of the ontology can be seen in Figure 2.4

The ontology states that a `Thesis` is written by one `Student`, has at least one `Academic` as a supervisor and consists of `Chapters`. Instances of `Person` play different roles, such as `Academic` and `Student`. These roles are played by a `Person` for a limited time, but this can not be represented in Description Logic. Therefore, there is a dichotomy between what an ontology should model and the constraints posed by the chosen ontology language.

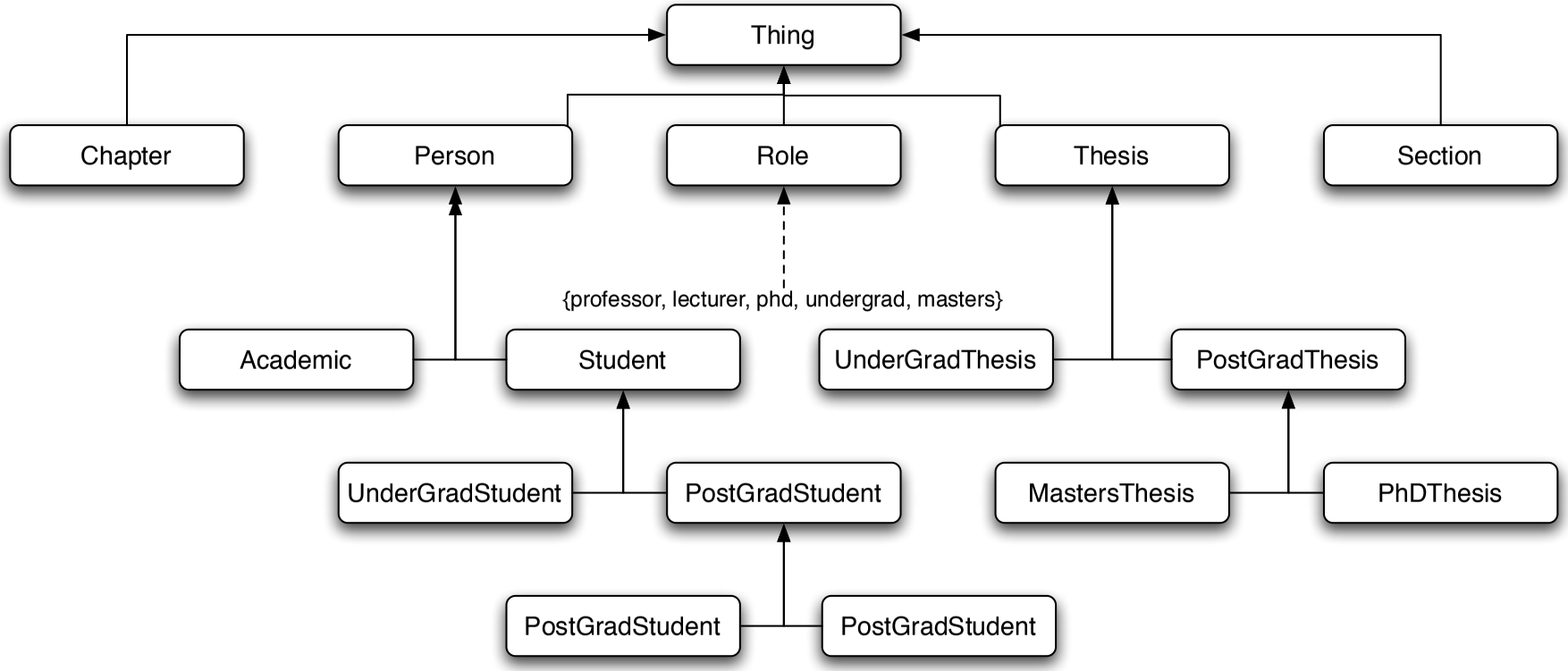

Figure 2.4: Graphical representation of the Thesis ontology taxonomy.

2.6 Ontologies In The Semantic Web

The Semantic Web aims to add a layer of meaning to the World Wide Web [134]. It is characterised as an open and distributed environment, where few assumptions can be made about the parties involved (for more on environments see Section 6.2.3). The result of this is to turn the Web from a web of documents into a web of knowledge, whereby the knowledge becomes machine understandable. For example, a query such as *"Give me a list of Jaguars built before 1980"*. This query on the 'syntactic' (current) Web is useless because the search engine has no notion of what the query is about; for that we need semantics. If all the pages containing information about jaguars were marked up in such a way as to describe the information they contained in a machine understandable way[11] then, it may be possible, for a computer to disambiguate the terms used in the query. For example, it would be able to tell the difference between Jaguar the car manufacturer and jaguar the animal. On the Semantic Web it is assumed that every resource is given a uniform resource identifier (URI) so that "anyone can link to it, refer to it or retrieve a representation of it" [116].

"The challenge of the Semantic Web, therefore, is to provide a language that expresses both data and rules for reasoning about the data" [134]. Thus, ontologies provide the mechanism by which we can assign semantics to the Web, but having an ontology is not enough. The adoption of a standard ontology language to encode ontologies is required to prevent the Web from becoming a set of "isolated monoliths" [57]. By agreeing upon a standard ontology language, for the Semantic Web this is OWL (see Section 2.3), we are agreeing upon both a shared syntax and a shared semantics to interpret this syntax. For example, if one ontology contains the following statement in OWL:

```
<owl:Class rdf:ID="PhDStudent">
<rdfs:subClassOf rdf:resource="#Student"/>
</owl:Class>
```

This states that `PhDStudent` is a subclass of `Student`. However, for this to be machine understandable this syntax needs to be given a standard semantics so that the notion of 'subclass' remains the same across machines. In the case of OWL, the standard semantics are Description Logic semantics, see Section 2.3, which in this case means that every instance of `PhDStudent` is also an instance of `Student` and it is possible, though not necessary, that `PhDStudent` and `Student` are equivalent. Thus, a standard syntax and semantics, or ontology language, provides a way to unambiguously interpret a set of symbols.

[11] This kind of markup is possible through the use of RDFa (`http://www.w3.org/TR/xhtml-rdfa-primer/`) as it allows RDF statements to be embedded in an HTML page.

In Semantic Web ontologies facilitate interoperability by allowing the different participants to agree on the meaning of the terms they use to communicate. An important step towards this is to achieve a form of consensus. Section 6.4 demonstrates a mechanism by which agents are able to reach consensus over a set of mappings between their ontologies. However, consensus can be achieved in another way and that is through the use of a shared ontology, with the shared ontology representing a shared conceptualization. For example, the Friend of a Friend (FOAF)[12] ontology is an ontology for describing people and the links between them. Thus, if two separate ontologies reuse FOAF to describe the people in their ontology then they will be interoperable with respect to the terms described by FOAF; each ontology will be able to interpret the description of a person provided by the other ontology.

[12] `http://www.foaf-project.org`

Chapter 3

Ontology Modularization

'Peace comes from within. Do not seek it without.' - Buddha

Summary This chapter details the research relating to ontology modularization, the process of identifying subsets of an ontology. Both ontology partitioning and ontology module extraction are approaches to ontology modularization. The different ontology modularization techniques are discussed, including both logical and traversal based ontology module extraction techniques. A classification of ontology module extraction techniques based on their properties is also provided. Work covering the issue of producing a common framework for ontology module extraction is also discussed.

3.1 Motivation

Ontologies are increasingly being used in knowledge management systems, e-Science, and bio-informatics [124]. As such, ontologies are used to perform a diverse range of tasks, such as reasoning, query answering, reuse, etc.; with each of these tasks placing different constraints upon the ontology. However, the design, maintenance, reuse, and integration of ontologies are complex tasks [24]. In order to provide efficiency gains for these tasks one can use ontology modularization.

Ontology modularization overcomes the problem of identifying a fragment of an existing ontology to be reused and is listed as one of the principles for building good ontologies in Ontology Engineering good practices [6]. It enables ontology developers to include only those concepts and relations that are relevant for the application they are modelling an ontology for. In essence, why use the whole ontology when an ontology module would suffice?

Consider the example ontology presented in Section 2.5 that presents an ontology about theses, and in the application under consideration there is only a need to talk about `PhDStudent`. There are several choices available:

- *Reuse the whole ontology.* This requires all the definitions in the existing ontology

to be included in the ontology for the application, such as those concerning *Academic*. This creates an unnecessary overhead, which is even bigger if the ontology being reused is large. The overhead being in terms of reasoning and maintenance.

- *Build a new ontology.* If the overhead in reusing the whole of the existing ontology is large then one solution would be to create the definitions needed in the ontology for the application. This requires much work on the part of the developer and, as mentioned above, ontology design is a complex task. Furthermore, if the ontology being reused has definitions that are popular or a standard, such as the Systematised Nomenclature of Medicine Clinical Terms (SNOMED CT)[1], then it makes little sense to reinvent what is already there.

- *Reuse part of the existing ontology.* This allows for the reuse of part, a module, of the existing ontology. It can be seen as a compromise between the above two. The application ontology only gets added to it what is needed and existing resources can be taken advantage of.

With this in mind Rector *et al.* [108] present the following goals for ontology modularization:

1. *Scalability.* This is concerned with the scalability of Description Logic (DL) reasoning (see Section 2.3.2). It is widely understood that, in general, the performance of DL reasoners degrade as the size of the ontology grows.[2] Thus, there is a motivation to reduce the size of the ontology that needs to be reasoned over to that which is necessary, i.e. an ontology module. The scalability issue also concerns the evolution of the ontology, the aim being to localise the change within an ontology module.

2. *Complexity Management.* With human designed ontologies it becomes increasingly difficult to control the accurateness of the ontology. Ontology modularization allows the designer to just focus on the relevant portion of the ontology.

3. *Understandability.* Intuitively smaller modules are easier to understand than larger ones. This is the case for both humans and agents (for more on agents see Section 6.2).

4. *Reuse.* This is common practice in Software Engineering and Ontology Engineering would benefit from such an approach; Chapter 5 addresses this goal. This goal emphasises the need for mechanisms to produce modules in such a way that increases their chances of being reused; i.e., they only contain what is relevant and useful.

[1] `http://www.ihtsdo.org/snomed-ct/`

[2] It is also possible that small ontologies can degrade performance, for example by including numerous general concept inclusion (GCI) axioms. This is discouraged in practice.

This Chapter details the principles underlying the many techniques for ontology modularization. Indeed, the existence of numerous techniques for ontology modularization has motivated research into creating a common framework for ontology modularization; this work is detailed in Section 3.6.

3.2 Ontology Modularization

An ontology O can be defined as a pair

$$O = (Ax(O), Sig(O))$$

where $Ax(O)$ is a set of axioms (intensional and extensional), remember that Section 2.1.2 defined axioms as consisting of concepts, relations and functions. $Sig(O)$ is the signature of O [3]. The signature of an ontology O is the set of entity names (both concepts and properties) used by O, i.e., its vocabulary. Ontology modularization is therefore the process of defining a module $M = (Ax(M), Sig(M))$; where M is a subset of O, $M \subseteq O$, so $Ax(M) \subseteq Ax(O)$ and $Sig(M) \subseteq Sig(O)$. No assumptions beyond this are made at this point about the nature of a module.

The aim of modularization in general, regardless of the task, is to some extent to reduce the size of an ontology, but this is not an end in itself because it introduces the paradox that the optimum module size is 0; Section 4.3.1 discusses this in more detail.

Ontology modularization can be split into two distinct tasks: ontology partitioning and ontology module extraction. Ontology partitioning divides an ontology into a set of subsets with each subset being termed a partition, see Section 3.3; whilst ontology module extraction extracts a subset of an ontology, an ontology module, see Section 3.4. It should be noted that ontology partitioning is not the focus of this study, but is included for completeness.

Throughout this section reference will be made to the example 'Thesis' ontology introduced in Section 2.5.

3.3 Ontology Partitioning

Ontology partitioning is the task of splitting O into a set of, not necessarily disjoint[4], modules $\mathcal{M}= \{M_1, M_2, ..., M_n\}$. The union of all the modules should be equivalent to the ontology O that was partitioned, $\{M_1 \cup M_2 \cup ... \cup M_n\} = O$. Thus, a function $partition(O)$ can be defined as follows:

[3] This definition is agnostic with respect to the ontology language used to represent the ontology, but it should be noted that the modularization techniques detailed in this section assume a description logic representation.

[4] This is in contrast to the mathematical definition of partitioning that requires partitions to be disjoint.

Definition (Ontology Partitioning Function)

$$partition(O) \rightarrow \mathcal{M} = \{\{M_1, M_2, ..., M_n\} | \{M_1 \cup M_2 \cup ... \cup M_n\} = O\}$$

3.3.1 Stuckenschmidt and Klein

Stuckenschmidt and Klein [128] present a method for automatically partitioning ontologies based on the structure of the class hierarchy. The underlying assumption of the approach is that dependencies between concepts can be derived from the structure of the ontology; as such, the ontology is represented as a weighted graph $O = \langle C, D, w \rangle$ where nodes (C) represent concepts and edges (D) represent links between concepts that represent different kinds of dependencies that can be weighted (w) according to the strength of the dependency. The dependencies are based on the representation language, but include features such as subclass relations between concepts. The partitioning method can be broken down into three steps, the first step is to extract the dependency graph, which is a subgraph of the original ontology. The second step is to calculate the strength of the dependencies between the concepts; this is done by calculating the proportional strength (p_{ij}) between two concepts(c_i and c_j) where a_{ij} is the weight preassigned to the link:

$$p_{ij} = \frac{a_{ij} + a_{ji}}{\sum_k a_{ik} + a_{ki}}$$

The intuition here being that the fewer the individual social contacts then the more important they are. The assumption being that concepts in a module should be more interconnected, so being able to identify those with fewer connections means that they can be separated out. To determine the partitions, the last step, a network analysis algorithm is applied; this induces a connected subgraph where the vertices inside the subgraph are more strongly related among themselves than with the neighboring vertices. The size of these subgraphs is given as a parameter to the partitioning algorithm. The limiting factors of this approach are the dependency measures and the termination point of the partitioning process. The dependency measures are a limiting factor because they are agnostic with respect to the semantics of the ontology and the context for which the ontology partitions are being generated for. Stuckenschmidt and Klein [128] suggest that context-aware or semantics-based measures could replace the existing dependency measure. In addition, the termination point is a limiting factor because it is somewhat arbitrary and it could take several runs of the algorithm to generate partitions that are satisfactory, but whilst the termination point is arbitrary it is plausible due to the fact that one motivation for performing partitioning is that the ontology is too large in itself.

3.3.2 Cuenca Grau *et al.*

Cuenca Grau *et al.* [25] address the problem of partitioning an OWL ontology ($\mathcal{O}$) into an $\mathcal{E}$-connection (Σ). $\mathcal{E}$-connections [83] allow the interpretation domains of n combined systems (here each system can be seen as a description logic knowledge base) to be disjoint, where these domains are connected by means of n-ary 'link relations'. These 'link relations' allow connections to be drawn between the different partitions, as such reasoning can be done on each partition individually or reasoning can be done over a combination of linked partitions. Kutz *et al.* [83] show that Distributed Description Logics [9] are a special case of $\mathcal{E}$-connections linking a finite number of DL knowledge bases.

The partitions produced by Cuenca Grau*et al.* [25] are both *structurally* ($\Sigma \sim \mathcal{O}$) and *semantically* ($\Sigma \approx \mathcal{O}$) compatible. Structural compatibility ensures that no entities or axioms are added, removed or altered during partitioning; that is every axiom that exists in the $\mathcal{E}$-connection also exists in the ontology. Semantic compatibility is a desirable relation between the input and the output of a partitioning process as it ensures that the interpretation of the ontology with partitions is equivalent to the interpretation of the ontology without partitions. Thus, it ensures that equivalent KBs have exactly the same set of compatible $\mathcal{E}$-connections. Thus, consistency is preserved and existing subsumptions are ensured in the $\mathcal{E}$-connection. The algorithm to produce the partitions, ensuring structural and semantic compatibility, identifies the properties (roles) in $\mathcal{O}$ that can link $\mathcal{O}$ to Σ_i or Σ_i to $\mathcal{O}$; to ensure this is maximal roles are transformed into links whenever possible.

The two partitioning approaches detailed in the above approach the problem from totally different perspectives. Stuckenschmidt and Klein [128] do not consider the semantics of the ontology, so their approach is applicable across different ontology languages; whereas Cuenca Grau *et al.* [25] consider fully the Description Logics giving their approach guarantees about not altering the definitions contained in the partitions with respect to the original ontology. However, Cuenca Grau *et al.*'s approach is a one-shot approach, it is possible that an ontology can not be partitioned using $\mathcal{E}$-connections; but Stuckenschmidt and Klein's technique allows for the parameters to be tweaked in order to obtain a set of partitions that meet ones requirements.

3.4 Ontology Module Extraction

In contrast to ontology partitioning, ontology module extraction is the task of extracting a module M from an ontology O that covers a specified signature $Sig(M)$, such that $Sig(M) \subseteq Sig(O)$. M is the relevant part of O that is said to cover the elements defined by $Sig(M)$, as such $M \subseteq O$. M is an ontology itself and it is possible that further modules could be extracted from it. Thus, a function $extract(O, Sig(M))$ can

be defined as follows:

Definition (Ontology Module Extraction Function)

$$extract(O, Sig(M)) \rightarrow \{M | M \subseteq O\}$$

The techniques for ontology module extraction in the literature can be further subdivided into two distinct groups: *traversal approaches* and *logical approaches*. Traversal approaches [27, 33, 103, 115] represent the extraction as a graph traversal, with the module being defined by the conditional traversal of the graph, which implicitly considers the ontological semantics. Logical approaches [24, 82] focus on maintaining the logical properties of coverage and minimality, these approaches explicitly consider the ontological semantics when extracting an ontology module.

3.4.1 Traversal Based Extraction

All of the following methods for ontology module extraction can be considered as traversal based extraction techniques. Each represent the ontology as a graph and the ontology module is defined as a conditional traversal over this graph; Section 2.4 details how a description logic ontology can be a graph. It should be noted that traversal techniques could extract an ontology module that is not a proper subset of the axioms defined in the ontology.

d'Aquin *et al.*

d'Aquin *et al.* [27] describe an ontology module extraction technique that is integrated into the larger process of *knowledge selection*. Knowledge selection aims to dynamically retrieve the relevant components from online ontologies to automatically annotate a web page that is currently being viewed in a web browser. Knowledge selection comprises three steps:

1. *Selection of relevant ontologies.* Given a set of terms that the ontology is required to cover the appropriate ontologies can be identified.

2. *Modularization of selected ontologies.* Using the ontologies from the previous step an ontology modularization technique is used to obtain modules that are considered relevant for the current task.

3. *Merging of the relevant ontology modules.* This step merges the ontology modules obtained previously; d'Aquin *et al.*do not provide details on how this should be done.

The focus here is on 2. The principle used for the extraction of an ontology module is to include all the elements that participate in the definition, either directly or indirectly, of the already included entities. That is if a concept or property is involved in a definition not already included in the module then that definition is included. This is similar to Seidenberg and Rector [115]. However, there are two distinct characteristics to this approach:

- **Inferences** are used during the extraction rather than just taking the input as it is allowing the method to extract both implicit and explicit knowledge. For example, the transitivity of the *subClassOf* edge allows new subclass relations to be inferred in the input ontology or definitions of superconcepts can be inherited by subconcepts being included in the module. It should be noted that other techniques, such as Doran *et al.* [33], assume that the input ontology is the inferred model, i.e. that all inferences are made *a priori* to extraction, as the constraints of the application they consider does not place time constraints upon the extraction process.

- **Shortcuts** are taken in the class hierarchy by including only the named classes that are the most specific common super-classes of the included classes. This is done by restricting the possible values of the Least Common Subsumer (LCS) algorithm [20] to the classes in the ontology; the LCS being the most specific concept that subsumes two other concepts already in the module.

In addition, it is possible that instances also get included into the module. An instance is included in the module when they form an enumerated class definition, when they are part of a role and the other instance involved is already in the module, or when they are instances of a concept included in the input signature.

Doran *et al.*

Doran *et al.* [33] tackle the problem of ontology module extraction from the perspective of an Ontology Engineer wishing to reuse part of an existing ontology. The approach aims to extract an ontology module about a single user supplied concept that is self-contained, concept centred and consistent; defined as follows:

- Self-contained. Ontology modules should be a self-contained subset of a parent ontology. Given a set of relations the ontology module should be transitively closed with respect to these relations. Transitive closure means that all relations in between two concepts, even if the relation identifies an intermediate concept, are included.

- Concept centred. The ontology module contains enough information to describe the start concept. Direct superclasses are considered unimportant because they

only place the start concept in context. It is assumed that the Ontology Engineer already has a context in mind for the ontology module. In addition including superclasses would increase the chances of the ontology module being equal to the whole ontology.

- Consistent. Ontology modules should be consistent. Given a consistent ontology to extract a module from, the module produced should be consistent.

Doran *et al.*'s approach is agnostic with respect to the language the ontology is represented in, but the ontology language must be able to be transformed into the *Abstract Graph Model* they present. The conditional traversal is done via two sets: one set of edges to traverse and one set of edges not to traverse; with exceptions allowed in the first iteration of the algorithm. For example, when an extracting an ontology module from an OWL DL ontology the `owl:disjointWith` edges are not traversed in the first iteration, but in subsequent iterations they are. Evidently, the labels placed on the edges can be changed to suit the ontology language.

Doran *et al.* [33] provide an ontology module extraction method that is language neutral. The *Abstract Graph Model* proposed is an edge-labeled directed graph G, given an alphabet $\sum_E$, is an ordered pair $G = (V, E)$ where:

- V is a finite set of vertices,
- $E \subseteq V \times \sum_E \times V$ is a ternary relation describing the edges (including label). (N.B. E is not symmetric which gives us direction. Therefore to properly capture the definitions of 'disjoint' and 'equivalent' two edges are required.)

Using this abstract model, it is possible to define an ontology module as $G_M = (V_M, E_M)$, where $V_M \subseteq V \wedge V_M \neq \emptyset$ and $E_M \subseteq E$. This implies that $G_M \subseteq G$.

Doran *et al.* reduce module extraction to the traversal of a graph given a starting point x such that $x \in V_G$. The only exception being that there is no need to traverse '*disjoint*' labeled edges of x in the first iteration. Thus, the module is a graph $G_M = (V_M, E_M)$ where V_M and E_M are the sets of traversed vertices and edges respectively. The minimum number of possible G_M derivable from G should be equal to the number of elements in V. This is because a single module could be generated for each concept.

Algorithm 1[5] presents the pseudo-code description of the ontology module extraction method. The complexity of the algorithm is $O(n^2)$; in this case the graph is a complete graph (each distinct pair of vertices is connected by an edge) thus requiring as many as n traversals from each node. The graph is conditionally traversed to extract the ontology module.

In the case of OWL DL in the first iteration of the extraction process the disjoint relation is not traversed, but in subsequent iterations it is. This exception is justified

[5]Implementation available at: `http://code.google.com/p/modtool/`

Algorithm 1 Module Extraction

INPUT

- A directed graph $G = (V, E)$
- s a starting vertex such that $s \in V_G$
- Excluded - a container of E not to be followed
- Visited - a container of V that have been visited
- ToVisit - a container of V to be visited.

OUTPUT

- A directed graph $G_M = (V_M, E_M)$

procedure $extractModule$(Vertex s)
if $s \notin$ Visited **then**
  insert s into Visited
  create container $X = \{e \in E | s \times \sum_E \times v\}$
  while X is not empty **do**
    y =first element of X
    if $y \notin$ Excluded **then**
      $y \cup E_M$
      insert r such that $y = s \times \sum_E \times r$ into ToVisit
    end if
    if ToVisit is not empty **then**
      t =first element of ToVisit
      remove t from ToVisit
      $extractModule(t)$
    else
      output G_M
    end if
  end while
end if

because the user explicitly chooses the concept that the process will start on. If this concept has disjoint sibling concepts the assumption is made that the user is not interested in these concepts. The reason behind this assumption is that disjointedness requires that the concepts have no instances in common. If the user had wished to include the disjoint concepts they should have started the process on their common superclass.

An interesting result of applying this algorithm to generate an ontology module, is that the ontology module produced will be transitively closed with respect to the relations that are traversed. The only caveat is that the ontology being used to obtain the module must be transitively closed with respect to these relations in order to guarantee that the module will also have transitive closure.

It is important to note that there is no upward navigation of the subclass hierarchy from the concept the process was started on. Again this is justified because the user chooses the starting concept. Furthermore, allowing upward navigation of the subclass hierarchy would substantially increase the chance of extracting a module that is equal to the whole ontology. This choice does not seem to impact upon the quality of the modules in the task based evaluation considered in Section 4.5.

The abstract graph model means that the module extraction process is independent of the language. For example, the alphabet for OWL-DL is

$$\sum_{E} = \{subClassOf, disjointWith, equivalentTo, subPropertyOf, property\}$$

and for RDFS it is

$$\sum_{E} = \{subClassOf, subPropertyOf, property\}$$

Notionally these labels correspond to the primitives of OWL-DL apart from '*property*'. If an edge is labelled '*property*' then it means the starting vertex is the domain and the ending vertex is the range.

A Walkthrough Example. Section 2.5 presents a simple ontology about Theses, see Figure 3.1 From this ontology we shall extract an ontology module about 'PostGradStudent', this is the starting concept.

Iteration 1 'PostGradStudent' is added to Visited. 'PostGradStudent' is disjoint with 'UnderGradStudent' so 'UnderGradStudent' is not added to the container ToVisit. 'PostGradStudent' has two subclasses, 'MastersStudent' and 'PhDStudent', these are added to ToVisit. 'PostGradStudent' has no more edges to traverse and is removed from ToVisit; the extraction now continues with 'MastersStudent' as the concept of focus.

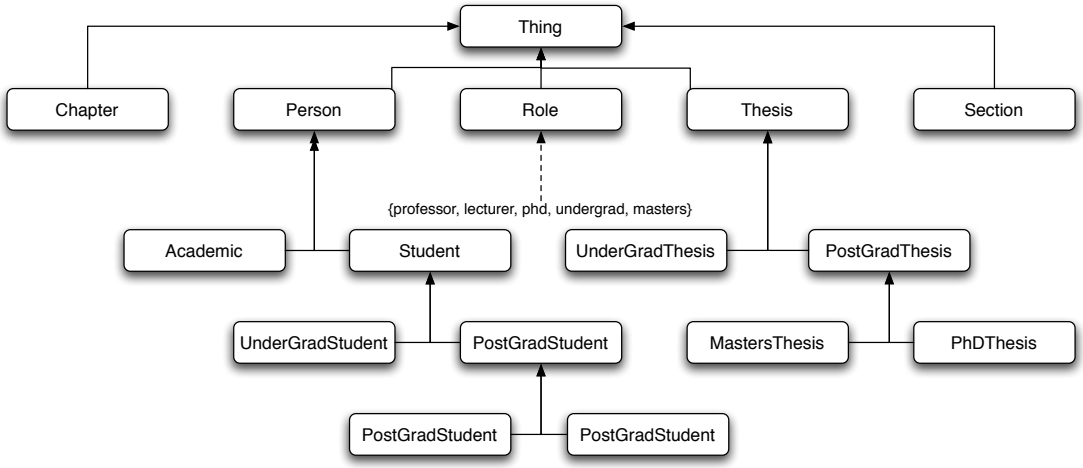

Figure 3.1: Graphical representation of the Thesis ontology taxonomy.

Iteration 2 'MastersStudent' is added to Visited. 'MastersStudent' has no more edges to traverse and is removed from ToVisit; the extraction now continues with 'PhD-Student' as the concept of focus.

Iteration 3 'PhDStudent' is added to Visited. 'PhDStudent' has no more edges to traverse and is removed from ToVisit. ToVisit is now empty; the extraction process ends and the ontology module is outputted.

Noy and Musen

Noy and Musen [103] define the notion of *traversal view extraction*, which defines an *ontology view*. An ontology view is analogous to an ontology module because it encapsulates a subset of the original ontology. As such, this technique can be considered as an ontology module extraction technique.

Starting from one class of the ontology being considered, relations from this class are recursively traversed to include the related entities. The relations to be traversed are selected by the user and for each relation selected a depth of traversal is assigned, a *traversal directive*, when this depth is reached the algorithm will stop 'traversing' this relation. A traversal directive is defined as a pair $\langle C, \mathcal{PT} \rangle$, where C is a concept in the ontology, the start point for the traversal, and $\mathcal{PT}$ is a set of property directives. A property directive is a pair $\langle P, n \rangle$, where P is a property in the ontology, the property to be traversed, and n is a non-negative integer specifying the depth of the traversal.

Thus, a *traversal view specification* is defined as a set of traversal directives. The result of a traversal view specification is a *traversal view* which is the union of all the results of the traversal directives in the traversal view specification. A traversal view, therefore, contains all classes and instances encountered on the path of the specified traversal.

This technique is interactive, and incorporated into PROMPT [101], which is a plugin for the Protege ontology editor that allows the user to manage multiple ontologies by allowing them to compare versions, merge them and extract ontology modules. This is a very flexible approach and allows an Ontology Engineer to iteratively construct the ontology module that they require by extending the current 'view', but it can require the Ontology Engineer to have a deep understanding of the ontology that is being used in order to define the appropriate traversal directives.

Seidenberg and Rector

Seidenberg and Rector [115] developed a technique specifically for extracting an ontology module from the GALEN[6,7] medical ontology. However, the core of the technique is generic and can be applied to other ontologies. The technique takes one or more classes of the ontology as input, the $Sig(M)$, and anything that participates, even indirectly, to the definition of an included class is added to the ontology module too. The algorithm can be broken down as follows, assume we have a $Sig(M) = \{A\}$. Firstly the hierarchy is upwardly traversed (analogous to Upper Cotopy defined in [93], which calculates the set of super-concepts of a concept), so all of the A's superclasses are included. Next the hierarchy is downwardly traversed so that all the A's subclasses are included. It should be noted that the sibling classes of A are not included, they could be included by explicitly adding them to the $Sig(M)$. The restrictions, intersection, union and equivalent classes of the already included classes can now be added to the module. Lastly, properties across the hierarchy from the previously included classes are traversed; the target of these links are only upwardly traversed.

The traversal via properties is terminated when a *boundary class* is reached. A boundary is class is reached when a certain recursion depth is reached in the property traversal resulting in all links from this class to be removed. It is important to note that the named superclass of a boundary class must be included in order to retain the correct hierarchical structure.

Seidenberg and Rector [115] tailored their approach to the GALEN ontology. As such, there are certain features of this technique which may only be applicable to GALEN; such as property filtering. The properties are filtered by removing all restrictions in which they occur. The result of property filtering can lead to class definitions becoming equivalent, whilst this is not incorrect it does introduce unnecessary definitions as they are turned into primitive concepts by the algorithm.

[6] `http://www.co-ode.org/galen/`
[7] `http://www.opengalen.org/index.html`

3.4.2 Logical Based Extraction

In contrast to the traversal based ontology module extraction techniques the logical based extraction techniques are based on the notion of conservative extension [90]. An ontology module is a subset of the ontology is was extracted from, this ontology is a conservative extension if the entailments regarding the ontology module are captured totally within its signature. More formally Lutz *et al.* [90]present the following definition:

Definition (Conservative Extension) Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes formulated in a DL $\mathcal{L}$, and let $\Gamma \subseteq sig(\mathcal{T}_1)$ be a signature. Then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a Γ-conservative extension of $\mathcal{T}_1$ if for all $C_1, C_2 \in \mathcal{L}(\Gamma)$, we have $\mathcal{T}_1 \models C_1 \sqsubseteq C_2$ iff $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$.

Thus, all the entailments regarding the signature of the ontology module are the same as if you take the union of the ontology module and the ontology it was taken from. In essence, the ontology contains nothing more about the signature of the ontology module. In this sense, the ontology module is minimal. Unfortunately, Lutz *et al.* [90] also show that deciding if an O is a conservative extension is undecidable for OWL DL. However, Konev *et al.* [82] have developed an algorithm, MEX, for extracting conservative extensions from acyclic terminologies formulated in $\mathcal{ALCI}$ or $\mathcal{ELI}$. Whilst these restrictions limit this approaches use, it can be successfully used on large real world ontologies such as SNOMED CT.

Grau *et al.* [24] overcome the limitations of conservative extensions for more expressive description logics by considering a relaxation of the minimality constraint; they term these modules as locality-based modules. *Coverage* and *safety* are the properties that locality-based modules can guarantee, but this is done at the expense of minimality which is in addition guaranteed by conservative extensions. Coverage and safety [23] are defined in terms of a module being imported by a local ontology ($\mathcal{L}$) as follows:

Coverage Extract everything the ontology says about the specified terms. The module O' covers the ontology O for terms from some signature X if for all classes A and B built from terms in X, such that if $\mathcal{L} \cup O \models A \sqsubseteq B$ then $\mathcal{L} \cup O' \models A \sqsubseteq B$.

Safety The meaning of the extracted terms is not changed. $\mathcal{L}$ uses the terms from X safely if for all classes A and B built from terms in X, such that if $\mathcal{L} \cup O' \models A \sqsubseteq B$ then $O' \models A \sqsubseteq B$.

Two different variants of locality are described by Grau *et al.* [56]. Syntactic locality can be computed in polynomial time, but semantic locality is PSPACE-complete. Syntactic locality is computed based on the syntactic structure of the axiom whereas semantic locality is computed based on the interpretation ($\mathcal{I}$) of the axiom. Jimènez-Ruiz *et al.* [79] propose two different locality conditions for extracting ontology modules.

$\bot$-locality (upper module) extracts a module that is suitable for refinement; it should contain all the super-concepts of the signature. Whereas, $\top$-locality (lower module) extracts a modules that is suitable for generalisation; it should contain all the sub-concepts of the signature. For example, consider a TBox with the following three axioms:

$$\texttt{Periodical} \sqsubseteq \texttt{Publication}$$

$$\texttt{Newspaper} \sqsubseteq \texttt{Periodical}$$

$$\texttt{Journal} \sqsubseteq \texttt{Periodical}$$

Now consider that we want to extract a module about `Periodical` ($Sig(\texttt{Periodical})$), a $\bot$-locality module would include only the first axiom and a $\top$-locality module would include only the second and third axioms.

The issue concerning the syntactic locality is that syntactically different but semantically equivalent axioms can be treated differently. For example, Borgida and Giunchiglia [8] raise this issue of the syntactic approximation via the following example; consider the two sets of axioms $\{A \sqsubseteq (B \sqcap C)\}$ and $\{A \sqsubseteq B, A \sqsubseteq C\}$, these axioms are semantically equivalent but the syntactic difference will effect the extraction process; this would also pose a problem to the traversal based ontology module extraction techniques. The syntactic locality also can not handle tautologies, but this is unlikely to affect real world applications as ontologies with tautologies would be considered badly engineered.

3.5 Classifying Ontology Module Extraction Techniques

As a diverse range of techniques exist for ontology module extraction, it is useful to provide a comparative summary to classify the techniques. First, an anecdotal comparison is made amongst the techniques, based on the example ontology presented in Section 2.5. However, as the starting assumptions made by the traversal and logical approaches are diametrically opposed, that is the starting assumptions made by both are different , it would be misleading to compare their features. For example, as traversal techniques are not designed to provide minimality it is not possible to state such assertions or guarantees about these techniques. As such, a comparison is made within each category, traversal and logical, before providing some general remarks in summary.

3.5.1 Anecdotal Comparison

For the following five techniques (described in Section 3.4) we extracted a module using the signature `PhDStudent` ($Sig(\{\texttt{PhDStudent}\})$): Doran *et al.* [33], d'Aquin *et al.* [27], Seidenberg and Rector [115] and the two variants of the technique proposed by Cuenca Grau *et al.* [24].

		Traversal			Logical	
Thesis Ontology		Doran	d'Aquin	Seidenberg	Cuenca U	Cuenca L
Classes	Chapter				✓	✓
	Person	✓		✓	✓	✓
	Academic	✓			✓	✓
	Student	✓		✓	✓	✓
	PostGradStudent	✓		✓	✓	✓
	MastersStudent	✓			✓	✓
	PhDStudent	✓	✓	✓	✓	✓
	UnderGradStudent	✓			✓	✓
	Role	✓		✓	✓	✓
	Section					
	Thesis				✓	✓
	PostGradThesis				✓	✓
	MastersThesis				✓	✓
	PhDThesis				✓	✓
	UnderGradThesis				✓	✓
Properties	authorOf				✓	✓
	chapterOf				✓	✓
	hasAuthor				✓	✓
	hasChapter				✓	✓
	hasRole	✓		✓	✓	✓
	hasSection					
	hasSubSection					
	hasSupervisor				✓	✓
	hasFirstSupervisor				✓	✓
	hasSecondSupervisor				✓	✓
	supervisorOf				✓	✓

Table 3.1: Anecdotal comparison of module extraction techniques using the Thesis ontology.

Table 3.1 shows the classes and properties that were included in the `PhDStudent` modules for the different techniques. It should be noted that the Thesis ontology is small and highly interconnected which biases it slightly against the Cuenca Grau variants. However, Table 3.1 still shows that d'Aquin aims to be as small as possible by including only one class due to the definitions of the super classes being moved to `PhDStudent`. Furthermore, Table 3.1 also shows the contrasting traversal conditions of Doran and Seidenberg because Doran includes classes that are linked to `PhDStudent` but Seidenberg only includes the super classes of `PhDStudent`.

3.5.2 Traversal Based Extraction Feature Comparison

	Interactive	Traversal Direction	Property Filtering	Use Reasoner
Whole Ontology	✗	N/A	✗	✗
d'Aquin *et al.*	✗	Up & Down	✗	✓
Doran *et al.*	✗	Down	✗	✗
Noy and Musen	✓	Up & Down	✗	✗
Seidenberg and Rector	✗	Up & Down	✓	✗

Table 3.2: Comparison of features for traversal based ontology module extraction.

Table 3.2 compares the following features of the traversal based extraction techniques:

- *Interactive.* Does the traversal based extraction technique require interaction from the user beyond specifying the signature of the ontology module?
- *Traversal Direction.* Does the traversal based extraction technique go up (including super concepts of the signature) or down (including sub concepts of the signature) the hierarchy?
- *Property Filtering.* Does the traversal based extraction technique carry out property filtering? That is removing the definitions in which the properties occur.
- *Use Reasoner.* Does the traversal based extraction technique use a description logic reasoner?

The techniques operate in a broadly similar way with a few important distinctions. d'Aquin is the only one who uses a reasoner. This could be major drawback if an ontology is encountered that is particularly difficult to reason over; for example one containing numerous GCIs. Seidenberg and Rector's is the only technique to do property filtering which is due to the technique being be designed to work over GALEN, where this feature was desirable. Perhaps the most important distinction to note here is that Doran is the only technique that does not consider upwards traversal. As this technique was intended for use by an Ontology Engineer in the context of ontology reuse where the assumption is that the Engineer knows the 'context' to place the module in.

3.5.3 Logical Based Extraction Feature Comparison

	Coverage	Minimality	DL Expressivity	Tractable
Whole Ontology	✓	✗	Any	✓
Locality Based	✓	✗	OWL DL (SHOIN(D))	✓
MEX	✓	✓	EL+	✓
Conservative Extension	✓	✓	Any	✗

Table 3.3: Comparison of features for logical based ontology module extraction.

Table 3.3 compares the following features of the logical based extraction techniques:

- *Coverage.* Do the module produced by the logical based extraction technique guarantee coverage? That is the module covers everything about the specified terms that is in the ontology.
- *Minimality.* Are the modules produced by the logical based extraction technique minimal? That is the module contains only that which is necessary.
- *DL Expressivity.* What is the maximum expressivity level that the logical based extraction technique will operate?
- *Tractable.* Is the logical based extraction technique tractable?

The important contrast to make when considering the logical techniques is the trade-off between the properties of minimality, coverage and tractability. If one wishes to preserve both properties then you must restrict the expressivity of the DL you use; if a higher expressivity is required then the minimality constraint must be removed to make the problem tractable.

3.5.4 Summary of Classification

Whilst it is possible to classify the techniques based on certain inherent properties, such as safety, the utility of such a classification is questionable due to the problem of deciding which technique to use for which task. Of course, the choice of task places different constraints on which technique is applicable.

However, this classification does allow one to draw immediate comparisons between the techniques within each area. Perhaps, more difficult is drawing comparisons between the logical and traversal approaches. Their different starting positions mean it is hard to draw fair comparisons. Whilst the logical techniques guarantee certain properties the proponents of the traversal techniques may question the utility of such properties for the scenario their technique was designed for. For example, d'Aquin's scenario of knowledge selection has a high tolerance for inaccuracies; but using conservative extensions for maintenance tasks of SNOMED has a low tolerance for inaccuracy.

3.6 Common Frameworks for Ontology Modularization

There are a plethora of techniques for carrying out ontology module extraction, see Section 3.4. All of these techniques are designed for different applications and contain different assumptions about the problem as whole. Thus, there is a need to draw this work together within a common framework; the key advantage of a common framework is the ability to select, adapt and combine the different approaches. This would greatly facilitate an objective evaluation and comparison of the different ontology modularization techniques; Chapter 4 presents an in-depth evaluation of ontology modularization, including a metric and task based evaluation.

Also, the development of new techniques is made easier, since common technical issues, such as whether reasoning is used or not, are already tackled in the framework and little effort is required by the developer.

3.6.1 Tell/Ask Interface

Borgida and Giunchiglia [8] present a Tell/Ask interface for 'importing knowledge', that is reusing knowledge from existing ontologies. This work can be viewed as a common framework for ontology module extraction. Inspired by Levesque's functional approach to knowledge representation [87], which allows the user to interact with a knowledge base via Tell and Ask operators. You 'Tell' the TBox facts and then 'Ask' the TBox queries about the facts it contains. Borgida and Giunchiglia cast this to Description Logics. In this context, Tell operations allow a TBox to be built and Ask operations allow the knowledge base to be interrogated.

Therefore, this work can be cast to an approach for a common framework for ontology module extraction. A series of Ask operations would simulate the extraction approach. The answers of these operations would form Tell operations to construct a new TBox, the ontology module. This allows for a flexible framework, but adding new operations is likely to be costly as each will have to be implemented separately.

3.6.2 Graph Transformations

d'Aquin *et al.* [26] suggest graph transformations as a possible common framework. Firstly, d'Aquin *et al.* [26] present a way to transform ontologies into a directed attributed graphs, which is a directed graph where attributes, in terms of types and values can be added to the nodes and edges. Thus, a node representing a concept C can have **type** *Class* and value **name** = C. For example, the axiom

$$PersonWithDogAndCat \equiv Person \sqcap \exists hasPet.Dog \sqcap \exists hasPet.Cat$$

would be transformed into the graph shown in Figure 3.2

The existing techniques ([27], [33], [103] and [115]) are then represented as a series of graph transformations. A graph transformation takes one graph as a pre-condition

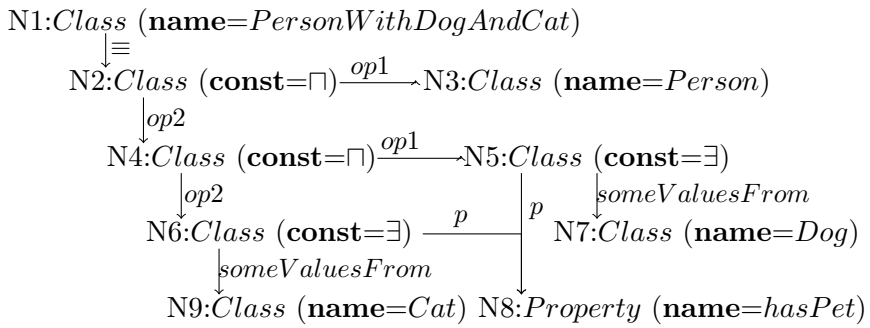

Figure 3.2: Attributed graph representation of the expression $PersonWithDogAndCat \equiv Person \sqcap \exists hasPet.Dog \sqcap \exists hasPet.Cat$

and one graph as a post-condition. For example, Figure 3.3 shows the graph transformation for the downwards traversal of the subclass hierarchy. This allows for an easily extensible framework, but graph transformations are not widely applied in the Ontology Engineering and Semantic Web communities, thus there could be a considerable user overhead in learning how they operate.

Premiss	**Transformation**
$\mathtt{C_1}$:*Class* (**inc**)	$\mathtt{C_1}$:*Class* (**inc**)
⊑	⊑ (**inc**)
$\mathtt{C_2}$:*Class*	$\mathtt{C_2}$:*Class* (**inc**)

Figure 3.3: An example graph transformation for downwards traversal of a subclass hierarchy.

3.6.3 SPARQL Based Extraction

The work by Borgida and Giunchiglia [8] and d'Aquin *et al.* [26] require the user to become familiar with non-standard formalisms, but the work by Doran *et al.* [32] uses the W3C standards of RDF and SPARQL as the basis for a common framework for ontology module extraction. All OWL ontologies can be represented as an RDF graph (see Section 2.4) and SPARQL is a query language for RDF. Thus, Doran *et al.* [32] present SOMET which shows that it is possible to cast the traversal based ontology module extraction approaches as a series of SPARQL queries upon an RDF graph. Thus, the selection, adaptation and combinations of the techniques are manipulations of SPARQL queries.

The SOMET framework (see Figure 3.4) already includes the SPARQL representations of the techniques presented in [27], [33], [103] and [115]. For example, some of the queries required for Doran *et al.*'s [33] technique, where ?c is the current concept

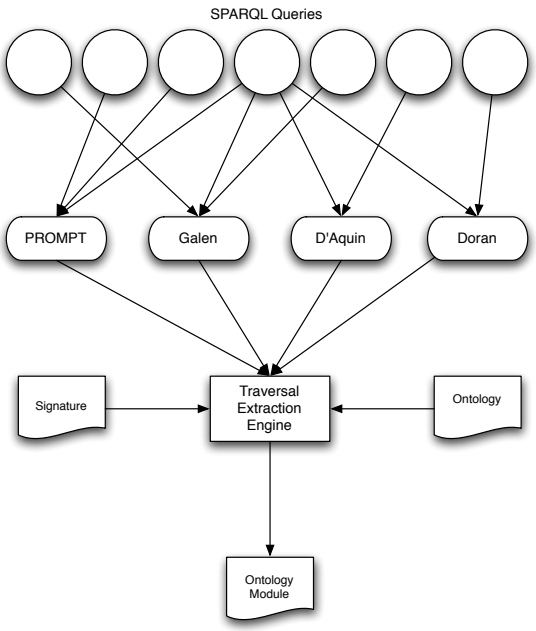

Figure 3.4: The SOMET framework. [32]

of focus, are:

- `DESCRIBE` $?c$

  Describes the current resource. That is the list of the statements in which the $?c.$ appears as subject, plus the closure computed from any blank nodes involved.

- `CONSTRUCT` $\{?y$ `rdfs : domain` $?c.\}$`WHERE`$\{?y$ `rdfs : domain` $?c.\}$

  Returns all the $?y$ where $?c.$ is the domain of a property.

- `DESCRIBE` $?y$ `WHERE` $\{?y$ `rdfs : subClassOf` $?c.\}$

  Returns all the subclasses of $?c.$

- `CONSTRUCT` $\{?y$ `owl : equivalentClass` $?c.\}$ `WHERE` $\{?y$ `owl : equivalentClass` $?c.\}$

  Returns all the $?y$ where $?y$ is an equivalent class to $?c.$

Furthermore, the framework allows these queries to be modified or new ones to be added. Thus, the framework is fairly flexible and allows for experimentation in a formalism that is standard and utilised within the Semantic Web community.

3.7 Modular Ontology

All of the work considered so far has considered taking a monolithic ontology and producing partitions or modules. The alternative to this is to design ontologies in a modular way from scratch, this is analogous to the notion of modularity in Software

Engineering. There has been some work in this area by Euzenat *et al.* [43] and Ensan and Du *et al.* [40, 41]; both approaches use the notion of interface for constructing a modular ontology.

Through the use of ontology module interfaces and ontology alignments (see Section 6.2.4) Euzenat *et al.* [43] are able to construct a modular ontology. An ontology module imports a set of interfaces from other ontology modules and a set of ontology alignments connects these interfaces to the local definitions; the ontology module should also declare what terms are in its export interface. This specification allows for the flexible composition of ontology modules to create a modular ontology.

Ensan and Du *et al.* [40, 41] define an interface as a triple: $\langle C_N, R_N, \mathcal{T} \rangle$, where $\mathcal{T}$ is the TBox of the interface and C_N and R_N are sets of concept and role names used in $\mathcal{T}$. A module then declares which interfaces it exposes and uses; thus allowing a modular ontology to be defined as a set of modules, a set of interfaces and a configuration function. The configuration function which chooses one exposing module for every utilizer module-interface pair.

These approaches are comparable to the use of Distributed Description Logics (DDL) [9]. DDL defines a modular ontology as a set of ontology modules connected through bridge rules, whereby a bridge rule constructs a mapping between two concepts in two different ontology modules.

Part III

Evaluation

Chapter 4

Evaluating Ontology Modularization and Ontology Modules

'The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I found it!) but 'That's funny ...' - Isaac Asimov

Summary The aim of this chapter is to evaluate the different ontology modularization techniques in terms of their performance. First, an overview of the existing literature on ontology evaluation is given. Then suitable metrics for evaluating ontology modularization are given, including an entropy inspired metric. These metrics are then applied to evaluate the different ontology modularization techniques. Lastly, a task based evaluation is carried out on the different ontology modularization techniques. Three tasks related to query answering are considered.

4.1 Motivation

Chapter 3 outlined the various techniques currently available in the literature for performing ontology modularization; this Chapter aims to look at the issue of deciding how 'good' an ontology module is and what 'good' means. The current literature on ontology modularization does not contain a definitive notion of what a 'good' module is. Thus, there is a need to define an objective measure, such as the entropy inspired metric presented in Section 4.3.3.

Whilst a number of different approaches for modularizing ontologies have been proposed in the literature, each with different characteristics and purposes, there are few efforts aimed at providing objective measures for the evaluation of the outcome of modularization techniques [28]. This motivates the need for objective criteria for evaluating the ontology modules produced as a result of modularization. Section 4.3 introduces

three objective metrics that have been proposed for evaluating modularization; these metrics are then used to carry out an evaluation in Section 4.4.

The prevalent measure to discriminate between ontology modules is the module *size*, that is the number of named concepts and properties that compose the module [56]. Other criteria have been proposed by Schlicht and Stuckenschmidt [112] that look at the structure of the ontology modules produced and attempt to assess the trade-off between maintainability and efficiency of reasoning in distributed systems. These criteria are [112]:

- *Redundancy.* The extent to which ontology modules overlap. The inclusion of redundancy in modules improves efficiency and robustness but in contrast it requires a higher effort to maintain the modules.

- *Connectedness*: The number of edges shared by the modules generated by a modularization approach. This criterion assumes that the modules are represented as a graph, where the vertices are the axioms in the ontology, and the edges are the properties (roles) connecting two axioms with a shared symbol. Connectedness estimates the degree of independence of the set of modules generated by the modularization approach.

- *Distance*: The process of modularization can simplify the structure of the module with respect to the ontology in input. Two different distance measures, *inter-module distance* and *intra-module distance* have been defined that count the number of modules that relate to entities, and the number of relations in the shortest path from two entities in a module, respectively.

All these criteria (including size) assess a specific aspect of the module obtained that depends on the task for which modularization is carried out [28]. However, there is no measure that attempts to capture the *combined effect* and aggregates the quantitative estimates of the dimensions represented by the criteria. This is the motivation behind the introduction of the entropy inspired measure in Section 4.3.3; which aims to facilitate a comparative analysis of the modularization approaches.

However, before considering this it is important to consider the general area of ontology evaluation; after all, ontology modules are ontologies. Uschold and Gruninger [138] introduce the notion of formal competency questions to evaluate the quality of an ontology. These are used to verify that what the ontology contains is sufficiently rich to answer some questions defined by the Ontology Engineers to limit the scope of the ontology; indeed these form part of Ontology Engineering methodologies, see Section 5.2.

4.2 Ontology Evaluation

With the increased availability of ontologies the need to evaluate the suitability of an ontology becomes more pressing; we need to have a way to decide that one ontology is better than another. The aim of ontology evaluation is to move away from the subjective to the objective. With this in mind Yu, Thom and Tam [148] classify the main approaches to ontology evaluation into 3 groups, these are:

1. **Gold standard evaluation.** One ontology is deemed the benchmark, or 'gold standard', and the ontology being evaluated is compared against it. For example, Maedche and Staab [93] present a collection of similarity measures to compare one ontology with a gold standard ontology. This kind of evaluation is typically carried out when assessing ontology learning algorithms, that is algorithms that can induce an ontological structure from a set of data [92].

2. **Criteria based evaluation.** Takes the ontology being evaluated and evaluates based on the proposed criteria [52]. These criteria are generally isolated from the applications; thus, meeting the criteria may not be enough to meet the needs of the application. Yu, Thom and Tam [148] also identify 8 distinct criteria, based on the criteria proposed in the literature [58, 60, 52, 62, 66], these are:
   (a) *Clarity.* Is the ontology clear and easy to understand?
   (b) *Consistency.* Is the ontology consistent or does it contain contradictions?
   (c) *Conciseness.* Is the ontology concise or are its defintions unnecessarily obfuscated?
   (d) *Expandability.* Is the ontology easy to expand?
   (e) *Correctness.* Is the ontology correct? The ontology may be correct for its conceptualization, but it could be incorrect with respect to your conceptualization.
   (f) *Completeness.* Is the ontology complete? Whilst this might be demonstrated it is unlikely that it could be proven.
   (g) *Minimal Ontological Commitment.* Is the ontological commitment weak or a strong? Strong ontological commitments may make the ontology harder to reuse.
   (h) *Minimal Encoding Bias.* Could the definitions in the ontology be easily translated to another ontology language?

3. **Task-based evaluation.** The ontology being evaluated has its competency checked in completing tasks. Here the evaluation is done in the context of the appli-

cation and the ontology's competence can be quantitatively measured. The downside being that the evaluation done for one task may not be applicable for another task.

4.2.1 Ontology Evaluation Methods

The above provides an overview of the main principles for ontology evaluation that are relevant in the context of this thesis. The following subsections detail the major ontology evaluation techniques in more detail.

OntoClean

OntoClean [66] presents a methodology based on formal ontology, the philosophical study of ontology, for the evaluation of a taxonomic structure; its focus has been on cleaning up taxonomies and it has been applied to the WordNet [95] taxonomy. The core of this methodology is based around the following four philosophical notions:

1. *Rigidity.* Based on the idea of essence. A property is essential to an individual if and only if it necessarily holds for that individual. Thus, a property is rigid (+R) if and only if it is essential to all its instances. Essential here means that the property is true in every possible world [89]. Thus, a property is non-rigid (-R) if and only if it is not essential to some of its instances, and anti-rigid if and only if it is not essential to all its instances.

2. *Unity.* Defined as an individual who is whole if and only if it is made by a set of parts unified by a relation R. A property is said to carry unity (+U) if there is a common unifying relation R such that all the instances of the property are wholes under R. A property carries anti-unity if all its instances can possibly be non-wholes.

3. *Identity.* The logical relation of numerical sameness, in which a thing stands only by itself. "Identity is related to the problem of distinguishing a specific instance of a certain class from other instances of that class by means of a characteristic property, which is unique for it (that whole instance)". [65]

4. *Dependence.* Allows us to distinguish between extrinsic and intrinsic properties based on whether they depend or not on the objects other than the one they are ascribed to. Intrinsic properties are those characterising an object and do not depend on another object; usually good to become identity conditions, for example inverse-functional properties in DL (see Section 2.3). Extrinsic properties are not inherent properties and a are usually given by some external agent.

The above are attached to concepts in the taxonomy as meta-relations to represent the behaviour of the concepts. OntoClean also contains a set of axioms that can be

used in conjunction with the meta-relations to suggest how a taxonomy can be cleaned. For example, one OntoClean axiom is that "a property carrying anti-unity has to be disjoint of a property carrying unity"; thus if you have meta-relations in contradiction to this then this is an area that should be cleaned. A comprehensive list of these axioms can be found in [65].

This method is useful for fixing an existing ontology, but it does not provide any mechanism by which different ontology modules could be compared. This makes it unsuitable for evaluating the results of different ontology modularization processes. Furthermore, if the ontologies have problems which are identified by OntoClean then it is likely that these problems will also pass into the module, this suggests the ontology requires fixing before the ontology module can be extracted.

OntoMetric

OntoMetric [133] presents a set of processes for the user to carry out to obtain the measure of how suitable an ontology is for a particular application. Five dimensions are considered in making this decision, these are:

1. *Ontology content.* This is what the ontology contains and how that contents is organised.

2. *Ontology language.* This is the language in which the ontology is encoded.

3. *Methodology followed to develop ontology.* This is the methodology followed to develop the ontology. Section 5.2 discusses several methodologies for Ontology Engineering.

4. *Software used to build the ontology.* The software tools used to develop the ontology. This includes software such as ontology editors and reasoners.

5. *Cost of using the ontology.* This considers the license of the ontology and the software needed. It also considers the hardware and software costs.

These dimensions are used to obtain the overall measure of suitability which is generated by following these processes:

1. *Specify objectives.* The developers should know the constraints of their environment, as such they should be able to rank the importance of the dimensions stated above.

2. *Build decision tree.* The root node is "select most appropriate ontology" and the first level nodes being the five dimensions stated above. Each dimension can then have different factors placed underneath it and then sub-trees of characteristics can be added. These characteristics will vary depending on the project and its constraints.

3. *Pairwise comparison matrixes.* Each set of brother nodes has a comparison matrix [110] computed. These comparisons depend on the objectives and aims identified in Process 1. This results in a weight to represent the relative importance of each criteria.

4. *Assess alternative ontologies.* For each alternative ontology its characteristics are assessed. Using the values calculated in the previous step we can ascend up the tree, building a vector as we go, until we reach a node.

5. *Combine vectors.* The vector of weights (Processs 3) is combined with the values of the alternatives (Process 4). Based on these results an appropriate choice can now be made.

This method allows a developer to justify their decision by forcing them to consider the importance of the project objectives, and to carefully study the characteristics of each ontology. However, it is a time-consuming method and requires a huge effort on the part of the developer. Considering there are numerous ontology modularization techniques it is likely that the developer would have to apply this to several ontologies making it unsustainable. Furthermore, it requires a human and cannot easily be followed by an agent of the type detailed in Section 6.2.2, whereby they are considered to be autonomous and proactive in achieving their goals.

Ontology Evaluation Framework

Gangemi *et al.* [46, 47] present a collection of metrics focusing on the structure, function and usability of an ontology the aim being to integrate the various ontology evaluation methods. They ground the various measures they present in an ontology of ontology evaluation and validation.

Numerous measures are presented by Gangemi *et al.* [46]; and here we present the most relevant to the subject of this thesis. The assumption underlying the measures being that the ontology structure is represented as a graph and that the relevant components can be retrieved from the graph as required. The measures are presented within the relevant category of structural, functional and usability.

Structural Measures These measure the structural dimension of an ontology, focusing on syntax and graph structure. Measures in this category include:

- *Measures for depth.* Depth is the cardinality of paths in a graph; here the edges are assumed to represent subclass relations.
- *Measures for breadth.* Breadth is the cardinality of levels in a graph; here the edges are assumed to represent subclass relations.

- *Measures for tangledness.* Tangledness relates to the multihierarchical nodes of a graph; here the edges are assumed to represent subclass relations. Tangledness computes the ratio of the number of nodes in the graph to the number of nodes with more than one outgoing subclass edge.
- *Measures for fan-outness.* Fan-outness is related to the dispersion of graph nodes, i.e. how much the graph spreads; here the edges are assumed to represent subclass relations.
- *Measures for density.* Density is defined as the presence of clusters of classes with many non-taxonomical relations holding between them.
- *Measures for modularity.* Modularity relates to the asserted modules of a graph whereby disjoint modules are related via one or more subclass edges.
- *Measures for logical adequacy.* Logical adequacy relates to the graph having formal semantics; for example the ratio of consistent classes to inconsistent classes.

Functional Measures These measure the functional dimension of an ontology aiming to find the extent to which an ontology mirrors a given expertise or competency.

- *O_Precision and O_Recall.* Analogous to the traditional precision and recall measures [140] that allow one to measure how well a document retrieval task performs based on the ratio of relevant and correct documents returned (see Section 4.3.2 for the formulation). The models of an ontology are considered rather than considering documents, so precision and recall become measures over the ontology models and the intended ontology models.
- *O_Accuracy.* This tries to measure the fitness of an intended conceptualization by mapping states of affairs, which are intended conceptualizations even those not possible due to a mismatch between cognitive and formal semantics, to possible worlds.

Usability Measures These measure the usability dimension of an ontology aiming to understand the relation between users and ontologies. This is done along three levels:

1. *Recognition.* This requires the ontology to have an adequate set of annotations to allow the user to access in an easy way information on using the ontology in an effective way. Thus, recognition requires having a properly documented ontology to ensure effective access. Therefore, the ontology requires the proper annotations. The assumption being that providing such information makes the ontology more usable.

2. *Efficiency.* This concerns enabling users to achieve their goals in an efficient manner. This includes annotations about the organisation, the commercial and the development. Having adequate annotations in these areas facilitates efficient use of the ontology within an organisation.

3. *Interfacing.* This concerns the problem of constructing a user interface to the ontology. For the purpose of ontology evaluation the ontology should include 'interface' annotations. For example (taken from [47]), a contract negotiation ontology might contain annotations to allow an implementation of a visual contract modelling language.

Whilst the framework provides numerous metrics for evaluating ontologies it does not provide a method for understanding which metrics are suitable in different situations. Furthermore, the majority of the metrics only consider the taxonomic structure and do not exploit the richer semantics offered by Description Logics (see Section 2.3).

4.3 Metrics for Evaluating Module Extraction

Section 4.2 provides an introduction to the general area of ontology evaluation. Many of the existing ontology evaluation approaches have a sizable cost in terms of time and effort. They require time to follow through the methodology and to properly, and accurately, interpret the results which are sometimes subjective. Therefore, this Section introduces and discusses the metrics for evaluating ontology modularization that have been proposed in the literature. These measures have a low cost and try to be as objective as possible.

It seems that ontology modularization requires a quick objective way to decide how 'good' a module is because there are numerous techniques for generating them. An Ontology Engineer needs to decide which module is best and the existing techniques for ontology evaluation are rather time-consuming.

4.3.1 Size

Nearly all evaluations carried out on ontology modularization techniques, see [33, 26, 115, 24, 127], consider size as a metric in their evaluations. Size is the number of entities, named classes and properties, in an ontology. Thus, the size of an ontology, O, can be calculated as follow:

$$size(O) = |Sig(O)| \tag{4.1}$$

It is possible that size may also be considered as just the number of named classes in an ontology. One important thing to note about the size metric is that it does not take account of the number of unnamed classes in an ontology; depending on how the

ontology is constructed this could be of great importance, for example if the ontology contains a few named classes but many class restrictions.

The Paradox Of Size

The aim of modularization in general is to reduce the size of an ontology, but this is not an end in itself because it introduces the obvious paradox that the optimum module size is 0 [32]. Whilst an ontology module of size 0 is highly reusable and the effort required to reuse it is negligible, it is evident that it is also fairly useless. This inverse relation between reusability and usability is noted by Gomez-Perez, Fernandez-Lopez and Corcho [54]. They note that whilst general, upper-level ontologies are highly reusable and applicable to different applications they are unlikely to prove useful, for a given application, without modification; whereas, application level ontologies are very useful for specific cases and this specificity decreases its reusablity.

Therefore size must be traded off with some other metric. The following will introduce two metrics, precision and recall; and entropy, that could be used in a trade off with size.

4.3.2 Precision & Recall

Doran *et al.* [33] adapt the precision and recall metrics used by Dellschaft & Staab [30], for evaluating ontology learning, to evaluate module extraction. These metrics are based on those used in Information Retrieval [140].

In information retrieval precision and recall are defined as:

$$\texttt{Precision}(p) = \frac{|\{\texttt{relevant documents}\} \cap \{\texttt{retrieved documents}\}|}{|\{\texttt{retrieved documents}\}|} \tag{4.2}$$

$$\texttt{Recall}(r) = \frac{|\{\texttt{relevant documents}\} \cap \{\texttt{retrieved documents}\}|}{|\{\texttt{relevant documents}\}|} \tag{4.3}$$

Precision tells us the proportion of documents retrieved that are relevant, i.e., what proportion of documents returned were correct, and recall tells us the proportion of documents that are relevant are actually retrieved, i.e., how many correct documents were retrieved. The harmonic mean of precision and recall can then be calculated to give the fMeasure as follows:

$$\texttt{fMeasure} = \frac{2(p \times r)}{(p + r)} \tag{4.4}$$

These metrics are adopted by Dellschaft & Staab [30] for performing gold standard evaluations of ontology learning where they use precision and recall for evaluating taxonomies. Therefore, the taxonomic precision of two concepts, $c_1 \in O_C$ and $c_2 \in O_R$, is defined as:

$$tp_{ce}(c_1, c_2, O_C, O_R) := \frac{|ce(c_1, O_C) \cap ce(c_2, O_R)|}{|ce(c_1, O_C)|} \tag{4.5}$$

The characteristic extract function ($ce()$) needed for the purpose of this thesis is the semantic cotopy(sc) [91][1], the set of all a given concepts super- and subconcepts, which given the concept $c \in C$, the set of all concepts, and the ontology O is defined as:

$$sc(c, O) := \{c_i | c_i \in C \wedge (c_i \leq c \vee c \leq c_i)\} \tag{4.6}$$

Doran *et al.* [33] adopt the Dellschaft & Staab metric for evaluating ontology modules and define what precision and recall mean in this context.

- Precision (p_M). All the taxonomical relations that are in the module are also in the parent ontology. Thus, given Equation 4.5 and 4.6, this can be expressed as:

$$p_M(c_1, c_2, O_M, O) := \frac{|sc(c_1, O_M) \cap sc(c_2, O)|}{|sc(c_1, O_M)|} \tag{4.7}$$

such that O_M is the module and O is the ontology it was extracted from and $c_1 \in O_M = c_1 \in O$.

- Recall (r_M). Everything that is in the parent ontology is in the module. This can be expressed as:

$$r_M(c_1, c_2, O_M, O) := \frac{|sc(c_1, O_M) \cap sc(c_2, O)|}{|sc(c_2, O)|} \tag{4.8}$$

Recall objectively measures how much of the original ontology is retained in the module, as such it could be seen as some measure of competence. Precision ensures that all the taxonomic relations that are in the module were in the ontology it was extracted from; i.e. no new taxonomic relations have been introduced.

4.3.3 Entropy Inspired Metric

The notion of entropy was been applied to information theory by Shannon [117] in order to provide a quantitative measure of the information contained in a message. Shannon defines entropy as a measure of the average information content the recipient is missing when they do not know the value of a random variable, that is the measure of uncertainty associated with the random variable, calculated as:

$$H(X) = -\sum_{i}^{n} p(x_i) \log p(x_i) \tag{4.9}$$

where $p(x_i) = Pr(X = x_i)$ and X is a discrete random variable. Calmet & Daemi [17] exploited this notion for measuring the reduction of uncertainty of one concept with respect to another, by considering possible target concepts in between them. This is represented through a *probability mass function*, $p(x_i)$, which is calculated for each

[1]For module extraction we do not need to worry about lexical changes to the concepts; this issue may be relevant for ontology learning and Dellschaft & Staab [30] propose a solution.

vertex in the graph (corresponding to some concept), by dividing the degree of the vertex; i.e. number of edges (i.e. properties) connected to that concept, with the sum of all degrees of V (where $v_i, v \in V$ are vertices):

$$p(v_i) = \frac{deg(v_i)}{\sum_{v \in V} deg(v)} \tag{4.10}$$

However, this entropy-based approach is limited as it considers all edges as equal. Doran *et al.* [34] show that this leads to some counter intuitive results. For example, consider the two graphs in Figure 4.1 where both graphs have an entropy value of 2.81.

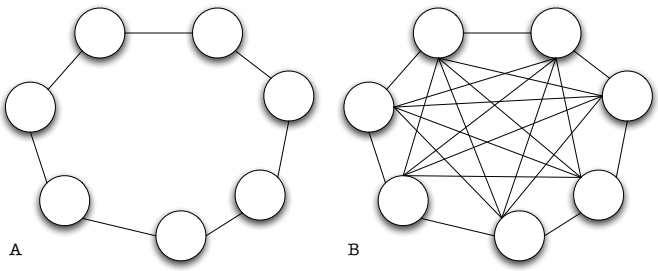

Figure 4.1: Two graphs with equal entropy.

This is counter intuitive, however, if we hypothesise that in one instance all the edges are `<owl:equivalentClass>` and in the other they are `<owl:ObjectProperty>` then the entropy values are expected to be different. When the edges represent `<owl:equivalentClass>`, then graph A represents the uninferred model and graph B the inferred model (i.e. the extra edges in graph B are implicit in graph A), where these edges have been made explicit and effect the entropy measure. Analogously, when each edge represents a different `<owl:ObjectProperty>` then the entropy values should be different because the second graph has many more properties linking the concepts.

Doran *et al.* [34] overcome the limitations of this approach via a reformulation of the entropy measure of Calmet & Daemi that accounts for the different types of relationships that can exist between concepts. This reformulation splits the entropy into two levels:

1. **Language Level.** Estimates the information content carried by the edges that represent language level constructs. These constructs are part of the ontological representation that is being used, for instance the OWL statements `<owl:equivalentClass>` or `<rdfs:subClassOf>`. These statements only require knowledge of the OWL semantics to be understood; whilst edges at this level may encode domain knowledge they do not require domain knowledge to be understood.

2. **Domain Level.** Concerned with the domain specific relationships; these are the constructs that allow an Ontology Engineer to tailor the ontology to their

domain. Such a construct in OWL would be the definition of an object property, `<owl:ObjectProperty>`. This level captures the information content that a relationship contributes to an ontology or module. The edges at this level require knowledge of the domain to be properly understood; for example, to fully comprehend an object property knowing the OWL semantics is insufficient. The label of the property will carry information regarding the conceptualization of the domain.

This reformulation requires the ontologies to be represented as an edge-labelled directed multigraph $G = (V, E)$ where:

- V is a finite set of vertices, representing the concepts defined in the ontology.
- $E = L \cup D$, where:
  - $L \subseteq V \times \Sigma_L \times V$ is a ternary relation whose elements (v_m, l_i, v_n) are language level edges, where $l \in \Sigma_L$, and Σ_L is the set of all the constructs in the ontology language that represent relationships between concepts.
  - $D \subseteq V \times \Sigma_D \times V$ is a ternary relation whose elements (v_t, d_j, v_u), where $d \in \Sigma_D$, and Σ_D is the set of relationships defined to capture links between domain entities.
- $\Sigma_L = \{l_1, ..., l_n\}$ and $\Sigma_D = \{d_1, ..., d_n\}$ are sets of labels which will label the edges of L and D respectively. Whilst the labels in Σ_L are defined by the specification of the ontology language used to represent the ontology[2], the labels in Σ_D are decided by the ontology developer.

The following functions assign a label to each edge from the respective alphabets:

- $label_l(L) : L \rightarrow \Sigma_L$
- $label_d(D) : D \rightarrow \Sigma_D$

For the purpose of this thesis we will consider Σ_L and Σ_D to be the following sets. It should be noted that these sets could be changed by the ontology developer if it is required.

- $\Sigma_L = \{$`<owl:disjointWith>`,`<owl:equivalentClass>`,`<owl:intersectionOf>`, `<owl:unionOf>`,`<rdfs:subClassOf>`$\}$
- $\Sigma_D = \{$`<owl:ObjectProperty>`$\}$

[2]In OWL, Σ_L is equivalent to the set of properties in the OWL vocabulary. The complete list of properties is available in Appendix C of the OWL reference document (http://www.w3.org/TR/owl-ref/)

Language Level Entropy - $H_L(X)$

The language level entropy ($H_L(X)$) calculates the entropy of the language level edges. Consider $G_L = (V, L)$ where $G_L \subseteq G$. We assume that all language level edges have equal weight and thus the probability mass function $p(v_i)$ is defined as:

$$p(v_i) = \frac{degOut(v_i)}{|L|} \tag{4.11}$$

where

$$degOut() : V \to \mathbb{R}$$

for each v that exists in V such that $degOut(v) = |L_v|$ where $L_v = \{(v, l, x)|v \in V\}$.

The function $degOut(v)$ counts the number of outgoing edges from a given v, *i.e.* the *degree* of the node (concept) v. Thus

$$\sum_{v \in V} degOut(V) = |L|$$

because for every element of V, the outgoing edges are considered and all the elements of L (v_i, l, v_m) must have $v \in V$ as the first element.

Domain Level Entropy - $H_D(X)$

The domain level entropy ($H_D(X)$) calculates the entropy associated with the domain level edges. We consider $G_D = (V, D)$ where $G_D \subseteq G$. We assume that the elements of Σ_D that appear more frequently in D split their information content evenly, thus the weight associated with these edges should be lower. For example, in an ontology modelling the relationships between a `PhD Student` and their `Supervisors` the relationships `coAuthorOf` can link a `PhD Student` and a `Supervisor`, or two `PhD Student`s, thus appearing more than once. Therefore, the information carried by this relationship is split between the contribution to the definition of `PhD Student`, and the contribution to the definition of `Supervisor`.

For every $d \in D$ we define a weighting function

$$w() : \Sigma_D \to \mathbb{R}$$

that assigns a real number corresponding to the weight to every element of the alphabet Σ_D. The weights $w_d, d \in \Sigma_D$ are defined as

$$w(d) = \frac{1}{|D_d|}$$

where

$$D_d = \{(x \times \sigma_D \times y)|label_d(d) = \sigma_D\}$$

that is, the weights are determined on the number of elements of the relationship $D \subseteq V \times \Sigma_D \times V$ for which the label $\sigma_D \in \Sigma_D$ is the same. The weights of the edges are normalised between 0 and 1, with the edges that appear more frequently getting a lower weight and the edges that appear less frequently getting a higher weight. The probability mass function $p(i)$ that we use for calculating $H_D(X)$ is:

$$p(i) = \frac{weightsFromNode(i)}{\sum_{v \in V} weightsFromNode(v)} \tag{4.12}$$

where

$$weightsFromNode() : V \to \mathbb{R}$$

for each v that exists in V such that

$$weightsFromNode(v) = \sum_{f \in F} w(f)$$

where F is the set of edges from D involving v. Thus, the weights of the edges outgoing from v are summed and divided by the sum of the weights of the outgoing edges for all elements of V.

Recombining The Entropy Measure

The ontology entropy measure $H(X)$ is calculated as the sum of the language and domain entropies:

$$H(X) = H_L(X) + H_D(X) \tag{4.13}$$

Depending on the semantics encoded in the graph it may be necessary to consider $\top$ and $\bot$. Assuming that $\top$ and $\bot$ are elements of V, then they are included in the above formula. However, one may just wish to consider the entropy amongst the user declared elements of V, as $\top$ and $\bot$ are usually required elements of the language (e.g., OWL). Thus, in this case, the entropy measure for the ontology would be:

$$H(X) = (H_L(X) + H_D(X)) - (H(\top) + H(\bot)) \tag{4.14}$$

which subtracts the entropy values associated with $\top$ and $\bot$ from the overall entropy value ($H(X)$).

Example of Entropy Computation

This example, based on the ontology given in Section 2.5, aims to illustrate what domain and language entropy are calculating and, thus, what contributes to the entropy of an ontology. We consider four variations of this ontology, whose entropy values can be seen in Table 4.1.

First consider the ontology in Figure 4.2. This ontology is simple a taxonomy, i.e. it only contains subclass relations. As such, it's score for $H_D(X)$ in Table 4.1 is 0 because this variant only contains language level edges.

Ontology	$H_L(X)$	$H_D(X)$	$H(X)$
Figure 4.2	2.459	0.000	2.459
Figure 4.3	0.000	1.057	1.057
Figure 4.4	2.459	1.057	3.516
Figure 4.5	3.533	2.654	6.187

Table 4.1: The entropy values generated for the different ontologies in Figures 4.2 - 4.5

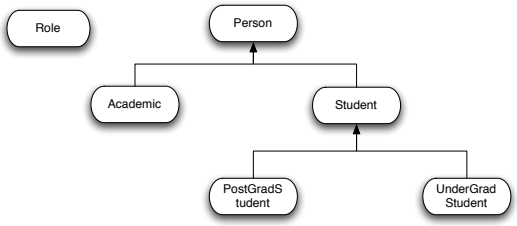

Figure 4.2: Ontology taxonomy.

Secondly consider the ontology in Figure 4.3. This ontology has no simple taxonomic relations, it only has three object properties. Thus, it's score for $H_L(X)$ in Table 4.1 is 0 because this variant only contains domain level edges.

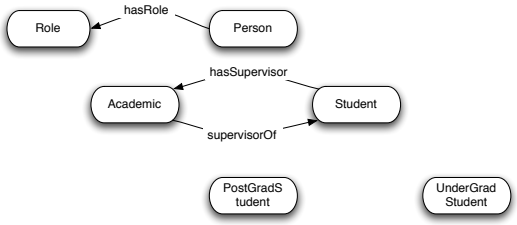

Figure 4.3: Ontology object properties.

The above two example show how the $H_L(X)$ and $H_D(X)$ are independent. Even when we consider the ontology in Figure 4.4, which is the union of Figure 4.2 and 4.3. The values in Table 4.1 now show that we have the $H_D(X)$ from Figure 4.3 and the $H_L(X)$ from Figure 4.2, which is to be expected. Note, however, that the $H(X)$ is now higher than in the previous two cases.

Lastly, consider the ontology in Figure 4.5 which is a further axiomatization of Figure 4.4. The addition of the extra restrictions results in an increase in both $H_L(X)$ and $H_D(X)$ as the restrictions contain both language and domain level elements. This is intuitive with respect to the definition of the entropy formula as it is a function over the edges in the graph. Essentially the more that is added to the graph the higher the entropy.

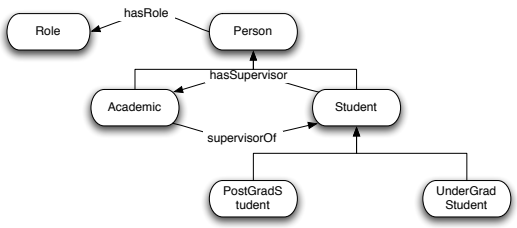

Figure 4.4: Ontology taxonomy and object properties.

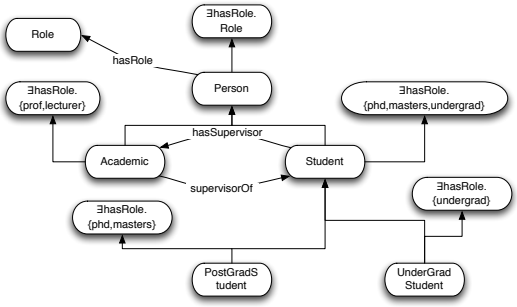

Figure 4.5: More axiomatized ontology. (NB Not all restrictions shown, the restrictions with {} would be split out so there is only one instance value per restriction)

4.4 Metric Based Evaluation

The aim of the metric based evaluation is to investigate the suitability of different objective measures for evaluating the results of an ontology modularization process. Firstly, precision and recall is investigated as it allows one to measure how much of the ontology is in the ontology module. Secondly, an entropy inspired metric is investigated which allows one to measure the information content of an ontology module.

4.4.1 Using Precision & Recall for Module Evaluation

Ontology	Species	Equivalent	Disjoint	Restriction
AKT-Portal	$\mathcal{ALCHIOF}(\mathcal{D})$		X	X
MindSwappers	$\mathcal{ALCHIF}(\mathcal{D})$		X	
Family	$\mathcal{ALC}$	X	X	X

Table 4.2: Table showing ontology properties.

Doran *et al.* [33] conducted experiments to evaluate their extraction method (see Section 3.4.1) using precision and recall on three ontologies: the AKT-Portal, MindSwappers and Family ontologies. These ontologies were chosen because of their varying expressiveness, see Table 4.2. Whilst the Family ontology is the least expressive it is highly interconnected and contains complex restrictions. For example, the Grandfather concept is defined as a father who has a child who is a parent.

A module was produced for each concept. The metric was then run to calculate the precision and recall, see Section 4.3.2, of the module with respect to the original ontology. The hypothesis is that precision will be high because the approach does not make any changes to the concepts that are placed in the module. In addition, intuitively, the recall should vary dependent on how large the module is and where it appears in the taxonomy. For example, large modules at the top of the taxonomy should have a high recall; whilst small modules at the bottom of the taxonomy should have a low recall. There should be little expectation that recall scores highly. Indeed it is undesirable to score highly on recall because this indicates that the ontology module is fairly similar to the original ontology, which nullifies the benefits of the module extractions process. Thus, it is expected that the score for recall will be lower than that for precision.

Ontology	Average Precision	Average Recall	Average fMeasure
Portal	1	0.54	0.68
MindSwappers	1	0.72	0.79
Family	1	0.84	0.9

Table 4.3: Experimental Results

The results are shown in Table 4.3. They concur with the expectations. Precision was high for all three ontologies; recall was lower, as expected.

The difference in recall between Portal and Mindswappers can largely be attributed to the difference in the average depth of the class tree. The average depth of the class tree for Portal is 5.89 and for Mindswappers it is 3.9. The average branching factor and the average number of object properties per concept are not significantly different. The average branching factor of Portal is 2.53 and for Mindswappers it is 2.58. Whilst the average number of object properties per concept for Portal is 1.56 and for Mindswappers it is 1.26.

The difference in the average depth of the class tree attributes to the difference in recall because of the way the subclass rule works in the extraction process. The greater the depth of the class tree means that as the process gets further down the hierarchy, then less of the hierarchy is placed into the ontology module. Thus, the less in the ontology module the lower the recall.

Because the precision and recall metric only considers the taxonomic structure of an ontology it is perhaps not best suited for evaluating ontology modules as a sizable amount of module content might be missed. Of course, if the ontology language is restricted to a taxonomy then precision and recall would have a higher value due to them taking account of the full expressivity of the ontology language being used. In this case, small modules would correspond to a low value for recall.

Ontology Name	# Cl.	Total # Prop.	# ObjProp	DL expressivity	Normalized $H(X)$
Conference	59	64	46	$\mathcal{ALCHIF}(\mathcal{D})$	0.1658
cmt	29	59	49	$\mathcal{ALCIF}(\mathcal{D})$	0.2928
confOf	38	36	13	$\mathcal{SIF}(\mathcal{D})$	0.2701
crs-dr	14	17	15	$\mathcal{SHIN}$	0.5127
edas	103	50	30	$\mathcal{ALCIF}(\mathcal{D})$	0.1097
ekaw	73	33	33	$\mathcal{SHIN}(\mathcal{D})$	0.1413
MICRO	31	26	17	$\mathcal{ALCIOF}(\mathcal{D})$	0.3030
OpenConf	62	45	24	$\mathcal{ALCIO}(\mathcal{D})$	0.1811
paperdyne	45	78	17	$\mathcal{ALCHIOF}(\mathcal{D})$	0.1946
PCS	23	38	24	$\mathcal{ELUIF}(\mathcal{D})$	0.3754
sigkdd	49	28	17	$\mathcal{ELI}(\mathcal{D})$	0.1744

Table 4.4: Classes, properties, expressivity, and normalized $H(X)$ values for each of the OntoFarm (http://nb.vse.cz/ svatek/ontofarm.html) ontologies used in the OAEI.

4.4.2 Using Entropy for Module Evaluation

The entropy evaluation is split into three parts, as follows:

1. *Intra-technique Evaluation.* Determines if the entropy based measures discriminate between modules of the same size when the signatures supplied to the algorithm were different.

2. *Inter-technique Evaluation.* Reflects the perspective of an Ontology Engineer wishing to reuse an ontology module; where there is a need to discriminate between two equally sized ontology modules produced by different techniques.

3. *Characterising the Entropy Measure.* Contrasts the different modularization techniques using the entropy metrics discussed in Section 4.3.3 over several different ontologies.

The Intra (1) and Inter (2) evaluations use the ontologies (AKT-Portal, Mindswappers and Family) used in the previous section, shown in Table 4.2. This dataset was chosen because it was small allowing the evaluation to be conducted in depth. The inter-evaluations was only run for the techniques of Doran and d'Aquin because they require the ontology modules to be inspected and it was felt that just two techniques would be sufficient.

However, the characterisation of the entropy measure (3) uses the ontologies listed in Table 4.4, complete with a brief characterisation in terms of the number of classes and properties, and the level of DL expressivity used to represent them. A larger dataset was required for this evaluation to provide more data in order to properly characterise the entropy measure. Indeed, this is the reason for also running the evaluation over the techniques by Cuenca-Grau *et al.*and Seidenberg and Rector.

Family ontology					
	Interval of Module Size	**OE**	$H_L(X)$	$H_D(X)$	$H(X)$
Doran	18	0.157	0.249	0.208	0.431
d'Aquin	5	0.004	0.241	1	1.241
Cuenca Lower	26	0.126	0	0	0
AKT Portal ontology					
Doran	1	0.018	0	0	0
	34	0.005	0	0.013	0.012
	36	0.015	0.002	0.018	0.018
	37	0.012	0.003	0.010	0.013
	38	0.088	0.064	0.050	0.112
	39	0.012	0	0.024	0.025
	186	0.121	0.137	0.156	0.291
d'Aquin	40	0.162	0.680	2.030	1.475
	41	0.283	0.709	2.053	1.511
	42	0.168	0.629	2.075	1.509
	43	0.202	0.702	2.071	1.511
	46	0.187	0.644	2.05	1.501
	48	0.176	0.690	1.396	0.933
Cuenca Upper	20	0.370	0.741	1.437	1.128
	21	0.416	0.849	1.182	0.346
	22	0.450	0.762	1.536	1.397
	23	0.507	0.843	1.573	1.258
	24	0.473	0.698	1.600	1.212
	25	0.504	0.733	1.642	1.415
	27	0.450	0.857	1.506	1.084
	28	0.644	1.005	1.427	1.027
	29	0.487	0.690	1.450	1.201
	30	0.483	1.036	1.025	0.383
	40	0.394	0.892	0.464	0.428
	42	0.396	0.606	0.996	0.692
Cuenca Lower	9	0.299	0.322	0	0.322
	14	0.243	0.590	0.722	1.000
	15	0.412	0.965	0.906	1.004
	24	0.216	0.319	0.066	0.384
Mindswap ontology					
Doran	1	0.142	0	0	0
	17	0.004	0.011	0	0.011
	19	0.018	0.024	0.300	0.309
	20	0.067	0.024	0.291	0.291
	23	0.002	0.016	0	0.016
d'Aquin	1	0	0	0	0
Cuenca Upper	11	0	0	0	0
Cuenca Lower	1	0	0	0	0

Table 4.5: Intervals in the entropies for the Family, AKT Portal, and Mindswap ontologies

Intra-technique Evaluation

Each of the module sets extracted were grouped by size, and the entropy was calculated (each of the four metrics under evaluation were used). An *interval* was then determined for a set of some given size, based on the difference between the maximal and minimal entropy calculations for the modules in that set. These intervals are listed in Table 4.5 for each of the three ontologies using the four module extraction techniques. No results are shown for those sets where the interval value was zero for all entropy metrics evaluated.

LCO Concept	d'Aquin			Doran		
Import Directives	none			support.owl		
Expressivity	$\mathcal{ALCF}$			$\mathcal{ALCOF}(\mathcal{D})$		
	Defined	Imported	Total	Defined	Imported	Total
Classes	22	0	22	9	17	26
Datatype Properties	0	0	0	4	10	14
Object Properties	10	0	10	1	5	6
Annotation Properties	1	0	1	1	3	4
Individuals	6	0	6	2	13	15
General Concept Inclusion	5	0	5	0	0	0
SubClass Axioms	18	0	18	23	0	23
Disjoint Axioms	0	0	0	7	0	7
	Defined	Imported		Defined	Imported	
Base Model Triple No.	359	0	359	64	284	348
Inferred Model Triple No.	572	0	572	120	575	692

Table 4.6: Comparison between d'Aquin and Doran approaches on *LCO*

For the Family ontology, the results show that in two cases the new entropy based metrics are more discriminating than the original entropy metric (OE), whilst in the case of Cuenca-Grau *et al*'s 'lower' technique, only the OE metric provided some discrimination (i.e. there was a difference of 0.126 between the highest and lowest entropy values for different modules of size 26). However, for many of the ontology modules produced (seven sets, which are not reported in the table), there was no difference in entropy values. This is due to the ontology being highly interconnected, it contains many concepts in terms of complex class restrictions (for example, $\texttt{Son} \equiv \texttt{Male} \sqcap \exists \texttt{hasParent.Parent}$).

For most of the modules generated from the AKT-Portal ontology, the improved entropy metric ($H(X)$) provided greater discrimination than the OE metric. This difference varied, depending on the module extraction technique; from an average of 0.039 (OE) compared to 0.067 ($H(X)$) for the Doran technique, to an average of 0.196 (OE) compared to an average of 1.407 ($H(X)$) for d'Aquin's technique. The OE metric identified the smallest intervals for the majority of module sets, and in general, the domain level entropy metric produced the greatest intervals.

The MindSwap ontology produced some anomalous results, with two of the four techniques (d'Aquin *et al* and Cuenca-Grau *et al*'s 'lower' technique)[3] producing modules of size 0[4] or 1. As the entropy metrics rely on there being edges between concepts, they fail on graphs with single (or a very small number of) concepts. Doran *et al*'s technique produced several modules with a range of sizes. However, unlike the intervals generated for the other ontologies, a greater number of sets had intervals at the language level rather than at the domain level, suggesting that the modules tend to contain the same domain level edges.

Inter-technique Evaluation

Two modules extracted from the Portal with the signature set to 'Learning Centered Organization' (LCO) by the d'Aquin and Doran approaches are described by means of some metrics computed with SWOOP[5] in Table 4.6. These results show that the two approaches generate similar modules w.r.t. size, but their entropy values are different (see Table 4.7) and indeed their content is largely different. This shows that two modules of the same size extracted by different techniques that produce an ontology module about the same concept are better discriminated objectively via an entropy based measure; and that the improved measure allows an Ontology Engineer to better identify where the difference is. The following now examines some of the differences between the two modules.

One important difference is the fact that the Doran approach leaves the *owl:imports* directives in the module. This helps to keep track of dependencies, as well as allowing the extracted module to be reused should the imported ontologies change, but importing large ontologies may lead to very large modules. The d'Aquin approach would require the ontology module to be rebuilt if any change occurs in the imported ontologies, but the module is self contained.

Another difference is the expressivity: d'Aquin does not include datatypes and nominals in the module. However, the relation between traversal modularization methods and expressivity needs deeper investigation before meaningful conclusions can be drawn. Looking at the specific differences, we note that the only common named concept between the two modules, not including the imported ontology, is the root of the model: LCO. Focusing on the differences, in the d'Aquin module 13 subclass relationships where LCO is the subject were included, one with *Organization* and 12 with anonymous classes, which represent the definition of LCO. In the Doran module, there is only one *isDefinedBy* property that states that LCO is defined according

[3]These results suggest that the extraction techniques of d'Aquin *et al* and Cuenca-Grau *et al* place strict criteria in certain circumstances on what is included within a module, and thus may fail to produce usable modules.

[4]An ontology module of size zero would typically contain either $\top$ or $\bot$. This may be of value when considering the modularization process itself, but of little pragmatic use to the Ontology Engineer.

[5]`http://code.google.com/p/swoop/`

to `http://www.aktors.org/ontology/portal`, and 7 statements relating LCO to its named subclasses. d'Aquin seems to capture the definition of the root concept, while Doran aims to capture the portion of the ontology that specialises the root concept; this is confirmed by the respective motives outlined in [33, 27].

Approach	OE	$H_L(X)$	$H_D(X)$	$H(X)$
Doran	4.048	4.963	3.864	8.826
d'Aquin	4.975	4.655	3.936	8.591

Table 4.7: Entropy values for LCO modules.

Characterising the Entropy Measure

The aim of this evaluation is to contrast the different modularization techniques using the entropy metrics discussed in Section 4.3.3 over several different ontologies. By utilizing the constituent entropies, Language Entropy ($H_L(X)$) and Domain Entropy ($H_D(X)$), further insights regarding the characteristics of the resulting modules should be possible over simply considering their size. Five ontology modularization techniques have been evaluated: the upper and lower variants of Cuenca Grau *et al.* [24], the technique proposed by Seidenberg and Rector [115], d'Aquin *et al.*'s approach [27] and the approach proposed by Doran *et al.* [33]. The eleven ontologies used in the evaluation were taken from the *OAEI 2007 Conference Track* repository[6] (with the exception of three ontologies[7]). The dataset is switched from the one used in the previous evaluations as the OAEI has a larger number of real world ontologies covering the same domain; this will aid the characterisation of the entropy measure.

The ontologies used in this evaluation are listed in Table 4.4, complete with a brief characterization in terms of the number of classes and properties, and the level of DL expressivity used to represent them. The Integrated Entropy ($H(X)$) has been calculated, and normalized with respect to the number of named classes, to indicate the mean $H(X)$ per named concept for each of the ontologies.

For each modularization evaluation, a module was created for each of the named concepts (i.e. the concept is used as the signature for each modularization method) within each ontology, resulting in a module set. The size of the module (in terms of the number of named concepts included within it) and the three entropy metrics ($H_L(X)$, $H_D(X)$ and $H(X)$) were calculated based on the inferred model for each of the modules and recorded. The entropy values were then normalized with respect to the size of the module to give a per-concept entropy.

Table 4.8 lists the results for the modules generated by each of the modularization techniques across each of the ontologies, as well as the number of named classes defined

[6] `http://oaei.ontologymatching.org/2007/conference/`
[7] These ontologies have memory requirements of more than 2GB.

Table 4.8: Comparison of the Module Size (in terms of named entities) and the resulting $H(X)$ values for each of the different modularization approaches. Both the number of modules generated containing more than two named concepts, and this value as a percentage of all modules for each ontology are given.

		DAQUIN			**DORAN**			**SEIDENBERG**			**CG-L**			**CG-U**		
Ontology	#	Modules (>1)			Modules (>1)			Modules (>1)			Modules (>1)			Modules (>1)		
Name	Cl.	Num	% Cl.	$H(X)$	Num	% Cl.	$H(X)$	Num	% Cl.	$H(X)$	Num	% Cl.	$H(X)$	Num	% Cl.	$H(X)$
Conference	59	26	44.1%	0.312	24	40.7%	0.433	49	83.1%	0.418	35	59.3%	0.404	35	59.3%	0.406
cmt	29	14	48.3%	0.347	11	37.9%	0.472	22	75.9%	0.459	9	31.0%	0.373	9	31.0%	0.375
confOf	38	7	18.4%	0.376	8	21.1%	0.469	33	86.8%	0.298	25	65.8%	0.242	25	65.8%	0.242
crs-dr	14	9	64.3%	0.426	3	21.4%	0.470	10	71.4%	0.448	0	0.0%	0.000	0	0.0%	0.000
edas	103	18	17.5%	0.442	25	24.3%	0.329	89	86.4%	0.474	102	99.0%	0.456	102	99.0%	0.462
ekaw	73	14	19.2%	0.535	23	31.5%	0.466	67	91.8%	0.489	15	20.5%	0.310	15	20.5%	0.311
MICRO	31	14	45.2%	0.327	6	19.4%	0.493	28	90.3%	0.217	24	77.4%	0.349	24	77.4%	0.350
OpenConf	62	12	19.4%	0.475	25	40.3%	0.254	60	96.8%	0.483	62	100.0%	0.326	62	100.0%	0.325
paperdyne	45	22	48.9%	0.179	8	17.8%	0.477	41	91.1%	0.171	36	80.0%	0.202	36	80.0%	0.202
PCS	23	13	56.5%	0.367	10	43.5%	0.456	17	73.9%	0.367	7	30.4%	0.316	7	30.4%	0.317
sigkdd	49	11	22.4%	0.404	14	28.6%	0.448	43	87.8%	0.375	8	16.3%	0.373	8	16.3%	0.374
Mean values for all ontologies:			**36.7%**	**0.381**		**29.7%**	**0.433**		**85.0%**	**0.382**		**52.7%**	**0.305**		**52.7%**	**0.306**

by each ontology. For each technique, the number of modules containing two or more named concepts is given, as well as the mean normalised $H(X)$ values. Cases where no modules can be generated, or where modules of size one are generated are not given. This is in part due to the fact that some techniques (such as DORAN and SEIDENBERG) are guaranteed to generate a module containing at least the concept specified in the module signature, whereas Cuenca Grau *et al.*'s techniques (CG-L and CG-U) can generate empty modules (i.e. of size zero, $\top$ and $\bot$ are not considered by the size measure; meaning no concept could be added to the module to conform with the safety condition, see Section 3.4.2). In addition, as the entropy metrics defined above determine the level of connectivity between concepts, modules of size less than two result in zero-based entropy values. Finally, the number of modules (of size > 1) generated for each ontology is given as a percentage of the total number of named concepts. The mean values across all the ontologies for the percentage of modules considered, and the normalised $H(X)$ values are given at the bottom of the table.

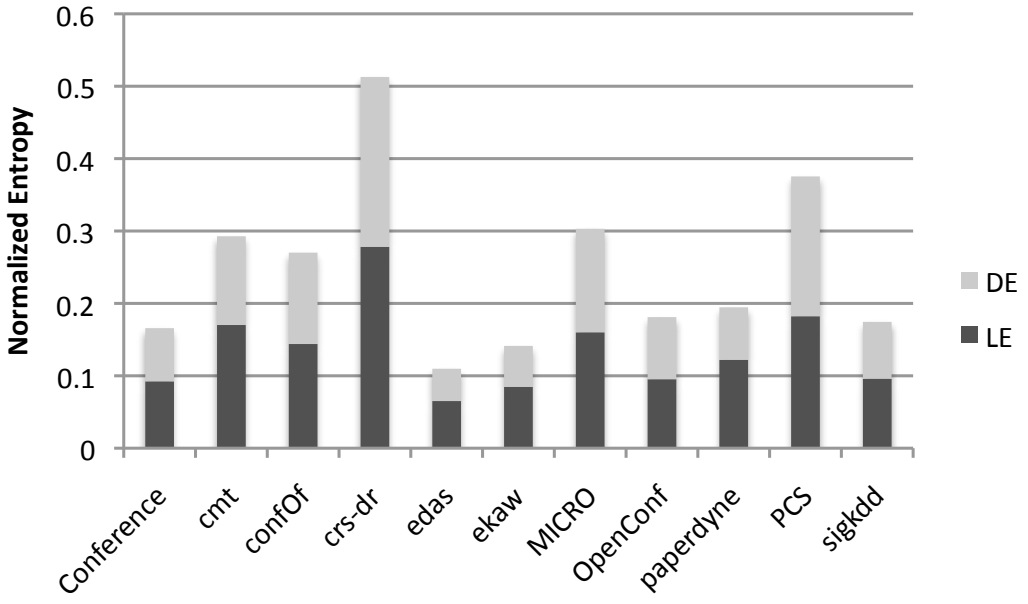

Figure 4.6: The Normalized Entropy (based on the $H_L(X)$ and $H_D(X)$ values) for each of the OAEI ontologies.

Discussion

The metrics given in Table 4.4 provide a baseline measure for each of the ontologies prior to modularization. The normalised entropy values indicate the level of connectivity associated with each of the concepts. Figure 4.6 illustrates this value graphically, in terms of the Language Entropy ($H_L(X)$), forming the lower segment of each bar, and Domain Entropy ($H_D(X)$), forming the upper segment. Whilst there is no clear

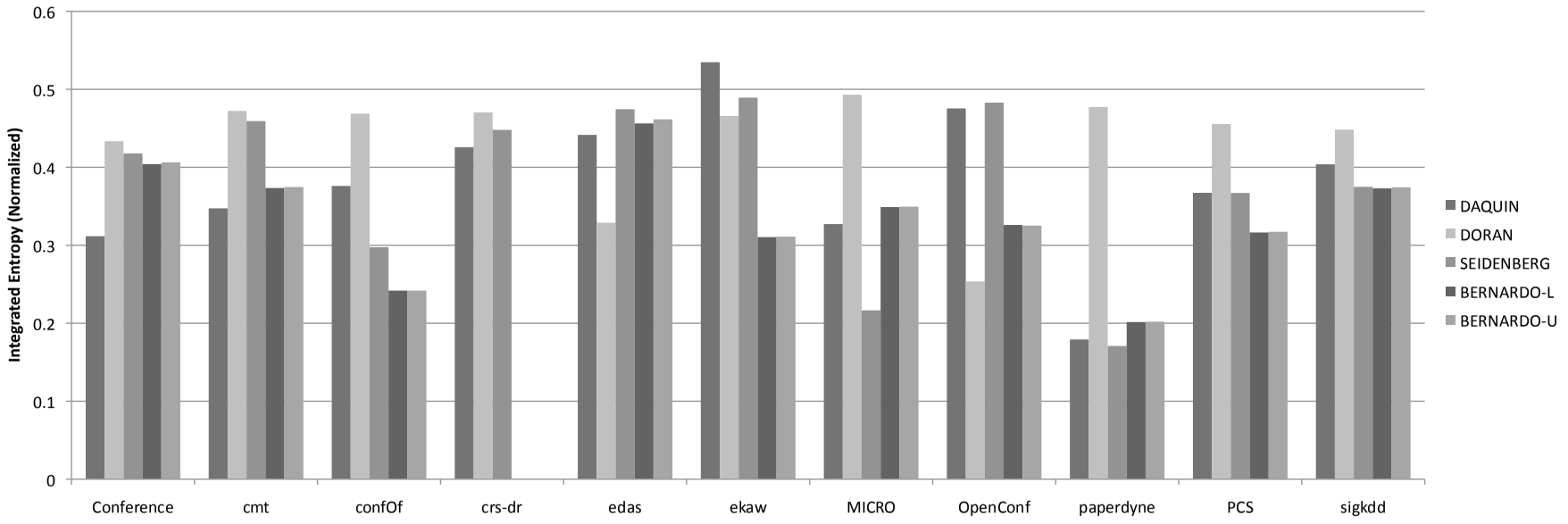

Figure 4.7: The mean Normalized Integrated Entropy for the modules generated for each of the ontologies.

correlation, the results suggest a possible weak trend whereby as the ratio of properties to concepts increases, then the entropy rises. However, this trend is difficult to verify as the size and complexity of ontologies increases, as the distribution of properties across the concepts is far from uniform.

The results in Table 4.8 suggest that SEIDENBERG produces the largest number of modules (and also the largest average module size[8]), varying between 71.4% and 96.8% of the concepts yielding modules, whereas the smallest number of modules were generated by DORAN (generating modules for only an average of 29.7% concepts). However, the corresponding normalised $H(X)$ values suggest that the modules generated by DORAN contain a greater number of edges (and hence a greater connectivity) than those modules created by other methods. Although the Cuenca Grau *et al.* variants produce the smallest mean $H(X)$ values across all the ontologies, in most cases the individual $H(X)$ value falls within the minimum and maximum $H(X)$ for a given ontology. The only outlier here is for the *crs-dr* ontology, whereby the Cuenca Grau *et al.* variants were unable to find any modules for any of the concepts tested. However, this was the smallest of the ontologies tested, and was highly interconnected; which is reflected by the ontology having the highest normalised $H(X)$ value (0.5127, Table 4.4). Figure 4.7 illustrates the individual normalised Integrated Entropy values for each of the modularization approaches (for each of the ontologies tested). In eight of the eleven ontologies, DORAN produces the highest $H(X)$ value; however, in two of the other three cases (*edas* and *OpenConf*), it produced the worst $H(X)$ results.

4.4.3 Critique

The metric based evaluation has shown that it is difficult to objectively assess how 'good' an ontology module is. Indeed, it is possible to define objective measures and use them to assess the modules; the problem then being in deciding what these measures mean. These problems suggested that a more focused evaluation was needed. We previously considered how the different ontology module extraction techniques were created for a particular task. Therefore, it was natural to consider a comparison of the different techniques in a task-based evaluation. This allows one to characterise the performance of the different techniques over a standard task. The next Section presents this evaluation.

4.5 Task Based Evaluation

The metric based evaluation (Section 4.4) considers the modularization process in isolation and the metrics attempt to objectively quantify how 'good' an ontology is. However, this does not tell us if an ontology module is fit for purpose. Therefore, we

[8]The average module size is not given in the Tables above.

have identified three problems that support a number of common tasks such as query answering or service retrieval:

1. Instance retrieval. Asking for instances of a concept in an ontology.

2. Subclass retrieval. Asking for the subclasses of a concept in an ontology.

3. Superclass retrieval. Asking for the superclasses of a concept in an ontology.

These three tasks are considered as statistical classification problems over the original ontology O, and the module M_i computed using a modularization technique whose input signature is $Sig_{M_i} = \{C_i\}$.

Let $O = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base; let $Ind(\mathcal{A})$ be the set of all individuals occurring in $\mathcal{A}$, and let $C = \{C_1, C_2, \cdots, C_s\}$ be the set of both primitive and defined concepts in O.

4.5.1 Instance Retrieval

The $InstanceRetrieval_{(O,C_i)}$ problem can be defined as follows: **given** an individual $a \in Ind(\mathcal{A})$, and a class $C_i \in C$ **determine** the set

$$I_O = \{Instance(C_i) \subseteq Ind(\mathcal{A}) | O \models C_i(a)\}$$

For the purpose of this evaluation M_i is to be considered an ontology itself, so the evaluation distinguishes between the task of instance retrieval in the original ontology O, $InstanceRetrieval_{(O,C_i)}$, from the task of instance retrieval in the module M_i where the signature of the module $S_{M_i} = \{C_i\}$, therefore the problem becomes determining

$$I_{M_i} = \{Instance(C_i) \subseteq Ind(\mathcal{A}) | M_i \models C_i(a)\}$$

4.5.2 Subclass Retrieval

The $SubclassRetrieval_{(M_i,C_i)}$ problem can be defined as follows: **given** a class $C_i \in C$ **determine** the set

$$Sub_O = \{\{X_1, X_2, \cdots, X_m\} \subseteq C | \forall X_j, j = 1, \cdots m : O \models \mathcal{I}^{X_j} \subseteq \mathcal{I}^{C_i}\}$$

Analogously, we define the problems $SubclassRetrieval_{(M_i,C_i)}$, with the set Sub_{M_i} for the module M_i as follows:

$$Sub_{M_i} = \{\{X_1, X_2, \cdots, X_m\} \subseteq C | \forall X_j, j = 1, \cdots m : M_i \models \mathcal{I}^{X_j} \subseteq \mathcal{I}^{C_i}\}$$

4.5.3 Superclass Retrieval

The $SuperclassRetrieval_{(O,C_i)}$ problem can be defined as follows: **given** a class $C_i \in C$ **determine** the set

$$Sup_O = \{\{X_1, X_2, \cdots, X_m\} \subseteq C | \forall X_j, j = 1, \cdots m : O \models \mathcal{I}^{C_i} \subseteq \mathcal{I}^{X_j}\}$$

Analogously, we define the problems $SuperclassRetrieval_{(M_i,C_i)}$, with the set Sup_{M_i} for the module M_i as follows:

$$Sup_{M_i} = \{\{X_1, X_2, \cdots, X_m\} \subseteq C | \forall X_j, j = 1, \cdots m : M_i \models \mathcal{I}^{C_i} \subseteq \mathcal{I}^{X_j}\}$$

4.5.4 Capability Evaluation

To evaluate the capability of a modularization technique, a named concept C_i is selected in O; the corresponding module M_i is then built. Then we consider the following three problems:

1. $InstanceRetrieval_{(O,C_i)}$ and $InstanceRetrieval_{(M_i,C_i)}$. These define respectively I_O and I_{M_i}, the set of instances of C_i in O and in M_i.

2. $SubclassRetrieval_{(O,C_i)}$ and $SubclassRetrieval_{(M_i,C_i)}$. These define respectively Sub_O and Sub_{M_i}, the set of subclasses of C_i in O and in M_i.

3. $SuperclassRetrieval_{(O,C_i)}$ and $SuperclassRetrieval_{(M_i,C_i)}$. These define respectively Sup_O and Sup_{M_i}, the set of superclasses of C in O and in I_M.

For the above three pairs of results, (I_O,I_{M_i}) and (Sub_O,Sub_{M_i}) and (Sup_O,Sup_{M_i}), the following are computed:

- *True Positive.* This is the number of entities (classes or instances) that are classified correctly (i.e. as Subclass, Superclass or Instance of a class C_i). For the pair (I_O,I_{M_i}) this is defined as:

  $$truepositive = |I_O \cap I_{M_i}|$$

  Giving us the number of instances of C_i in both O and M_i. This can be computed for (Sub_O, Sub_M) and (Sup_O, Sup_M) by substituting them for (I_O,I_{M_i}).

- *False Positive.* This is the number of entities that were incorrectly classified as positive. For the pair (I_O,I_{M_i}) this is defined as:

  $$falsepositive = |I_O \setminus I_M|$$

  Giving us the number of instances of C_i in O which are not instances of C_i in M_{C_i}. This can be computed for (Sub_O, Sub_M) and (Sup_O, Sup_M) by substituting them for (I_O,I_{M_i}).

- *False Negative.* This is the number of correct entities that were missed (i.e. entities that were not classified as belonging to the true positive class but should have been). For the pair (I_O,I_{M_i}) this is defined as:

$$falsenegative = |I_M \setminus I_O|$$

Giving us the number of instances of C_i in M_i which are not instances of C_i in O. This can be computed for (Sub_O, Sub_M) and (Sup_O, Sup_M) by substituting them for (I_O,I_{M_i}).

False negatives can occur when the modularization approach does not preserve all the constraints on class definitions by not taking a proper subset of the axioms that were present in the original ontology. Thus new class definitions are generated and are included in the module, for instance a disjunction axiom is lost in the modularization process (this can occur in those modularization approaches that do not guarantee *safety*, that is to leave the concept definitions unchanged, as discussed in 3.4.2). If a disjunction axiom is lost between two concepts then it is now possible that the instances of these concepts could be inferred to be instances of the other concept. This is discussed in more in detail in Section 4.5.5.

These values enable classical precision and recall measures (see Section 4.3.2) to be computed for the three problems in consideration, where precision and recall are defined as follows:

$$precision = \frac{truepositive}{truepositive + falsepositive} \tag{4.15}$$

$$recall = \frac{truepositive}{truepositive + falsenegative} \tag{4.16}$$

To synthesise the values of precision and recall into a single value, the F-measure is used:

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \tag{4.17}$$

4.5.5 Evaluation Setup

The dataset for this evaluation consists of eleven ontologies from the OAEI 2007 Conference Track[9]; the full list, as well as some metrics for these ontologies, such as expressivity, number of named concepts and roles, and number of anonymous concepts, is available in Table 4.9.

The modularization techniques available (see Section 3.4) for the experiment are:

1. Cuenca Grau *et al* [24], *lower variant*, shortened in the following as CG^L ;
2. Cuenca Grau *et al* [24], *upper variant*, shortened as CG^U ;

[9] `http://oaei.ontologymatching.org/2007/conference/`

Ontology Name	# Concepts	Total # Properties	# Object Properties	# Anon Concepts	DL Expressivity
Conference	59	64	46	33	$\mathcal{ALCHIF}(\mathcal{D})$
cmt	29	59	49	11	$\mathcal{ALCIF}(\mathcal{D})$
confOf	38	36	13	42	$\mathcal{SIF}(\mathcal{D})$
crs-dr	14	17	15	0	$\mathcal{SHIN}$
edas	103	50	30	30	$\mathcal{ALCIF}(\mathcal{D})$
ekaw	73	33	33	27	$\mathcal{SHIN}(\mathcal{D})$
MICRO	31	26	17	33	$\mathcal{ALCIOF}(\mathcal{D})$
OpenConf	62	45	24	63	$\mathcal{ALCIO}(\mathcal{D})$
paperdyne	45	78	17	109	$\mathcal{ALCHIOF}(\mathcal{D})$
PCS	23	38	24	26	$\mathcal{ELUIF}(\mathcal{D})$
sigkdd	49	28	17	15	$\mathcal{ELI}(\mathcal{D})$

Table 4.9: Classes, properties, and expressivity for each of the OAEI ontologies.

3. d'Aquin *et al* [27], shortened as DAQ ;

4. Doran *et al* [33], shortened as DOR ;

5. Seidenberg and Rector [115], shortened as $SEID$.

For each one of these techniques, the implementation made available by the authors has been used to guarantee the behaviour of each approach as intended by the original authors. Wrappers have been used to fit the techniques into the testing framework, such as changes to the expected input method, from a URL for the ontology to load to a local file. However, these do not modify the data, and have no affect on the results of the evaluation.

For each ontology, the set of named concepts has been considered. For each named concept, each technique has been used to produce the related module; the modularization signature was in each case the single named concept. The total number of modules obtained in this way is the sum of the *# Concepts* column in Table 4.9, multiplied by the number of techniques; this gives a total of 2630 modules, of which 526 were generated for each technique.

For each module, precision, recall and F-measure have been computed, as outlined in Section 4.5.4. The results have then been presented by ontology and technique (Table A.1, A.2 and A.3).

Table 4.10 lists the results for the modules generated by each of the modularization techniques across each of the ontologies, as well as the number of named classes defined by each ontology. For each technique, the number of modules containing two or more named concepts is given. Cases where no modules can be generated, or where modules of size one are generated are not given. This is in part due to the fact that some techniques (such as DOR and $SEID$) are guaranteed to generate a module containing

Ontology Name	# Cl.	DAQ Mod. (> 1) % Cl.	DOR Mod. (> 1) % Cl.	SEID Mod. (> 1) % Cl.	CG-L Mod. (> 1) % Cl.	CG-U Mod. (> 1) % Cl.
Conference	59	44.1%	40.7%	83.1%	59.3%	59.3%
cmt	29	48.3%	37.9%	75.9%	31.0%	31.0%
confOf	38	18.4%	21.1%	86.8%	65.8%	65.8%
crs-dr	14	64.3%	21.4%	71.4%	0.0%	0.0%
edas	103	17.5%	24.3%	86.4%	99.0%	99.0%
ekaw	73	19.2%	31.5%	91.8%	20.5%	20.5%
MICRO	31	45.2%	19.4%	90.3%	77.4%	77.4%
OpenConf	62	19.4%	40.3%	96.8%	100.0%	100.0%
paperdyne	45	48.9%	17.8%	91.1%	80.0%	80.0%
PCS	23	56.5%	43.5%	73.9%	30.4%	30.4%
sigkdd	49	22.4%	28.6%	87.8%	16.3%	16.3%
Mean values:		**36.7%**	**29.7%**	**85.0%**	**52.7%**	**52.7%**

Table 4.10: Comparison of the Module Size (in terms of named entities) for each of the different modularization approaches. The percentage size of all modules for each ontology are given. See Table A.4 for more detail.

at least the concept specified in the module signature, whereas Cuenca Grau *et al.*'s two techniques (CG^L and CG^U) can generate empty modules (i.e. of size zero). Finally, the number of modules (of size > 1) generated for each ontology is given as a percentage of the total number of named concepts. The mean values across all the ontologies for the percentage of modules considered.

Monotonicity in DL and False Negatives

Some of the extracted modules cause some instances to be misclassified with respect to the whole ontology, i.e., there is some concept C_i for which there are instances in M_i that are not instances of C in O; this seems counterintuitive given the monotonic nature of Description Logics. According to monotonicity, all the inferences that can be drawn from a subset of axioms can be drawn against the whole set of axioms in a knowledge base, so no inferences that can be drawn from a module M_i should be lost when considering O. This intuition, however, is based on the assumption that the module is a proper subset of the original axioms, but this is not guaranteed by all the techniques considered.

The same problem can arise with subclasses and superclasses; in the current evaluation, these problems have been detected in 31 cases, involving two ontologies: Paperdyne and OpenConf. These two ontologies are both complex and highly interconnected thus increasing the chances that an extraction technique will miss something when extracting concept definitions.

4.5.6 Discussion

The results presented demonstrate that all the modularization techniques evaluated perform reasonably well in terms of precision and recall across all but two of the considered ontologies. However, all the approaches but Seidenberg's experienced some degradation in performance when applied to OpenConf and Paperdyne. This could be due to the fact that these two ontologies are both very complex and interconnected ontologies that cause all the approaches to degrade.

Furthermore, we note that Seidenberg's technique seems to have the greatest degree of variation with respect to the considered ontology, with many cases in which the precision is either 0 or 100%. This result seems to indicate that some of the heuristics used by Seidenberg's modularization approach might have been overly specific to the characteristics of the GALEN ontology, and thus are not so well suited for ontologies that have different characteristics with respect to GALEN.

One interesting result is that there is no discernible difference between the logic based approaches and the traversal based approaches in terms of precision and recall. However, the modules differ in size and the percentage of modules with 0 or one concept only. This seems to indicate that users need to look at the characteristics of the task they have in mind in order to choose the most appropriate modularization approach; thus, making up the intuition outlined in Section 3.2 that the different techniques have different purposes. Hence, for instance, we might want to distinguish the task of single instance retrieval from the more generic task of Instance Retrieval. The former is typical of queries where a single instance of a concept is required. For example, in service provision, where a request is made for a service that is of a certain class, such as *"Give me a service that is an instance of Weather service"*. The Instance Retrieval task provides *all* the instances of a class. In the first case, any of the modularization approaches with high precision results (Cuenca Grau upper and lower variants, d'Aquin and Doran) would perform equally well; whilst Doran's has the lowest precision, it is still within a 0.05% error. Recall, in this scenario would not be as important as finding just one correct instance which would suffice to satisfy the user request.

Conversely, if we are looking at the problem of generalized instance retrieval, then recall becomes important, and in this case Doran has a better recall (whilst maintaining a good performance) followed by d'Aquin, and Cuenca Grau's variants, whose recall values are very similar.

If the problem consists of retrieving all the subclasses, then Doran once again performs better than the others. This is an artifact of the type of traversal used by the approach (see Section 3.4.1), that traverses mainly downwards from the signature concept. Interestingly enough, the results for subclass retrieval and superclass retrieval on this dataset seem to indicate that the content of a module is defined by the definition of the subclasses of the signature concept, whilst the superclasses seem to provide a

lesser contribution to the module definition. For this reason Doran's approach, that includes only the constraints from the superclasses of the signature that are inherited down the hierarchy, performs as well as other approaches like d'Aquin or Cuenca Grau.

Other considerations that a user might want to take into account when choosing a module are related to the specific characteristics of the task. If the module produced is to be used for extensive reasoning, then Cuenca Grau's approach is to be preferred as it is the only one amongst those considered that guarantees safety. This guarantee, for example, would be important in critical systems, such as medical systems, where a misclassification could have serious consequences. However, if safety is not a concern, then Doran and d'Aquin are good candidates.

4.6 Conclusions

The metric-based evaluation (Section 4.4) shows that the intra-technique evaluation of the entropy inspired measure in most cases is a preferable discriminating factor than size. Furthermore, the entropy measure presented allows the Ontology Engineer to assess what contributes to the overall entropy of the ontology module by calculating the entropy at the domain and language levels. The characterisation of the entropy measures suggests that as the number of properties (with respect to concepts) increase, then the entropy values also increase. However, no clear trend was observed, suggesting that further investigation is needed.

In addition, the assumptions made at the moment (such as all language level edges being equal, etc.) could possibly be relaxed to further improve the entropy measure. For example, it is possible to argue that a 'disjoint' edge carries more information than a 'subclass' edge and, as such, should be weighted differently. This may further enhance the discriminating power of the measure and allow for a clearer characterisation.

The metrics could also be extended in future work to link information content to a notion of usability and reusability. In principle, an ontology with low entropy has less information content, and thus is likely to be highly *reusable*, as it is unlikely to contain in depth axiomatizations making it widely applicable; but not highly *usable* for an application because it is unlikely to contain enough information. Whereas, an ontology with high entropy, in contrast, will be more likely to be *usable* because it should contain more information. Such an entropy metric could help to bridge the gap between the subjective evaluation of an Ontology Engineer and the objective measures available to them, and provide additional insight into the characteristics of different modules.

The task-based evaluation (Section 4.5) was conducted because there has been little systematic analysis and comparison of the modularization approaches with respect to a set of common tasks. Objective measures such as size or entropy give some information about a module, but fail to capture task-related information, such as whether the

module is fit for purpose, or can lose information (with respect to using the original ontology). To this end, the task-based evaluation was a systematic and extensive empirical evaluation of various module extraction approaches, from the perspective of their suitability for a specific task. Three related problems were identified that supported a number of common tasks such as query answering or service retrieval: *Instance retrieval*, *Subclass retrieval*, and *Superclass retrieval*.

The results suggest that pragmatic, heuristic approaches such as those that assume graph traversal may be as good as logical-based approaches for some scenarios. Whilst better for tasks that may require safety guarantees or extensive reasoning, logical based approaches may not offer many benefits when used for generalized instance retrieval. However, in nearly all cases, little utility is gained by considering the definition of concepts that are more general than those appearing in the signature. Future work could extend this analysis to better identify boundary cases whereby certain techniques may be more suitable than others.

Part IV

Practice

Chapter 5

Applying Ontology Module Extraction to Ontology Reuse

'Consider fully, act decisively.' - Jigoro Kano

Summary This chapter considers the practical application of ontology modularization to the problem of ontology reuse. Firstly, an overview of existing Ontology Engineering methodologies is given, which covers how they include steps for the reuse of existing ontologies. Then two methodologies for reusing ontology modules are presented. These methodologies explicitly consider the problem of reusing ontology modules and the associated issues that come with this.

5.1 Motivation

Ontologies have been successfully employed in order to solve problems deriving from the management of shared, distributed knowledge, and the efficient integration of information across applications [38]. Much of this success depends on the ability to share and reuse existing ontologies [54]. Ontology construction is deemed to be a time consuming and labour intensive task, but is mediated by the possibility of reusing existing ontologies [33]. This is greatly facilitated by the existence of Ontology Libraries (for example, the DAML Ontology Library[1]), and the emergence of search engines such as Swoogle[2] and Watson[3] which support the retrieval of web ontologies.

Many ontology development methods and methodologies, such as the Ontology 101 method [100] and Methontology [55], include a reuse step in the ontology development lifecycle that allow Ontology Engineers to integrate into the ontology they are currently designing and implementing an ontology that has already been developed. The reuse of existing ontologies can occur at the design stage and at the implementation stage. Some ontology editors, for example Protégé 3 [102], allow the reuse of another ontology by

[1] http://www.daml.org/ontologies/
[2] http://swoogle.umbc.edu/
[3] http://watson.kmi.open.ac.uk/WatsonWUI/

including it in the model that is being designed. In Protégé this happens through the inclusion of other projects, which operates at the knowledge level. The web ontology language OWL[4] offers the possibility of importing an OWL ontology by means of the `<owl:imports>` statement, and many ontology development tools allow the import of OWL files. In both cases, the whole ontology is included, which can create a huge overhead. However, ontology developers might only be interested in a portion of the original ontology, especially when the ontology being reused is very large. For example, a developer might only be interested in concepts about and relating to diabetes; but they have to import a whole medical ontology, such as UMLS [5], in order to get what was required. Thus, there is a need for methods that allow for part of an ontology to be specified and reused; these ontology modularization methods are covered in Section 3.2.

Thus, an Ontology Engineer now has the means to extract a subset of the ontology, with the subset being the portion of the ontology that they wish to reuse. However, none of the existing Ontology Engineering methodologies provides guidance in how the reuse of an ontology module should be mediated; hence, the emergence of methodologies for ontology module reuse (Section 5.3).

5.2 Methodologies for Ontology Engineering

Ontology Engineering is to ontology as Software Engineering is to software. As such, Ontology Engineering "refers to the set of activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies" [54]. The literature provides many examples of Ontology Engineering Methodologies and the following Sections shall describe some of them.

5.2.1 Ontology 101

The Ontology 101 method [100] prescribes a series of steps for an Ontology Engineer to follow to iteratively construct an ontology. Those steps are the following:

Step 1: Determine the domain and scope of the ontology. This step should define what domain the ontology is going to cover, what use the ontology is going to be put to, what demands will be placed on the ontology and who will maintain it. A set of competency questions should also be written in this step; these are the questions that the ontology must answer.

Step 2: Consider reusing existing ontologies. In this step the Ontology Engineer should check if there are existing ontologies that cover the domain of interest or

[4] http://www.w3.org/2004/OWL/#specs

[5] An OWL translation is available from: http://swpatho.ag-nbi.de/owldata/ swpatho1/umlssn.owl

sources that can be extended and refined. This could be of particular concern if the ontology under development must be integrated into a larger system.

Step 3: Enumerate important terms in the ontology. The Ontology Engineer should list all the terms that the ontology needs to talk about and what needs to be said about these terms.

Step 4: Define the classes and the class hierarchy. Uschold and Gruninger [138] highlight several approaches to developing a class hierarchy: top-down, bottom-up and a combination of both. The Ontology 101 Methodology does not advocate one approach above another; but the class hierarchy should be based on the terms defined in Step 3.

Step 5: Define the properties of classes. It is likely that the classes alone will not meet the needs of the ontology so internal structure can be given to them via properties. Again inspiration for properties should be drawn from the terms defined in Step 3. The Methodology notes that the property should be attached to the most general class that is possible.

Step 6: Define the facets of the slots. This step attaches cardinalities to the properties previously defined. Data types of the relevant properties should also be set.

Step 7: Create instances. The last step is to populate the ontology with individuals. This should allow the ontology to be tested against the competency questions defined in Step 1.

These steps can be repeated and they do not necessarily have to be done in sequence. For example, property definitions could be added to the classes at the same time the class hierarchy is being created.

5.2.2 METHONTOLOGY

The METHONTOLOGY methodology [55] groups the development activities into three groups, which are:

Management Activities. These activities include scheduling, quality assurance and control. As these are not concerned directly with building the ontology they will not be discussed here; for more information see [55]

Support Activities. These activities are aimed to support the Development Activities. They include knowledge acquisition, integration, evaluation, etc. In this step the ontologies that are candidates for reuse are selected and evaluated; but no detail on how to do this is given in the methodology.

Development Activities. The five activities within this group form the backbone of the process to construct the ontology. These activities are:

1. *Specification.* This step identifies the users and uses of the ontology. This should determine the granularity level and scope of the ontology.
2. *Conceptualization.* This activity involves structuring the knowledge. METHONTOLOGY defines eleven tasks to follow in order to structure the knowledge.
   - *Task 1: Build glossary of terms.* A glossary of terms relevant to the domain is constructed. It is possible that multiple terms refer to the same concept.
   - *Task 2: Build concept taxonomies.* From the terms identified in the previous task a concept hierarchy is constructed. The constructed hierarchy should be checked for errors before moving on to the next task.
   - *Task 3: Build ad hoc binary relation diagrams.* This task identifies binary relations between the concepts in the previously built hierarchy. These relations include things such as saying that `Thesis hasAuthor Person`.
   - *Task 4: Build concept dictionary.* The concept dictionary contains all the domain concepts, their relations, their instances and their attributes.
   - *Task 5: Describe ad hoc binary relations.* Provides a description to the relation contained in the concept dictionary. This should add a name, the source and target concepts, the cardinality, and any mathematical properties that the relation might have.
   - *Task 6: Describe instance attributes.* Provides a description of the instance attributes contained in the concept dictionary. This should identify which concepts the attribute belongs to and specify the value and type of the relation.
   - *Task 7: Describe class attributes.* Provides a description of the class attributes contained in the concept dictionary. These attributes belong to the class rather than a particular instance, but their description is the same as for the previous task. An example of a possible class attribute is, `numberOfAuthors` for the class `PhDThesis` where the value would be 1.
   - *Task 8: Describe constants.* Provides a description of the constants contained in the concept dictionary. Constants specify information related to the domain and always take the same value. A possible constant for a thesis ontology is `maxNumberOfPages`, which would set the maximum number of pages in a thesis to its value.

- *Task 9: Describe formal axioms.* Formal axioms are logical expressions that are always true [54]. The description should include both a natural language description and a logical formulation of the axiom.
- *Task 10: Describe rules.* Rules are used to infer new knowledge in the ontology. The description is the same as for the formal axioms, but the logical expression should specify pre and post conditions; that is the set of conditions necessary to infer the new knowledge.
- *Task 11: Describe instances.* After all the above tasks have been completed instances can be added to the conceptualization. A description of an instance includes its name, the concept it is an instance of, and the attributes and their values that belong to the instance.

These tasks are in a similar vein to the steps defined by the Ontology 101 methodology (see Section 5.2.1).

3. *Formalization.* The conceptualization is put into a formal model. Depending on how the ontology is developed then this step maybe skipped. For example, one could use an ontology editor in the conceptualization step; thus also formalizing and implementing the ontology at the same time.
4. *Implementation.* The formal ontology model is implemented into an ontology language. Essentially the knowledge level model is transformed into a language level model.
5. *Maintenance.* Whilst stating that this activity should be carried out METHONTOLOGY does not prescribe a particular way to perform it. This is due to the fact that the METHONTOLOGY methodology can be used to develop ontologies for a large variety of applications/tasks/purposes and, as such, the maintenance strategy for each would need to be different.

5.2.3 On-To-Knowledge

The On-To-Knowledge methodology [125] is aimed at creating ontologies for intelligent access to large volumes of semi-structured and textual sources; as such the ontologies developed are dependent on the application. The methodology defines five processes to follow to, theses are:

Process 1: Feasibility Study. This serves as a basis for the next process. The feasibility study should be conducted following the method in Schreiber *et al.* [113]. This should cover both technical feasibility, organisational feasibility and project feasibility.

Process 2: Kickoff. This process defines the ontology requirements that describes: the domain and goal of the ontology, the design guidelines, available knowledge

sources, possible users and use cases. Competency questions could be created to add detail to the requirements. Doing this should informally identify the most important concepts and relations.

Process 3: Refinement. This process should produce a mature ontology conforming to the specification. This process is divided into two activities:

1. *Knowledge Elicitation.* The ontology defined in the previous step should be refined with the help of domain experts. The axioms added in this step should be agreed by all the experts.
2. *Formalization.* This activity encodes the ontology into the ontology language of choice.

Process 4: Evaluation. This process should show that the ontology developed in the previous Processes is useful. To do this two activities are required:

1. *Checking the requirements and competency questions.* Simply checking if the ontology meets the requirements outlined and if all the competency questions are answered correctly.
2. *Testing the ontology in the target application.* This should highlight possible issues in the context of the ontology being used. This might lead to a further refinement of the ontology.

Process 5: Maintenance. On-To-Knowledge proposes that the maintenance of the ontology is folded into the system software maintenance. However, it is still important to clarify who should do the maintenance and how it should be done; but no concrete steps are provided by the methodology.

5.2.4 NEON Methodology

The NEON methodology [53, 131, 130] is a scenario based methodology; rather than focusing on building a single isolated entity it is focussed on building a network of ontologies. A network of ontologies is a collection of ontologies related via relations; these relations can be via: mapping (see Section 6.2.4), modularization (see Chapter 3), version, etc. The scenarios fall around the backbone identified by METHONTOLOGY (see Section 5.2.2) with each scenario being broken down into different processes.

The nine scenarios identified, not intended to be exhaustive, for building ontology networks are:

1. *From scratch without reusing existing knowledge resources.* Development starts from scratch with no reuse of existing resources. The developers should clearly specify the requirements of the ontology, this should result in the ontology requirements specification document (ORSD). This can then be used to check that there are no relevant resources that could be reused.

2. *By reusing and reengineering non ontological resources.* Developers should decide which existing non ontological resource can be reused; this should been done according to the ORSD defined in the scenario above.

3. *By reusing ontological resources.* Existing ontology resources are reused, these resources include whole ontologies, possibly ontology modules, etc.

4. *By reusing and re-engineering ontological resources.* Developers reuse and re-engineer existing ontological resources.

5. *By reusing and merging ontological resources.* Arises only when several resources from the same domain are selected for reuse and there is a need to create a new resource.

6. *By reusing, merging and reengineering ontological resources.* Similar to the scenario above except that the developers decide to re-engineer the set of resources; for example, by translating the ontology language.

7. *By reusing ontology design patterns.* Developers access ontology design patterns [48] in repositories to reuse them[6]. Ontology design patterns, analogous to design patterns [45] in Software Engineering, provide templates that encode best practice principles for ontology engineering.

8. *By restructuring ontological resources.* Developers restructure an existing resource to integrate it into the network being built. One possible way to restructure is via ontology modularization (see Chapter 3).

9. *By localizing ontological resources.* Developers obtain a multilingual ontology by adapting an existing ontology.

5.3 Methodologies for Ontology Module Reuse

Whilst the methodologies described in Section 5.2 consider some notion of ontology reuse none of them explicitly consider how one should reuse an ontology module. Evidently there are similarities between reusing an ontology and an ontology module, after all, they are both ontologies. However, it is necessary to consider extra issues when reusing an ontology module, for example evaluating its competency or whether concepts mean the same; the following will explain two methodologies for reusing ontology modules and will outline the associated issues.

[6] `http://ontologydesignpatterns.org` provides an explanation and repository of ontology design patterns.

5.3.1 Doran *et al.* Methodology

Doran *et al.* [33] present a methodology for reusing an ontology module, see Figure 5.1. This descriptive methodology gives an Ontology Engineer guidance in how to best tackle the problem of ontology reuse via ontology module extraction. Whilst the steps in the methodology are fairly self-explanatory it is worth saying more about the 'ontology evaluation' and 'check competency' steps. Firstly, ontology evaluation is non-trivial and various methods for evaluating ontologies have been proposed, in addition to Chapter 4 both [13] and [51] provide useful surveys of ontology evaluation; see Section 6.7 for metrics to evaluate ontology modules. Secondly, checking the competency of an ontology module is an issue solely raised by Doran *et al.* [33]; an ontology module is competent if it fits the requirements stated for it, these requirement are usually, within Ontology Engineering, expressed as a set of competency questions.

With this in mind Doran *et al.*'s methodology consists of the following steps:

Define competency of module Before the module extraction process begins the Ontology Engineer needs to define competency questions for the module; these should state what the Ontology Engineer wants the ontology module to express. The competency questions could be expressed in terms of SPARQL queries. Defining the competency of the ontology module is important so the Ontology Engineer can ensure the extracted ontology module meets their requirements.

Ontology Selection The Ontology Engineer needs to select the relevant ontology that they wish to extract a module from, as the number of ontologies available continues to increase this step will become increasingly important, especially as ontologies can be discovered by browsing the available repositories or by using search engines like Swoogle[7] or Watson[8]. Sabou *et al.* [111] provide an extensive review on this area of research and also distinguish the elements of the ontology selection process; these are:

- *Information Need.* The application scenario that the ontology is being selected for will place different needs upon the information that is required of it. For example, it may be required that the ontology can fulfil a set of queries, or that it contains certain logical axioms.
- *Selection Criteria.* The ontologies need to be evaluated to ensure they meet selection criteria such as topic coverage, ontology structure, ontology popularity, etc.
- *Ontology Library.* Ontologies can be selected from a pre-existing library, for example Swoogle or Watson. The libraries should facilitate the selection of

[7] `http://swoogle.umbc.edu/`
[8] `http://watson.kmi.open.ac.uk`

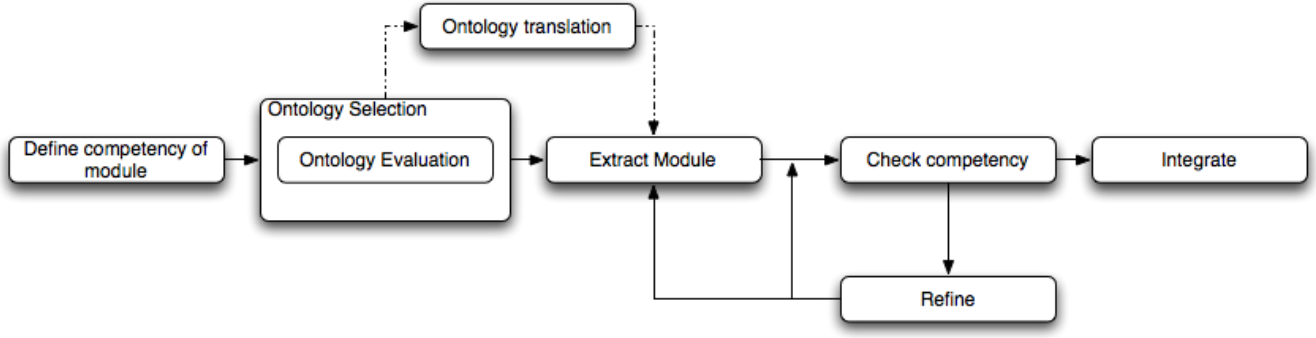

Figure 5.1: A methodology for reusing an ontology module. [33]

an ontology by allowing ontology statistics to be compared. For example, which ontology is smaller or the expressivity levels of the different ontologies.

- *Output.* This should be a list of ontologies that could possibly be selected for reuse, possibly ranked according to criteria suggested by the Ontology Engineer.

The major effort in this task will be carrying out the ontology evaluation, for more on ontology evaluation see Section 4.2.

- Ontology Evaluation The Ontology Engineer needs to evaluate any ontology that is discovered to see how suitable it is for reuse. There are several criteria that can be applied to evaluate an ontologies suitability for reuse. One criterion could be the ontologies popularity; if an ontology is popular and widely used then it is likely to be correct and of some quality. Another criterion could be to check the ontology against the competency questions the Ontology Engineer defined for the ontology module. If the ontology does not meet the competency questions then the ontology module will most likely not either.

Ontology Translation This step aims to change the representational language used to implement the ontology. This may be necessary to make the ontology compatible with the module extraction process being used. However, the application of this step requires some care to be taken by the Ontology Engineer. Translating an ontology from a more expressive formalism such as OWL-DL to RDFS will cause the loss of some ontological features, such as restrictions on the classes or cardinality constraints. Conversely, the translation from a less expressive formalism into a more expressive one will cause the Ontology Engineer to perform a further enrichment step that is needed to complement the less expressive model obtained through translation. Ontolingua [58] and WebODE [21] provide intermediary ontologies that allow an ontology in one formalism to be translated into a different formalism.

A complete treatment of the problems associated to ontology translation can be found in [21]; they are too numerous too detail in this thesis and are worthy of separate treatment.

Extract module The ontology module is extracted from the selected ontology. The amount of effort required by the Ontology Engineer in this step will depend on the method of extraction that is being used, see Section 3.4 for the details of different extraction techniques.

Check Competency The extracted ontology module is checked against the competency questions, in order to see if they are met. One possible way to automatically

check whether the competency questions are met is by formulating them as a set of SPARQL queries. If the competency questions are met then the Ontology Engineer can integrate the ontology module into the ontology that they are constructing. If the competency questions are not met then the Ontology Engineer should refine the ontology module.

Refine When the competency questions are not satisfied then the Ontology Engineer should refine the module in a way that ensures the competency questions are met. This refinement could take two possible forms. Firstly, the Ontology Engineer could alter the ontology module, i.e. by adding or removing statements, in order to make it meet the requirements defined by the competency questions. However, under certain circumstances the Ontology Engineer may decide that this would take too much effort and thus decide to run the module extraction process again, or indeed run a different module extraction process.

Integrate Once the ontology module satisfies the competency questions it can be integrated into the ontology being constructed.

5.3.2 Logic Based Methodology

Jimenez-Ruiz *et al.* [79] present a methodology[9] which provides Ontology Engineers with a precise set of guidelines to follow in order to ensure that certain logical properties such as safety, the meaning of the extracted terms remains the same, and coverage, everything the ontology says about the extracted terms is in the module; see Section 3.4.2) for the definitions.

The methodology is split into two phases: *offline* and *online*, see Figure 5.2. The two phases are detailed below:

Offline Phase The Ontology Engineer has the ontology, $\mathcal{T}$, that is being developed. The next step is for the Ontology Engineer to specify the signature, $\mathbf{S}$, that is to be reused from the external ontologies; each element of the signature is associated to the external ontology it was taken from. Thus, $\mathbf{S} = \mathbf{S}_1 \uplus \ldots \uplus \mathbf{S}_i$; where each $\mathbf{S}_i \subseteq \mathbf{S}$ represents the symbols from the external ontology $\mathcal{T}_i'$.

In the next step the Ontology Engineer needs to decide how each $\mathbf{S}_i$ will be reused. Will it be generalised ($\top$-locality) or specialised ($\bot$-locality)[10]? This step is required to ensure safety.

Online Phase In this step the Ontology Engineer imports the locality based module from each of the external ontologies identified in the previous step. Once the external ontology is loaded, implying that the Ontology Engineer is committing

[9] This methodology is implemented as a Protégé plugin

[10] See Section 3.4.2 for more information about $\top$-locality and $\bot$-locality

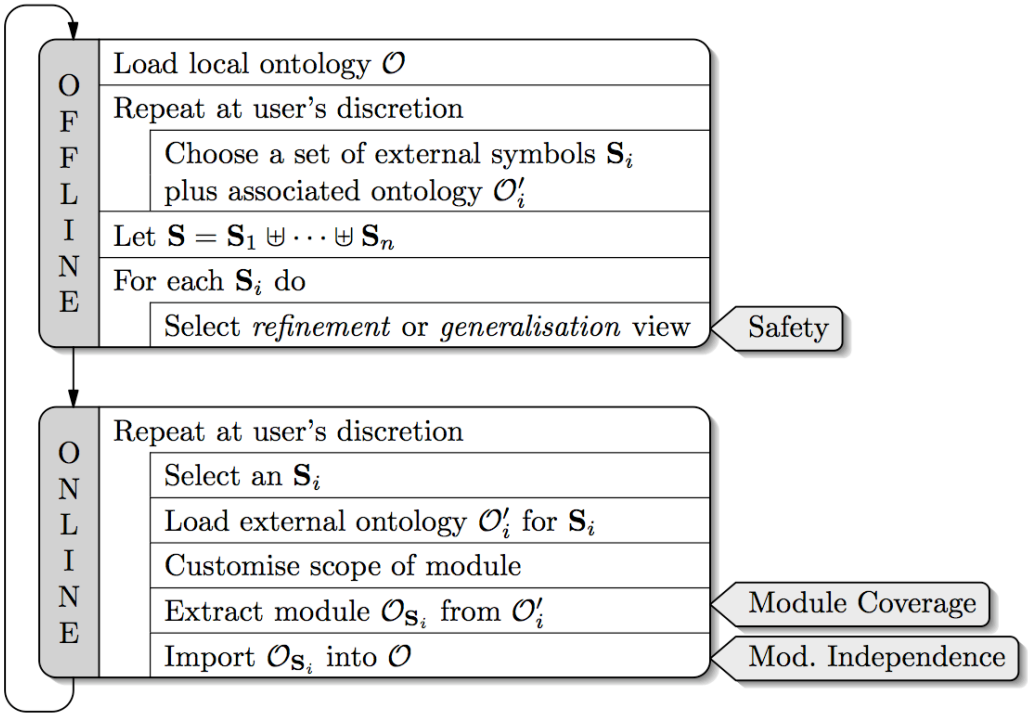

Figure 5.2: Logic based methodology. Taken from [79]

to this particular version, there is an opportunity for $\mathbf{S}_i$ to be extended. Then the actual module, $\mathcal{T}_{\mathbf{S}_i}$, is extracted from the external ontology; which is guaranteed to cover the elements specified by $\mathbf{S}_i$.

The final step is to import the module previously extracted into the ontology; thus we now have a new ontology $\mathcal{T} \cup \mathcal{T}_{\mathbf{S}_i}$. It is possible that this inclusion compromises the safety guarantee of the other external ontologies. This is considered undesirable so a *module independence* guarantee is required; this states that given an ontology $\mathcal{T}$ and two signatures $\mathbf{S}_1$, $\mathbf{S}_2$ then $\mathcal{T}$ guarantees module independence if for all $\mathcal{T}_1$ with $Sig(\mathcal{T}) \cap Sig(\mathcal{T}) \subseteq \mathbf{S}_1$ it holds that $\mathcal{T} \cup \mathcal{T}_1$ is safe for $\mathbf{S}_2$.

Chapter 6

Applying Ontology Modularization to the Dynamic Selection of Ontology Alignments in Multi-Agent Systems

'Arguments are to be avoided; they are always vulgar and often convincing.' - Oscar Wilde

Summary This chapter considers the practical, and novel, application of ontology modularization to the problem of the dynamic selection of ontology alignments in multi-agent systems where modularization is used as as space reduction mechanism. First some preliminary information is given concerning agents, multi-agent systems and how ontology alignments can overcome the problem of semantic heterogeneity. Next the definition of the argumentation framework is given, including the value-based argumentation framework that is used in the rest of the chapter. From this it is possible to show how argumentation can be used to argue over ontology alignments and, thus, how modularization can be used as a space reduction mechanism to this process. It is possible that there is information loss when applying modularization to this problem. This is discussed and two solutions are proposed. Finally, an evaluation is presented that shows that modularization successfully reduces the space, but does not have a negative impact on the quality of the alignment agreed.

6.1 Motivation

Interacting systems are now the norm in the everyday computing world, even trivial systems contain sub-systems, termed agents, that need to interact [146]. Furthermore, as these systems are likely to be distributed and have decentralised management, systems such as the Internet, then it is challenging to impose global constraints upon the system. This type of environment is called an open environment.

An open environment places no constraints on an agent's, i.e. a software component, ability to enter or leave the environment, furthermore no assumptions can be made about the agents that will be encountered now or in the future. We assume that each agent has its own ontology, its own model of the world. Each agent having its own ontology, about which no assumptions can be made, leads to the problem of *semantic heterogeneity*. That is agents have differing specifications or conceptualizations of concepts in their ontology [142].

In order to enable inter-agent communication semantic heterogeneity must be overcome. Effective communication within open and dynamic environments is dependent on the ability of *agents* to reach a mutual understanding over a set of messages, where no prior assumptions can be made on the vocabulary used to communicate. Unlike small, closed environments (where all the components are known at design time), open, Web-scale environments are typically characterised by large numbers of services which are continually evolving or appearing, and where semantic heterogeneity is the norm.

Semantic heterogeneity can be divided into two levels: language and ontology [81]. Language level heterogeneity occurs when two ontologies are written in different languages, for example description logic and first-order logic. Ontology level heterogeneity occurs when either there is a mismatch in the conceptualization of the ontology, for example a difference in the extensions of the same concept; or a difference in the way that the conceptualization was specified, for example giving the same name to two different concepts.

Thus, semantic heterogeneity means few assumptions can be made about the services on offer at any time, the way in which they are modeled, or the terminology or vocabulary that they use. In such cases, it becomes imperative to specify the explicit vocabularies or ontologies used to facilitate meaningful communication as environments open up, or the heterogeneity of large systems increases. This has been facilitated by the emergence of standards for representing ontologies and optimised reasoners capable of processing them within a tractable timeframe [134].

In addition, transactions should be interpreted by both service providers and consumers based on the underlying semantics of the messages themselves, and thus these agents should resolve any type of mismatch that may exist due to the use of different, but conceptually overlapping ontologies. However, this reconciliation has to be achieved automatically and at run-time (without human intervention) if such components are to transact as the size of the environment grows.

Early systems avoided the problem of ontological heterogeneity by relying on the existence of a shared ontology, or simply assuming that a canonical alignment, possibly defined at design time, could be used to resolve the mismatches. However, such assumptions work only when the environment is (semi-) closed and carefully managed, and no longer hold in open environments where a plethora of ontologies exist. How-

ever, semantic heterogeneity can be overcome by using ontology alignment (see Section 6.2.4). An ontology alignment defines a set of relations between the entities of two ontologies, thus, allowing entities in one ontology to be explicitly related to entities in another ontology. Unfortunately, however, the techniques for ontology alignment generation either take a long time, for example [76], or are user-led, for example [101]. These constraints prevent the agents from dynamically generating the alignments as and when they are needed.

However, we can assume that the ontology alignments exist somewhere in the environment, for example from an Ontology Alignment Service (OAS) [84], thus, the agents will now be able to acquire the relevant alignment when needed. Relevant alignments are those that align the ontologies of the two agents who wish to cooperate. There is a problem, however, as it is possible that more than one alignment exists between the two agent ontologies, and so the agents now have the problem of deciding which to use. Laera *et al.* [85] define a framework, based on argumentation (see Section 6.3). Argumentation allows the agents to argue over the ontology alignments and reach a mutually acceptable agreement. They are able to argue for or against a mapping based on their preferences. However, the complexity of this process can reach Π_2^p-complete [37] making it impractical.

In this Chapter we propose a new task for ontology modularization (see Section 3.2) that is to reduce the search space that the agents have to argue over in order to reach an agreement over ontology alignments. However, before it is necessary to consider what agents and multi-agent systems are, as well as characterising the environment they operate in. This is detailed in Section 6.2 along with an outline of the problem of semantic heterogeneity and an explanation of how ontology alignments overcome it. Section 6.3 provides an introduction to argumentation and details the value-based argumentation framework. It is then possible in Section 6.4 to show how agents can argue over ontology alignments and how modularization can act as a space reduction mechanism. An illustrative example is given in Section 6.5. The application of modularization to argumentation over ontology alignments can lead to information loss and this is outlined in Section 6.6, alongside two possible solutions. Finally, an evaluation of applying ontology modularization to the problem of arguing over ontology alignments is provided in Section 6.7.

6.2 Preliminaries

Before detailing how ontology modularization can be used as a space reduction mechanism for ontology modularization it is necessary to cover some preliminary information concerning multi-agent systems, specifically those in open distributed environments, and ontology alignment. The purpose here is not to give a complete overview of these fields, but to cover those aspects relevant to this Chapter.

6.2.1 What is an Agent?

Wooldridge and Jennings [147] present two notions of agent: *weak* and *strong*. The weak notion of agency is characterised by the following properties:

- *Autonomy.* Autonomous agents must be capable of carrying out actions without direct human intervention. This is essential for *situated* agents; those agents in an environment shared by others.
- *Social ability.* Social agents interact and co-operate with each other in order to share resources or achieve their goals.
- *Reactivity.* Reactive agents perceive their environment and can react to changes that occur in the environment in a timely manner. (See Section 6.2.3 for more on agent environments.)
- *Pro-activeness.* Pro-active agents have goal-directed behaviour and do not merely react to their environment but create opportunities to pursue their goals.

The stronger notion of agency ascribes mentalistic properties to the agents, such as knowledge, belief, intention and obligation [118]. Subsets of theses properties will be used depending on the situation. For the purposes of this thesis the only property we need to consider an agent to have is knowledge; that is knowledge encoded in a description logic ontology.

The above is nicely summarised by Wooldridge [146] and this is the definition of agent adopted for the this Chapter.

> "An *agent* is a computer system that is *situated* in some environment, and that is capable of *autonomous action* in this environment in order to meet its design objectives."

This thesis assumes that one of the design objectives of the agents is *rationality*. A rational agent takes the action that will cause it to the be the most successful [109].

6.2.2 What is a Multi-Agent system?

Interacting systems are the norm in the everyday computing world, even trivial systems contain sub-systems that need to interact [146]. Hence, a multi-agent system (MAS) is a system composed of multiple (semi-) autonomous components [78]. Sycara [132] identifies the following main characteristics of MAS:

- *Incomplete Information.* Each agent has a limited viewpoint and does not possess the information or capabilities needed to solve the problem by itself.
- *No Global Control.* The agents behaviour is a result of social rules and the interactions that result, rather than being dictated by a central authority.

- *Decentralised.* Resources in the system are divided and distributed.

This advances the definition beyond a simple collection of agents, who happen to share an environment, toward that of an organization with rules and interactions that allow the members to co-operate and collaborate.

Before moving on to characterise open distributed environments it is important to note what class of MAS this Chapter assumes. Zambonelli, Jennings and Wooldridge [149] identify the following two classes of MAS:

- *Distributed Problem Solving Systems.* Agents are explicitly designed to *co-operate* to achieve a common goal.

- *Open Systems.* A term first coined by Hewitt [71] in open systems agents are not necessarily explicitly designed to share a common goal and can dynamically leave and enter the system. Here the agents can be said to be in *competition*.

This Chapter assumes that the MAS under consideration are open systems because no assumptions are being made about the nature of the agents in the system and the ontologies that they possess. This means that the solutions proposed here could be applicable in real-world scenarios such as the Semantic Web. The Semantic Web [134] is an open, distributed environment where few assumptions are made about the participants. The participants can enter and leave the environment at will and have their own separate internal ontological models.

6.2.3 Characterisation of an Open Environment

Section 6.2.2 introduced open multi-agent systems (MAS) as systems where the agents do not necessarily share a goal and are free to enter and leave the system as they wish. Now we need to consider what characterises this type of environment in order to understand the constraints imposed upon the solution to dynamic reconciliation presented in this Chapter. Russell and Norvig [109] provide the following classification of environment properties.

Fully observable vs. partially observable. If an environment is fully observable then an agent is able to obtain the complete state of the environment at any point in time. Most real-world environments, such as the Internet, are partially-observable.

Deterministic vs. stochastic. An environment is deterministic if the next state of the environment is completely determined by the current state and the action executed by the agent; there is no uncertainty with respect to the result of the agent's action. Conversely, in stochastic environments agents only have partial control over their environment; they can not predict what actions other agents will take and how this will affect the environment.

Episodic vs. sequential. If an environment is episodic then the agent's experience is divided into atomic episodes where an action taken in the current episode does not depend on previous actions. Whereas, in sequential environments an agent's past decisions can affect future decisions; thus, requiring the agent to think ahead. For example, consider a game of chess where early moves affect how the game progresses at a later stage.

Static vs. dynamic. If the environment can change whilst an agent is deciding what to do then it is dynamic, but if it only changes as a result of the agent's action then it is static. The environment can be termed *strategic* if it is deterministic except for the actions of other agents.

Discrete vs. continuous. If an environment is discrete then there are a fixed, finite number of states and actions. A continuous environment has a range of continuous values that change over time. For example, consider driving a car in a city.

Single agent vs. multiagent. A single agent system is one in which only one agent exists in the environment. For example, a crossword puzzle solving agent. Section 6.2.1 characterises these agents as being in an environment where they are capable of action to achieve their goal. Whereas, multiagent environments are those in which there are a collection of agents each capable of autonomous action, as detailed in Section 6.2.2.

Thus, we are now able to properly characterise the type of environment under consideration in this thesis. It is defined as follows:

Definition (Open Multi-Agent System) An open multi-agent system is an open system, where agents are free to enter and leave as they choose, and the environment is characterised as partially observable, stochastic, sequential, dynamic and continuous.

6.2.4 Semantic Heterogeneity

As noted in Section 6.1 the problem of semantic heterogeneity can be divided into language level and ontology level mismatches. For this Chapter we are not concerned with language level mismatches as we assume that the agents adhere to the OWL standard (see Section 2.3). Thus, we will only cover ontology level mismatches here.

Klein [81] defines ontology level mismatches as the differences that occur when either there is a mismatch in the conceptualization of the ontology, for example a difference in the extensions of the same concept; or a difference in the way that the conceptualization was specified, for example giving the same name to two different concepts. Visser *et al.* [142] postulate that ontology level mismatches can occur in the conceptualization and explication stages of ontology design.

An explication mismatch would occur when the concepts are specified in a different way. Whereas, a conceptualization mismatch would occur when there is a difference in the way the domain is modeled, which could result in different concepts or different relations among those concepts. For example, one ontology concerning `Animals` organises itself around the concepts `Birds` and `Mammals`, and another ontology concerning `Animals` organises itself around the concepts `Herbivores` and `Carnivores` [142]. This results in two ontologies with a conceptualization mismatch, they are modelling the same thing in different ways.

Klein [81] further subdivides both explication and conceptualization mismatches. Explication mismatches are subdivided into the following:

- *Paradigm.* Two ontologies model the same domain but with different representation paradigms. For example, different paradigms can be used to model time, is it a continuum or a set of discrete points? This mismatch could also occur because a different knowledge representation paradigm has been adopted. For example, using temporal logic or Allen's interval algebra [2] to model time.

- *Concept description.* Two ontologies model the same concept in a different way. See the `Animals` example above.

- *Synonym terms.* Two ontologies have the same entity but with different labels. For example, `Car` and `Automobile`

- *Homonym terms.* Two ontologies have an entity with the same label but different semantics. For example, `Graph` in Computer Science has a different meaning to the widely used meaning of a diagram showing relations between variables.

- *Encoding values.* Two ontologies encode the same values in different ways. For example, in the US dates are represented as `mm/dd/yy`, but in the UK they are represented as `dd/mm/yy`.

Conceptualization mismatches are subdivided into the following:

- *Concept scope.* Two ontologies have a different extension (set of instances) for the same concept. For example, different UK Universities would have different concept scopes for the `Lecturer` concept.

- *Model coverage and granularity.* Two ontologies cover different portions of the same domain. For example, a difference in coverage would mean that an ontology about accommodation might or might not include `Hostels`, and a difference in granularity would mean that the ontology does or does not distinguish between the different types of `Room`.

Euzenat and Shvaiko [42] reduce these types of heterogeneity to those that which are addressed by ontology alignment, these are:

- *Terminological.* This refers to synonym and homonym mismatches.
- *Conceptual.* This refers to differences in coverage, granularity and scope.

What is Ontology Alignment?

When few constraints are placed on the agents in the environment it is unreasonable to assume that they share the same ontology, it is reasonable to assume that each agent has its own ontology. This leads to semantic heterogeneity because each agent has its own model of the world and possesses no way to express what it knows in a way that the other agents can understand. An ontology alignment expresses the mapping[1] between entities belonging to two different ontologies [42]. Thus, ontology alignments should help overcome semantic heterogeneity (see Section 6.2.4) by allowing agents to express the terms they wish to communicate in terms the other agent understands. Thus, an ontology mapping [42] is defined as:

Definition (Ontology Mapping) Given two ontologies O and O' a mapping is the tuple: $m = \{e, e', n, r\}$, such that:

- $e \in O$ and $e' \in O'$ where e and e' are concepts, roles or individuals;
- n is a degree of confidence in the mapping m;
- r is the relation ($\equiv, \sqsubseteq$, etc.) holding between e and e'.

It is possible that multiple mappings exist between e and e', with different relations, r, or confidence values, n[2]. An ontology alignment can now be defined as a set of ontology mappings.

Definition (Ontology Alignment) Given two ontologies O and O' an alignment is a set of mappings.

Thus, an ontology alignment allows agents with different ontologies to express relations between entities in their ontologies. However, it is possible that multiple ontology alignments exist for a pair of ontologies, as such, agents now have the problem of deciding which to use to overcome the problem of semantic heterogeneity.

An Example of an Ontology Alignment

This Section provides a brief example of an ontology alignment to illustrate how they overcome semantic heterogeneity. Figure 6.1 depicts two ontologies in the domain of publishing; they express similar terms using different conceptualizations. In addition,

[1] N.B. Correspondence is used a synonym for mapping in the literature.

[2] Note that the n does not provide a fuzzy characterisaton of the expressed relation, but reflects a confidence value in that relation holding between the entities.

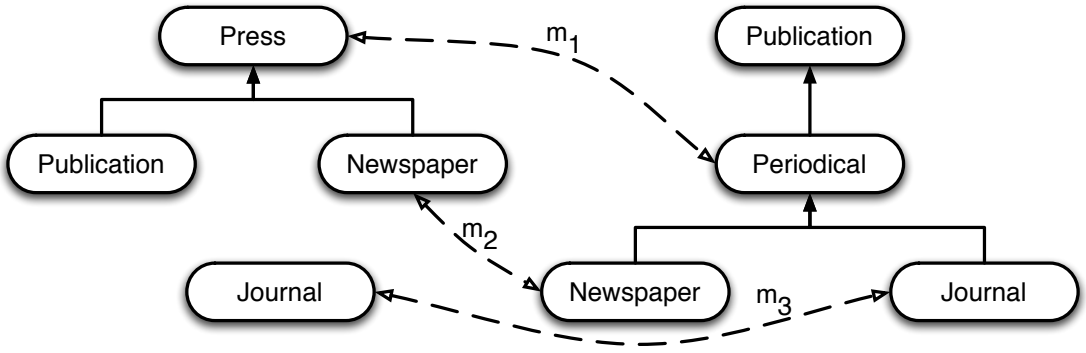

Figure 6.1: An example of an ontology alignment.

Figure 6.1 also shows an example of an ontology alignment consisting of the following three ontology mappings:

1. $m_1 = \{\texttt{PressPeriodical}, 0.45, \equiv\}$ - Maps `Press` to `Periodical` with an $\equiv$ relation with a confidence of 0.45.

2. $m_2 = \{\texttt{Newspaper}, \texttt{Newspaper}, 0.85, \equiv\}$ - Maps `Newspaper` to `Newspaper` with an $\equiv$ relation with a confidence of 0.85.

3. $m_3 = \{\texttt{Journal}, \texttt{Journal}, 0.85, \equiv\}$ - Maps `Journal` to `Journal` with an $\equiv$ relation with a confidence of 0.85.

These mappings overcome the problem of semantic heterogeneity by mapping entities in one ontology to entities in another. Thus, when the agents wish to communicate using these two ontologies they can do so by using the mappings to translate the messages into entities that they understand.

Generating Ontology Alignments

There are numerous systems for generating ontology alignments, for example [76, 101, 1, 39]. Each alignment system has different underlying methods for computing the alignments; but, generally speaking, they can be classified according to the following categories suggested by Euzenat and Shvaiko [42]:

Terminological Methods. Compares the lexical labels of the entities, including the name and comments. This comparison can be done using string similarity [19] or by using some linguistic knowledge, for example by using WordNet [95].

Internal Structural Methods. Compares the internal structure of the entities. In DL (see Section 2.3) terms this means considering criteria such as the value range of properties, their cardinality, and their transitivity, etc.

External Structural Methods. Compare the relations of the entities with other entities. For example, their position within a taxonomy. The intuition here is that if entities in two ontologies are similar then their neighbours may also be similar.

Extensional Methods. Compare the known extension of the entities. In DL (see Section 2.3) terms this means comparing the given instances of the entities.

Semantic Methods. Compare the interpretations of the entities. In DL (see Section 2.3) terms this means comparing the interpretations ($\mathcal{I}$). These methods are limited because they require the existence of some mapping(s) between the ontologies to compute more; if no mappings are given then these methods cannot derive them.

These methods can be combined by an ontology alignment technique in different ways and, therefore, many ontology alignment techniques exist in the literature. The existence of several alternative methods led to the establishment of the Ontology Alignment Evaluation Initiative (OAEI)[3]. The OAEI aims to compare the different alignment techniques through controlled experimentation over a standardised data set. The OAEI has run a competition annually since 2004.

6.3 What is Argumentation?

Ontology alignments (Section 6.2.4) can overcome the problem of semantic heterogeneity. However, in open environments (see Section 6.2.3) it is possible that multiple alignments will exist between the ontologies of two agents. Thus, the agents need to rationally reach an agreement. Various techniques allow agents to reach an agreement, but in this thesis we focus on the use of argumentation. Argumentation is a process of systematic reasoning in support of an idea, action or theory.

Argumentation facilitates the agents in reaching an agreement over the mappings they will use to communicate. Within Artificial Intelligence it has been applied in a number of areas as a way to approach and frame problems, and to develop novel solutions. In the context of this thesis it is used as a way to bring about a mutually acceptable agreement, where agents have incomplete knowledge [84].

6.3.1 Argumentation Framework

This thesis adopts the framework used by Laera *et al.* [85], which is based upon Bench-Capon's Value-Based Argument Framework (VAF) [5], that introduces the notions of *audience* and *preference values*. An audience represents a group of agents who share the same preferences over a set of values, with a single value being assigned to each argument. The VAF is based on the seminal work by Dung [36]. Dung showed that

[3] `http://oaei.ontologymatching.org/`

many forms of non-monotonic reasoning and logic programming are special forms of his argumentation theory.

An Argumentation Framework [36] is defined as follows:

Definition (Argumentation Framework) An Argumentation Framework (AF) is a pair $AF = \langle AR, A \rangle$, where AR is a set of arguments $A \subset AR \times AR$ is the *attack* relation 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".

Let R, S be subsets of AR, we say that:

(a) $s \in S$ is *attacked* by R if there is some $r \in R$ such that $\langle r, s \rangle \in A$.

(b) $x \in AR$ is *acceptable* with respect to S if for every $y \in AR$ that attacks x there is some $z \in S$ that attacks y.

(c) S is *conflict-free* if no argument in S is attacked by any other argument in S.

(d) A conflict-free set is *admissible* if every $y \in S$ is acceptable with respect to S.

(e) S is a *preferred extension* if it is a maximal (with respect to set inclusion, $\subseteq$) admissible set.

A preferred extension represents a consistent position within an AF; it is defensible against all attacks and it cannot be further extended without becoming inadmissible. It is important to note that AF can be represented as a directed graph where the vertices correspond to elements of AR and edges correspond to elements of A. For example, *"x attacks y"* would give the attack graph shown in Figure 6.2.

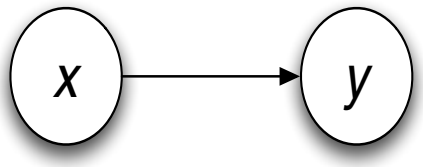

Figure 6.2: A simple example of an attack graph.

6.3.2 Value-Based Argumentation Framework (VAF)

In Dung's framework [36] attacks always succeed; in essence they are all given equal value. For deductive arguments this suffices, but in our scenario, ontology alignment negotiation, the persuasiveness of an argument could change depending on the audience, where an audience represents a certain set of preferences. One alternative is to use an extension of Dung's framework called Value-Based Argumentation Framework (VAF) [5], which assigns 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. Thus, it is possible to systematically relate strengths of arguments to their motivations and to accommodate different audience interests.

Definition (Value-Based Argumentation Framework) A Value-Based Argumentation Framework (VAF) is defined as $\langle AR, A, \mathcal{V}, \eta \rangle$, where:

- $\langle AR, A \rangle$ is an argumentation framework;
- $\mathcal{V}$ is a set of k *values* which represent the types of arguments;
- $\eta : AR \rightarrow \mathcal{V}$ is a mapping that associates a value $\eta(x) \in \mathcal{V}$ with each argument $x \in AR$.

The notion of *audience* is central to the VAF. Audiences are individuated by their preferences over the values. Thus, potentially, there are as many audiences as there are orderings of $\mathcal{V}$[4]. The set of arguments is assessed by each audience in accordance to its preferences. An audience is defined as follows:

Definition (Audience) An *audience* for a VAF is a binary relation $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V}$ whose irreflexive transitive closure, $\mathcal{R}^*$, is asymetric, 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 $(v_i, v_j) \in \mathcal{R}^*$

This notion allows us to consider that different agents (represented by an audience) can have different perspectives on the same candidate mapping. Thus, the VAF [5] defines what it means for an argument to be acceptable relative to some audience; it is defined as follows:

Definition (Argument Acceptability) Let $\langle AR, A, \mathcal{V}, \eta \rangle$ be a VAF, with R and S as subsets of AR, and an audience $\mathcal{R}$:

(a) For $x, y \in AR$, x is a *successful attack* on y with respect to $\mathcal{R}$ if $(x, y) \in A$ and $\eta(y) \not\succ_{\mathcal{R}} \eta(x)$.

(b) $x \in AR$ is *acceptable* with respect to S with respect to $\mathcal{R}$ 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) S is *conflict-free* with respect to $\mathcal{R}$ if for every $(x, y) \in S \times S$, either $(x, y) \notin A$ or $\eta(y) \succ_{\mathcal{R}} \eta(x)$

(d) A conflict-free set S is *admissible* with respect to $\mathcal{R}$ if every $x \in S$ is acceptable to S with respect to $\mathcal{R}$

[4]Number of audiences $= |\mathcal{V}|!$

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

(f) $x \in AR$ is *subjectively acceptable* if and only if x appears in the preferred extension for some specific audience.

(g) $x \in AR$ is *objectively acceptable* if and only if x appears in the preferred extension for every specific audience.

(h) $x \in AR$ is *indefensible* if it is neither subjectively nor objectively acceptable.

6.4 Argumentation over Ontology Alignments

Laera *et al.* [85] adopt the VAF (see Section 6.3.2), allowing agents to express preferences for different mapping types, and restrict the arguments to those concerning ontology mappings allowing agents to explicate their mapping choices. An agent is considered to be:

Definition (Agent) An agent, Ag_i, is characterised by the tuple $< O_i, VAF_i, Pref_i, \epsilon_i >$ where O_i is an ontology, VAF_i is the Value-based argumentation framework, $Pref_i$ is the private pre-ordering of preferences over the possible values, $\mathcal{V}$, and ϵ_i is the private threshold value.

Laera *et al.*define the arguments as follows:

Definition (Mapping Argument) An argument $x \in AR$ is a triple $x = \langle G, m, \sigma \rangle$ where:

- m is a mapping
- G is the grounds justifying the *prima facie* belief that the mapping does or does not hold
- σ is one of $\{+, -\}$ depending on whether the argument is that m does or does not hold

Laera [84] presents an algorithm for the agents to generate the arguments. The agents will argue for $(+)$ a mapping if it is the agent's most preferred value in $\mathcal{V}$ and the degree of confidence, n, of the mapping is greater than the agents private threshold, ϵ; otherwise the agent will argue against $(-)$ the mapping.

Laera *et al.* also address the notion of attack; x is attacked by the assertion of its negation, $\neg x$, this counter-attack is defined as follows:

Definition (Counter Attack) An argument $x \in AR$ attacks an argument $y \in AR$ if x and y are arguments for the same mapping, m, but with different σ.

For example, if $x = \langle G_1, m, + \rangle$ and $y = \langle G_1, m, - \rangle$, x counter-argues y and vice-versa.

Laera *et al.*also provide the elements required for instantiating $\mathcal{V}$, these relate to the methods used to generate the mappings outlined in Section 6.2.4,they are as follows:

M - Semantic. The sets of models for two entities do or do not compare.

IS - Internal Structural. Two entities share more or less internal structure.

ES - External Structural. The set of relations each of the two entities have with other entities do or do not compare.

T - Terminological. The names of the two entities share more or less lexical features.

E - Extensional. The known extensions of the two entities do or do not compare.

The agents can now express, and exchange, their arguments about ontology mappings and decide from their perspective, audience, what arguments are in their preferred extension; but the agents still need to reach a mutually acceptable position with regards to what ontology alignment they actually agree upon. Laera *et al.*define the notion of *agreed* and *agreeable* alignment as follows:

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

Definition (Agreeable Alignment) An *agreeable alignment* extends the agreed alignments with those mappings supported by arguments in some preferred extensions of every agent.

Thus, a mapping is *rejected* if it is in neither the agreed nor agreeable alignment. Given the context of agent communication it is rational for the agents to accept as many candidate mappings as possible [85], thus both sets of alignments are considered. The agents should only completely disagree when they want the opposite, indeed, the agents gain little by arguing and not reaching some kind of agreement.

6.4.1 Combining Ontology Modularization and Argumentation

Ontology modularization (see Chapter 3) can be used as a pre-processing step to improve the efficiency of an argumentation framework, when used to search the space of all candidate ontology mappings [35].When two agents communicate, only the initiating agent (Ag_1) is aware of its task, and consequently, what concepts are relevant to this task. It can therefore select these relevant concepts within the signature of the desired ontology module. The signature of the resulting ontology module can then be used to filter the correspondences, and consequently the number of arguments necessary within the argumentation process. The steps below describe this process whilst

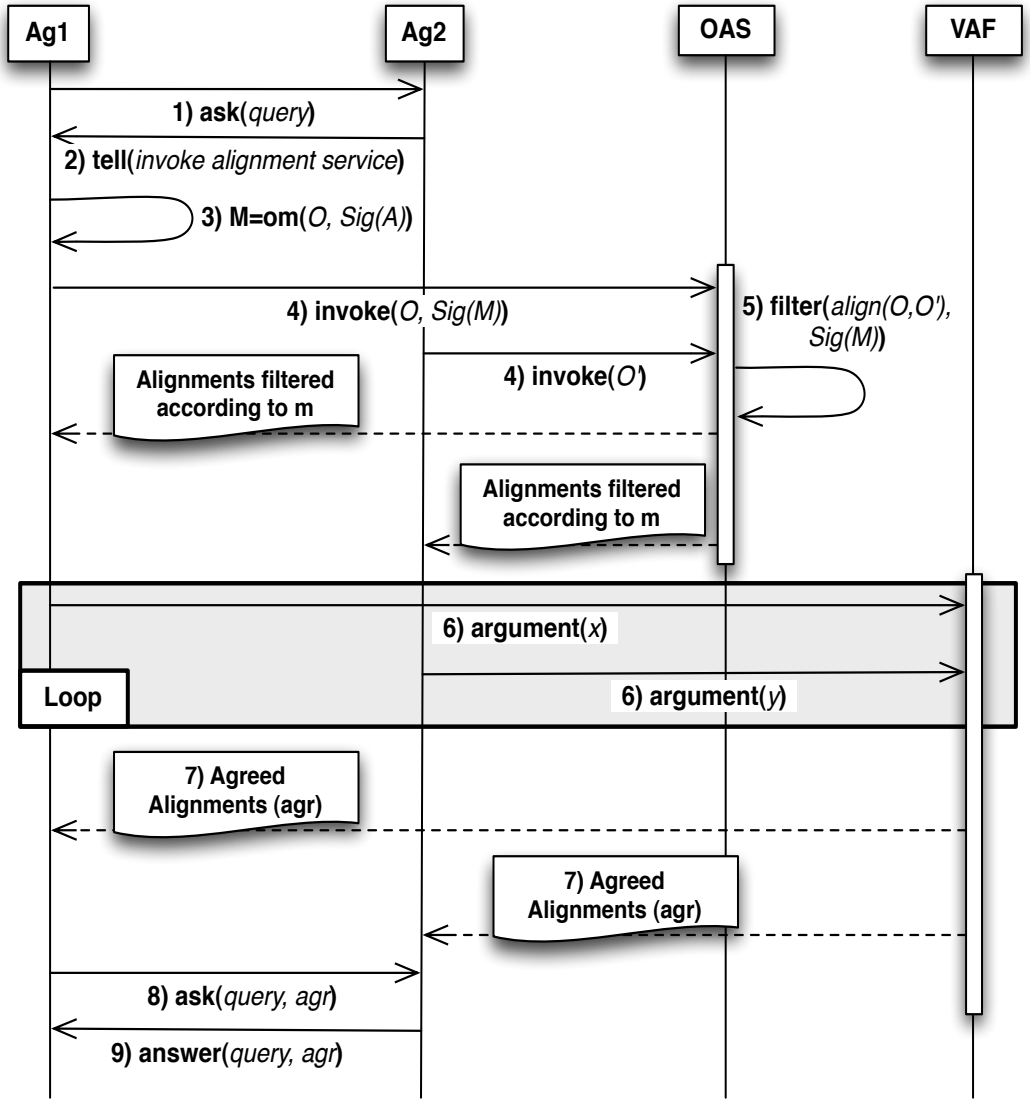

Figure 6.3: UML Sequence Diagram of Ontology Modularization and Argumentation.

Figure 6.3 depicts the process as a UML Sequence Diagram; it is assumed that two agents, Ag_1 and Ag_2 have ontologies O and O' respectively.

1. Ag_1 asks a query, $query(A \in Sig(O))$, to Ag_2.

2. Ag_2 does not understand the query, $A \notin Sig(O')$, and informs Ag_1 they need to use an Ontology Alignment Service (OAS)

3. Ag_1 produces, $om(O, Sig(A))$, an ontology module, M, to cover the concepts required for its task.

4. Ag_1 and Ag_2 invoke the OAS. Ag_1 sends its ontology, O and the signature of M, $Sig(M)$.

5. The OAS aligns the two ontologies and filters the correspondences according to M. Only those correspondences featuring an entity from M are returned to both agents. The set of ontology correspondences are filtered according to the following function:

   Definition (Mapping Filtering Function) A *filtering function*, filter(), filters the set of candidate mappings prior to argumentation, Z, into a subset $Z' \subseteq Z$ such that:

   $$filter(Z, Sig(M)) : Z \rightarrow Z' \mid \forall m \in Z',\ m = \langle e, e', n, R \rangle \wedge e \in Sig(M)$$

6. The agents begin the Meaning-Based Argumentation process, and iterate it, with each agent generating arguments and counter-arguments.

7. The iteration terminates when the agents reach an agreement on a set of correspondences, and this set is returned to both agents.

8. Ag_1 asks a query to Ag_2 but uses the correspondences so that Ag_2 understands, query$(A \in Sig(O) \wedge B \in Sig(O'))$ where A and B are aligned.

9. Ag_2 answers the query making use of the resulting alignment.

Steps 6 and 7 represent a black-box process, which is the argumentation process. Modularization is therefore used to filter the correspondences that are passed to this process. The combination of these two processes reduces the cost of reaching an agreement over the set of correspondences, by reducing the size of the set of correspondences, and hence the number of arguments that can be made. Thus, the agent is able to overcome semantic heterogeneity with a mutual agreement that is cheaper to obtain.

6.5 An Illustrative Example

This succinct example illustrates the ideas presented previously and relates them to the steps identified in Section 6.4.1. Assume that we have two agents; Ag_1 wishes to ask a query of Ag_2(Step 1), Ag_1 wants to know the instances of `Press`. Ag_1 uses O_1 a BibTex ontology[5] and Ag_2 uses O_2 the General University Ontology[6] ontology. Here we only consider a subset of these ontologies, see Figure 6.4. Until the agents have aligned their ontologies Ag_2 (Step 2) will be unable to fulfil the request of Ag_1.

Ag_1 knows the concepts that are relevant for its task and extracts an ontology module (Step 3), M, in this example the M will include the concepts `Press`, `Publication`

[5] http://www.cs.toronto.edu/semanticweb/maponto/ontologies/BibTex.owl
[6] http://http://www.mondeca.com/owl/moses/univ.owl

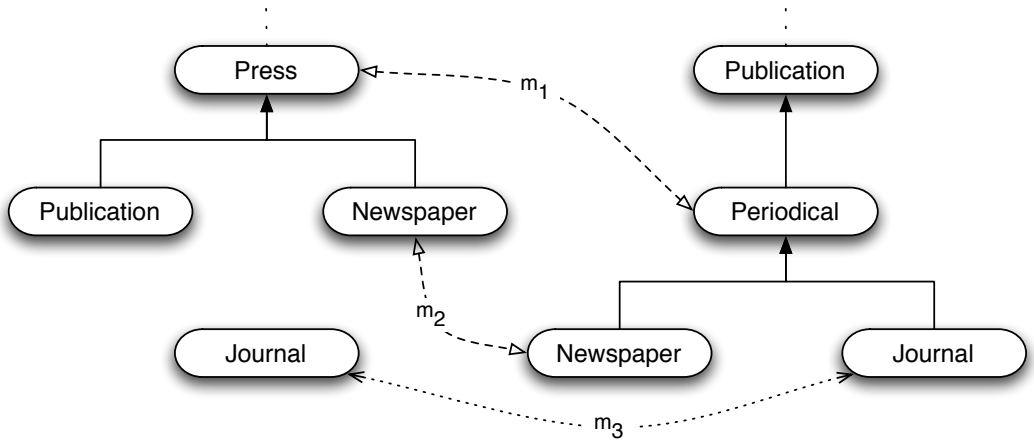

Figure 6.4: Example ontology for combining argumentation and modularization.

and `Newspaper`. Now when the agents invoke the Ontology Alignment Service (OAS) Ag_1 will send its ontology O_1 and the signature of M(Step 4).

$$Sig(M) = \{\texttt{Paper\_Author}, \texttt{Publication}, \texttt{Newspaper}\}$$

The OAS produces the following set of possible correspondences:
$m_1 = \langle O_1 : \texttt{Press}, O_2 : \texttt{Periodical}, 0.45, = \rangle$
$m_2 = \langle O_1 : \texttt{Newspaper}, O_2 : \texttt{Newspaper}, 0.85, = \rangle$
$m_3 = \langle O_1 : \texttt{Journal}, O_2 : \texttt{Journal}, 0.85, = \rangle$

The OAS will now filter the alignments according to the Sig(M) using the function defined in Section 6.4.1(Step 5). The result of this process is the following reduced set of ontology correspondences:
$m_1 = \langle O_1 : \texttt{Press}, O_2 : \texttt{Periodical}, 0.45, = \rangle$
$m_2 = \langle O_1 : \texttt{Newspaper}, O_2 : \texttt{Newspaper}, 0.85, = \rangle$

This reduced set of ontology correspondences will now be used as input to the argumentation process(Step 6). The preference ordering that the agents possess affects how the argumentation phase advances. However, this preference should not affect the premise that the fewer alignments there are to argue over then the fewer arguments that are generated. If we assume now that Ag_1 prefers terminological to external structure (T $\succ$ ES), whilst Ag_2 prefers external structure to terminological (ES $\succ$ T).

The arguments and counter arguments made during the argumentation phase are shown in Table 6.1. This set of arguments allows the agents to build the argument graph, shown in Figure 6.5; whereby the nodes represent the arguments and the arcs represent the attacks relation, with the direction indicating the direction of attack.

The graph shows that arguments B and D support m_1 and m_2 respectively, but that arguments A and C argue against m_1. Given the different preferences of the two

Id	Argument	A	$\mathcal{V}$	Agent
A	$\langle \nexists m = \langle$superconcept(Press), superconcept(Periodical), $n, \equiv\rangle, m_1, -\rangle$	B, 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_2
D	$\langle \exists m = \langle$superconcept(Newspaper), superconcept(Newspaper), $n, \equiv\rangle, m_2, +\rangle$		ES	Ag_2

Table 6.1: Arguments made by Ag_1 and Ag_2 , along with the arguments they attack($\mathcal{A}$) and the value($\mathcal{V}$) of the argument itself.

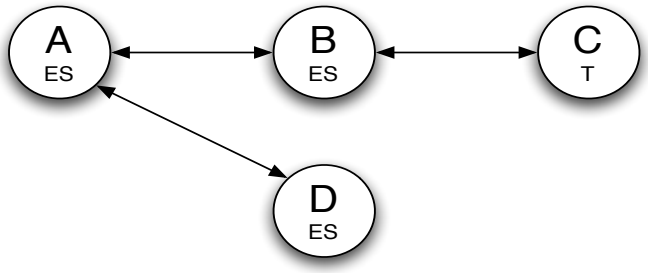

Figure 6.5: Attack graph.

agents, the preferred extensions are shown in Table 6.2. Since no argument is in every preferred extension then there is no *agreed alignment*; this need not always be the case. However, we can say that the from the set of arguments $\{A, B, C, D\}$[7] there is the *agreeable alignment* is $\{m_1, m_2\}$, which is returned to both agents (Step 7) is .

Agent	Preferred Extensions
Ag_1	$\{A,C\}$,$\{B,D\}$, $\{C,D\}$
Ag_2	$\{A,C\}$,$\{B,D\}$

Table 6.2: Arguments for and against m_1 and m_2

Now Ag_1 can ask Ag_2 the query (Step 8) and Ag_2 is able to answer (Step 9) due to the agreed set of alignments.

6.6 Possibility for Information Loss

In the work presented so far it's assumed that an agent *knows all* the concepts that are relevant to its task. However, in practice it may be possible that this assumption does not hold. Consider the scenario presented in Section 6.5. In the case where no modularization is used, the following alignments are used:
$m_1 = \langle O_1 : \texttt{Press}, O_2 : \texttt{Periodical}, 0.45, =\rangle$
$m_2 = \langle O_1 : \texttt{Newspaper}, O_2 : \texttt{Newspaper}, 0.85, =\rangle$
$m_3 = \langle O_1 : \texttt{Journal}, O_2 : \texttt{Journal}, 0.85, =\rangle$

If Ag_1 were to ask Ag_2 to give them all the instances of `Press` then Ag_1 would

[7]Remember that for the agreeable alignment we take the union of all the preferred extensions.

be able to also make sense of the concept `Journal`, even if it does not know a property that connects them.

The example in Section 6.5 used modularization and Ag_1extracted an ontology that didn't include `Journal`. Thus, in this case, m_3 is discarded and this mapping does not enter the argumentation phase. However, when Ag_2 answers the query "*Give me all the instances of* `Press`" it can decide whether or not to return the information regarding `Journal`. In the case that it decides not to then there is no problem, in this case the assumption is that Ag_1 knows all the concepts that are relevant to its task. In the other case where it does return the information regarding `Journal` then Ag_1will not be able to make full use of it as m_3 is missing from the alignment; a mapping such as m_3 would be useful in an information gathering scenario.

6.6.1 Preventing Information Loss

A possible way to deal with the possible loss of information is for Ag_2 to also carry out a modularization step. Once the alignments have been filtered Ag_2 uses the entities identified as a signature for modularization and filters the mappings according to its module. There are two possible ways for Ag_2 to carry out this process:

Sol. 1 Ag_1 filters the alignment according to the signature of its module, $Sig(M)$, which would produce the set of alignments A. Ag_2 then uses A as the input to its own modularization step, $om()$. Ag_2 now filters the alignments according to the signature of its module, $Sig(M')$.

Sol. 2 Rather than using A Ag_2 uses a subset defined by the input signature to Ag_1's modularization step.

Either of these solutions would solve the problem presented above. However, since Ag_2 may identify new, possibly relevant, entities Ag_1 should now also include the previously missing entities to the signature of its ontology module. Evidently this recursive process could be expensive, and both agents could end up with ontology modules equal to the original ontology. One possible way to prevent this would be via conservative extensions [90] that guarantee inferential completeness, but due to their undecidability above $\mathcal{EL}^{++}$ it would require the relaxation of the assumption made in this thesis. The effect of the two possible solutions above are investigated further in Section 6.7; showing that both perform adequately.

6.6.2 Revisiting The Example

Revisiting the example detailed in Section 6.5 with the above solutions in mind would lead to m_3 being included in the argumentation phase; thus, possibly, overcoming the information loss if the agents produce arguments which result in m_3 being acceptable.

6.7 Evaluation

The evaluation has two main objectives:

1. to quantify the impact of the use of modularization techniques on the number of mappings that the argumentation process receives as input

2. to quantitatively evaluate the quality of the resulting alignments compared with the alignments obtained without modularization.

These two objectives are explained in the following sections.

6.7.1 Ontologies and Tracks

Objective (1) requires a diverse set of ontologies, ideally covering different domains, and a diverse set of alignment techniques for the generation of alignments. This is required to overcome the bias deriving from the alignment technique being used, since a specific technique might produce extremely small or extremely large modules, thus skewing the reduction results, but the modules produced might not be useful for the agents task. On the other hand, it is not easy to find extensive sets of ontologies covering different domains and for which there are reliable or verifiable mappings available, so a trade off must be chosen.

The eleven ontologies used in the evaluation were taken from the *OAEI 2007 Conference Track* repository (with the exception of three ontologies whose memory requirements for reasoning were over 2 GB). Table 6.3 recalls again the ontologies used, complete with a brief characterization in terms of the number of classes and properties, and the level of DL expressivity used to represent them.

The alignment techniques available are those used by each system participating in the OAEI tracks; in order to simplify the experimental setup, the systems themselves are not involved in the evaluation; the Alignment API[8] is used instead to access the submitted results. For the chosen track, five systems have submitted a sufficient number of alignments for an overall comparison, i.e., they align each ontology with the others; some systems also provide reverse alignments, i.e., the alignments are of the form $O_A \leftrightarrow O_B$ and $O_B \leftrightarrow O_A$.

The modularization techniques used in this evaluation are some of those described in Sect. 6.4.1; in particular, Cuenca Grau's *lower* and *upper* techniques, d'Aquin's, Doran's and Seidenberg's have also been used; the implementations for these techniques are all available, and only minor modifications to the code have been implemented, to enable them to work in our experimental framework. For clarity, the techniques have been labelled $Techs = \{BASELINE, CUENCAGRAU^U, CUENCAGRAU^L, DAQUIN, DORAN, SEIDENBERG\}$ (shortened to $\{B, CG^U, CG^L, DAQ, DOR,$

[8] `http://alignapi.gforge.inria.fr/`

$SEID$}); $BASELINE$ corresponds to the original alignment, i.e., the alignment as is produced by the alignment techniques where no modularization is used, and is used as the baseline for averaging the results.

6.7.2 Quality of Alignments

Objective (2) requires either a gold standard alignment as a reference to evaluate the resulting mappings, or a method to compare the reduced alignments with the original ones, i.e., a way to verify whether a reduced alignment is equivalent to the original one with respect to correctness and completeness of the results; for the latter evaluation, the following use case is being considered, based on query answering, this follows a similar task style to the evaluation carried out in Section 4.5.

Agent A_1 and agent A_2 engage in communication; A_1 is asking queries of the kind

$$Q_I \ = \ \{x \mid x \ is \ a \ Y\}$$

i.e., instance retrieval queries, to A_2; alternatively, queries of the kind

$$Q_{sup} \ = \ \{x \mid x \ is \ a \ superclass \ of \ Y\}$$

and

$$Q_{sub} \ = \ \{x \mid x \ is \ a \ subclass \ of \ Y\}$$

i.e., queries that explore the concepts' structure.

The original alignment M_0 enables A_2 to give the set of answers X_0; given a reduced alignment M_i, the corresponding set of answers X_i is computed and compared to X_0.

To do this, three *retention* measures are defined:

Definition (Instance Retention) Given an OWL ontology O, an OWL class C not defined in O and two alignments M_0 and M_i, with $M_i \subseteq M_0$, such that:

$$IR : \{O, C, M_0, M_i\} \rightarrow [0, 1]$$

is the function defined as

$$IR(O, C, M_0, M_i) = \frac{|\{x \mid \texttt{instanceOf}(x, C) \in O \sqcup M_i\}|}{|\{x \mid \texttt{instanceOf}(x, C) \in O \sqcup M_0\}|}$$

the number of instances of C described in $O \sqcup M_i$ divided by the number of instances of C described in $O \sqcup M_0$.

Similar definitions can be given for *Subclass Retention* and *Superclass Retention*:

Definition (Subclass Retention) Given an OWL ontology O, an OWL class C not defined in O and two alignments M_0 and M_i, with $M_i \subseteq M_0$, such that:

$$SubR : \{O, C, M_0, M_i\} \rightarrow [0, 1]$$

is the function defined as

$$SubR(O, C, M_0, M_i) = \frac{|\{x \mid \texttt{subClass}(x, C) \in O \sqcup M_i\}|}{|\{x \mid \texttt{subClass}(x, C) \in O \sqcup M_0\}|}$$

the number of subclasses of C defined in $O \sqcup M_i$ divided by the number of subclasses of C defined in $O \sqcup M_0$.

Definition (Superclass Retention) Given an OWL ontology O, an OWL class C not defined in O and two alignments M_0 and M_i, with $M_i \subseteq M_0$, such that:

$$SupR : \{O, C, M_0, M_i\} \rightarrow [0, 1]$$

is the function defined as

$$SupR(O, C, M_0, M_i) = \frac{|\{x|\texttt{superClass}(x, C) \in O \sqcup M_i\}|}{|\{x|\texttt{superClass}(x, C) \in O \sqcup M_0\}|}$$

the number of superclasses of C defined in $O \sqcup M_i$ divided by the number of superclasses of C defined in $O \sqcup M_0$.

These functions can be used to quantify what information is lost by using a smaller alignment M_i, obtained by using a modularization technique, in place of the original alignment M_0, taking into account the task at hand, which is represented by the OWL classes used as a signature for the modularization process. The closesr the result of a retention measure is to 1, the less information is being lost due to the reduction in the alignment size.

Retention functions work well under the hypothesis that for all possible OWL classes on which the measure is computed there are instances available (respectively, subclasses and superclasses). In the ontologies used in the evaluation, this is not always true, i.e., some concepts do not have instances and some classes do not have superclasses or subclasses defined in the ontology. Artificial instances can be added, but it is not meaningful to add artificial subclasses or superclasses in the cases in which they are missing; those cases are left out of the evaluation presented in this thesis.

It is possible that for some concepts there are no alignments that generate answers, i.e., there is no correspondence that generates an answer for C with respect to $O \sqcup M_0$, therefore the retention measures give $0/0$ indetermined form. These cases are not averaged in the evaluation, since they depend on the quality of the alignment itself, i.e., on the suitability of the original alignment for communication, and not on the modularization techniques or on the argumentation process. Evaluating and discussing this issues would be outside the scope of this thesis as it is not concerned with evaluating the alignment techniques themselves.

A corner case is raised when the alignment M_i is empty, i.e., when the modularization techniques generate modules that do not refer to any concept mentioned in the

mappings. In this case, the retention measures can only report 0 or 0/0 indetermined form.

A value of 0 would mean that some answers that were available with respect to M_0 have been lost by reducing the alignment, while a 0/0 would instead mean that there was no retrievable answer; in this second case, reducing the alignment to 0 has the advantage of skipping the argumentation and the query answering phases, since it is already known that no answers will be generated.

The number of cases in which this last hypothesis is true has been evaluated in the current experiments, and there is indeed evidence that this hypothesis is wrong only in 0.15% of the cases, i.e., trusting an empty alignment obtained from modularization to signify that no answers are available is wrong in one case over six hundred. More details are given in the following.

Ontology name	# of classes	# of properties	DL expressivity
cmt	31	64	ALCHIF(D)
Conference	61	69	ALCHIF(D)
confOf	40	41	SHIF(D)
crs_dr	16	22	ALCHIF(D)
edas	105	55	ALCHIF(D)
ekaw	75	38	SHIN
MICRO	33	31	ALCHOIF(D)
OpenConf	64	50	ALCHOI(D)
paperdyne	47	83	ALCHOIF(D)
PCS	25	43	ALCHIF(D)
sigkdd	51	33	ALCHI(D)

Table 6.3: Classes, properties and DL expressivity for the OAEI ontologies.

6.7.3 Evaluation Setup

The experiments are divided into *runs*; each run is described as a tuple $< O_1, O_2, A, T, S, I >$, where:

- O_1 and O_2 are distinct ontologies from the same track (the order is important, so each track produces $n * (n - 1)$ pairs);
- A is the alignment being produced by a specific alignment technique on the pair of ontologies, and therefore a set of mappings;
- T is a modularization technique, with S being the signature used for the extraction process;
- I is the technique being used to fix the information loss problem described in Section 6.6.

	CG^U	CG^L	DAQ	DOR	$SEID$	Total
ALC	32	32	2	31	145	242
ALC(D)	2	2	0	0	0	4
ALCF	1	1	25	0	0	27
ALCF(D)	17	17	0	0	170	204
ALCH(D)	10	10	0	0	0	20
ALCHI	18	18	0	4	0	40
ALCHIF	32	32	0	38	0	102
ALCHIF(D)	6	6	0	0	0	12
ALCHIN	7	7	0	0	0	14
ALCHOF(D)	0	0	0	0	45	45
ALCHOI	10	10	0	17	0	37
ALCHOIF	25	25	0	0	0	50
ALCN	0	0	0	0	73	73
ALCO	0	0	0	0	62	142
ALCOI	53	53	0	5	31	62
EL	196	198	349	363	0	1106
EL+	90	90	103	69	0	352
SHI	7	7	0	0	0	14

Classes	6.02	6.04	2.22	14.78	62.36	18.82
stdev	8.42	8.42	3.38	28.02	25.82	28.78
Object Properties	1.94	1.95	0	1.15	11.49	3.33
stdev	5.16	5.17	0	3.75	7.06	6.4
Datatype Properties	0.28	0.28	0	0	2.81	0.68
stdev	1.17	1.17	0	0	3.78	2.15
Anonymous Classes	4.98	5	0.43	3.08	33.52	9.81
stdev	11.13	11.14	1.42	10.48	25.96	19.01

Table 6.4: Modules statistics: DL expressivity and number of modules (upper section) and average and standard deviation for number of classes, object and datatype properties, and anonymous classes (lower section).

The modularization technique T is one of the elements in $Techs$; when $T = B$, i.e., the baseline, no modularization happens, and therefore the signature S is ignored, as well as I. For the other values of T, three new runs are generated for each named concept C in O_1, with $S = \{C\}$ and I being one of $\{NONE, COMPLETE, SIGNATURE-ONLY\}$ (shortened to $\{N, C, SO\}$), see Section 6.6.1.

Called $NC(O)$ the number of named concepts in ontology O and n the number of ontologies in a track, therefore, the number of runs with O as first ontology for a track is $6 * (3 * NC(O) + 1) * (n - 1)$.

Information being recorded includes:

- for each ontology, number of concepts, properties, anonymous concepts and DL expressivity (in Table 6.3);
- for each module with $|S| \geqslant 1$, number of concepts, properties, anonymous concepts and DL expressivity;
- for each pair of ontologies (O_1, O_2)
  - Modules extracted from O_1, and percentage reduction in concept and property number with respect to the original ontology, these values are given in Table 6.4);
  - Number of mappings and arguments being generated without modularization (baseline technique B, values in columns *Original size (B)* and *Original alignments* in Table A.5 and Table A.6);
  - Number of mappings and arguments resulting from modularization with each value for T and I (Table A.5 and Table A.6);
  - Number of mappings being accepted and rejected by the argumentation process for all cases above (Table A.7 and Table A.8);

Where relevant, tables report also the values for each run excluding the cases in which the reduced alignments had size 0, i.e. no correspondences were found to be relevant.

6.7.4 Results Discussion

The results presented in Table 6.5 and in Table 6.6 show that the overall impact of using a modularization technique for reducing the amount of candidate correspondences can vary from 57% to 95% (*candidate* column in both tables). Three broad categories emerge from this: (i) those cases where modularization yields a significant reduction in the number of correspondences argued; (ii) those cases where no reduction occurs; and (iii) those cases where no correspondences are identified, and thus no argumentation occurs. This third category was unexpected, and corresponds to those scenarios

	accepted / %	candidate / %	arguments / %
DAQ	3.16 / 12.73%	2.46 / 8.50%	6.37 / 11.00%
DOR	2.67 / 10.75%	2.15 / 7.43%	5.47 / 9.44%
CG^U	1.93 / 7.77%	1.35 / 4.66%	3.92 / 6.77%
CG^L	1.93 / 7.77%	1.35 / 4.66%	3.92 / 6.77%
$SEID$	12.65 / 50.89%	12.41 / 42.87%	26.26 / 45.36%
C	4.51 / 21.74%	3.95 / 19.20%	10.40 / 24.67%
SO	5.07 / 24.55%	3.95 / 19.20%	9.28 / 21.55%
N	3.82 / 19.47%	3.95 / 19.20%	7.89 / 19.20%

Table 6.5: Average over all runs for each modularization technique (upper half) and for each information loss solution (lower half)

	accepted / %	candidate / %	arguments / %
DAQ	6.05 / 24.34%	4.98 / 17.19%	12.61 / 21.78%
DOR	5.96 / 24.00%	5.80 / 20.02%	13.50 / 23.32%
CG^U	3.96 / 15.92%	3.65 / 12.60%	9.09 / 15.70%
CG^L	3.96 / 15.92%	3.65 / 12.60%	9.09 / 15.70%
$SEID$	19.08 / 76.78%	23.70 / 81.85%	48.84 / 84.34%
C	7.89 / 38.84%	8.35 / 33.37%	20.37 / 42.27%
SO	8.66 / 43.03%	8.35 / 33.37%	18.80 / 37.59%
N	6.85 / 34.70%	8.35 / 33.37%	16.71 / 33.37%

Table 6.6: Average over all runs for each modularization technique (upper half) and for each information loss solution (lower half). Alignments of size zero are not included in the average.

whereby no suitable correspondences exist for the alignment of the original signature, and thus any argumentation would be redundant.

Technique	IR	stdev	$SubR$	stdev	$SupR$	stdev
DAQ	99.87%	0.61%	100%	0%	100%	0%
DOR	99.97%	0.26%	100%	0%	100%	0%
CG^U	99.59%	2.85%	99.73%	2.79%	99.73%	2.79%
CG^L	99.59%	2.85%	99.73%	2.79%	99.73%	2.79%
$SEID$	100%	0%	100%	0%	100%	0%
Overall average	99.80%	1.83%	99.89%	1.77%	99.89%	1.77%

Table 6.7: Instance, Subclass and Superclass Retention values

Depending on the modularization technique being chosen; the impact of the information loss solution is more contained: whichever technique is being used, the expected reduction in candidate alignments is close to 80% and the search space for the argumentation is reduced from 75% to 80%.

Paired with the results presented in Table 6.7, that evaluate the quality of the resulting alignments in terms of the retention measures defined in Sect. 6.7.2, the data outlines the following conclusion: there is a trade off between reduction of the argu-

O_1	O_2	System	I	MT	IR	$SubR$	$SupR$
OpenConf	paperdyne	Ontodna	N	CG^L CG^U	50.00%	50.00%	50.00%
OpenConf	Conference	Ontodna	N	CG^L CG^U	66.67%	66.67%	66.67%
OpenConf	cmt	Lily	N	CG^L CG^U	71.43%	71.43%	71.43%
OpenConf	crs_dr	Lily	N	CG^L CG^U	78.57%	78.57%	78.57%
OpenConf	paperdyne	Lily	N	CG^L CG^U	82.00%	82.50%	82.50%
OpenConf	edas	Lily	N	CG^L CG^U	82.05%	82.05%	82.05%
OpenConf	confOf	Lily	N	CG^L CG^U	87.50%	87.50%	87.50%
OpenConf	paperdyne	Falcon	N	CG^L CG^U	87.50%	87.50%	87.50%

Table 6.8: A snapshot of the lower retention values

mentation search space and retention, where a reduction of the search space close to 95% (for the CG^U and CG^L techniques) corresponds to the highest loss on retention, averaged at 0.05%. Some examples, reported in Table 6.8, show that in a very restricted number of cases these techniques can lead to a higher loss in retention, when coupled with a lack of information loss solutions. On the other hand, there is a lower impact on the number of candidate correspondences, such as in the $SEID$ technique, which produces maximal retention values. Thus, guaranteeing that no information is being lost, but the average reduction in the search space for this technique (see Table 6.9 and Table A.9) is slightly lower than 24%, and, as shown in the more detailed breakdown in Table A.6 when $SEID$ is used over the results of the Ola system, it can be as small as 5 %.

These possibilities suggest a flexible architecture able to use more than one modularization technique, where a lower than expected number of results to a query can trigger the use of a more conservative modularization technique; this would enable the system to guarantee the lowest possible loss of information, while ensuring maximum reduction of the search space in the average case.

The quality of the resulting alignments in terms of the retention measures is quite high: only a handful of cases show retention lower than 0.95, and a great majority show retention equal to 1, therefore granting that the choice of modularization technique and information loss solution can safely be done on the basis of the size of the resulting alignment.

Moreover, there are a large number of cases in which the use of modularization yields an empty alignment, as shown in Table A.10. Experimental evaluation shows that in the largest majority of these cases (99.85%) an empty alignment is correlated with no answers being available for the concepts in the signature, even with the complete original alignment. This confirms that there are cases in which the argumentation process can be skipped altogether without hampering the reliability of the system in terms of retention; it is in fact always higher than 99% on average. Thus, preventing agents from wasting time arguing over mappings that will not help them achieve their task.

Average original alignment size (B): 24.84						
		DAQ	DOR	CG^U	CG^L	$SEID$
Accepted alignment size	C	3.26	2.97	1.89	1.89	12.53
	SO	3.78	2.94	2.57	2.57	13.47
	N	2.43	2.08	1.32	1.32	11.92
Accepted alignment size (%)	C	16.00%	13.10%	8.88%	8.88%	61.85%
	SO	18.39%	13.39%	11.64%	11.64%	67.70%
	N	13.28%	10.25%	6.94%	6.94%	59.91%
Average candidate alignment size: 28.95						
Average # of arguments: 57.90						
		DAQ	DOR	CG^U	CG^L	$SEID$
Avg candidates with mod.	C	2.46	2.15	1.35	1.35	12.41
	SO	2.46	2.15	1.35	1.35	12.41
	N	2.46	2.15	1.35	1.35	12.41
Avg # args with mod.	C	7.62	6.01	5.22	5.22	27.93
	SO	6.57	6.08	3.85	3.85	26.04
	N	4.92	4.30	2.70	2.70	24.83
Avg candidates with mod. (%)	C	13.34%	10.27%	6.97%	6.97%	58.45%
	SO	13.34%	10.27%	6.97%	6.97%	58.45%
	N	13.34%	10.27%	6.97%	6.97%	58.45%
Avg # args with mod. (%)	C	18.49%	13.66%	11.98%	11.98%	67.22%
	SO	16.04%	13.26%	8.94%	8.94%	60.56%
	N	13.34%	10.27%	6.97%	6.97%	58.45%

Table 6.9: Average accepted alignment sizes (averaged by modularization technique)

6.8 Conclusion

Agents need to reconcile ontological differences, especially within the context of open and dynamic environments where no *a priori* assumptions about the nature of the ontology can be made. Negotiation frameworks (such as the value-based argumentation framework) allow agents to negotiate over different ontology correspondences, and identify those alignments that are mutually acceptable. However, this collaborative search is computationally costly, as the complexity of the decision problems reach $\Pi_2^{(p)}$-complete. This Chapter proposed the use of ontology modularization as a mechanism to reduce the size of the search space for finding acceptable alignments.

The use of ontology modularization as a filter-based pre-processing stage was evaluated empirically over eleven ontologies used in the OAEI initiative. The results show that the use of ontology modularization can significantly reduce the average number of correspondences presented to the argumentation framework, and hence the size of the search space – in some cases by up to 97%, across a number of different ontology pairs. In addition, three patterns emerged:

- where no reduction in size occurred;
- where the number of correspondences was reduced;

- where modules of size zero were found.

This latter case corresponded to failure scenarios; i.e. where the subsequent transaction could fail due to insufficient alignment between the ontologies. Thus, we demonstrated that ontology modularization not only reduces the cost of negotiating over correspondences and establishing communication, but that it can be effectively used to predict cases where negotiation will fail to identify relevant correspondences to support meaningful queries.

Furthermore, the experiments demonstrated that in most cases the reduction in the search space did not affect the quality of the alignment. Indicating that agents can argue over less and have comparable performance. Considering that the argumentation is computationally complex then the agents can reduce the cost of this step without adversely affecting their performance.

Part V

Synopsis

Chapter 7

Conclusions and Future Work

'Every day you may make progress. Every step may be fruitful. Yet there will stretch out before you an ever-lengthening, ever-ascending, ever-improving path. You know you will never get to the end of the journey. But this, so far from discouraging, only adds to the joy and glory of the climb.' - Sir Winston Churchill

Summary This chapter provides a review of the contributions of this thesis along with a number of avenues for future work. Initially the contributions are discussed along with the implications of the key achievements. The discussion of future work concludes this chapter and addresses possible further contributions emerging from this thesis.

7.1 Review of Contributions

This section shall review the contributions of this thesis in light of the two research questions given in Section 1.2; they were:

1. How can part of an ontology be reused instead of the whole?
2. How can the ontology modules, obtained as a result of ontology modularization, be used in practice?

Below the work carried out is summarised with respect to its contribution and how it addressed the above questions.

In Chapter 2 the required background was introduced. This included a definition of what an ontology is (see Section 2.1.1) and how they can be represented. This included an explanation of Description Logics (see Section 2.3) and their reasoning tasks. Lastly, an original example ontology is given (see Section 2.5) which is used at various parts throughout the thesis.

A comprehensive review of the ontology modularization literature is conducted in Chapter 3. This outlines the *principles* that underlie the differing modularization

techniques. These principles maybe different depending on whether a logical extraction technique (see Section 3.4.2) or a traversal extraction technique (see Section 3.4.1) is being used. This chapter also includes the *contribution* of a sound heuristic for extracting ontology modules via a graph traversal. In addition, a classification of the different techniques is provided to highlight the differences that exist between the disparate techniques. It was considered unreasonable to provide an overall classification of the techniques because of the diverse starting assumptions made. For example, classifying the traversal based techniques in terms of their logical properties would have little value.

The notion of a common framework is also explored in Chapter 3. A common framework would allow the different modularization techniques to be implemented on a common framework to facilitate a more objective comparison of performance. Three possibilties for common frameworks are provided (see Section 3.6). The pursuit of a common framework would greatly facilitate further comparisons between the various modularization techniques than is currently possible.

One issue not addressed in this thesis or by any of the ontology modularization techniques in the literature is the practical issue of what URI to assign to a module? Whilst the URI should be different to the ontology it was extracted from, it will contain concepts with the same URI as the original ontology. Unless the module has been extracted using a logical extraction method (see Section 3.4.2) it is possible that the concepts will have different definitions; essentially meaning that there are two different concepts with the same URI. This could present itself as a problem if those two concepts ever end up in the same ontology.

Chapter 4 is devoted to the issue of evaluating modularization techniques and the ontology modules that they produce. Firstly, the existing literature on ontology evaluation is reviewed to highlight why it is unsuitable for evaluating modularization. Several metrics are then introduced as possible candidates for evaluating ontology modularization. These include the *contribution* of an entropy inspired metric which aims to capture the information content of an ontology module; as such, it better represents a module contents than size.

Two evaluations are then conducted: an evaluation of the different metrics when comparing different modularization techniques, and a task based evaluation which assesses the suitability of different modularization techniques for three tasks. Whilst the metric based evaluation showed that entropy was more discriminate than size it was unable to show a proper characterisation of the entropy measure. From the evaluation carried out it is clear that entropy has some value, but it is rather tricky to properly characterise it. In order to obtain this characterisation the weighting function and the mechanics of the measure need to be deeply scrutinised. However, the task based evaluation showed that the traversal based approaches have a comparable per-

formance to the logical approaches; at least on the tasks considered. Whilst, of course, this could just be luck it could also suggest that under certain condition/circumstances that the traversal extraction methods also have some desirable formal properties. Thus, it would be interesting to consider if it was possible to observe comparable performance over different tasks.

Ontology reuse is a popular scenario for considering the use of ontology modularization and this is examined in Chapter 5. Ontology Engineering methodologies whilst including a step for ontology reuse do not prescribe a methodology for reuse via ontology modularization. Thus, two methodologies for reusing ontology modules are presented; one is a general purpose methodology and the other is a methodology for reusing locality based modules. The general purpose methodology can be followed regardless of the modularization technique being used, but the locality based methodology requires the modularization technique being used to provide the appropriate guarantees.

Chapter 6 contributes a novel application of how ontology modularization can be used in *practice* as a space reduction mechanism for the problem of dynamically selecting ontological alignments in open multi-agent systems. This shows that modularization is an effective space reduction mechanism for the complex argumentation process. A problem with information loss is noted and two solutions are provided. The effectiveness of this application of modularization is demonstrated by the evaluation carried out in Section 6.7 which shows an 80% reduction in the search space when using modularization with argumentation and that this reduction does not reduce the quality of the produced alignment, particularly when the information loss (see Section 6.6) solutions are used. In addition, agents were also able to identify those cases where argumentation would have been a waste of their time. This suggests that not only can ontology modularization reduce the cost of reaching an agreement, but that it can also help identify those cases where agents would have been unable to reach an agreement.

In summary then this thesis makes four original contributions, these are:

1. **Sound Heuristic for Module Extraction.** Section 3.4.1 presented a detailed explanation of a graph traversal based technique for ontology module extraction. This technique allows an ontology module to be extracted that is centered around a specified concept. The technique presented is ontology language agnostic, as long as the ontology language being used can be transformed into the abstract graph model that this technique operates over.

2. **Entropy Metric.** Section 4.3.3 presented a detailed explanation of the entropy inspired metric for evaluating ontology modules. This metric allows one to evaluate the information content of an ontology module, allowing the full expressivity of the ontology language to be considered.

3. **Comprehensive Evaluation.** Sections 4.4 details an evaluation of the different ontology modularization techniques. The evaluation shows that whilst the entropy measure is a more discriminate measure than size it is still difficult to evaluate how 'good' an ontology module is based solely on an isolated objective metric. In addition, Section4.5 details a task-based evaluation of the different ontology modularization techniques. The tasks considered is query answering and the results show that the traversal based extraction techniques have comparable performance to the logical based extraction techniques, at least, for the task considered.

4. **Novel Application of Modularization.** Chapter 6 details a new application of ontology modularization. The Chapter explains how ontology modularization can be successfully applied to the problem of dynamically reaching an agreement over ontology alignments. Ontology modularization allows the search space for the argumentation process to be reduced. The evaluation conducted shows that ontology modularization successfully reduces the search space whilst not having an adverse effect upon the quality of the agreed alignment. Furthermore, it showed that agents were able to identify cases where an agreement could not be achieved prior to the argumentation phase preventing the agents from wasting time.

7.2 Future Work

There are numerous issues that stem from the research in this thesis that, unfortunately, due to time constraints there has not been opportunity to investigate further. These issues broadly fall into three areas: ontology modularization, ontology module evaluation and the dynamic selection of ontology alignments. They are presented here as we consider them worthy of future research.

7.2.1 Ontology Modularization

As discussed in Section 3.2 there are numerous techniques in the literature for ontology modularization broadly split into logical extraction (3.4.2) and traversal extraction (3.4.1). The logical extraction methods have formal properties, such as safety, coverage, etc. This leads to the natural question as to whether the traversal extraction methods also have desirable formal properties. Interest in this issue is further piqued due to the task based evaluation (see Section 4.5) carried out in this thesis, as it shows that the traversal extraction methods, at least on the task evaluated, show comparable performance to the logical extraction methods. This aspect is worthy of future investigation to fully explore the reasons for the performance of the traversal based techniques in the evaluation.

Despite an increasing number of mechanisms proposed for ontology modularization, there are few studies, to the best of our knowledge only Palmisano *et al.* [105], that have systematically evaluated different approaches over a number of ontologies. There is a distinct need for there to be a structured evaluation of modularization techniques over a standard data set, similar to the Ontology Alignment Evaluation Initiative (OAEI). Such an initiative for modularization would allow the strengths and weaknesses of the different systems to be assessed. It should enable the performance of the different techniques across a number of objective measures which would be of benefit to people who have the problem of deciding what technique they should use.

Furthermore, the establishment of an initiative similar to the OAEI for modularization should increase communication and foster collaboration among ontology modularization researchers. More importantly, perhaps, is that it should promote the development and uptake of improved evaluation techniques; this is a distinct weakness in the current literature.

Given that nobody has addressed the issue of what URI to assign a module then future work could include the development of a protocol for controlling the reuse of extracted modules. For example, if the same module is extracted from an ontology by two different people then it could be convenient if they assigned them the same URI.

7.2.2 Ontology Module Evaluation

As well as requiring a more systematic and structured approach to evaluate ontology modularization in general, the development of the entropy measure brought up some interesting areas of future work. Firstly, whilst an extensive evaluation of the entropy measure was carried out there is a need for there to be a user-study to be conducted. It would be of particular interest to consider how understandable an ontology is. Intuitively an ontology, or module, with low entropy should be easier to understand.

Perhaps it would be interesting to implement numerous evaluation metrics, give them to users and collect feedback on which they found useful, intuitive, informative, etc. Having them implemented would also allow them to be applied in application scenarios where no human is involved, such as the one considered in Section 6.4.1.

Given that a proper characterisation of the entropy measure was not possible it would be interesting to explore a more 'semantic' entropy measure. The current entropy measure produces different entropy values for axioms that are semantically equivalent but syntactically different; intuitively a 'semantic' entropy measure would give the same value. One possible way to do this might be by considering how much it takes to explain why certain axioms are entailed; some work on computing justifications for OWL already exists in the literature, for example Horridge, Parsia and Sattler [73] present an algorithm to compute laconic and precise justifications.

Section 4.5 carried out a task based evaluation of the ontology module extraction

techniques showing that traversal approaches can have a comparable performance to the logical approaches. It would be interesting to extend this analysis to better identify boundary cases whereby certain techniques may be more suitable than others. This task is likely to benefit by considering some of the issues raised in Section 7.2.1.

7.2.3 Dynamic Selection of Ontology Alignments

In Section 6.4.1 modularization was used as a space reduction mechanism for the dynamic selection of ontological alignments. Whilst this novel application did not bring to light many issues concerning modularization it did bring to light numerous issues regarding the problem of dynamically reaching an agreement in open environments.

Firstly, the actual use of argumentation raises several issues. Currently, Laera *et al.* [85] only consider one-stop argumentation and it would be interesting to consider iterative argumentation, thus allowing the agents to argue and counter-argue in a more gregarious manner. However, introducing the iterative argumentation would raise the need to consider strategies for argumentation. For example, is it possible for one agent to argue in a certain way to affect the outcome in its favour?

Furthermore, the actual arguments that are possible, based on the schema presented by Laera [84], are limited. As such, it would be interesting to investigate richer grounds for the arguments themselves. For example, an agent would be able to argue against a mapping because it would make its ontology unsatisfiable. This would require the agents to be able to place DL entailments in the grounds of the arguments; Moguillansky and Wassermann [97] use arguments of this form to reason about inconsistencies.

One overriding concern, and indeed the motivation for applying modularization, is the high complexity of the approach presented in [85]. Thus, it would be useful to characterise precisely the cases (i.e. the structure of the attack graph) where this worst case complexity actually occurs. This would be useful to know either for the agents to avoid creating attack graphs that trigger the worst case complexity or by being able to design a protocol for the argumentation which prevents worst case attack graphs from occurring. Whilst being able to circumvent the worst case may seem to lessen the incentive for using modularization this is not the case because modularization has value in that it restricts the argumentation to those elements of the ontologies that are relevant.

In addition to the issues concerning the argumentation aspect noted above some general issue about dynamic agreement have also been raised, particularly in open environments where the agents are assumed to be rational. The first issue concerning agent utility; a bounded rational agent should only reach an agreement when it is going to be of benefit, i.e. bring about an increase in its utility. Under the scenario being considered here requires answers to the following three questions:

1. **What is the cost of reaching an agreement?** There is evidently a cost in

reaching an agreement in terms of complexity, hence the application of modularization to the problem. However, this concerns only one dimension of the cost. Other costs that an agent might consider would be communication and time. When the agents engage in an argumentation process there is a communication overhead, this could be significant if the ontologies are large or the argumentation process proceeds iteratively. Both the complexity and communication cost directly affect the next dimension, time. In a highly dynamic fast changing environment an agent may prefer 'quick' agreements to 'correct' agreements so that it can capitalise on the current state of affairs more effectively. Conversely if an agent is going to commit a sizable amount of time to reach an agreement then the following two points become more important.

2. **What is the cost of the agreement?** Once an agreement has been reached then an agent may need to reclassify its ontology, this incurs a cost. Depending on this cost the agent may choose to make the agreement temporary or permanent. These issues fall under ontology evolution and some work in the area of open environments can be found in Palmisano *et al.* [104] where they present an algorithm for estimating the cost of including a new axiom in your existing ontology.

3. **What is the value of the agreement?** So far we have considered the cost of getting the agreement and of the agreement itself, but this cost can be offset by the value of the agreement. Value can be derived from the agreement because it should facilitate the agent in completing its tasks. An agreement would be of greater value to an agent if it were robust and long lasting because it would delay the need to reach an agreement again. The notion of robustness for an agreement could be linked to the notion of objective acceptability in argumentation; in that objectively acceptable arguments are accepted by all parties and are not a source of dispute. It maybe more troublesome to evaluate the lifespan of an agreement, particularly given the nature of open environments.

The notion of minimal agreement is interesting; and modularization may play a part in achieving them. An agreement is minimal if it is the minimum number of mappings required to successfully complete the agents task. The motivation for applying modularization was to prevent the agents from wasting time by arguing over irrelevant concepts. The same idea could be applied to the agreement; why should the agents agree on more than they need? This is more pertinent when you consider the issues raised above. The agreement should just be enough to allow the agent to do what they need to do, a sub-minimal agreement would mean that the agent wouldn't be able to

do what it wanted and a non-minimal agreement would add unnecessary overhead for the agent in achieving its task. In essence the agents need the adequate amount of the relevant knowledge.

Part VI

Appendices

Appendix A

Experimental Results

A.1 Detailed Tables For The Task Based Evaluation (Section4.5)

	CG^L			CG^U			DAQ			DOR			$SEID$		
	P	R	FM	P	R	FM	P	R	FM	P	R	FM	P	R	FM
Conference	1.00	0.84	0.912	1.00	0.84	0.912	1.00	0.80	0.888	1.00	1.00	1.000	0.83	0.72	0.773
cmt	1.00	0.76	0.862	1.00	0.76	0.862	1.00	0.78	0.875	1.00	1.00	0.998	0.76	0.60	0.670
confOf	1.00	0.83	0.909	1.00	0.83	0.909	1.00	0.83	0.909	1.00	1.00	1.000	0.87	0.74	0.801
crs_dr	1.00	0.84	0.911	1.00	0.84	0.911	1.00	0.84	0.911	1.00	1.00	1.000	0.71	0.71	0.714
edas	1.00	0.86	0.926	1.00	0.86	0.926	1.00	0.83	0.907	1.00	0.99	0.995	0.87	0.81	0.838
ekaw	1.00	0.76	0.865	1.00	0.76	0.865	1.00	0.76	0.865	1.00	1.00	1.000	0.92	0.73	0.813
MICRO	1.00	0.87	0.928	1.00	0.87	0.928	1.00	0.82	0.903	1.00	0.97	0.987	0.94	0.81	0.869
OpenConf	0.90	0.78	0.835	0.90	0.78	0.835	1.00	0.73	0.841	0.89	1.00	0.942	0.97	0.81	0.885
paperdyne	0.96	0.82	0.884	0.96	0.82	0.884	0.99	0.87	0.924	0.84	1.00	0.910	0.91	0.82	0.862
PCS	1.00	0.75	0.860	1.00	0.75	0.860	1.00	0.76	0.864	1.00	1.00	1.000	0.74	0.61	0.667
sigkdd	1.00	0.81	0.893	1.00	0.81	0.893	1.00	0.81	0.893	0.99	1.00	0.996	0.88	0.76	0.816
Average	0.987	0.810	0.889	0.987	0.810	0.889	0.999	0.802	0.889	0.974	0.997	0.984	0.854	0.739	0.792

Table A.1: Results broken down by ontology and technique; all the modules for a single ontology and a single technique are averaged together; this table only reports the *instances* results.

	CG^L			CG^U			DAQ			DOR			$SEID$		
	P	R	FM	P	R	FM	P	R	FM	P	R	FM	P	R	FM
Conference	1.00	0.87	0.928	1.00	0.87	0.928	1.00	0.84	0.910	1.00	1.00	1.000	0.83	0.74	0.784
cmt	1.00	0.80	0.892	1.00	0.80	0.892	1.00	0.82	0.901	1.00	1.00	0.999	0.76	0.64	0.692
confOf	1.00	0.86	0.924	1.00	0.86	0.924	1.00	0.86	0.924	1.00	1.00	1.000	0.87	0.76	0.813
crs_dr	1.00	0.87	0.929	1.00	0.87	0.929	1.00	0.87	0.929	1.00	1.00	1.000	0.71	0.71	0.714
edas	1.00	0.88	0.939	1.00	0.88	0.939	1.00	0.87	0.929	1.00	1.00	1.000	0.87	0.82	0.845
ekaw	1.00	0.80	0.889	1.00	0.80	0.889	1.00	0.80	0.889	1.00	1.00	1.000	0.92	0.77	0.835
MICRO	1.00	0.89	0.941	1.00	0.89	0.941	1.00	0.88	0.934	1.00	1.00	1.000	0.94	0.83	0.882
OpenConf	0.90	0.81	0.852	0.90	0.81	0.852	1.00	0.83	0.907	0.98	1.00	0.992	0.97	0.90	0.931
paperdyne	0.96	0.84	0.896	0.96	0.84	0.896	0.99	0.89	0.937	0.98	1.00	0.990	0.91	0.83	0.871
PCS	1.00	0.81	0.893	1.00	0.81	0.893	1.00	0.81	0.896	1.00	1.00	1.000	0.74	0.63	0.682
sigkdd	1.00	0.84	0.912	1.00	0.84	0.912	1.00	0.84	0.912	0.99	1.00	0.997	0.88	0.78	0.826
Average	0.987	0.842	0.909	0.987	0.842	0.909	0.999	0.845	0.915	0.996	1.000	0.998	0.854	0.766	0.807

Table A.2: Results broken down by ontology and technique; all the modules for a single ontology and a single technique are averaged together; this table only reports the *subclasses* results.

	CG^L			CG^U			DAQ			DOR			$SEID$		
	P	R	FM	P	R	FM	P	R	FM	P	R	FM	P	R	FM
Conference	0.87	0.82	0.845	1.00	0.53	0.689	0.98	0.70	0.821	0.71	0.54	0.612	1.00	0.64	0.783
cmt	1.00	0.61	0.758	1.00	0.55	0.711	1.00	0.53	0.693	1.00	0.67	0.806	1.00	0.75	0.860
confOf	0.74	0.65	0.689	0.99	0.56	0.718	0.88	0.80	0.837	0.83	0.71	0.767	1.00	0.77	0.871
crs_dr	1.00	0.55	0.707	1.00	0.72	0.839	1.00	0.77	0.871	1.00	0.56	0.718	1.00	0.61	0.758
edas	1.00	0.52	0.684	1.00	0.83	0.906	0.97	0.83	0.892	1.00	0.53	0.689	1.00	0.51	0.678
ekaw	1.00	0.66	0.792	0.94	0.91	0.922	0.99	0.57	0.722	0.92	0.77	0.837	1.00	0.84	0.912
MICRO	0.76	0.63	0.687	0.95	0.79	0.863	1.00	0.47	0.643	1.00	0.60	0.752	1.00	0.51	0.676
OpenConf	1.00	0.84	0.912	1.00	0.55	0.708	1.00	0.57	0.724	0.90	0.85	0.877	1.00	0.57	0.723
paperdyne	0.90	0.73	0.810	1.00	0.89	0.940	1.00	0.55	0.711	1.00	0.55	0.708	1.00	0.55	0.708
PCS	1.00	0.89	0.940	0.99	0.61	0.759	0.87	0.69	0.769	0.90	0.73	0.810	0.99	0.65	0.786
sigkdd	1.00	0.56	0.718	1.00	0.55	0.706	1.00	0.60	0.753	1.00	0.64	0.783	0.95	0.79	0.863
Average	0.934	0.677	0.777	0.988	0.680	0.796	0.972	0.644	0.767	0.933	0.651	0.760	0.994	0.654	0.783

Table A.3: Results broken down by ontology and technique; all the modules for a single ontology and a single technique are averaged together; this table only reports the *superclasses* results.

Ontology Name	# Cl.	DAQ Modules (> 1) Num	DAQ Modules (> 1) % Cl.	DOR Modules (> 1) Num	DOR Modules (> 1) % Cl.	SEID Modules (> 1) Num	SEID Modules (> 1) % Cl.	CG-L Modules (> 1) Num	CG-L Modules (> 1) % Cl.	CG-U Modules (> 1) Num	CG-U Modules (> 1) % Cl.
Conference	59	26	44.1%	24	40.7%	49	83.1%	35	59.3%	35	59.3%
cmt	29	14	48.3%	11	37.9%	22	75.9%	9	31.0%	9	31.0%
confOf	38	7	18.4%	8	21.1%	33	86.8%	25	65.8%	25	65.8%
crs-dr	14	9	64.3%	3	21.4%	10	71.4%	0	0.0%	0	0.0%
edas	103	18	17.5%	25	24.3%	89	86.4%	102	99.0%	102	99.0%
ekaw	73	14	19.2%	23	31.5%	67	91.8%	15	20.5%	15	20.5%
MICRO	31	14	45.2%	6	19.4%	28	90.3%	24	77.4%	24	77.4%
OpenConf	62	12	19.4%	25	40.3%	60	96.8%	62	100.0%	62	100.0%
paperdyne	45	22	48.9%	8	17.8%	41	91.1%	36	80.0%	36	80.0%
PCS	23	13	56.5%	10	43.5%	17	73.9%	7	30.4%	7	30.4%
sigkdd	49	11	22.4%	14	28.6%	43	87.8%	8	16.3%	8	16.3%
Mean values:			**36.7%**		**29.7%**		**85.0%**		**52.7%**		**52.7%**

Table A.4: Comparison of the Module Size (in terms of named entities) for each of the different modularization approaches. Both the number of modules generated containing more than two named concepts, and this value as a percentage of all modules for each ontology are given.

A.2 Tables Argumentation

System	MT	Original size (B)	Original arguments	Retained alignment size and # of arguments						Avg percent					
				C M	C A	SO M	SO A	N M	N A	C M	C A	SO M	SO A	N M	N A
Asmov	CG^L	13.22	26.45	0.94	2.98	0.94	2.18	0.93	1.87	7.10%	11.27%	7.10%	8.26%	7.10%	7.10%
	CG^U			0.94	2.98	0.94	2.18	0.93	1.87	7.10%	11.27%	7.10%	8.26%	7.10%	7.10%
	DAQ			1.49	4.38	1.49	3.55	1.49	2.99	11.31%	16.56%	11.31%	13.44%	11.31%	11.31%
	DOR			1.41	4.01	1.41	3.59	1.41	2.83	10.70%	15.18%	10.70%	13.60%	10.70%	10.70%
	$SEID$			8.68	21.00	8.68	17.88	8.68	17.36	65.64%	79.38%	65.64%	67.59%	65.64%	65.64%
Falcon	CG^L	13.10	26.21	1.07	4.28	1.07	2.90	1.07	2.14	8.17%	16.35%	8.17%	11.09%	8.17%	8.17%
	CG^U			1.07	4.28	1.07	2.90	1.07	2.14	8.17%	16.35%	8.17%	11.09%	8.17%	8.17%
	DAQ			2.59	6.67	2.59	5.86	2.59	5.19	19.81%	25.46%	19.81%	22.37%	19.81%	19.81%
	DOR			2.21	5.47	2.21	5.23	2.21	4.42	16.88%	20.89%	16.88%	19.95%	16.88%	16.88%
	$SEID$			10.23	23.12	10.23	21.11	10.23	20.47	78.09%	88.21%	78.09%	80.55%	78.09%	78.09%
Lily	CG^L	46.94	93.89	3.67	16.30	3.67	11.90	3.67	7.35	7.83%	17.37%	7.83%	12.68%	7.83%	7.83%
	CG^U			3.67	16.30	3.67	11.90	3.67	7.35	7.83%	17.37%	7.83%	12.68%	7.83%	7.83%
	DAQ			6.71	23.55	6.71	20.26	6.71	13.43	14.31%	25.09%	14.31%	21.59%	14.31%	14.31%
	DOR			5.81	17.71	5.81	18.81	5.81	11.63	12.39%	18.87%	12.39%	20.03%	12.39%	12.39%
	$SEID$			33.45	74.76	33.45	71.67	33.45	66.90	71.26%	79.63%	71.26%	76.34%	71.26%	71.26%
Ola	CG^L	65.56	131.12	0.40	0.81	0.40	0.81	0.40	0.81	0.62%	0.62%	0.62%	0.62%	0.62%	0.62%
	CG^U			0.40	0.81	0.40	0.81	0.40	0.81	0.62%	0.62%	0.62%	0.62%	0.62%	0.62%
	DAQ			0.25	0.50	0.25	0.50	0.25	0.50	0.39%	0.39%	0.39%	0.39%	0.39%	0.39%
	DOR			0.70	1.40	0.70	1.40	0.70	1.40	1.07%	1.07%	1.07%	1.07%	1.07%	1.07%
	$SEID$			5.62	11.25	5.62	11.25	5.62	11.25	8.58%	8.58%	8.58%	8.58%	8.58%	8.58%
Ontodna	CG^L	5.91	11.83	0.65	1.69	0.65	1.42	0.65	1.31	11.13%	14.31%	11.13%	12.02%	11.13%	11.13%
	CG^U			0.65	1.69	0.65	1.42	0.65	1.31	11.13%	14.31%	11.13%	12.02%	11.13%	11.13%
	DAQ			1.23	2.95	1.23	2.65	1.23	2.47	20.89%	24.95%	20.89%	22.43%	20.89%	20.89%
	DOR			0.61	1.45	0.61	1.37	0.61	1.22	10.32%	12.31%	10.32%	11.64%	10.32%	10.32%
	$SEID$			4.06	9.50	4.06	8.25	4.06	8.12	68.70%	80.28%	68.70%	69.75%	68.70%	68.70%

Table A.5: Average candidate alignment sizes with and without modularization

System	MT	Original size (B)	Original arguments	Retained alignment size and # of arguments						Avg percent					
				C M	C A	SO M	SO A	N M	N A	C M	C A	SO M	SO A	N M	N A
Asmov	CG^L	13.23	26.45	1.99	6.21	1.99	4.74	1.99	3.97	15.01%	23.47%	15.01%	17.93%	15.01%	15.01%
	CG^U			1.99	6.21	1.99	4.74	1.99	3.97	15.01%	23.47%	15.01%	17.93%	15.01%	15.01%
	DAQ			3.47	10.51	3.47	8.47	3.47	6.94	26.23%	39.74%	26.23%	32.01%	26.23%	26.23%
	DOR			3.69	10.40	3.69	9.37	3.69	7.38	27.91%	39.30%	27.91%	35.43%	27.91%	27.91%
	$SEID$			8.68	21.00	8.68	17.88	8.68	17.36	65.64%	79.38%	65.64%	67.59%	65.64%	65.64%
Falcon	CG^L	13.11	26.22	2.21	8.22	2.21	5.89	2.21	4.43	16.89%	31.34%	16.89%	22.47%	16.89%	16.89%
	CG^U			2.21	8.22	2.21	5.89	2.21	4.43	16.89%	31.34%	16.89%	22.47%	16.89%	16.89%
	DAQ			5.36	14.46	5.36	12.59	5.36	10.71	40.85%	55.14%	40.85%	48.02%	40.85%	40.85%
	DOR			5.43	13.15	5.43	12.58	5.43	10.86	41.43%	50.15%	41.43%	47.97%	41.43%	41.43%
	$SEID$			10.24	23.13	10.24	21.12	10.24	20.47	78.09%	88.21%	78.09%	80.55%	78.09%	78.09%
Lily	CG^L	46.94	93.89	4.58	20.58	4.58	15.11	4.58	9.15	9.75%	21.92%	9.75%	16.09%	9.75%	9.75%
	CG^U			4.58	20.58	4.58	15.11	4.58	9.15	9.75%	21.92%	9.75%	16.09%	9.75%	9.75%
	DAQ			9.76	35.56	9.76	30.47	9.76	19.52	20.79%	37.87%	20.79%	32.45%	20.79%	20.79%
	DOR			8.97	26.35	8.97	27.85	8.97	17.94	19.11%	28.06%	19.11%	29.66%	19.11%	19.11%
	$SEID$			33.45	74.76	33.45	71.67	33.45	66.91	71.26%	79.63%	71.26%	76.34%	71.26%	71.26%
Ola	CG^L	65.56	131.12	7.66	15.33	7.66	15.33	7.66	15.33	11.69%	11.69%	11.69%	11.69%	11.69%	11.69%
	CG^U			7.66	15.33	7.66	15.33	7.66	15.33	11.69%	11.69%	11.69%	11.69%	11.69%	11.69%
	DAQ			3.16	6.32	3.16	6.32	3.16	6.32	4.82%	4.82%	4.82%	4.82%	4.82%	4.82%
	DOR			8.62	17.24	8.62	17.24	8.62	17.24	13.15%	13.15%	13.15%	13.15%	13.15%	13.15%
	$SEID$			61.90	123.80	61.90	123.80	61.90	123.80	94.41%	94.41%	94.41%	94.41%	94.41%	94.41%
Ontodna	CG^L	5.91	11.83	1.81	4.48	1.81	3.97	1.81	3.61	30.54%	37.86%	30.54%	33.57%	30.54%	30.54%
	CG^U			1.81	4.48	1.81	3.97	1.81	3.61	30.54%	37.86%	30.54%	33.57%	30.54%	30.54%
	DAQ			3.14	7.75	3.14	6.95	3.14	6.28	53.08%	65.53%	53.08%	58.77%	53.08%	53.08%
	DOR			2.27	5.37	2.27	5.06	2.27	4.54	38.35%	45.37%	38.35%	42.75%	38.35%	38.35%
	$SEID$			4.22	9.87	4.22	8.57	4.22	8.44	71.34%	83.37%	71.34%	72.43%	71.34%	71.34%

Table A.6: Average candidate alignment sizes with and without modularization (excluding alignments of size 0)

		Avg size with mod.			Avg retained size		
System	MT	C	SO	N	C	SO	N
Asmov	CG^L	1.09	1.49	0.93	9.23%	12.08%	8.05%
	CG^U	1.09	1.49	0.93	9.23%	12.08%	8.05%
B size	DAQ	1.77	2.19	1.49	14.16%	17.27%	11.87%
13.22	DOR	1.79	2.01	1.41	13.97%	15.33%	10.98%
	$SEID$	8.94	10.50	8.68	71.64%	81.22%	69.96%
Falcon	CG^L	1.45	2.14	1.07	11.38%	16.36%	8.40%
	CG^U	1.45	2.14	1.07	11.38%	16.36%	8.40%
B size	DAQ	2.93	3.33	2.59	22.26%	25.53%	19.69%
13.11	DOR	2.61	2.73	2.21	20.69%	21.42%	17.61%
	$SEID$	10.55	11.56	10.23	82.37%	89.50%	79.99%
Lily	CG^L	5.95	8.15	3.67	12.70%	17.13%	7.88%
	CG^U	5.95	8.15	3.67	12.70%	17.13%	7.88%
B size	DAQ	10.13	11.77	6.71	22.05%	25.55%	14.50%
46.95	DOR	9.40	8.85	5.81	18.95%	17.81%	11.69%
	$SEID$	35.83	37.38	33.45	76.32%	79.53%	71.55%
Ola	CG^L	0.25	0.25	0.25	0.70%	0.70%	0.70%
	CG^U	0.25	0.25	0.25	0.70%	0.70%	0.70%
B size	DAQ	0.15	0.15	0.15	0.41%	0.41%	0.41%
45.05	DOR	0.38	0.38	0.38	1.03%	1.03%	1.03%
	$SEID$	3.20	3.20	3.20	8.60%	8.60%	8.60%
Ontodna	CG^L	0.71	0.84	0.65	10.40%	11.94%	9.70%
	CG^U	0.71	0.84	0.65	10.40%	11.94%	9.70%
B size	DAQ	1.32	1.47	1.23	21.14%	23.19%	19.96%
5.92	DOR	0.68	0.72	0.61	10.88%	11.34%	9.94%
	$SEID$	4.12	4.75	4.06	70.30%	79.62%	69.44%

Table A.7: Average alignment sizes with and without modularization

		Avg size with mod.			Avg retained size		
System	MT	C	SO	N	C	SO	N
Asmov	CG^L	2.36	3.09	1.99	19.00%	23.95%	16.15%
	CG^U	2.36	3.09	1.99	19.00%	23.95%	16.15%
B size	DAQ	4.23	5.26	3.47	33.89%	41.64%	27.49%
13.22	DOR	4.69	5.20	3.69	36.87%	40.32%	28.92%
	$SEID$	8.94	10.50	8.68	71.64%	81.22%	69.96%
Falcon	CG^L	2.93	4.07	2.21	23.59%	31.68%	17.92%
	CG^U	2.93	4.07	2.21	23.59%	31.68%	17.92%
B size	DAQ	6.30	7.23	5.36	47.96%	55.35%	40.80%
13.11	DOR	6.29	6.57	5.43	50.66%	52.45%	44.05%
	$SEID$	10.56	11.56	10.24	82.37%	89.50%	79.99%
Lily	CG^L	7.45	10.13	4.58	16.19%	21.62%	10.11%
	CG^U	7.45	10.13	4.58	16.19%	21.62%	10.11%
B size	DAQ	15.23	17.78	9.76	34.23%	39.89%	21.87%
46.95	DOR	13.93	13.17	8.97	29.40%	27.79%	19.06%
	$SEID$	35.84	37.38	33.45	76.32%	79.53%	71.55%
Ola	CG^L	4.85	4.85	4.85	13.55%	13.55%	13.55%
	CG^U	4.85	4.85	4.85	13.55%	13.55%	13.55%
B size	DAQ	1.87	1.87	1.87	5.44%	5.44%	5.44%
45.05	DOR	4.68	4.68	4.68	13.20%	13.20%	13.20%
	$SEID$	35.20	35.20	35.20	94.61%	94.61%	94.61%
Ontodna	CG^L	1.97	2.23	1.81	36.32%	39.30%	33.57%
	CG^U	1.97	2.23	1.81	36.32%	39.30%	33.57%
B size	DAQ	3.48	3.88	3.14	60.01%	65.71%	54.97%
5.92	DOR	2.53	2.68	2.27	44.20%	46.22%	40.90%
	$SEID$	4.29	4.93	4.22	73.00%	82.68%	72.11%

Table A.8: Average alignment sizes with and without modularization (excluding alignments of size 0)

Average original alignment size (B): 24.84						
		DAQ	DOR	CG^U	CG^L	$SEID$
Accepted alignment size	C	6.22	6.42	3.91	3.91	18.96
	SO	7.20	6.46	4.87	4.87	19.92
	N	4.72	5.01	3.09	3.09	18.36
Accepted alignment size (%)	C	36.31%	34.87%	21.73%	21.73%	79.59%
	SO	41.61%	36.00%	26.02%	26.02%	85.51%
	N	30.11%	29.23%	18.26%	18.26%	77.64%
Average candidate alignment size: 28.95						
Average # of arguments: 57.90						
		DAQ	DOR	CG^U	CG^L	$SEID$
Avg candidates with mod.	C	4.98	5.80	3.65	3.65	23.70
	SO	14.92	14.50	10.96	10.96	50.51
	N	4.98	5.80	3.65	3.65	23.70
Avg # args with mod.	C	12.96	14.42	9.01	9.01	48.61
	SO	4.98	5.80	3.65	3.65	23.70
	N	9.95	11.59	7.30	7.30	47.40
Avg candidates with mod. (%)	C	29.16%	27.99%	16.78%	16.78%	76.15%
	SO	29.16%	27.99%	16.78%	16.78%	76.15%
	N	29.16%	27.99%	16.78%	16.78%	76.15%
Avg # args with mod. (%)	C	40.62%	35.21%	25.26%	25.26%	85.00%
	SO	35.22%	33.79%	20.35%	20.35%	78.26%
	N	29.16%	27.99%	16.78%	16.78%	76.15%

Table A.9: Average accepted alignment sizes (averaged by modularization technique, excluding alignments of size 0)

MT	System	Size 0	Size ≠ 0	Total	Size 0 %
DAQ	Asmov	1446	737	2183	66.24%
	Falcon	1725	904	2629	65.61%
	Lily	1034	1581	2615	39.54%
	Ola	4924	336	5260	93.61%
	Ontodna	3838	1276	5114	75.05%
	Average				68.01%
DOR	Asmov	1374	809	2183	62.94%
	Falcon	1624	1005	2629	61.77%
	Lily	977	1638	2615	37.36%
	Ola	4924	336	5260	93.61%
	Ontodna	3902	1212	5114	76.30%
	Average				66.40%
CG^U	Asmov	1235	948	2183	56.57%
	Falcon	1370	1259	2629	52.11%
	Lily	713	1902	2615	27.27%
	Ola	5041	219	5260	95.84%
	Ontodna	3548	1566	5114	69.38%
	Average				60.23%
CG^L	Asmov	1235	948	2183	56.57%
	Falcon	1370	1259	2629	52.11%
	Lily	713	1902	2615	27.27%
	Ola	5041	219	5260	95.84%
	Ontodna	3548	1566	5114	69.38%
	Average				60.23%
$SEID$	Asmov	0	2183	2183	0.00%
	Falcon	0	2629	2629	0.00%
	Lily	0	2615	2615	0.00%
	Ola	4880	380	5260	92.78%
	Ontodna	208	4906	5114	4.07%
	Average				19.37%
Over all techniques					
	Asmov	5290	5625	10915	48.47%
	Falcon	6089	7056	13145	46.32%
	Lily	3437	9638	13075	26.29%
	Ola	24810	1490	26300	94.33%
	Ontodna	15044	10526	25570	58.83%
	Overall average				54.85%

Table A.10: Percentage of empty alignments by modularization technique and alignment system

Appendix B

Thesis Ontology

B.1 Thesis Ontology Axiomatization

Classes

Academic

Academic $\equiv \exists$ hasRole {lecturer} $\sqcup \exists$ hasRole {professor}

Academic $\sqsubseteq$ Person

Chapter

Chapter $\sqsubseteq$ Thing

Chapter $\sqsubseteq \neg$ Person

Chapter $\sqsubseteq \neg$ Role

Chapter $\sqsubseteq \neg$ Section

Chapter $\sqsubseteq \neg$ Thesis

Chapter $\sqsubseteq \neg$ Person

Role $\sqsubseteq \neg$ Person

Section $\sqsubseteq \neg$ Person

Thesis $\sqsubseteq \neg$ Person

Chapter $\sqsubseteq \neg$ Role

Person $\sqsubseteq \neg$ Role

Section $\sqsubseteq \neg$ Role

Thesis $\sqsubseteq \neg$ Role

Chapter $\sqsubseteq \neg$ Section

Person $\sqsubseteq \neg$ Section

Role $\sqsubseteq \neg$ Section

Thesis $\sqsubseteq \neg$ Section

Chapter $\sqsubseteq \neg$ Thesis

Person $\sqsubseteq \neg$ Thesis

Role $\sqsubseteq \neg$ Thesis

Section $\sqsubseteq \neg$ Thesis

MastersStudent

MastersStudent $\equiv \exists$ hasRole {masters}

MastersStudent $\sqsubseteq$ PostGradStudent

MastersThesis

MastersThesis $\equiv \exists$ hasAuthor MastersStudent

MastersThesis $\sqsubseteq$ PostGradThesis

MastersThesis $\sqsubseteq \neg$ PhDThesis

Person

Person $\sqsubseteq \exists$ hasRole Role

Person $\sqsubseteq$ Thing

Person $\sqsubseteq \neg$ Chapter

Role $\sqsubseteq \neg$ Chapter

Section $\sqsubseteq \neg$ Chapter

Thesis $\sqsubseteq \neg$ Chapter

Person $\sqsubseteq \neg$ Chapter

Person $\sqsubseteq \neg$ Role

Person $\sqsubseteq \neg$ Section

Person $\sqsubseteq \neg$ Thesis

Chapter $\sqsubseteq \neg$ Role

Person $\sqsubseteq \neg$ Role

Section $\sqsubseteq \neg$ Role

Thesis $\sqsubseteq \neg$ Role

Chapter $\sqsubseteq \neg$ Section

Person $\sqsubseteq \neg$ Section

Role $\sqsubseteq \neg$ Section

Thesis $\sqsubseteq \neg$ Section

Chapter $\sqsubseteq \neg$ Thesis

Person $\sqsubseteq \neg$ Thesis

Role $\sqsubseteq \neg$ Thesis

Section $\sqsubseteq \neg$ Thesis

PhDStudent

PhDStudent $\equiv \exists$ hasRole {phd}

PhDStudent $\sqsubseteq$ PostGradStudent

PhDThesis

PhDThesis $\equiv \exists$ hasAuthor PhDStudent

PhDThesis $\sqsubseteq$ PostGradThesis

PhDThesis $\sqsubseteq \neg$ MastersThesis

PostGradStudent

PostGradStudent $\equiv \exists$ hasRole {masters} $\sqcup \exists$ hasRole {phd}

PostGradStudent $\sqsubseteq$ Student

PostGradThesis

PostGradThesis $\equiv \exists$ hasAuthor PostGradStudent

PostGradThesis $\equiv =$ hasFirstSupervisor Academic

PostGradThesis $\equiv =$ hasSecondSupervisor Academic

PostGradThesis $\sqsubseteq$ Thesis

PostGradThesis $\sqsubseteq \neg$ UnderGradThesis

Role

Role $\equiv$ {professor} $\sqcup$ {lecturer} $\sqcup$ {phd} $\sqcup$ {undergrad} $\sqcup$ {masters}

Role $\sqsubseteq$ Thing

Person $\sqsubseteq \neg$ Chapter

Role $\sqsubseteq \neg$ Chapter

Section $\sqsubseteq \neg$ Chapter

Thesis $\sqsubseteq \neg$ Chapter

Chapter $\sqsubseteq \neg$ Person

Role $\sqsubseteq \neg$ Person

Section $\sqsubseteq \neg$ Person

Thesis $\sqsubseteq \neg$ Person

Role $\sqsubseteq \neg$ Chapter

Role $\sqsubseteq \neg$ Person

Role $\sqsubseteq \neg$ Section

Role $\sqsubseteq \neg$ Thesis

Chapter $\sqsubseteq \neg$ Section

Person $\sqsubseteq \neg$ Section

Role $\sqsubseteq \neg$ Section

Thesis $\sqsubseteq \neg$ Section

Chapter $\sqsubseteq \neg$ Thesis

Person $\sqsubseteq \neg$ Thesis

Role $\sqsubseteq \neg$ Thesis
Section $\sqsubseteq \neg$ Thesis

Section

Section $\sqsubseteq$ Thing
    Person $\sqsubseteq \neg$ Chapter
Role $\sqsubseteq \neg$ Chapter
Section $\sqsubseteq \neg$ Chapter
Thesis $\sqsubseteq \neg$ Chapter
Chapter $\sqsubseteq \neg$ Person
Role $\sqsubseteq \neg$ Person
Section $\sqsubseteq \neg$ Person
Thesis $\sqsubseteq \neg$ Person
Chapter $\sqsubseteq \neg$ Role
Person $\sqsubseteq \neg$ Role
Section $\sqsubseteq \neg$ Role
Thesis $\sqsubseteq \neg$ Role
Section $\sqsubseteq \neg$ Chapter
Section $\sqsubseteq \neg$ Person
Section $\sqsubseteq \neg$ Role
Section $\sqsubseteq \neg$ Thesis
Chapter $\sqsubseteq \neg$ Thesis
Person $\sqsubseteq \neg$ Thesis
Role $\sqsubseteq \neg$ Thesis
Section $\sqsubseteq \neg$ Thesis

Student

Student $\equiv \exists$ hasRole {masters} $\sqcup \exists$ hasRole {phd} $\sqcup \exists$ hasRole {undergrad}
    Student $\sqsubseteq$ Person

Thesis

Thesis $\equiv \exists$ hasChapter Chapter
    Thesis $\equiv =$ hasAuthor Person
    Thesis $\equiv \geq 1$ hasSupervisor Academic
    Thesis $\sqsubseteq$ Thing
    Thesis $\sqsubseteq =$ hasAuthor Person

Person $\sqsubseteq \neg$ Chapter
Role $\sqsubseteq \neg$ Chapter
Section $\sqsubseteq \neg$ Chapter
Thesis $\sqsubseteq \neg$ Chapter
Chapter $\sqsubseteq \neg$ Person
Role $\sqsubseteq \neg$ Person
Section $\sqsubseteq \neg$ Person
Thesis $\sqsubseteq \neg$ Person
Chapter $\sqsubseteq \neg$ Role
Person $\sqsubseteq \neg$ Role
Section $\sqsubseteq \neg$ Role
Thesis $\sqsubseteq \neg$ Role
Chapter $\sqsubseteq \neg$ Section
Person $\sqsubseteq \neg$ Section
Role $\sqsubseteq \neg$ Section
Thesis $\sqsubseteq \neg$ Section
Thesis $\sqsubseteq \neg$ Chapter
Thesis $\sqsubseteq \neg$ Person
Thesis $\sqsubseteq \neg$ Role
Thesis $\sqsubseteq \neg$ Section

Thing

UnderGradStudent

UnderGradStudent $\equiv \exists$ hasRole {undergrad}
UnderGradStudent $\sqsubseteq$ Student

UnderGradThesis

UnderGradThesis $\equiv \exists$ hasAuthor UnderGradStudent
UnderGradThesis $\sqsubseteq$ Thesis
UnderGradThesis $\sqsubseteq \neg$ PostGradThesis

Object properties

authorOf

authorOf $\equiv$ hasAuthor$^-$

chapterOf

chapterOf $\equiv$ hasChapter$^-$

hasAuthor

$\text{authorOf} \equiv \text{hasAuthor}^{-}$

$\exists\ \text{hasAuthor}\ \text{Thing} \sqsubseteq \text{Thesis}$

$\top \sqsubseteq \forall\ \text{hasAuthor}\ \text{Student}$

hasChapter

$\text{chapterOf} \equiv \text{hasChapter}^{-}$

$\exists\ \text{hasChapter}\ \text{Thing} \sqsubseteq \text{Thesis}$

$\top \sqsubseteq \forall\ \text{hasChapter}\ \text{Chapter}$

hasFirstSupervisor

$\sqsubseteq \text{hasSupervisor}$

hasRole

$\exists\ \text{hasRole}\ \text{Thing} \sqsubseteq \text{Person}$

$\top \sqsubseteq \forall\ \text{hasRole}\ \text{Role}$

hasSecondSupervisor

$\sqsubseteq \text{hasSupervisor}$

hasSection

$\exists\ \text{hasSection}\ \text{Thing} \sqsubseteq \text{Chapter}$

$\top \sqsubseteq \forall\ \text{hasSection}\ \text{Section}$

hasSubSection

$\exists\ \text{hasSubSection}\ \text{Thing} \sqsubseteq \text{Section}$

$\top \sqsubseteq \forall\ \text{hasSubSection}\ \text{Section}$

hasSupervisor

$\text{supervisorOf} \equiv \text{hasSupervisor}^{-}$

$\exists\ \text{hasSupervisor}\ \text{Thing} \sqsubseteq \text{Thesis}$

$\top \sqsubseteq \forall\ \text{hasSupervisor}\ \text{Academic}$

supervisorOf

$\text{supervisorOf} \equiv \text{hasSupervisor}^{-}$

Individuals

lecturer

lecturer : Role
{professor} ≠ {lecturer} ≠ {phd} ≠ {undergrad} ≠ {masters}

masters

masters : Role
{professor} ≠ {lecturer} ≠ {phd} ≠ {undergrad} ≠ {masters}

phd

phd : Role
{professor} ≠ {lecturer} ≠ {phd} ≠ {undergrad} ≠ {masters}

professor

professor : Role
{professor} ≠ {lecturer} ≠ {phd} ≠ {undergrad} ≠ {masters}

undergrad

undergrad : Role
{professor} ≠ {lecturer} ≠ {phd} ≠ {undergrad} ≠ {masters}

B.2 Thesis Ontology TBox in TURTLE

```
@prefix : <http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#> .
@prefix owl2xml: <http://www.w3.org/2006/12/owl2-xml#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl2: <http://www.w3.org/2006/12/owl2#> .
@prefix thesis: <http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@base <http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl> .

<http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl> rdf:type owl:Ontology .

#################################################################
#
#    Object Properties
```

```
#
#################################################################


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#authorOf

:authorOf rdf:type owl:ObjectProperty ;

          owl:inverseOf :hasAuthor .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#chapterOf

:chapterOf rdf:type owl:ObjectProperty ;

           owl:inverseOf :hasChapter .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasAuthor

:hasAuthor rdf:type owl:ObjectProperty ;

           rdfs:range :Student ;

           rdfs:domain :Thesis .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasChapter

:hasChapter rdf:type owl:ObjectProperty ;

            rdfs:range :Chapter ;

            rdfs:domain :Thesis .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasFirstSupervisor

:hasFirstSupervisor rdf:type owl:ObjectProperty ;
```

```
                    rdfs:subPropertyOf :hasSupervisor .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasRole

:hasRole rdf:type owl:ObjectProperty ;

         rdfs:domain :Person ;

         rdfs:range :Role .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasSecondSupervisor

:hasSecondSupervisor rdf:type owl:ObjectProperty ;

                     rdfs:subPropertyOf :hasSupervisor .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasSection

:hasSection rdf:type owl:ObjectProperty ;

            rdfs:domain :Chapter ;

            rdfs:range :Section .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasSubSection

:hasSubSection rdf:type owl:ObjectProperty ,
                        owl:TransitiveProperty ;

               rdfs:range :Section ;

               rdfs:domain :Section .
```

```
###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasSupervisor

:hasSupervisor rdf:type owl:ObjectProperty ;

               rdfs:range :Academic ;

               rdfs:domain :Thesis .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#supervisorOf

:supervisorOf rdf:type owl:ObjectProperty ;

              owl:inverseOf :hasSupervisor .




#################################################################
#
#    Data properties
#
#################################################################


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#endDate

:endDate rdf:type owl:DatatypeProperty ;

         rdfs:domain :Role ;

         rdfs:range xsd:date .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#startDate

:startDate rdf:type owl:DatatypeProperty ;

           rdfs:domain :Role ;
```

```
          rdfs:range xsd:date .



#################################################################
#
#    Classes
#
#################################################################


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Academic

:Academic rdf:type owl:Class ;

          owl:equivalentClass [ rdf:type owl:Class ;
                                owl:unionOf ( [ rdf:type owl:Restriction ;
                                                owl:onProperty :hasRole ;
                                                owl:hasValue :lecturer .
                                              ]
                                              [ rdf:type owl:Restriction ;
                                                owl:onProperty :hasRole ;
                                                owl:hasValue :professor .
                                              ]
                                            ) .
                              ] ;

          rdfs:subClassOf :Person .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Chapter

:Chapter rdf:type owl:Class ;

         rdfs:subClassOf owl:Thing .
```

```
###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#MastersStudent

:MastersStudent rdf:type owl:Class ;

                owl:equivalentClass [ rdf:type owl:Restriction ;
                                      owl:onProperty :hasRole ;
                                      owl:hasValue :masters .
                                    ] ;

                rdfs:subClassOf :PostGradStudent .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#MastersThesis

:MastersThesis rdf:type owl:Class ;

               owl:equivalentClass [ rdf:type owl:Restriction ;
                                     owl:onProperty :hasAuthor ;
                                     owl:someValuesFrom :MastersStudent .
                                   ] ;

               rdfs:subClassOf :PostGradThesis ;

               owl:disjointWith :PhDThesis .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Person

:Person rdf:type owl:Class ;

        rdfs:subClassOf owl:Thing ,
                        [ rdf:type owl:Restriction ;
                          owl:onProperty :hasRole ;
                          owl:someValuesFrom :Role .
                        ] .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#PhDStudent
```

```
:PhDStudent rdf:type owl:Class ;

            owl:equivalentClass [ rdf:type owl:Restriction ;
                                  owl:onProperty :hasRole ;
                                  owl:hasValue :phd .
                                ] ;

            rdfs:subClassOf :PostGradStudent .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#PhDThesis

:PhDThesis rdf:type owl:Class ;

           owl:equivalentClass [ rdf:type owl:Restriction ;
                                 owl:onProperty :hasAuthor ;
                                 owl:someValuesFrom :PhDStudent .
                               ] ;

           rdfs:subClassOf :PostGradThesis .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#PostGradStudent

:PostGradStudent rdf:type owl:Class ;

                 owl:equivalentClass [ rdf:type owl:Class ;
                                       owl:unionOf ( [ rdf:type owl:Restriction ;
                                                       owl:onProperty :hasRole ;
                                                       owl:hasValue :masters .
                                                     ]
                                                     [ rdf:type owl:Restriction ;
                                                       owl:onProperty :hasRole ;
                                                       owl:hasValue :phd .
                                                     ]
                                                   ) .
                                     ] ;

                 rdfs:subClassOf :Student .
```

```
###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#PostGradThesis

:PostGradThesis rdf:type owl:Class ;

                owl:equivalentClass [ rdf:type owl:Restriction ;
                                      owl:onProperty :hasFirstSupervisor ;
                                      owl2:onClass :Academic ;
                                      owl:cardinality "1"^^xsd:nonNegativeInteger .
                                    ] ,
                                    [ rdf:type owl:Restriction ;
                                      owl:onProperty :hasAuthor ;
                                      owl:someValuesFrom :PostGradStudent .
                                    ] ,
                                    [ rdf:type owl:Restriction ;
                                      owl:onProperty :hasSecondSupervisor ;
                                      owl2:onClass :Academic ;
                                      owl:cardinality "1"^^xsd:nonNegativeInteger .
                                    ] ;

                rdfs:subClassOf :Thesis ;

                owl:disjointWith :UnderGradThesis .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Role

:Role rdf:type owl:Class ;

      owl:equivalentClass [ rdf:type owl:Class ;
                            owl:oneOf ( :professor
                                        :lecturer
                                        :phd
                                        :undergrad
                                        :masters
                                      ) .
                          ] ;
```

```
      rdfs:subClassOf owl:Thing .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Section

:Section rdf:type owl:Class ;

         rdfs:subClassOf owl:Thing .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Student

:Student rdf:type owl:Class ;

         owl:equivalentClass [ rdf:type owl:Class ;
                               owl:unionOf ( [ rdf:type owl:Restriction ;
                                               owl:onProperty :hasRole ;
                                               owl:hasValue :masters .
                                             ]
                                             [ rdf:type owl:Restriction ;
                                               owl:onProperty :hasRole ;
                                               owl:hasValue :phd .
                                             ]
                                             [ rdf:type owl:Restriction ;
                                               owl:onProperty :hasRole ;
                                               owl:hasValue :undergrad .
                                             ]
                                           ) .
                             ] ;

         rdfs:subClassOf :Person .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Thesis

:Thesis rdf:type owl:Class ;

        owl:equivalentClass [ rdf:type owl:Restriction ;
                              owl:onProperty :hasChapter ;
```

```
                              owl:someValuesFrom :Chapter .
                            ] ,
                            [ rdf:type owl:Restriction ;
                              owl:onProperty :hasAuthor ;
                              owl2:onClass :Person ;
                              owl:cardinality "1"^^xsd:nonNegativeInteger .
                            ] ,
                            [ rdf:type owl:Restriction ;
                              owl:onProperty :hasSupervisor ;
                              owl2:onClass :Academic ;
                              owl:minCardinality "1"^^xsd:nonNegativeInteger .
                            ] ;

        rdfs:subClassOf owl:Thing ,
                        [ rdf:type owl:Restriction ;
                          owl:onProperty :hasAuthor ;
                          owl2:onClass :Person ;
                          owl:cardinality "1"^^xsd:nonNegativeInteger .
                        ] .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#UnderGradStudent

:UnderGradStudent rdf:type owl:Class ;

                  owl:equivalentClass [ rdf:type owl:Restriction ;
                                        owl:onProperty :hasRole ;
                                        owl:hasValue :undergrad .
                                      ] ;

                  rdfs:subClassOf :Student .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#UnderGradThesis

:UnderGradThesis rdf:type owl:Class ;

                 owl:equivalentClass [ rdf:type owl:Restriction ;
                                       owl:onProperty :hasAuthor ;
```

```
                                        owl:someValuesFrom :UnderGradStudent .
                                    ] ;

                rdfs:subClassOf :Thesis .


###  http://www.w3.org/2002/07/owl#Thing

owl:Thing rdf:type owl:Class .




#################################################################
#
#    Individuals
#
#################################################################


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#lecturer

:lecturer rdf:type :Role .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#masters

:masters rdf:type :Role .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#phd

:phd rdf:type :Role .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#professor

:professor rdf:type :Role .
```

```
###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#undergrad

:undergrad rdf:type :Role .




#################################################################
#
#    General axioms
#
#################################################################


[ rdf:type owl2:AllDisjointClasses ;
  owl2:members ( :Chapter
                 :Person
                 :Role
                 :Section
                 :Thesis
               ) .
][ rdf:type owl:AllDifferent ;
   owl:distinctMembers ( :professor
                         :lecturer
                         :phd
                         :undergrad
                         :masters
                       ) .
 ]
```

B.3 Thesis Ontology ABox in TURTLE

```
@prefix : <http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#> .
@prefix owl2xml: <http://www.w3.org/2006/12/owl2-xml#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl2: <http://www.w3.org/2006/12/owl2#> .
@prefix thesis: <http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@base <http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl> .

<http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl> ;
          rdf:type owl:Ontology owl:imports null:thesis.owl .

#################################################################
#
#    Object Properties
#
#################################################################


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasAuthor

thesis:hasAuthor rdf:type owl:ObjectProperty .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasChapter

thesis:hasChapter rdf:type owl:ObjectProperty .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasFirstSupervisor

thesis:hasFirstSupervisor rdf:type owl:ObjectProperty .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasRole

thesis:hasRole rdf:type owl:ObjectProperty .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasSecondSupervisor

thesis:hasSecondSupervisor rdf:type owl:ObjectProperty .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasSection
```

```
thesis:hasSection rdf:type owl:ObjectProperty .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#hasSubSection

thesis:hasSubSection rdf:type owl:ObjectProperty .




#################################################################
#
#    Classes
#
#################################################################


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Chapter

thesis:Chapter rdf:type owl:Class .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Person

thesis:Person rdf:type owl:Class .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Section

thesis:Section rdf:type owl:Class .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#Thesis

thesis:Thesis rdf:type owl:Class .
```

```
#################################################################
#
#    Individuals
#
#################################################################


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#chapter1

:chapter1 rdf:type thesis:Chapter ;

          thesis:hasSection :section1 ,
                            :section2 .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#chapter2

:chapter2 rdf:type thesis:Chapter .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#florianagrasso

:florianagrasso rdf:type thesis:Person ;

                thesis:hasRole thesis:lecturer .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl
###  #ontologymodularization

:ontologymodularization rdf:type thesis:Thesis ;

                        thesis:hasChapter :chapter1 ,
                                          :chapter2 ;

                        thesis:hasSecondSupervisor :florianagrasso ;

                        thesis:hasAuthor :pauldoran ;
```

```
                        thesis:hasFirstSupervisor :valentinatamma .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#pauldoran

:pauldoran rdf:type thesis:Person ;

           thesis:hasRole thesis:phd .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#section1

:section1 rdf:type thesis:Section ;

          thesis:hasSubSection :subsection1 ,
                               :subsection2 .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#section2

:section2 rdf:type thesis:Section .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#subsection1

:subsection1 rdf:type thesis:Section ;

             thesis:hasSubSection :subsubsection1 .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#subsection2

:subsection2 rdf:type thesis:Section .


###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#subsubsection1

:subsubsection1 rdf:type thesis:Section .
```

```
###  http://www.csc.liv.ac.uk/~pdoran/ontologies/thesisInstances.owl#valentinatamma

:valentinatamma rdf:type thesis:Person ;

                thesis:hasRole thesis:lecturer .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#lecturer

 .


###  http://www.csc.liv.ac.uk/~pdoran/ontology/thesis.owl#phd

 .
```

Bibliography

[1] Sivan Albagli, Rachel Ben-Eliyahu-Zohary, and Solomon Eyal Shimony. Markov network based ontology matching. In Boutilier [11], pages 1884–1889.

[2] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge, 2007.

[4] Claudine Beaumont. The web that thinks for itself. *The Daily Telegraph*, page 33, March 20, 2008.

[5] Trevor J.M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

[6] Amaia Bernaras, Iñaki Laresgoiti, and Jose Manuel Corera. Building and reusing ontologies for electrical network applications. In Wolfgang Wahlster, editor, *ECAI*, pages 298–302. John Wiley and Sons, Chichester, 1996.

[7] George S. Boolos, John P. Burgess, and Richard C. Jeffrey. *Computability and logic*. Cambridge University Press, 2007.

[8] Alex Borgida and Fausto Giunchiglia. Importing from functional knowledge bases - a preview. In Cuenca-Grau et al. [22].

[9] Alex Borgida and Luciano Serafini. Distributed description logics: Directed domain correspondences in federated information sources. In R. Meersman, Z. Tari, and et al, editors, *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE : Confederated International Conferences CoopIS, DOA, and ODBASE 2002. Proceedings*, volume 2519 of *Lecture Notes in Computer Science*, pages 36–53. Springer Berlin, 2002.

[10] Willem N. Borst. *Construction of Engineering ontologies for knowledge sharing and reuse.* PhD thesis, Centre for Telematica and Information Technology, University of Twente, 1997.

[11] Craig Boutilier, editor. *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 2009.

[12] Ronald J. Brachman and James G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2):171 – 216, 1985.

[13] Janez Brank, Marko Grobelnik, and Dunja Mladenic. A survey of ontology evaluation techniques. In *Proceedings of 8th International multi-conference Information Society*, pages 166–169, 2005.

[14] John G. Breslin, Andreas Harth, Uldis Bojars, and Stefan Decker. Towards semantically-interlinked online communities. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2005.

[15] Christopher Brewster, Kieron O'Hara, Steve Fuller, Yorick Wilks, Enrico Franconi, Mark A. Musen, Jeremy Ellman, and Simon Buckingham Shum. Knowledge representation with ontologies: The present and future. *IEEE Intelligent Systems*, 19(1):72–81, 2004.

[16] Bruce G. Buchanan and Edward H. Shortliffe. *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* The Addison-Wesley series in artificial intelligence. Addison-Wesley, 1984.

[17] Jacques Calmet and Anusch Daemi. From entropy to ontology. In *AT2AI-4 - Fourth International Symposium "From Agent Theory to Agent Implementation" at 17th European Meeting on Cybernetics and Systems Research, Vienna, April 2004.*, 2004.

[18] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. Open knowledge base connectivity 2.0. Technical Report KSL-98-06, Knowledge Systems Laboratory, Stanford University, 1998.

[19] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Proceedings of KDD Workshop on Data Cleaning and Object Consolidation*, pages 73–78, 2003.

[20] William W. Cohen, Alexander Borgida, and Haym Hirsh. Computing least common subsumers in description logics. In *AAAI*, pages 754–760, 1992.

[21] Oscar Corcho. *A Layered Declarative Approach to Ontology Translation with Knowledge Preservation (Frontiers in Artificial Intelligence and Applications)*. IOS Press, US, 2005.

[22] Bernardo Cuenca-Grau, Vasant Honavar, Anne Schlicht, and Frank Wolter, editors. *Proceedings of the 2nd International Workshop on Modular Ontologies, WOMO'07, Whistler, Canada, October 28, 2007*, 2007.

[23] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Just the right amount: Extracting modules from ontologies. In *WWW 2007, Proceedings of the 16th International World Wide Web Conference, Banff, Canada, May 8-12, 2007*, pages 717–727, 2007.

[24] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research (JAIR)*, 31:273–318, 2008.

[25] Bernardo Cuenca-Grau, Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Automatic Partitioning of OWL Ontologies Using E-Connections. In *Proceedings of the 2005 International Workshop on Description Logics (DL-2005)*, 2005.

[26] Mathieu d'Aquin, Paul Doran, Enrico Motta, and Valentina Tamma. Towards a parametric ontology modularization framework based on graph transformation. In Cuenca-Grau et al. [22].

[27] Mathieu d'Aquin, Marta Sabou, and Enrico Motta. Modularization: a key for the dynamic selection of relevant knowledge components. In *First International Workshop on Modular Ontologies, ISWC 2006, First International Workshop on Modular Ontologies, ISWC 2006*, Athens, Georgia, USA., 2006.

[28] Mathieu d'Aquin, Anne Schlicht, Heiner Stuckenschmidt, and Marta Sabou. Ontology modularization for knowledge selection: Experiments and evaluations. In *Database and Expert Systems Applications, 18th International Conference, DEXA 2007, Regensburg, Germany, September 3-7, 2007, Proceedings*, pages 874–883, 2007.

[29] R. Davis, H. Shrobe, and P. Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[30] Klaas Dellschaft and Steffen Staab. On how to perform a gold standard based evaluation of ontology learning. In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 228–241. Springer, 2006.

[31] Oxford Dictionaries. *Oxford English Dictionary*. Oxford University Press, 2008.

[32] Paul Doran, Ignazio Palmisano, and Valentina Tamma. SOMET: Algorithm and Tool for SPARQL Based Ontology Module Extraction. In Ulrike Sattler and Andrei Tamilin, editors, *Proceedings of the Workshop on Ontologies: Reasoning and Modularity (WORM-08)*. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-348/, 2008.

[33] Paul Doran, Valentina A. M. Tamma, and Luigi Iannone. Ontology module extraction for ontology reuse: an ontology engineering perspective. In Silva et al. [119], pages 61–70.

[34] Paul Doran, Valentina A. M. Tamma, Ignazio Palmisano, Terry R. Payne, and Luigi Iannone. Evaluating ontology modules using an entropy inspired metric. In Jain [77], pages 918–922.

[35] Paul Doran, Valentina A. M. Tamma, Terry R. Payne, and Ignazio Palmisano. Dynamic selection of ontological alignments: A space reduction mechanism. In Boutilier [11], pages 2028–2033.

[36] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321 – 357, 1995.

[37] Paul E. Dunne and Trevor J.M. Bench-Capon. Complexity in value-based argument systems. In *In Proceedings 9th European Conference, JELIA*, volume 3229 of *Lecture Notes in Artificial Intelligence*, pages 360–371. Springer Verlag, 2004.

[38] John Davies (Editor), Dieter Fensel (Editor), and Frank van Harmelen (Editor). *Towards the Semantic Web: Ontology-driven Knowledge Management*. John Wiley and Sons Ltd, UK, 2002.

[39] Marc Ehrig and Steffen Staab. QOM – Quick Ontology Mapping. In *The Semantic Web – ISWC 2004: Third International Semantic Web Conference, Hiroshima, Japan, November 7-11, 2004. Proceedings*, volume 33298 of *Lecture Notes in Computer Science*, page 683. Springer Berlin, 2004.

[40] Faezeh Ensan. Formalizing ontology modularization through the notion of interfaces. In Aldo Gangemi and Jérôme Euzenat, editors, *EKAW*, volume 5268 of *Lecture Notes in Computer Science*, pages 74–82. Springer, 2008.

[41] Faezeh Ensan and Weichang Du. An interface-based ontology modularization framework for knowledge encapsulation. In Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad

Thirunarayan, editors, *International Semantic Web Conference*, volume 5318 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2008.

[42] Jerome Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, 2007.

[43] Jérôme Euzenat, Antoine Zimmermann, and Frederico Luiz Gonçalves de Freitas. Alignment-based modules for encapsulating ontologies. In Cuenca-Grau et al. [22].

[44] Dieter Fensel, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. Oil: An ontology intrastructe for the semantic wen. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[45] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns : elements of reusable object-oriented software*. Addison Wesley, 1995.

[46] Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, Jos Lehmann Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, Jos Lehmann Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann. Ontology evaluation and validation: an integrated formal model for the quality diagnostic task. Technical report, Laboratory for Applied Ontology, 2005.

[47] Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann. Modelling ontology evaluation and validation. In *ESWC*, pages 140–154, 2006.

[48] Aldo Gangemi and Valentina Presutti. *Handbook of Ontologies (2nd edition)*, chapter Ontology Design Patterns. Springer: Berlin, 2009.

[49] Michael R. Genesereth and Richard E. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Technical Rerot Logic-92-1, Computer Science Department, Stanford University, 1992.

[50] Michael R. Genesereth and Nils J. Nilsson. *Logical foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.

[51] Alain Giboin, Diana Maynard, Jens Hartmann, Maria del Carmen Suarez-Figueroa, Roberta Cuel, and York Sure. Methods for ontology evaluation. KWeb Deliverable D1.2.3, University of Karlsruhe, DEC 2004.

[52] Asunción Gómez-Pérez. Towards a framework to verify knowledge sharing technology. *Expert Systems With Applications*, 11(4):519–529, 1996.

[53] Asunción Gómez-Pérez and María del Carmen Suárez-Figueroa. Scenarios for building ontology networks within the neon methodology. In Yolanda Gil and Natasha Fridman Noy, editors, *K-CAP*, pages 183–184. ACM, 2009.

[54] Asuncion Gomez-Perez, Mariano Fernandez-Lopez, and Oscar Corcho. *Ontological Engineering*. Springer, London, 2003.

[55] Asunción Gómez-Pérez and Dolores Rojas-Amaya. Ontological reengineering for reuse. In Dieter Fensel and Rudi Studer, editors, *EKAW*, volume 1621 of *Lecture Notes in Computer Science*, pages 139–156. Springer, 1999.

[56] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. A logical framework for modularity of ontologies. In Veloso [141], pages 298–303.

[57] Thomas R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In *KR*, pages 601–602, 1991.

[58] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[59] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995.

[60] Michael Grüninger and Mark Fox. Methodology for the design and evaluation of ontologies. In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13*, 1995.

[61] Nicola Guarino. Formal ontology in information systems. In *Proceedings of FOIS'98*, pages 3–15. IOS Press, 1998.

[62] Nicola Guarino. Some ontological principles for designing upper level lexical resources. In *Proceedings of the First International Conference on Lexical Resources and Evaluation*, pages 527–534, 1998.

[63] Nicola Guarino and Luca Boldrin. Ontological requirements for knowledge sharing. In *IJCAI93 Workshop on Knowledge Sharing and Information Interchange*, 1993.

[64] Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases: Towards a terminological classification. In N Mars, editor, *Towards very large knowledge bases: knowledge building and knowledge sharing (KBKS'95)*. IOS Press, 1995.

[65] Nicola Guarino and Christopher Welty. Supporting ontological analysis of taxonomical realtionships. *Data and knowledge engineering*, 39(1):51–74, 2001.

[66] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with ontoclean. *Commun. ACM*, 45(2):61–65, 2002.

[67] Glen Hart, Martina Johnson, and Catherine Dolbear. Rabbit: Developing a control natural language for authoring ontologies. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 348–360. Springer, 2008.

[68] Patrick Hayes. Rdf semantics, 2004.

[69] Patrick J. Hayes. The logic of frames. In J. Brachman and Hector J. Levesque, editors, *Readings in Knowledge Representation*. Morgan Kaufmann, 1985.

[70] Martin Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW2008)*, volume 5268 of *LNCS*, pages 332–347. Springer, 2008.

[71] Carl Hewitt. Offices are open systems. *ACM Transactions on Information Systems*, 4(3):271–287, 1986.

[72] David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik (Principles of Mathematical Logic)*. Springer-Verlag, 1928.

[73] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in owl. In *ISWC 08 The International Semantic Web Conference 2008, Karlsruhe, Germany*, 2008.

[74] Ian Horrocks. Ontologies and the semantic web. *Communications of the ACM*, 51(12):58–67, 2008.

[75] Ian Horrocks and Peter Patel-Schneider. Reducing owl entailment to description logic satisfiability. *Web Semantics*, 1(4):345 – 357, 2004. International Semantic Web Conference 2003.

[76] Wei Hu and Yuzhong Qu. Falcon-ao: A practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):237 – 239, 2008. World Wide Web Conference 2007Semantic Web Track.

[77] L. Jain, editor. *2008 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2008, 9-12 December 2008, Sydney, NSW, Australia, Main Conference Proceedings*. IEEE, 2008.

[78] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[79] Ernesto Jimenez-Ruiz, Bernardo Cuenca Grau, Ulrike Sattler, Thomas Schneider, and Rafael Berlanga. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *The Semantic Web: Research and Applications*, volume 5021 of *LNCS*, pages 185 – 199. Springer, 2008.

[80] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 1995.

[81] Michel Klein. Combining and relating ontologies: An analysis of problems and solutions. In *Proceedings of the IJCAI'01 Workshop on Ontologies and Information Sharing*, 2001.

[82] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter. Semantic modularity and module extraction in description logics. In *Proceedings of ECAI-2008: 18th European conference on Artificial Intelligence*, 2008.

[83] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[84] Loredana Laera. *Toward Shared Understanding - An Argumentation Based Approach for Communication in Open Multi-Agent Systems.* Phd thesis, University of Liverpool, 2008.

[85] Loredana Laera, Ian Blacoe, Valentina A. M. Tamma, Terry R. Payne, Jérôme Euzenat, and Trevor J. M. Bench-Capon. Argumentation over ontology correspondences in mas. In Edmund H. Durfee, Makoto Yokoo, Michael N. Huhns, and Onn Shehory, editors, *AAMAS*, page 228. IFAAMAS, 2007.

[86] Douglas B. Lenat and Edward A. Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47:185–250, 1991.

[87] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23(2):155–212, 1984.

[88] Hector J. Levesque. Is it enough to get the behavior right? In Boutilier [11], pages 1439–1444.

[89] E.J. Lowe. *Kinds of being. A study of individuation, identity and the logic of sortal terms.* Basil Blackwell, 1989.

[90] Carsten Lutz, Dirk Walther, and Frank Wolter. Conservative extensions in expressive description logics. In Veloso [141], pages 453–458.

[91] Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. Technical report, University of Karlsruhe, 2001.

[92] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[93] Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In Asunción Gómez-Pérez and V. Richard Benjamins, editors, *EKAW*, volume 2473 of *Lecture Notes in Computer Science*, pages 251–263. Springer, 2002.

[94] Deborah L. McGuinness. Ontologies come of age. In *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2002.

[95] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11), 1995.

[96] Marvin Minsky. A framework for representing knowledge. Ai laborartory memo, MIT, 1974.

[97] Martín O. Moguillansky and Renata Wassermann. Inconsistent-tolerant dl-lite reasoning: An argumentative approach. In *Automated Reasoning about Context and Ontology Evolution - ARCOE 2009 (at IJCAI 2009)*, 2009.

[98] Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1):87 –127, 1982.

[99] Allen Newell and Herbert A. Simon. Computer science as empirical inquiry: symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[100] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics (SMI), Department of Medicine, Stanford University School of Medicine, 2001.

[101] Natalya F. Noy and Mark A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6):983–1024, December 2003.

[102] Natalya Fridman Noy, Ray W. Fergerson, and Mark A. Musen. The knowledge model of protégé-2000: Combining interoperability and flexibility. In Rose Dieng and Olivier Corby, editors, *EKAW*, volume 1937 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2000.

[103] Natalya Fridman Noy and Mark A. Musen. Specifying ontology views by traversal. In *International Semantic Web Conference*, pages 713–725, 2004.

[104] Ignazio Palmisano, Valentina A. M. Tamma, Luigi Iannone, Terry R. Payne, and Paul Doran. Dynamic change evaluation for ontology evolution in the semantic web. In Jain [77], pages 34–40.

[105] Ignazio Palmisano, Valentina A. M. Tamma, Terry R. Payne, and Paul Doran. Task oriented evaluation of module extraction techniques. In Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, editors, *International Semantic Web Conference*, volume 5823 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2009.

[106] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax, 2004.

[107] Charles S. Peirce. On the algebra of logic. *American Journal of Mathematics*, 3(1):15–57, 1880.

[108] Alan Rector, Amedeo Napoli, Giorgos Stamou, Giorgos Stoilos, Holger Wache, Jeff Pan, Mathieu d'Aquin, Stefano Spaccapietra, and Vassilis Tzouvaras. Report on modularization of ontologies. Technical report, Knowledge Web Deliverable D2.1.3.1, 2005.

[109] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (International Edition)*. Pearson Education, 2003.

[110] Thomas Saaty. A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology*, 15:234–281, 1977.

[111] Marta Sabou, Vanessa Lopez, Enrico Motta, and Victoria Uren. Ontology selection: Ontology evaluation on the real semantic web. In *Proceedings of the EON'2006 Workshop, "Evaluation of Ontologies on the Web", held in conjunction with WWW'2006*, 2006.

[112] Anne Schlicht and Heiner Stuckenschmidt. Towards structural criteria for ontology modularization. In *First International Workshop on Modular Ontologies, ISWC 2006, First International Workshop on Modular Ontologies, ISWC 2006*, Athens, Georgia, USA., 2006.

[113] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert de Hoog, Nigel Shadbolt, Walter Van de Velde, and Bob Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press, 1999.

[114] John R. Searle. Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3):417–457, 1980.

[115] Julian Seidenberg and Alan Rector. Web ontology segmentation: analysis, classification and use. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 13–22, New York, NY, USA, 2006. ACM Press.

[116] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, Jan-Feb 2006.

[117] Claude E. Shannon. A mathematical theory of communication. Technical Report 27:379-423, 623-656, Bell System Technical Report, July and October 1948.

[118] Yoav Shoham. Agent-oriented programming. *Artificial Intelligenct*, 60(1), 1993.

[119] Mário J. Silva, Alberto H. F. Laender, Ricardo A. Baeza-Yates, Deborah L. McGuinness, Bjørn Olstad, Øystein Haug Olsen, and André O. Falcão, editors. *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*. ACM, 2007.

[120] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2), 2007.

[121] David Smith. Focus: Internet explorers: www.thenewrevolutionaries after youtube and myspace, what next? *The Observer*, November 26, 2006.

[122] John F Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[123] John F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Cole, 2000.

[124] Steffen Staab and Rudi Studer, editors. *Handbook on Ontologies*. Springer, 2004.

[125] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34, 2001.

[126] Clifford Stoll. The internet? bah! *Newsweek*, February 1995.

[127] Heiner Stuckenschmidt and Michel C. A. Klein. Integrity and change in modular ontologies. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 900–908. Morgan Kaufmann, 2003.

[128] Heiner Stuckenschmidt and Michel C. A. Klein. Structure-based partitioning of large concept hierarchies. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 289–303. Springer, 2004.

[129] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering, principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[130] Mari Carmen Suarez-Figueroa, Eva Blomqvist, Mathieu D'Aquin, Mauricio Espinoza, Asuncion Gomez-Perez, Holger Lewen, Igor Mozetic, Raul Palma, Maria Poveda, Margherita Sini, Boris Villazon-Terrazas, Fouad Zablith, and Martin Dzbor. Revision and extension of the neon methodology for building contextualized ontology networks. NeOn Deliverable D5.4.2, UPM, 2009.

[131] Mari Carmen Suarez-Figueroa, Guadalupe Aguado de Cea, Carlos Buil, Klaas Dellschaft, Mariano Fernandez-Lopez, Andres Garcia, Asuncion Gomez-Perez, German Herrero, Elena Montiel-Ponsoda, Marta Sabou, Boris Villazon-Terrazas, and Zheng Yufei. Neon methodology for building contextualized ontology networks. NeOn Deliverable D5.4.1, UPM, 2008.

[132] Katia P. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

[133] Adolfo Lozano Tello and Asunción Gómez-Pérez. Ontometric: A method to choose the appropriate ontology. *Journal Database Management*, 15(2):1–18, 2004.

[134] James Hendler Tim Berners-Lee and Ora Lassila. The semantic web. *Scientific American*, May 2001.

[135] Dmitry Tsarkov and Ian Horrocks. Fact++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

[136] Alan M. Turing. *Computing machinery and intelligence*, chapter 1, pages 12–35. Computers and Thought. McGraw-Hill, Inc., New York, NY, USA, 1963.

[137] M. Uschold. Where is the semantics in the semantic web? In *Proceedings of the Workshop on Ontologies in Mutli-Agent Systems at the 5th Conference on Autonomous Agents*, 2001.

[138] Michael Uschold and Michael Grüninger. Ontologies: principles, methods and applications. *Knowledge Engineering Review*, 11(2), 1996.

[139] Gertjan van Heijst, A. Th. Schreiber, and Bob J. Wielinga. Using explicit ontologies in kbs development. *Int. J. Hum.-Comput. Stud.*, 46(2-3):183–292, 1997.

[140] Cornelis J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

[141] Manuela M. Veloso, editor. *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 2007.

[142] Pepijn R.S Visser, Dean M. Jones, T.J.M. Bench-Capon, and M.J.R. Shave. Assessing heterogeneity by classifying ontology mismatches. In Nicola Guarino, editor, *Formal Ontology In Information Systems*, pages 148–162. IOS Press, 1998.

[143] Raphael Volz, Daniel Oberle, Steffen Staab, and Boris Motik. Kaon server - a semantic web management system. In *Alternate Track Proceedings of the Twelfth International World Wide Web Conference, WWW2003, Budapest, Hungary, 20-24 May 2003*. ACM, 2003.

[144] Richard Waters. World-wise web? finally on the horizon are computers that can reason. *Financial Times*, March 3, 2008.

[145] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*. Cambridge University Press, 1910-13.

[146] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2002.

[147] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[148] Jonathan Yu, James A. Thom, and Audrey M. Tam. Ontology evaluation using wikipedia categories for browsing. In Silva et al. [119], pages 223–232.

[149] Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Organizational abstractions for the analysis and design of multi-agent system. In *First international workshop, AOSE 2000 on Agent-oriented software engineering*, pages 235–251, Secaucus, NJ, USA, 2001. Springer-Verlag New York, Inc.