2007

Representing and reasoning with modular ontologies

Jie Bao
Iowa State University

Representing and reasoning with modular ontologies

by

Jie Bao

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Vasant Honavar, Major Professor
Drena Dobbs
Gary Leavens
James McCalley
Giora Slutzki
Wallapak Tavanapong
Doina Caragea

Iowa State University
Ames, Iowa
2007

TABLE OF CONTENTS

LIST OF TABLES vii

LIST OF FIGURES viii

ACKNOWLEDGEMENTS x

ABSTRACT xi

CHAPTER 1. Motivation And Overview 1

1.1 Motivation 1

1.2 The Proposed Solution 2

1.2.1 Outline of the Theoretical Approach 2

1.2.2 Applications 4

1.3 Content Guide 6

CHAPTER 2. Preliminaries 8

2.1 Description Logics 8

2.1.1 Basic Notions 8

2.1.2 The Basic Description Logics $\mathcal{ALC}$ 10

2.1.3 Expressive Description Logics - the $\mathcal{SH}$ family 11

2.2 Reasoning with Description Logics 13

2.2.1 Reasoning Tasks for Description Logics 13

2.2.2 Basic Tableau Algorithms 14

2.2.3 Tableau Algorithms for Expressive DLs 17

2.3 Web Ontology Language - OWL 20

CHAPTER 3. Modular Ontologies: Desiderata and Abstract Description . . 24

3.1 Modular Ontology Desiderata 24

3.1.1 Syntactic Modularity 25

3.1.2 Semantic Modularity . . . 28
3.1.3 Other Modularity Considerations . . . 32
3.2 Modular Ontology Formalisms: Required Features . . . 34
3.2.1 Semantic Requirements . . . 34
3.2.2 Expressivity Requirements . . . 39
3.3 A General Framework for Modular Ontologies . . . 41
3.3.1 Semantics of Modular Ontologies at a Glance . . . 42
3.3.2 Abstract Modular Ontology . . . 43
3.3.3 Expressivity of AMO . . . 46
3.3.4 Semantic Requirements of AMO . . . 52
3.4 Discussion and Related Work . . . 56
3.4.1 Local Model Semantics and Distributed First Order Logic . . . 57
3.4.2 Propositional Logic of Context . . . 58
3.4.3 Epistemic Semantics for Peer-to-Peer Databases . . . 59
3.4.4 Modular Ontologies based on Conservative Extensions . . . 62
CHAPTER 4. Package-based Description Logics . . . 64
4.1 Overview . . . 64
4.2 Package-based Description Logics . . . 65
4.2.1 Syntax of P-DL . . . 65
4.2.2 Semantics . . . 67
4.2.3 $\mathcal{SHOIQP}$ Examples . . . 74
4.2.4 Reduction to Ordinary DL . . . 76
4.2.5 Properties of Semantic Importing . . . 77
4.2.6 Discussion on the P-DL Semantics . . . 79
4.3 Adopt OWL as the Syntax for P-DL . . . 84
4.3.1 Limitations of OWL Importing . . . 84
4.3.2 A Modular Semantics for OWL . . . 86
4.3.3 A Modular Syntax for OWL . . . 88
4.3.4 Summary and Discussion . . . 92
4.4 Related Work . . . 93
4.4.1 Other Modular Ontology Languages . . . 93

4.4.2 Semantics of Modular Ontology Languages 98
4.4.3 Limitations of Existing Approaches 106
4.4.4 Relation between Other Formalisms and P-DL 114
4.4.5 Syntax Extensions to OWL 116

CHAPTER 5. Distributed Reasoning with P-DL 123

5.1 Overview 123
5.2 Reasoning in $\mathcal{ALCP}_{\mathcal{C}}^{-}$ 125
5.2.1 $\mathcal{ALCP}_{\mathcal{C}}^{-}$ 125
5.2.2 Distributed Tableaux for $\mathcal{ALCP}_{\mathcal{C}}^{-}$ 128
5.2.3 A Tableau Algorithm for $\mathcal{ALCP}_{\mathcal{C}}^{-}$ 130
5.2.4 Soundness, Completeness, Termination and Complexity 142
5.3 A Reasoning Algorithm for $\mathcal{ALCP}_{\mathcal{C}}$ 144
5.3.1 Extended Subset Blocking 144
5.3.2 Correctness and Complexity 146
5.4 Asynchronous Federated Reasoning for $\mathcal{ALCP}_{\mathcal{C}}$ 147
5.5 Reasoning in $\mathcal{SHIQP}$ 150
5.5.1 Overview 150
5.5.2 P-DL $\mathcal{SHIQP}$ 151
5.5.3 A Tableau for $\mathcal{SHIQP}$ 153
5.5.4 An Asynchronous Tableau Algorithm for $\mathcal{SHIQP}$ 155
5.5.5 Example 160
5.6 Related Work 160

CHAPTER 6. Reasoning with Hidden Knowledge 164

6.1 Overview 164
6.2 Motivating Examples 165
6.3 Privacy-Preserving Reasoning: General Framework 166
6.3.1 Partially Hidden Knowledge 166
6.3.2 Privacy-Preserving Inference 168
6.3.3 General Strategies 171
6.4 Privacy-Preserving Reasoning with $\mathcal{SHIQ}$ Ontologies 173
6.5 Privacy-Preserving Reasoning with Hierarchical Ontologies 175

6.6 Discussion: Privacy-Preserving Reasoning in P-DL . . . 178
6.6.1 Overview . . . 178
6.6.2 Distributed Privacy-Preserving Reasoning: General Setting . . . 179
6.6.3 Requirement for Distributed Privacy-preserving Reasoners . . . 184
6.7 Related Work . . . 185
6.7.1 Policy Languages . . . 185
6.7.2 Preventing Unwanted Inference . . . 186
6.7.3 Epistemic Semantics . . . 187
CHAPTER 7. Collaborative Building of Modular Ontologies . . . 188
7.1 General Desiderate of Collaborative Ontology Building . . . 188
7.1.1 Motivations . . . 188
7.1.2 Requirement of COB Environments . . . 190
7.2 CVS-based Collaboration and its Limitations . . . 191
7.3 COB-Editor . . . 194
7.3.1 Organizing Ontologies into Packages . . . 194
7.3.2 Benefits of Modular Organization for COB . . . 196
7.3.3 The COB-Editor . . . 197
7.4 WikiOnt: Wiki-based Modular Ontology Editor . . . 199
7.4.1 Overview . . . 199
7.4.2 Features . . . 200
7.5 Related Work . . . 203
CHAPTER 8. Conclusion and Discussion . . . 206
8.1 Contributions and Impacts . . . 206
8.2 Limitations and Future Work . . . 210
8.2.1 Modular Ontology Study in General . . . 210
8.2.2 Extending P-DL . . . 211
8.2.3 Reasoning Algorithms for Modular Ontologies . . . 212
8.2.4 Privacy-Preserving Reasoning in Modular Ontologies . . . 213
8.2.5 Applications of Modular Ontologies . . . 214

CHAPTER Appendix: Proof of Lemmas and Theorems 215

A.1 Proofs for Chapter 4 216

A.2 Proofs for Chapter 5 242

CHAPTER Bibliography 256

CHAPTER Index 275

LIST OF TABLES

2.1 Syntax and Semantics of DLs 10
2.2 Negation Normal Form Transformation 15
2.3 OWL DL Class Constructors, Axioms and Facts 21

3.1 Semantic Connection Expressivity of Modular Ontology Languages . . . 41
3.2 Possible AMO Expressivity Features 51

4.1 Semantics of Modular Ontology Languages 99
4.2 Comparison of Expressivity 120
4.3 Comparison of Semantic Properties of Modular Ontology Languages . . 121
4.4 Comparison of Expressivity of Modular Ontology Languages 122

LIST OF FIGURES

2.1 An Example of Description Logic Knowledge Base 8

3.1 Organizational Structure vs. Semantic Structure 26
3.2 Total Reuse vs. Syntactical Partial Reuse 27
3.3 Reasoning Exactness 35
3.4 Directionality of Modular Ontologies 37
3.5 Knowledge Transitive Reusability 37
3.6 Domain Relation 45
3.7 Image Domain Relation 47
3.8 Concept Image 47
3.9 Role Image 48
3.10 Propagation of Knowledge Among Agents 52
3.11 Two Types of Inconsistencies between Agents 53
3.12 Agent Consensus 54
3.13 Reusability 55

4.1 One-to-One Domain Relation 71
4.2 Compositionally Consistent Domain Relation 72
4.3 Cardinality Preservation for Roles 73
4.4 Evolution of Modular Ontology Languages 97
4.5 Semantics of DDL Bridge Rules 108

5.1 $\mathcal{ALCP}^{-}_{\mathcal{C}}$ Distributed Tableaux Example 131
5.2 Subset Blocking Example in $\mathcal{ALCP}^{-}_{\mathcal{C}}$ 134
5.3 The Need for Token Blocking 135
5.4 Transitive Subsumption Propagation in $\mathcal{ALCP}^{-}_{\mathcal{C}}$ 139

5.5 Detect Inter-module Unsatisfiability in $\mathcal{ALCP}^{-}_{\mathcal{C}}$ 140

5.6 Reasoning from Local Point of View in $\mathcal{ALCP}^{-}_{\mathcal{C}}$ 142

5.7 Non-termination Caused by Cyclic Importing 144

5.8 Example of $\mathcal{SHIQP}$ Tableau Expansion 160

5.9 Completion Graph in the DDL Tableau Algorithm 162

5.10 Completion Graph in P-DL Tableau Algorithms 162

5.11 Completion Graph in $\mathcal{E}$-Connections Tableau Algorithms 163

6.1 Safety of Hierarchical Ontologies 177

7.1 Non-collaborative Ontology Building 188

7.2 Collaborative construction of an Animal Trait Ontology 190

7.3 Collaborative Ontology Building with CVS: Gene Ontology 192

7.4 COB Editor 198

7.5 Collaborative Ontology Building with Package-extended Ontology 199

7.6 The Architecture of WikiOnt 201

7.7 A Wiki Page in the WikiOnt System 202

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me in one way or another on conducting research and the writing of this dissertation.

I want to express my heartfelt thanks to my advisor, Vasant Honavar. The work presented in this dissertation is not possible without his support and advice.

I gratefully acknowledge the help from many members of the ISU Computer Science department and the Artificial Intelligence Research Laboratory, as well as from other institutions: several papers that the dissertation is based upon are helped by Giora Slutzki, George Voutsadakis and Doina Caragea; the work on collaborative ontology building is in collaboration with LaRon Hughes, Zhiliang Hu, Peter Wong and James Reecy; the work on ontology building in medical domain is in collaboration with Yu Cao and Wallapak Tavanapong; the work on semantic data integration is in collaboration with Doina Caragea, Jyotishman Pathak, Changhui Yan and Neeraj Koul. I also want to thank useful discussion and inspiration from Gary Leavens, Dae-ki Kang, Rafael-Armando Jordan, Adrian Silvescu, Kewei Tu, Jun Zhang, Jaime Reinoso-Castillo, Feihong Wu, Hua Pei and Hua Ming.

I also wish to thank many people in the semantic web community for useful discussions: Jeff Pan, Yimin Wang, Luciano Serafini, Andrei Tamilin, Zhengxiang Pan and Jing Mei.

I cannot finish without saying "thanks" to my wife and parents, who are always my strong support in all these years.

This Research is supported by assistantships offered by the Department of Computer Science funded through grants from the National Science Foundation (IIS 0219699,0639230) and National Institutes of Health (GM 066387), funding from Center for Integrated Animal Genomics (ISU), and grants from USDA NAGRP Bioinformatics Coordination Project.

ABSTRACT

The success of the world wide web can be attributed to the network effect: The absence of central control on content and organization of the web allows thousands of independent actors to contribute resources (web pages) that are interlinked to constitute the web. Recent efforts to extend the web into a semantic web are aimed at enriching the web with machine interpretable content and interoperable resources and services. Realizing the full potential of the semantic web requires the large-scale adoption and use of ontology based approaches to sharing of information and resources. In such a setting, instead of a single, centralized ontology, it is much more natural to have multiple distributed ontologies that cover different, perhaps partially overlapping, domains (e.g., biology, medicine, pharmacology). Such ontologies represent the local knowledge of the ontology designers, that is, knowledge that is applicable within a specific context. Hence, many application scenarios, such as collaborative construction and management of complex ontologies, distributed databases, and large knowledge base applications, present an urgent need for ontology languages that support localized and contextualized semantics, partial and selective reuse of ontology modules, flexible ways to limit the scope and visibility of knowledge (as needed for selective knowledge sharing), federated approaches to reasoning with distributed ontologies, and structured approaches to collaborative construction of large ontologies. Against this background, this dissertation develops a family of description logics based modular ontology languages, namely Package-based Description Logics (P-DL), to address the needs of such applications. The main contributions of this dissertation include:

- The identification and theoretical characterization of the desiderata of modular ontology languages that can support selective sharing and reuse of knowledge across independently developed knowledge bases;

- The development of a family of ontology languages called P-DL, which extend the classical description logics (DL) to support selective knowledge sharing through a novel semantic

importing mechanism and the establishment of a minimal set of restrictions on the use of imported concepts and roles to support localized semantics, transitive propagation of imported knowledge, and different interpretations from the point of view of different ontology modules;

- The development of a family of sound and complete tableau-based federated reasoning algorithms for distributed, autonomous, P-DL ontologies including $\mathcal{ALCP}$ and $\mathcal{SHIQP}$, i.e., P-DL onologies where the individual modules are expressed in the P-DL counterpart of DL $\mathcal{ALC}$ and $\mathcal{SHIQ}$ respectively, that can be used to efficiently reason over a set of distributed, autonomous, ontology modules from the point of view of any specific module, that avoid the need to integrate ontologies using message exchanges between modules as needed.

- The formulation of criteria for answering queries against a knowledge base using hidden or private knowledge, whenever it is feasible to do so without compromising hidden knowledge, and the development of privacy-preserving reasoning strategies for the case of the commonly used hierarchical ontologies and $\mathcal{SHIQ}$ ontologies, along with a theoretical characterization of the conditions under which they are guaranteed to be privacy-preserving.

- The development of some prototype tools for collaborative development of large ontologies, including support for concurrent editing and partial loading of ontologies into memory.

CHAPTER 1. Motivation And Overview

1.1 Motivation

Ontologies that explicitly identify objects, properties, and relationships of interest in specific domains of inquiry are essential for collaborations that involve sharing of data, knowledge, or resources (e.g., web services) among autonomous individuals or groups in open environments. Consequently, there has been a significant body of recent work on languages for specifying ontologies, software environments for editing ontologies, algorithms for reasoning with, aligning, and merging ontologies (Gomez-Perez et al., 2002). In particular, ontologies play important roles in realizing the *Semantic Web* vision (Berners-Lee et al., 2001): extending existing World-Wide Web with machine understandable meta data (knowledge) to facilitate more intelligent information search and data sharing.

The rapid growth and adoption of Web was possible in part because it allowed a large community of individuals around the world to contribute to its construction by linking autonomous pages via hyperlinks. We expect that effective mechanisms and tools that would enable individuals with expertise in specific areas to contribute ontology modules that can be conceptually linked into larger ontologies would significantly accelerate the realization of the semantic web vision.

Semantic Web ontologies have several important characteristics, as follows:

1. Constructing large ontologies typically requires collaboration among multiple individuals or groups with expertise in specific areas, with each participant contributing only a part of the ontology. Therefore, instead of a single, centralized ontology, in most domains, there are multiple distributed ontologies covering parts of the domain.

2. Because no single ontology can meet the needs of all users under every conceivable scenario, the ontology that meets the needs of a user or a group of users needs to be assembled

from several independently developed ontology modules. Since different ontologies or different modules of a single ontology are developed by people with diverse points of view, *contextual* inconsistencies or conflicts between such modules are inevitable. Consequently, there is a need for mechanisms for resolving or managing such contextual conflicts.

3. While ontologies are often used to facilitate sharing of knowledge, data, and resources, many real-world scenarios also call for selectively *hiding* certain parts of an ontology (or conversely, selectively sharing certain parts of an ontology). The need for knowledge hiding may arise due to privacy and security concerns, or for managing and knowledge engineering purposes.

Unfortunately, the current state of the art in ontology engineering is reminiscent of the state of software engineering nearly four decades ago: Today's ontology languages, like the very early programming languages, are largely unstructured, and offer little support for modular design of ontologies and selective knowledge sharing between ontology modules. As a consequence, many existing ontologies are difficult to reuse in a larger context, leading to an *ontology engineering bottleneck*, which is a significant hurdle in the large-scale design, development, and deployment of semantic web applications. We need to come to terms with the characteristics of web ontologies. Specifically, next generation ontology languages need to support modular structure, collaborative construction, selective sharing and use of ontologies. Against this background, this study introduces the framework of Package-based Description Logics to meet this need.

1.2 The Proposed Solution

Package-based Description Logics (P-DL) (Bao et al., 2006f,c, 2007e) is aimed at solving several problems presented in existing approaches for modular ontology formalisms and collaborative ontology building. P-DL language features are aimed at providing fine-grained modular organization and controllable, selective knowledge sharing of an ontology.

1.2.1 Outline of the Theoretical Approach

In a P-DL ontology, an ontology is composed of a set of modules, called *packages*. P-DL syntax adopts a *semantic importing* approach that allows a subset of symbols defined in one package to be directly used in other packages, and imported symbols can be used to construct

concepts. In particular, this work studies the P-DL language $\mathcal{SHOIQP}$ as well as several of its subsets. The resulting modular ontology languages:

- Allow each ontology module to use a subset of Description Logic (DL) $\mathcal{SHOIQ}$ (Horrocks and Sattler, 2005), i.e., $\mathcal{ALC}$ augmented with transitive roles, role inclusion, role inversion, qualified number restriction and nominal concepts, hence covers a significant fragment of OWL-DL.

- Provide strong modeling ability compared with existing approaches (e.g., Distributed Description Logics (DDL) (Borgida and Serafini, 2002) and $\mathcal{E}$-connections (Grau, 2005)), using the mechanism of semantic import of names (including concept, role and nominal names) across ontology modules.

- Contextualize the interpretation of reused knowledge, hence the contextual inconsistency between different ontology modules can be largely controlled. A natural consequence of contextualized interpretation is that inferences are always drawn *from the point of view* of a *witness* module. Thus, different modules might infer different consequences, based on the knowledge that they import from other modules.

- Ensure that the result of reasoning is always the same as that obtained from a standard reasoner over an integrated ontology resulting from combining the relevant knowledge in a context-specific manner. This ensures the *monotonicity* of inference in the distributed setting.

- Avoid many of the known reasoning difficulties of the existing approaches (e.g. knowledge transitive reusability).

The reasoning procedure for P-DL is extended from existing DL tableau algorithms (Bao et al., 2006a,e). We adopt a distributed tableau-based reasoning approach that can strictly avoid reasoning with an integrated ontology, thus ensure the autonomy of constituting modules (e.g., without direct exposing of sensitive information) and improve the scalability of the reasoning process. The whole reasoning process is preformed by a federation of local reasoners, each has access only to the local knowledge in a specific module, to construct a collection of local tableaux instead of a single global tableau by exchanging a set of messages between local reasoners.

P-DL allows selective knowledge hiding in ontology modules to address the needs of privacy, copyright, and security concerns in ontologies. In such cases, an agent might want to hide a part of its ontology while sharing the rest. However, prohibiting any use of the hidden part of the ontology in answering queries from other agents may be overly restrictive. P-DL supports *scope limitation modifiers* (SLM) that can be associated with terms and axioms defined in a package (Bao et al., 2006f). A SLM (such as *public* and *private*) controls the visibility of the corresponding term or axiom to entities (e.g. a user, a reasoner) on the web, in particular, to other packages. We show how an agent can safely answer queries against its ontology based on inferences drawn using both the hidden and visible part of its ontology, without inadvertently revealing the hidden knowledge. In particular, we investigated such privacy-preserving reasoning process in the case of the commonly used partial order ontologies (i.e. hierarchies) and the $\mathcal{SHIQ}$ family of description logics. We also discussed the design desiderata for distributed privacy-preserving algorithms of P-DL ontologies.

1.2.2 Applications

The work presented in this dissertation may have potential impacts in several application scenarios (among others), including the following:

- **Collaborative Ontology Building**. The building process of an ontology is usually a collaborative process that involves cooperation among multiple domain experts, and the generation, management and integration of multiple components of the ontology in a principled fashion. P-DL provides language features needed for efficient *collaborative* construction of large, modular ontologies. We have developed WikiOnt (Bao and Honavar, 2004a) and COB-Editor (Bao et al., 2006g) software prototypes that provide 'proof of concept' of this approach.

- **Partial Ontology Reuse**. Knowledge bases, and in particular ontologies, are very likely to be reused. However, the lack of modularity in current ontology languages forces an ontology to be totally reused or not be reused at all. An ontology has to be completely reused even if only a small fragment of it is actually needed. Modular ontologies will facilitate more flexible and efficient reuse of existing ontologies (Bao et al., 2006f).

- **Web Security**. Many semantic web applications, such as web services and online personal information repositories, require *selective* sharing of ontologies between autonomous entities. The privacy-preserving inference algorithms provided in this work offer practical solutions to *semantic* knowledge hiding such that hiding knowledge may still be used in "safe" query answering process without potential compromising such hiding knowledge. Our work provides the necessary privacy-preserving inference support on the top of *syntactic* knowledge hiding as offered by access control or encryption of web resources.

- **Semantic Data Integration**. Data integration problems may require explicit description of the data semantics for involved data sources. Instead of relying on a single global ontology, such applications typically require distributed, connected, multiple ontologies, each capturing a subset of the domain of discourse. Based on the inference infrastructure provided by the modular ontology framework, we investigated the data integration and query translation problem when data users and (possibly multiple) data sources have multiple ontologies associated with the data (Bao et al., 2007a).

Potential applications of the work presented in this dissertation are no necessarily limited to the ones listed above. Some other applications include:

- **Modular Web Ontology Language**. OWL (Patel-Schneider et al., 2003), the current web ontology language, has limited provision for handling modular ontologies. Specifically, the `owl:imports` construct for linking ontology modules lacks support for partial reuse of or localized semantics for the linked ontology modules. The proposed work provides an alternative to `owl:imports`, namely, *semantic importing*, for linking multiple modular ontologies with support for localized semantics, partial ontology reuse, and distributed reasoning.

- **Scalable Inference Support for Large and Distributed Ontologies**. The Semantic Web will very likely utilize the *networked effect* that has been proven useful for the success of the existing Web: there is no authoritarian central control of the huge amount of resources (including ontologies) on the Web; instead, those resources are usually autonomously created and maintained, and are interlinked to each other. Hence, inference support in such a setting is necessarily distributed to meet the scalability challenge, the

lack of global knowledge (from an integrated knowledge base), and the context-specific nature of involved ontologies. The P-DL framework may provide useful tools and methods for such a need.

- **Knowledge Representation and Query Process in Peer-to-Peer Applications**. Peer-to-Peer (P2P) applications demand selective data and knowledge sharing among a set of autonomous peers. The modular ontology approach we presented here may be adopted as the logic foundation for P2P applications to support distributed querying of P2P resources and preserving of peer privacies.

- **Ontology Mapping**. The proposed techniques permits mappings between ontology modules to be modeled as axioms in those modules or in independent mapping modules. In particular, the general module transitive reusability in P-DL allows ontology mappings to be safely composed to form new mappings.

- **Ontology Evolution**. The methods we presented for partial ontology reuse may also be applied in ontology evolution, such that an existing ontology module can be "patched" by a new module, and axioms in the existing module are selectively reused, replaced or obsoleted in the new version of the ontology.

1.3 Content Guide

The dissertation contains the following chapters (in additional to this introduction chapter):

- Chapter 2 introduces the basic knowledge about Description Logics (DL) and web ontology languages. We focus on the basic DL $\mathcal{ALC}$ and the $\mathcal{SH}$ family of DL languages on their syntax, semantics and tableau-based reasoning algorithms. We also introduce the OWL web ontology language.

- Chapter 3 presents desiderata and the general framework for modular ontologies. We first enumerate the needs for modularity in ontologies from the aspects of ontology organizational structure, decentralized nature of web ontologies and contextuality of ontologies. A set of desiderata for modular ontology languages, including decidability, reasoning exactness, knowledge transitive reusability, and directional semantic connection, are introduced. Then, we present an abstract framework of modular ontologies based on local

model semantics. It serves as the foundation for the P-DL proposal and the comparison between P-DL and other formalisms.

- Chapter 4 presents the syntactical and semantic features of P-DL. We also give a brief review on the evolution of modular ontology formalism and compare P-DL with several other formalisms.

- Chapter 5 discusses reasoning algorithms for P-DL. We investigate sound and complete tableau-based reasoning algorithms for several P-DL languages, including: $\mathcal{ALCP}_{\mathcal{C}}$, which extends $\mathcal{ALC}$ with concept name importing between packages, and $\mathcal{SHIQP}$, which extends DL $\mathcal{SHIQ}$ with concept or role name importing between packages. These algorithms allow the reasoning process to be distributed relying on local reasoning services offered by each ontology module.

- Chapter 6 examines privacy-preserving reasoning in ontologies and, in particular, in P-DL modular ontologies. We precisely formulate the problem of "privacy-preserving inference". We exploit the indistinguishability of hidden knowledge and incomplete knowledge under the Open World Assumption (OWA) to develop an approach to *safe* use of hidden knowledge in query answering without the risk of unintended disclosure of hidden knowledge. We offer "history-safe" reasoning strategies for commonly used hierarchical ontologies and the $\mathcal{SHIQ}$ family of description logic ontologies.

- Chapter 7 is concerned with collaborative ontology building exploiting the notion of modular ontologies (or ontology packages). We describe two ontology editing tools, WikiOnt and COB Editor, which can support sharing, reuse, and collaborative editing of ontologies. Both tools allow ontology developers to create a community-shared ontology server, with the support for concurrent browsing and editing of the ontology. Multiple users can work on the same ontology on different packages (through locking mechanisms), without inadvertent overwriting the work of others.

- Chapter 8 summarizes the work on its contributions and limitations, and gives several open problems as the future work.

The dissertation also contains an appendix which gives detailed proofs of some original lemmas and theorems given in the dissertation.

CHAPTER 2. Preliminaries

This chapter introduces the basic notions of description logics (DL) on its syntax, semantics and reasoning algorithms, as well as the DL-based web ontology language OWL. Readers that are familiar with DL may skip this chapter. For more details about DL, please refer the *Description Logics Handbook* (Baader et al., 2003).

2.1 Description Logics

2.1.1 Basic Notions

Description Logics (DL) (Baader and Nutt, 2003) is a family of knowledge representation languages which defines concepts of a domain, and then use these concepts to specify properties of objects and individuals occurring in the domain. The basic syntactic blocks in DL are atomic concepts (unary predicates), atomic roles (binary predicates), and individuals (constants). A DL provides a set of constructors which allows to form complex concepts and roles from atomic concepts and roles. For example, a simple ontology about animals is represented in DL as in Figure 2.1:

$$\mathsf{Dog} \sqsubseteq \mathsf{Carnivore} \tag{2.1}$$
$$\mathsf{Carnivore} \sqsubseteq \mathsf{Animal} \sqcap \forall \mathsf{eats}.\mathsf{Animal} \tag{2.2}$$
$$\mathsf{Dog}(\mathsf{goofy}) \tag{2.3}$$
$$\mathsf{eats}(\mathsf{goofy}, \mathsf{foo}) \tag{2.4}$$

Figure 2.1 An Example of Description Logic Knowledge Base

In the example, $\mathsf{Dog}, \mathsf{Carnivore}, \mathsf{Animal}$ are concept names, eats is a role name, and $\mathsf{goofy}, \mathsf{foo}$ are individual names; $\sqcap$ (intersection) and $\forall$ (value restriction) are constructors. The ontology asserts that a Dog is a $\mathsf{Carnivore}$ (2.1); a $\mathsf{Carnivore}$ is an Animal that only eats Animal (2.2); goofy is a Dog (2.3); and that individual goofy eats another individual foo (2.4).

A description logic knowledge base may consist of a *TBox* (terminology box) and an *ABox* (assertion box), where the TBox is a finite set of general concept inclusion (GCI) axioms in the form of $C \sqsubseteq D$ (read as D subsumes C), and the ABox is a finite set of assertions in the form of $C(a)$ (a concept assertion) or $R(a, b)$ (a role assertion). In the ontology given above, the TBox contains axiom (2.1)-(2.2), and the ABox contains axiom (2.3)-(2.4). We may use $C \equiv D$ as the abbreviation of both $C \sqsubseteq D$ and $D \sqsubseteq C$.

Some DLs (e.g., the $\mathcal{SH}$ family introduced in section 2.1.3) also have a RBox (role box) that contains role inclusion axioms of the form $R_1 \sqsubseteq R_2$ where R_1, R_2 are roles.

The precise meaning of a DL language can be defined in the model-theoretical semantics:

Definition 2.1 (DL Semantics) *An **interpretation** of a description logic knowledge base is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, (.)^{\mathcal{I}} \rangle$, where the interpretation domain $\Delta^{\mathcal{I}}$ contains a nonempty set of objects and the interpretation function $(.)^{\mathcal{I}}$ maps each concept name C to a subset of the domain $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role name P to a binary relation $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ over the domain $\Delta^{\mathcal{I}}$, and each individual name i to an element in the domain $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.*

*An interpretation $\mathcal{I}$ **satisfies** a GCI $C \sqsubseteq D$, denoted by $\mathcal{I} \models C \sqsubseteq D$, iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. $\mathcal{I}$ is a **model** of a TBox $\mathcal{T}$, denoted by $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I}$ satisfies all GCIs in $\mathcal{T}$.*

An interpretation $\mathcal{I}$ satisfies the concept assertion $C(a)$, denoted by $\mathcal{I} \models C(a)$, iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and it satisfies the role assertion $R(a, b)$, denoted by $\mathcal{I} \models R(a, b)$, iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it satisfies all the concept and role assertions in $\mathcal{A}$.

An interpretation $\mathcal{I}$ satisfies a role inclusion axiom $R_1 \sqsubseteq R_2$ iff $R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$. $\mathcal{I}$ is a model of a RBox $\mathcal{R}$, denoted by $\mathcal{I} \models \mathcal{R}$, if $\mathcal{I}$ satisfies all role inclusions in $\mathcal{R}$.

An interpretation $\mathcal{I}$ is a model of a knowledge base $\mathcal{K} = \{\mathcal{T}, \mathcal{R}, \mathcal{A}\}$, denoted by $\mathcal{I} \models \mathcal{K}$, where $\mathcal{T}$ is the TBox, $\mathcal{R}$ is the RBox, $\mathcal{A}$ is the ABox, iff $\mathcal{I}$ is model of $\mathcal{T}$,$\mathcal{R}$, and $\mathcal{A}$.

*An knowledge base $\mathcal{K}$ **entails** an axiom γ, denoted by $\mathcal{K} \models \gamma$, if every model $\mathcal{I}$ of K satisfies γ.*

For example, a model of the ontology in Figure 2.1 is $\mathcal{I} = \langle \Delta^{\mathcal{I}}, (.)^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}} = \{a, b\}$, $(.)^{\mathcal{I}}$ maps concept "Dog" to $\mathsf{Dog}^{\mathcal{I}} = \{a\} \subseteq \Delta^{\mathcal{I}}$, "Carnivore" to $\mathsf{Carnivore}^{\mathcal{I}} = \{a\} \subseteq \Delta^{\mathcal{I}}$, "Animal" to $\mathsf{Animal}^{\mathcal{I}} = \{a, b\} \subseteq \Delta^{\mathcal{I}}$, role "eats" to $\mathsf{eats}^{\mathcal{I}} = \{(a, b)\} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, individual "goofy" to $\mathsf{goofy}^{\mathcal{I}} = a \in \Delta^{\mathcal{I}}$, and individual "foo" to $\mathsf{foo}^{\mathcal{I}} = b \in \Delta^{\mathcal{I}}$.

2.1.2 The Basic Description Logics $\mathcal{ALC}$

One of the most influential DLs, which is the foundation of many other DLs, is $\mathcal{ALC}$ (**A**ttributive **L**anguage with **C**omplements) (Schmidt-Schauß and Smolka, 1991). $\mathcal{ALC}$ provides boolean concept constructors ($\neg$: negation or complement, $\sqcap$: intersection, and $\sqcup$: union) plus the existential restriction (in the form of $\exists R.C$) and universal restriction (in the form of $\forall R.C$, also called value restriction) constructors. We also use the notation $\top$ (universal top concept) as the abbreviation for $A \sqcup \neg A$, and $\bot$ (bottom concept) as the abbreviation for $A \sqcap \neg A$, where A is any concept name.

The semantics of $\mathcal{ALC}$ constructors is given in the Table 2.1.

	Constructor	**Syntax**	**Semantics**
$\mathcal{ALC}$	top	$\top$	$\Delta^{\mathcal{I}}$
	bottom	$\bot$	$\varnothing$
	atomic concept	C	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
	abstract role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
	intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
	union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
	value restriction	$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b, (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
	existential quantification	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b, (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
	negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
$\cdot_{R^+}$	transitive role	R_+	$(a,b) \in R^{\mathcal{I}} \wedge (b,c) \in R^{\mathcal{I}} \rightarrow (a,c) \in R^{\mathcal{I}}$
$\mathcal{S}$	$= \mathcal{ALC}_{R^+}$		
$\mathcal{H}$	role hierarchies	$R \sqsubseteq S$	$(a,b) \in R^{\mathcal{I}} \rightarrow (a,b) \in S^{\mathcal{I}}$
$\mathcal{I}$	inverse role	R^-	$\{(b,a) \in (R^-)^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}$
$\mathcal{Q}$	qualified number restriction	$\bowtie {}^a n R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \bowtie n\}$
$\mathcal{N}^b$	number restriction	$\bowtie nR$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \bowtie n\}$
$\mathcal{F}^c$	functional roles	$\leqslant 1R$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \leqslant 1\}$
		$> 2R$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} > 2\}$
$\mathcal{O}$	nominal	o	$\#\{o^{\mathcal{I}}\} = 1$
	concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

[a] $\bowtie$ is one of $=, \geqslant, \text{or} \leqslant$
[b] can be seen as a limited case of $\mathcal{Q}$, ie. with $C = \top$
[c] can be seen as a limited case of $\mathcal{N}$, ie. with $n = 1$

Table 2.1 Syntax and Semantics of DLs

For example, the animal ontology in Figure 2.1 can be extended to the following $\mathcal{ALC}$ TBox:

$$\begin{aligned} \mathsf{Dog} &\sqsubseteq \mathsf{Carnivore} \\ \mathsf{Carnivore} &\sqsubseteq \mathsf{Animal} \sqcap \forall \mathsf{eats}.\mathsf{Animal} \\ \mathsf{Dog} &\sqsubseteq \exists \mathsf{eats}.\mathsf{Rabbit} \qquad (2.5) \\ \mathsf{Plant} &\sqsubseteq \neg \mathsf{Animal} \qquad (2.6) \\ \mathsf{Plant} \sqcup \mathsf{Animal} &\sqsubseteq \mathsf{Life} \qquad (2.7) \end{aligned}$$

which further states that a Dog eats some Rabbit (2.5), Plants are not Animals (2.6), and both Plants and Animals are types of Life (2.7).

2.1.3 Expressive Description Logics - the $\mathcal{SH}$ family

There are several extensions (with additional sets of concept and role constructors) on the top of $\mathcal{ALC}$ to meet the needs of applications that require more expressivity. One of the most important extensions is the $\mathcal{SH}$ DL family, which is closely related to several web ontology languages, such as DAML+OIL (Horrocks, 2002) and OWL (Schreiber and Dean, 2004).

The logic $\mathcal{SH}$ (Horrocks and Sattler, 1999) is extended from $\mathcal{ALC}$ with transitive roles (denoted as $_{R+}$) and role inclusions ($\mathcal{H}$)[1]. We denote by $\mathsf{Trans}(r)$=true if a role r is transitive. For example, in an ontology about people that in the logic of $\mathcal{SH}$, we may have transitive role hasSibling, role hasBrother and a role inclusion axiom:

$$\mathsf{hasBrother} \sqsubseteq \mathsf{hasSibling}$$

Hence, if an ABox of the ontology contains assertions $\mathsf{hasBrother}(a, b)$ and $\mathsf{hasSibling}(b, c)$, then we must have that $\mathsf{hasSibling}(a, b)$ (from the role inclusion) and $\mathsf{hasSibling}(a, c)$ (from the transitivity of hasSibling) are true.

The logic $\mathcal{SHOIQ}$ (Horrocks and Sattler, 2005), which serves as the logic foundation of the current web ontology language OWL (Horrocks et al., 2003), is further extended from $\mathcal{SH}$ with nominals ($\mathcal{O}$), inverse roles ($\mathcal{I}$), and qualified number restrictions ($\mathcal{Q}$).

More precisely, a *nominal* is a concept that has a singleton interpretation, i.e., there is one and only one individual in the interpretation of a nominal. For example, the country name FRANCE can be modeled as a nominal, such that the cardinality $\#(\mathsf{FRANCE}^{\mathcal{I}})$ will always be

[1] The logic $\mathcal{ALC}_{R+}$ is commonly denoted by the mnemonic $\mathcal{S}$ for its correspondence with the modal logic S4.

1 in any interpretation $\mathcal{I}$. We may use nominals to define a concept with explicitly enumerated members, such as

$$\mathsf{WeekEnd} \equiv \{\mathsf{SATURDAY}, \mathsf{SUNDAY}\}$$

where SATURDAY and SUNDAY are nominals.

Inverse roles allow us to use a role in "both directions". For example, suppose we have a role advises to denote the relation between a faculty member and a student; we wish to define a concept of students that are advised by a faculty member, then we can describe it with an inverse role as

$$\mathsf{Student} \sqcap \exists \mathsf{advises}^{-}.\mathsf{Faculty}$$

We may use $\mathsf{Inv}(R) = R^{-}$ as the inverse of a role R, and $R^{--} = R$. In the presence of role inversions, $\mathsf{Trans}(R)$=true iff R is transitive or $\mathsf{Inv}(R)$ is transitive. A role is called *symmetric* if it is equivalent to its own inverse, i.e., $R \equiv R^{-}$. For example, hasSibling is a symmetric role.

Qualified number restriction ($\mathcal{Q}$) is in the form of $\geq nR.C$, $\leq nR.C$ or $= nR.C$, where R is a simple role such that it is neither transitive nor has transitive sub-roles[2]. For example, to assert that every people must have exactly two parents who are also people, a woman is a mother if she has at least one child which is a people, and one people is married to at most one people, we have the following axioms:

$$\mathsf{People} \sqsubseteq (= 2\ \mathsf{hasParent.People}) \tag{2.8}$$

$$\mathsf{Mother} \sqsubseteq \mathsf{Woman} \sqcap (\geq 1\ \mathsf{hasChild.People}) \tag{2.9}$$

$$\mathsf{People} \sqsubseteq \leq 1\ \mathsf{marries.People} \tag{2.10}$$

Unqualified number restriction (denoted by $\mathcal{N}$) is a special case of $\mathcal{Q}$ such that the qualification concept C is always the top concept $\top$, therefore we only have restrictions in the form of $\geq nR$, $\leq nR$ and $= nR$. For example, it may not be necessary to specify that a child is a people to define motherhood, hence, axiom (2.9) can be reformulated as

$$\mathsf{Mother} \sqsubseteq \mathsf{Woman} \sqcap (\geq 1\ \mathsf{hasChild})$$

[2]Requiring roles to be simple here is necessary since it is known that qualified number restriction on non-simple role may lead to undecidability in $\mathcal{SHIQ}$ (Horrocks et al., 1999). Such a restriction may be relaxed to so called "admissible" RBoxes in $\mathcal{SHQ}$ (Kazakov et al., 2007).

Functional roles (denoted by $\mathcal{F}$) is further restricted from $\mathcal{N}$ such that $n = 1$. Therefore, concepts like $(= 2\ \mathsf{hasParent.People})$ will not be allowed in DLs with $\mathcal{F}$ constructors. In particular, $\mathcal{SHIF}$ (Horrocks et al., 2000) is important because it, when extended with data types, is the DL language corresponding to OWL Lite.

Several other commonly used DLs of the $\mathcal{SH}$ family include the following:

- $\mathcal{SHIQ}$ ($\mathcal{ALCHIQ}_{R+}$) (Horrocks et al., 1999) is obtained from $\mathcal{SHOIQ}$ by disallowing the use of nominals;
- $\mathcal{SHOQ}$ ($\mathcal{ALCHOQ}_{R+}$)(Horrocks and Sattler, 2001) is obtained from $\mathcal{SHOIQ}$ by disallowing the use of inverse roles;
- $\mathcal{SHIO}$ ($\mathcal{ALCHIO}_{R+}$) (Hladik, 2004) is obtained from $\mathcal{SHOIQ}$ by disallowing the use of (qualified) number restrictions.

The syntax and semantic of the $\mathcal{SH}$ family DLs are summarized in the Table 2.1.

2.2 Reasoning with Description Logics

2.2.1 Reasoning Tasks for Description Logics

DLs provide a good trade-off between the expressivity power and computational complexity. Many inference tasks with DLs can be solved efficiently with highly optimized DL reasoners, such as FaCT++ (Tsarkov and Horrocks, 2004), RACER (Haarslev and Möller, 2001) and Pellet (Sirin et al., 2007).

Typical reasoning tasks in DL include the follows:

- Subsumption: to test if a concept C is subsumed by another concept D, i.e., if $C \sqsubseteq D$;
- Satisfiability: to test if a concept C is satisfiable, i.e., if there is an individual in an interpretation such that it is an instance of C;
- Equivalency: to test if two concepts are equivalent, i.e., $C \equiv D$;
- Disjointness: to test if two concepts must have no shared instances;
- Membership: to test if an individual a is an instance of a concept C, i.e., $C(a)$;

Many reasoning problems can be reduced to other reasoning problems. For example, some of them can be reduced to subsumption:

- C and D are equivalent $\Leftrightarrow C \sqsubseteq D$ and $D \sqsubseteq C$
- C and D are disjoint $\Leftrightarrow C \sqcap D \sqsubseteq \bot$.
- a is a member of C $\Leftrightarrow \{a\} \sqsubseteq C$

Subsumption can also be reduced to satisfiability:

- $C \sqsubseteq D \Leftrightarrow C \sqcap \neg D$ is unsatisfiable

It is true because if $C \sqcap \neg D$ is satisfiable, there must an interpretation $\mathcal{I}$ and an element $x \in C^{\mathcal{I}}$ but $x \notin D^{\mathcal{I}}$ therefore $C \sqsubseteq D$ cannot hold. If no such an interpretation can be found, $C \sqcap \neg D$ is unsatisfiable, hence $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in any $\mathcal{I}$, therefore $C \sqsubseteq D$ is true.

Hence, all reasoning problems mentioned above can be in effect reduced to concept satisfiability checking. In the following discussion, we will focus on concept satisfiability checking only.

2.2.2 Basic Tableau Algorithms

Modern DLs exploit Tableau Algorithms (Baader and Sattler, 2001; Hollunder et al., 1990) for practical reasoning support. The basic idea of tableau algorithm is to check concept satisfiability (and hence also for concept subsumption) w.r.t. a knowledge base by constructing a common model of the concept and the knowledge base.

A tableau algorithm for a specific DL language contains the following main elements:

- A *completion graph*, or a *tableau* that represents a model of the DL language. Such a completion graph typically has the "tree model" property (Vardi, 1996).
- A set of *tableau expansion rules* to construct a complete and consistent completion graph.
- A set of *blocking rules* to detect infinite cyclic models and ensure termination.
- A set of *clash conditions* to detect logic contradictions.

In this subsection, we will demonstrate the basic process of tableau algorithms with the DL $\mathcal{ALC}$. For an $\mathcal{ALC}$ TBox $\mathcal{T}$ and an $\mathcal{ALC}$ -concept C_0, a tableau algorithm will construct a common model for both O and C_0 to checking the satisfiability of C_0 w.r.t. $\mathcal{T}$. If one such model (i.e., a completion graph) is found, C_0 is satisfiable, otherwise C is unsatisfiable.

Before the reasoning process starts, the concepts in $\mathcal{T}$ and C_0 should be transformed into the *Negation Normal Form* (NNF), i.e., with negation only occurs in front of atomic concepts. It can be done with rewriting rules in Table 2.2. We use $\doteq C$ to denote the NNF of $\neg C$.

$$\begin{aligned}
\neg\neg C &\equiv C \\
\neg(C \sqcap D) &\equiv \neg C \sqcup \neg D \\
\neg(C \sqcup D) &\equiv \neg C \sqcap \neg D \\
\neg\exists R.C &\equiv \forall R.\neg C \\
\neg\forall R.C &\equiv \exists R.\neg C \\
\neg \leqslant nR.C &\equiv \geqslant (n+1)R.C \\
\neg \geqslant (n+1)R.C &\equiv \leqslant nR.C \\
\neg \geqslant 0R.C &\equiv C \sqcap \neg C
\end{aligned}$$

Table 2.2 Negation Normal Form Transformation

Reasoning w.r.t. a TBox $\mathcal{T}$ can be reduced to reasoning w.r.t. an empty TBox with the *internalization* technique. Given $\mathcal{T}$, a concept $C_{\mathcal{T}}$ is defined as $C_{\mathcal{T}} = \prod_{(C_i \sqsubseteq D_i) \in \mathcal{T}} (\neg C_i \sqcup D_i)$. Any individual x in any model of $\mathcal{T}$ will be an instance of $C_{\mathcal{T}}$.

For an $\mathcal{ALC}$ knowledge base, a completion graph or a **tableau** $T = \langle V, E, \mathcal{L} \rangle$ is a tree, where V is the node set, E is the edge set, $\mathcal{L}$ is a function that assigns labels for each node and edge. Each node x in the tree represents an individual in the domain of the model, and the label $\mathcal{L}(x)$ contains all concepts of which x is an instance. Each edge $\langle x, y \rangle$ represents a set of role instances in the model, and the label $\mathcal{L}(\langle x, y \rangle)$ contains the names of those roles. If $R \in \mathcal{L}(\langle x, y \rangle)$, y is an *R-successor* of x. In an $\mathcal{ALC}$ -tableau:

- if $C \in \mathcal{L}(x)$, then $\neg C \notin \mathcal{L}(x)$,
- if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, then $C_1 \in \mathcal{L}(x)$ and $C_2 \in \mathcal{L}(x)$,
- if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, then $C_1 \in \mathcal{L}(x)$ or $C_2 \in \mathcal{L}(x)$,
- if $\forall R.C \in \mathcal{L}(x)$ and $R \in \mathcal{L}(\langle x, y \rangle)$, then $C \in \mathcal{L}(y)$,

- if $\exists R.C \in \mathcal{L}(x)$, then there is some y such that $R \in \mathcal{L}(\langle x, y \rangle)$ and $C \in \mathcal{L}(y)$.

Given a concept C and a TBox $\mathcal{T}$, the tableau is a tree expanded from an initial root node x_0, $\mathcal{L}(x_0) = C \sqcap C_{\mathcal{T}}$, with the following **expansion rules**:

- $\sqcap$-rule: if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not blocked, $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$, then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$;
- $\sqcap$-rule: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not blocked, $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$, then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1\}$ or $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_2\}$;
- $\exists$-rule: if $\exists R.C \in \mathcal{L}(x)$, x is not blocked, and x has no R-successor y with $C \in \mathcal{L}(y)$, then create a new node y with $\mathcal{L}(\langle x, y \rangle) = \{R\}$ and $\mathcal{L}(y) = \{C\}$;
- $\forall$-rule: if $\forall R.C \in \mathcal{L}(x)$, x is not blocked, and there is an R-successor y of x with $C \notin \mathcal{L}(y)$, then $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$;
- CE-rule: if $C_{\mathcal{T}} \notin \mathcal{L}(x)$, x is not blocked, then $\mathcal{L}(x) = \mathcal{L}(x) \cup C_{\mathcal{T}}$.

To ensure termination, a node can be blocked with the **subset blocking** strategy: for any node x, if there is an ancestor node y of x in the tree, and $\mathcal{L}(x) \subseteq \mathcal{L}(y)$, then x is blocked. No expansion rule will be applied to a blocked node. In fact, a blocked node prevents the cyclic application of tableau expansion rules, hence represents infinitely many similar individuals in the model.

An $\mathcal{ALC}$ tableau contains a **clash** if there is $\{C, \neg C\} \in \mathcal{L}(x)$ for some node x and concept C. A tableau is **consistent** (clash-free) if it contains no clash, and is **complete** if no expansion rule can be applied. The given concept is satisfiable if and only if the algorithm finds a consistent and complete tableau.

Note that the $\sqcup$-rule is *non-deterministic* in that it generates different possible tableaux. The algorithm needs to try multiple choices, i.e., *search* for different possible models. Once a chosen search path leads to a clash, the algorithm needs to track back to the tableau state before the choice, and try other remaining choices.

A tableau algorithm has to meet three requirements:

- Soundness: if a complete and consistent tableau is found by the algorithm, the tableau must satisfies the initial concept C_0.

- Completeness: if the initial concept C_0 is satisfiable, the algorithm can always find an complete and consistent tableau for it.

- Termination: the algorithm can terminate in finite steps with a result.

It can be proven that the aforementioned algorithm is terminating, sound and complete for $\mathcal{ALC}$ (Baader and Sattler, 2001).

2.2.3 Tableau Algorithms for Expressive DLs

Similar tableau algorithms can be designed for more expressive DL languages. In this subsection, we will briefly introduce such an algorithm for $\mathcal{SHOIQ}$ provided by (Horrocks and Sattler, 2005).

A $\mathcal{SHOIQ}$ completion graph is $T = \langle V, E, \mathcal{L}, \neq \rangle$. The symmetric binary relation $\neq$ is used to keep track of inequalities between nodes of T. The introduction of nominal concepts (the "$\mathcal{O}$" constructor) somehow relaxes the strict tree structure (Horrocks and Sattler, 2005) which is enjoyed by $\mathcal{ALC}$ -tableau: a $\mathcal{SHOIQ}$ completion graph contains two types of nodes, i.e., the *blockable nodes* which still form tree structures, and *nominal nodes* which may be arbitrarily interconnected. A nominal node is a node that has nominal names in its labels; such a node cannot be blocked since a blocked node represents infinitely many individuals while a nominal is only allowed to have singleton instances.

If $R \in \mathcal{L}(\langle x, y \rangle)$, y is said an *R-successor* of x and x is an *R-predecessor* of y. *Ancestor* is the transitive closure of predecessor, and *descendant* is the transitive closure of successor. A node y is called an *R-neighbor* of a node x if y is an R-successor of x or if x is an $\mathsf{Inv}(R)$-successor of y.

A node x is **directly blocked** iff none if its ancestors is blocked, and it has ancestors x', y and y' such that

1) x is a successor of x' and y is a successor of y',

2) y, x and all nodes on the path from y to x are blockable,

3) $\mathcal{L}(x) = \mathcal{L}(y)$ and $\mathcal{L}(x') = \mathcal{L}(y')$, and

4) $\mathcal{L}(\langle x', x \rangle) = \mathcal{L}(\langle y', y \rangle) \neq \emptyset$

A node y is **indirectly blocked** iff one of its safe ancestor is blocked. A node is **blocked** if either it is directly blocked or it is indirectly blocked.

An R-neighbor y of x is **safe** if x is blockable or if x is a nominal node and y is not blocked. It is safe in the sense enough R-neighbors for nominal nodes can be generated (Horrocks and Sattler, 2007).

We define $S^T(x,C) := \{y \in V | S \in \mathcal{L}(\langle x,y\rangle) \wedge C \in \mathcal{L}(y)\}$ as the set of S-successor of x with C in their labels. For an RBox $\mathcal{R}$, we denote $\sqsubseteq_{\mathcal{R}}$ as is the transitive-reflexive closure over $\mathcal{R} \cup \{\mathsf{Inv}(R) \sqsubseteq \mathsf{Inv}(S) | R \sqsubseteq S \in \mathcal{R}\}$.

The set of $\mathcal{SHOIQ}$ tableau expansion rules is given as the follows (Horrocks and Sattler, 2007) (the notions of Merge operation will be introduced later):

- $\sqcap$-rule: **if** $C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$, **then** $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$;
- $\sqcup$-rule: **if** $C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$, **then** $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$;
- $\exists$-rule: **if** $\exists S.C \in \mathcal{L}(x)$, x is not blocked, and x has no safe S-neighbor y of x with $C \in \mathcal{L}(y)$, **then** create a new node y with $\mathcal{L}(\langle x,y\rangle) = \{S\}$ and $\mathcal{L}(y) = \{C\}$;
- $\forall$-rule: **if** $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and there is an S-neighbor y of x with $C \notin \mathcal{L}(y)$, **then** $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$;
- $\forall_+$-rule: **if** $\forall S.C \in \mathcal{L}(x)$, x is not indirectly blocked, and there is some R with $\mathsf{Trans}(R)$, $R \sqsubseteq S$ and there is an R-neighbor y of x with $\forall R.C \notin \mathcal{L}(y)$, **then** $\mathcal{L}(y) = \mathcal{L}(y) \cup \{\forall R.C\}$;
- *choose*-rule: **if** $(\leq nS.C) \in \mathcal{L}(x)$, x is not indirectly blocked, and there is an S-neighbor y of x with $\{C, \dot{\neg}C\} \cap \mathcal{L}(y) = \emptyset$, **then** $\mathcal{L}(y) = \mathcal{L}(y) \cup \{E\}$ for some $E \in \{C, \dot{\neg}C\}$;
- $\geq$-rule: **if** $(\geq nS.C) \in \mathcal{L}(x)$, x is not blocked, and there are no n safe S-neighbors $y_1, ...y_n$ of x with $C \in \mathcal{L}(y_k)$ and $y_k \neq y_j$ for each $1 \leq k \leq j \leq n$, **then** create n new nodes $y_1, ...y_n$ with $\mathcal{L}(\langle x,y_k\rangle) = \{S\}$ and $\mathcal{L}(y_k) = \{C\}$ and $y_k \neq y_j$ for $1 \leq k \leq j \leq n$;
- $\leq$-rule: **if** $(\leq nS.C) \in \mathcal{L}(z)$, z is not indirectly blocked, $|S^T(x,C)| \geq n$ and there are two S-neighbors x, y of z with $C \in \mathcal{L}(x) \cap \mathcal{L}(y)$ and not $x \neq y$, **then**
  1. if x is a nominal node, then $\mathsf{Merge}(y,x)$,
  2. else if y is a nominal node or an ancestor of x, then $\mathsf{Merge}(x,y)$,
  3. else $\mathsf{Merge}(y,x)$;
- o-rule: **if** for some nominal o there are two nodes x, y with $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$ and not $x \neq y$, **then** $\mathsf{Merge}(x,y)$;
- NN-rule: **if** (1) $(\leq nS.C) \in \mathcal{L}(x)$, x is a nominal node, and there is a blockable S-neighbor

y of x such that $C \in \mathcal{L}(y)$ and x is a successor of y; (2) there is no m such that $1 \leq m \leq n$, $(\leq mS.C) \in \mathcal{L}(x)$ and there exists m nominal S-neighbors $z_1, ..., z_m$ of x with $C \in \mathcal{L}(z_k)$ and $z_k \neq z_j$ for $0 \leq k \leq j \leq m$, **then** (1) guess m with $1 \leq m \leq n$, set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\leq mS.C\}$; (2) create m new nodes $y_1, ...y_m$ with $\mathcal{L}(\langle x, y_k \rangle) = \{S\}$ and $\mathcal{L}(y_k) = \{C, o_k\}$ for some new nominal o_k and $y_k \neq y_j$ for $0 \leq k \leq j \leq n$;

- CE-rule: if $C_{\mathcal{T}} \notin \mathcal{L}(x)$, x is not indirectly blocked, then $\mathcal{L}(x) = \mathcal{L}(x) \cup C_{\mathcal{T}}$.

The Merge operation is used in "shrinking" rules ($\leq$- and o-rule) to merge one node into another node. More precisely, it contains the following operations (Horrocks and Sattler, 2007):

Pruning: The operation $\mathsf{Prune}(y)$ removes a node y and all blockable successors of y recursively. Formally, it performs the following operations:

1. for all successors z of y, remove $\langle y, z \rangle$ from E and, if z is blockable, $\mathsf{Prune}(z)$;
2. remove y from V.

Merging: Intuitively, the operation $\mathsf{Merge}(y, x)$ merges y into x by letting x inherits all predecessors and nominal successors of y, while prune y and its blockable sub-trees. More precisely, it has the following steps:

1. for all nodes z such that $\langle z, y \rangle \in E$
   (a) if $\{\langle x, z \rangle, \langle z, x \rangle\} \cap E = \emptyset$;, then add $\langle z, x \rangle$ to E and set $\mathcal{L}(\langle z, x \rangle) = \mathcal{L}(\langle z, y \rangle)$,
   (b) if $\langle z, x \rangle \in E$, then set $\mathcal{L}(\langle z, x \rangle) = \mathcal{L}(\langle z, x \rangle) \cup \mathcal{L}(\langle z, y \rangle)$,
   (c) if $\langle x, z \rangle \in E$, then set $\mathcal{L}(\langle x, z \rangle) = \mathcal{L}(\langle x, z \rangle) \cup \{\mathsf{Inv}(S) | S \in \mathcal{L}(\langle z, y \rangle)\}$, and
   (d) remove $\langle z, y \rangle$ from E;
2. for all nominal nodes z such that $\langle y, z \rangle \in E$
   (a) if $\{\langle x, z \rangle, \langle z, x \rangle\}$ $E = \emptyset$;, then add $\langle x, z \rangle$ to E and set $\mathcal{L}(\langle x, z \rangle) = \mathcal{L}(\langle y, z \rangle)$,
   (b) if $\langle x, z \rangle \in E$, then set $\mathcal{L}(\langle x, z \rangle) = \mathcal{L}(\langle x, z \rangle) \cup \mathcal{L}(\langle y, z \rangle)$,
   (c) if $\langle z, x \rangle \in E$, then set $\mathcal{L}(\langle z, x \rangle) = \mathcal{L}(\langle z, x \rangle) \cup \{\mathsf{Inv}(S) | S \in \mathcal{L}(\langle y, z \rangle)\}$, and
   (d) remove $\langle y, z \rangle$ from E;
3. set $\mathcal{L}(x) = \mathcal{L}(x) \cup \mathcal{L}(y)$;

4. add $x \neq z$ for all z such that $y \neq z$; and

5. Prune(y).

A $\mathcal{SHOIQ}$ tableau T contains a **clash** iff one of the following three situations occurs:

- $\{A, \neg A\} \subseteq \mathcal{L}(x)$, for some concept name A and a node x ;
- for some simple role S and a node x, $(\leqslant nS.C) \in \mathcal{L}(x)$ and there are $n+1$ S-neighbours $y_0, ...y_n$ of x such that $C \in \mathcal{L}(y_i)$ and $y_i \neq y_j$ for all $0 \leq i \leq j \leq n$;
- for some nominal name o, there are nodes $x \neq y$ with $o \in \mathcal{L}(x) \cap \mathcal{L}(y)$.

let O be the set of nominal names that occur in $\mathcal{T}$, then the tableau algorithm starts with the initial completion graph $T = (\{r_1, ..., r_l\}, \emptyset, \mathcal{L}, \emptyset)$ such that for every $o \in O$, there is a $r_i \in V$ with $\mathcal{L}(r_i) = \{o\}$. Then T is repeatedly expanded using the expansion rules introduced in the above, until no rule can be applied or a clash occurs.

2.3 Web Ontology Language - OWL

There has been a significant body of recent work on languages for specifying ontologies on the semantic web, including activities on the development of OIL (Ontology Inference Layer) (Fensel et al., 2001), DAML (DARPA Agent Markup Language) (Ouellet and Ogbuji, 2002), their combination DAML+OIL (Horrocks, 2002), and the recent OWL (Web Ontology Language) (Schreiber and Dean, 2004). In particular, OWL has been released as a W3C (World Wide Consortium) recommendation in February 2004, and the last three years witnesses the rapid development and adaption of OWL in a wide range of tools and services.

From the modeling point view, OWL has a strong correspondence to description logics (Horrocks et al., 2003). Concepts and roles in DLs correspond to *classes* and *properties* in OWL, respectively. Table 2.3 summaries OWL class constructors and axioms and their DL correspondences (except for features involving data types). The abstract syntax given in the table can be encoded in RDF/XML as its exchange syntax (Bechhofer et al., 2004). Hence, OWL shares many common features with RDF (Resource Description Framework), such as the use of Universal Resource Identifiers (URI) for the unambiguous reference of web resources. An alternative syntax, the Manchester Syntax (Horridge et al., 2006), is recently proposed to obtain a less verbose syntax that is more friendly for non-logician users.

Table 2.3: OWL DL Class Constructors, Axioms and Facts

Abstract Syntax	**DL Syntax**	**Semantics**
Class Constructors		
A (URI reference)	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
`owl:Thing`	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
`owl:Nothing`	$\bot$	$\bot^{\mathcal{I}} = \emptyset$
`intersectionOf`(C_1 C_2 ...)	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
`unionOf`(C_1 C_2 ...)	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
`complementOf`(C)	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
`oneOf`(o_1 ...)	$\{o_1, ...\}$	$\{o_1, ...\}^{\mathcal{I}} = \{o_1^{\mathcal{I}}, ...\}$
`restriction`(R `someValuesFrom`(C))	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y, \langle x, y\rangle \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$
`restriction`(R `allValuesFrom`(C))	$\forall R.C$	$(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y, \langle x, y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
`restriction`(R `hasValue`(o))	$R : o$	$(\forall R.o)^{\mathcal{I}} = \{x \mid \langle x, o^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\}$
`restriction`(R `minCardinality`(n))	$\geq nR$	$(\geq nR)^{\mathcal{I}} = \{x \mid \#(\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\}) \geq n\}$
`restriction`(R `maxCardinality`(n))	$\leq nR$	$(\leq nR)^{\mathcal{I}} = \{x \mid \#(\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\}) \leq n\}$
Axioms		
`class`(A `partial` C_1 ...C_n)	$A \sqsubseteq C_1 \sqcap ... \sqcap C_n$	$A^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}$
`class`(A `complete` C_1 ...C_n)	$A = C_1 \sqcap ... \sqcap C_n$	$A^{\mathcal{I}} = C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}$
`EnumeratedClass`(A o_1 ...o_n)	$A = \{o_1, ..., o_n\}$	$A^{\mathcal{I}} = \{o_1^{\mathcal{I}}, ..., o_n^{\mathcal{I}}\}$
`SubClassOf`(C_1 C_2)	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
`EquivalentClasses`(C_1 ...C_n)	$C_1 = ... = C_n$	$C_1^{\mathcal{I}} = ... = C_n^{\mathcal{I}}$
`DisjointClasses`(C_1 ...C_n)	$C_i \sqcap C_j = \bot$	$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \emptyset$
`ObjectProperty`(R `super`(R_1)...`super`(R_n)	$R \sqsubseteq R_i$	$R^{\mathcal{I}} \subseteq R_i^{\mathcal{I}}$
`domain`(C_1) ...`domain`(C_m)	$\geq 1R \sqsubseteq C_i$	$R^{\mathcal{I}} \sqsubseteq C_i^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
`range`(C_1) ...`range`(C_m)	$\top \sqsubseteq \forall R.C_i$	$R^{\mathcal{I}} \sqsubseteq \Delta^{\mathcal{I}} \times C_i^{\mathcal{I}}$

Continued on the next page

Table 2.3 – continued from the previous page

Abstract Syntax	DL Syntax	Semantics
[inverseOf(R_0)]	$R = R_0^-$	$R^{\mathcal{I}} = (R_0^{\mathcal{I}})^-$
[Symmetric]	$R = R^-$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^-$
[Functional]	$\top \sqsubseteq \leq 1R$	$\forall x, \#(\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\}) \leq 1$
[InverseFunctional]	$\top \sqsubseteq \leq 1R^-$	$\forall x, \#(\{y \mid \langle y, x\rangle \in R^{\mathcal{I}}\}) \leq 1$
[Transitive])	Trans(R)	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
EquivalentProperties(R_1 ...R_n)	$R_1 = ... = R_n$	$R_1^{\mathcal{I}} = ... = R_n^{\mathcal{I}}$
AnnotationProperty(S)		
Facts		
Individual(o type(C_1) ...type(C_n	$o \in C_i$	$o^{\mathcal{I}} \in C^{\mathcal{I}}$
value(R_1 o_1)...value(R_n o_n))	$\langle o, o_i\rangle \in R_i$	$\langle o^{\mathcal{I}}, o_i^{\mathcal{I}}\rangle \in R_i^{\mathcal{I}}$
SameIndividual(o_1 ...o_n)	$o_1 = ... = o_n$	$o_1^{\mathcal{I}} = ... = o_n^{\mathcal{I}}$
DifferentIndividuals(o_1 ...o_n)	$o_i \neq o_j$, for $i \neq j$	$o_i^{\mathcal{I}} \neq o_j^{\mathcal{I}}$, for $i \neq j$

OWL comes with three sub-languages with increasing expressivity power:

- **OWL-Lite**, which corresponds to the DL $\mathcal{SHIF}(D)$ (where (D) stands for data types), provides a useful subset of the OWL language features, such as concept hierarchies and property restrictions. The limitations on OWL Lite place it in a lower complexity class than OWL DL, making reasoning with OWL Lite being more efficient.

- **OWL-DL**, which corresponds to the DL $\mathcal{SHOIN}(D)$, provides the maximal subset of OWL with known decidability (hence ensure the existence of reasoners for OWL-DL) at the time when OWL was developed[3]. For example, OWL-DL requires strict type separation between classes, properties and individual, and cardinality constraints can only be placed on simple properties (roles).

- **OWL-Full** contains all the OWL language constructs and provides free, unconstrained use of RDF constructs. OWL Full also allows classes to be treated as individuals. How-

[3] As we have shown in the previous section, the DL $\mathcal{SHOIQ}$ is decidable and a practical reasoning algorithm was found for it after OWL was developed.

ever, use of the OWL Full features generally leads to the loss of decidability (except for under certain conditions, e.g., assuming contextual semantics (Motik, 2005)).

The most recent proposed extension to OWL is the OWL 1.1 language (Grau et al., 2006b), which provides several new features, such as syntactic sugars, increased expressivity in property constructs, increased datatype expressivity, meta-modeling ability, and semantic-free comments. The increased expressivity offers a more powerful yet still decidable underlying DL (than $\mathcal{SHOIN}(D)$) for OWL 1.1, i.e., the DL $\mathcal{SROIQ}(D)$ (Horrocks et al., 2006), which provides new constructors for acyclic complex role inclusions, disjoint roles (for simple roles), reflexive, irreflexive and asymmetric roles (for simple roles), universal roles, negated role assertions of the form $\neg R(a, b)$ in an ABox, and concept constructor $\exists R.\mathsf{Self}$ (i.e., to express the "local reflexivity" of a role).

OWL also provides a special construct, `owl:imports`, to bring information in different OWL ontologies into a single ontology. It offers modularity in limited sense such that an ontology can be *syntactically* divided into several files. However, as we will discuss later, OWL does not allow semantic modularity or preserving of knowledge context. In Chapter 4, we will discuss those limitations of OWL in more details and present solutions to adapt OWL as a modular ontology language.

CHAPTER 3. Modular Ontologies: Desiderata and Abstract Description

In section 3.1, we introduce the need for modular ontologies in a wide range of applications; in section 3.2, we give a set of informal requirements for modular ontology languages; in section 3.3, we present an Abstract Modular Ontology (AMO) language that is independent from concrete language construct features, and give a formal description of the semantic requirements on modular ontology languages specified in section 3.2. The AMO framework described in this chapter will serve as the semantic foundation for the package-based description logics (P-DL) that will be described in the next chapter, as well as for the comparison between P-DL and other modular ontology formalisms.

3.1 Modular Ontology Desiderata

Some have argued for a single, comprehensive, and cohesive upper level ontology, e.g., CYC (Lenat, 1995) or SUMO (Niles and Pease, 2001), to bridge the semantic gaps between different autonomous resources on the Semantic Web. However, since the web is a network of loosely coupled, distributed, autonomous entities, differences in *ontological commitments* or points of view are inevitable. In such a setting, no single, global ontology is unlikely to satisfy the needs of all users. In practice, the sheer huge volume of web information leads to serious scalability problem on building, storing and reasoning with a single ontology. Consequently, ontologies on the web are likely to be modular, distributed, and targeted to specific users, and are linked or (possibly only partly) reused as appropriate in a given setting.

Modularity has been a frequently mentioned, desirable feature to solve many of the existing problems in ontology engineering. However, the notion of "modularity" and what is a "good" module vary considerably in different applications. Different authors have given the requirements for modules from several specific contexts. Some of these requirements conflict with each other (Loebe, 2006). In this section, we will examine the major desiderata of modular

ontologies, which will guide the design of a practical modular ontology formalism.

Generally speaking, the need for modularity in ontologies can be viewed along two dimensions: the syntactical modularity and the semantic modularity. **Syntactic modularity** addresses the need to organize large ontologies in multiple, manageable, compact modules, so that syntactic interactions between ontology modules are well-controlled for more efficient ontology construction, revision and reuse. **Semantic modularity** addresses the need to allow localized and contextual points of view of autonomous contributors of different ontology modules and distributed reasoning. More details of these two types of modularity will be introduced in the following two subsections.

3.1.1 Syntactic Modularity

Modules represent ideally more or less self-contained units of an ontology that are loosely coupled. More precisely, syntactic modularity captures the need for the separation between ontology units such that a unit can be added or deleted from the whole knowledge base without modifying, or only needs minimal change of, the syntax of other units.

3.1.1.1 Loose Coupling

One of the major concerns in designing ontology modules is that the interaction between ontology modules should be minimized so that different modules are only *loosely coupled*. Syntactically, it can be measured by the *connectedness* (Schlicht and Stuckenschmidt, 2006) of ontology modules, i.e., the number of shared symbols between axioms in different modules. The intuition is that the communication overhead needed in distributed computation is heavily influenced by the exchange of messages that contain the shared symbols. Hence, in order for efficient reasoning to be possible, ontology modules must be only loosely connected (Amir and McIlraith, 2005). Such a notion of modularity also leads to several structural criteria for ontology modularity based on graph decomposition (Stuckenschmidt and Klein, 2004; Schlicht and Stuckenschmidt, 2006).

3.1.1.2 Organizational Structure

It is often useful to distinguish between two types of structures in ontologies: organizational structure and semantic structure. The *organizational structure* of an ontology consists of an

arrangement of symbols and axioms into moderate-sized units which is aimed at making the ontology easy to design, use, and reuse. The *semantic structure* of an ontology, on the other hand, deals with the relationship (e.g., concept hierarchy) between meanings of symbols in an ontology.

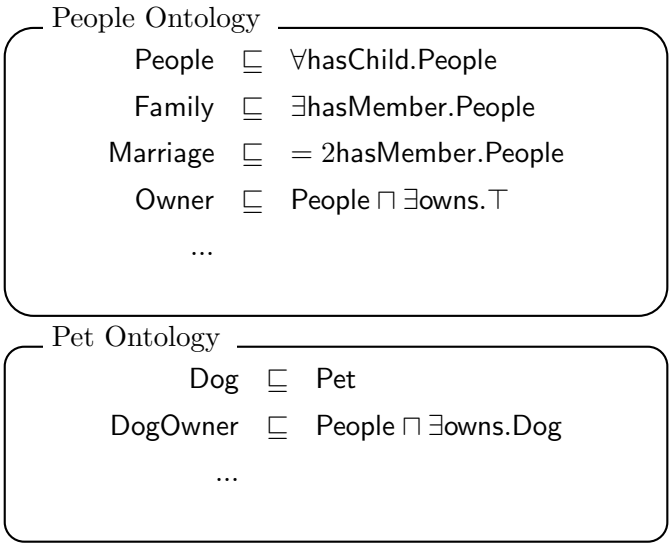


Figure 3.1 Organizational Structure vs. Semantic Structure

For example, suppose there are two ontologies where knowledge is *organized* in different modules about pet and people, respectively (Fig. 3.1). The division of knowledge (represented by axioms) in different units (typically each with a focused domain, e.g., pet or people) forms the organizational structure of the ontology. On the other hand, semantic structure of the ontology is specified by axioms (possibly from multiple modules) explicitly (e.g., a DogOwner is a People who owns Dog) or implicitly (e.g., a DogOwner is a Owner). Organizational substructure does not necessarily always correspond to the semantic structure: concept subsumptions that form one concept hierarchy may be from different modules (e.g., in an upper-level general ontology and a domain-specific ontology), and concepts that are not directly semantically related may be organized together in one module to describe a domain from multiple aspects (e.g., on marriage and ownership about people in Fig. 3.1).

The distinction between organizational and semantic hierarchies can be understood by drawing an analogy with object-oriented programming languages such as Java. In such languages, new classes can be derived from (and hence semantically related to) existing classes. Such class hierarchies offer an example of semantic structure. Java also has a notion of packages, which are

organized in a package hierarchy. Semantically unrelated classes can be organized into packages that bundle together the classes that are used in a specific class of applications (e.g., graphics).

Unfortunately, unlike the case of software engineering where modular design of programs is supported by formal methods, there still lacks principled ways to guide the design of ontology organizational structure.

3.1.1.3 Syntactical Partial Reuse

Ontologies are very likely to be reused. However, the lack of modularity in ontologies often leads to the "everything or nothing" choice in ontology reuse. For example, in creating a "MyPet" ontology, one may want to import the knowledge about pets from a comprehensive 'Animal' ontology. However, if the "Animal" ontology is only treated as a monolithic entity, one can only reuse the 'Animal' ontology in its entirety, although only a small part of it is needed. Modular structure would enable flexible and efficient partial reuse of the ontology. As shown in Fig. 3.2, a modularized version of the "Animal" ontology allows only the relevant parts of the whole "Animal" ontology being reused by the "MyPet" ontology, thereby avoiding the need to import *unwanted* ontology fragments. This is especially useful when the reused ontology is very large.

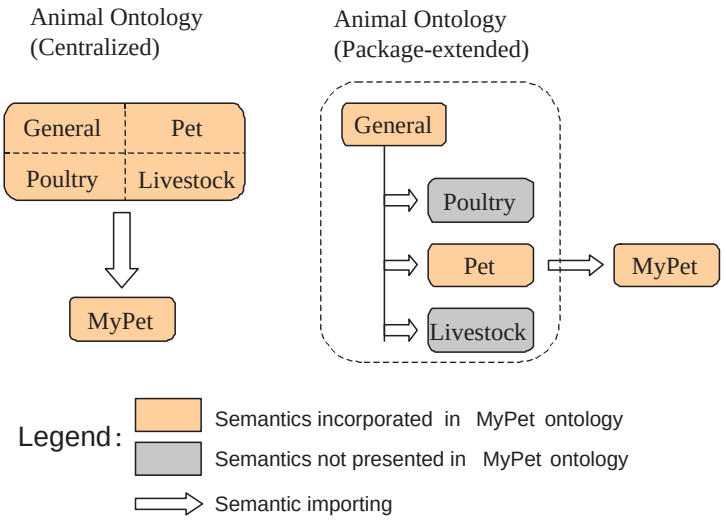


Figure 3.2 Total Reuse vs. Syntactical Partial Reuse

3.1.2 Semantic Modularity

3.1.2.1 Contextuality

Due to the distributed nature of the semantic web, ontologies on the semantic web typical capture only contextual knowledge. Such knowledge may depend on many implicit assumptions that the ontology users are not always aware of. Examples of contextual phenomena in web ontologies include the following:

- **Implicit Domain of Discourse**. Consider two independently developed ontologies: a *People* ontology and a *Work* ontology. The *People* ontology asserts (with an implicity context of people) "those that is not a man is a woman", and *Work* ontology asserts "an equal opportunity enterprise employs both men and women". Attempts to reuse that knowledge without regard to it applicable context can lead to unintended consequences. For example, an attempt to reuse the knowledge from the *People* ontology in the context of the *Work* ontology may lead to the absurd conclusion: "every enterprise is either a man or a woman".

- **Difference in the Universe of Discourse**. Suppose we will query two ontologies about people in two different departments. In the first ontology, the universe of people is explicitly enumerated by their names, which can be modeled using nominals in description logics, e.g., $\mathsf{People} = \{\mathsf{Alice}, \mathsf{Bob}\}$. In the second ontology, the universe of people is also explicitly enumerated but with a different set of names (which is disjoint from the ones in the first ontology), e.g., $\mathsf{People} = \{\mathsf{Carol}, \mathsf{Dave}, \mathsf{Eve}\}$. Hence, the two ontologies disagree on the members of People such that it has different interpretations in the two contexts.

- **Subjectivity**. Contextual differences also occur when there are conflicting political, cultural or social points of view of ontology producers (Guha et al., 2004). For example, in Western countries, the notion $\mathsf{Weekend}$ typically refers to Saturday and Sunday, while in Muslim countries it is Friday and Saturday. For another example, an animal ontology may assert a dog is a carnivore, a carnivore only eats animals, and a animal is not a plant:

$$\begin{aligned} \mathsf{Dog} &\sqsubseteq \mathsf{Carnivore} \\ \mathsf{Carnivore} &\sqsubseteq \forall \mathsf{eats}.\mathsf{Animal} \\ \mathsf{Animal} &\sqsubseteq \neg \mathsf{Plant} \end{aligned}$$

However, a dog ontology, which may reuse knowledge from the animal ontology, asserts that a sick dog sometimes eats grass which is plant

$$\begin{aligned} \mathsf{SickDog} &\sqsubseteq \mathsf{Dog} \sqcap \exists \mathsf{eats}.\mathsf{Grass} \\ \mathsf{Grass} &\sqsubseteq \mathsf{Plant} \end{aligned}$$

Both the modules capture truth from local points of view. There is no requirement for the author of the general animal ontology module to know all possible "exceptions" in the future, and the dog ontology module doesn't want to give up the whole general animal ontology module just because of few inconsistencies.

- **Spatio-Temporal Contexts**. The same sentence or symbol may have different validity or meanings in different places or at different time. For example, the sentence "Today is weekend" may be true or false in different time zones of the world at the same moment, or in the same place but in different days. Ontology evolution can also lead to temporal context change: For example, Hong Kong has recently changed from the 6-day work week system to the 5-day work week system, hence the meaning of Weekend on web pages made in different time in Hong Kong may refer only Sunday or both Sunday and Saturday.

Contexts in AI and KR (Knowledge Representation) has received increasing attention from early 1990s, including the pioneering work of McCarthy and Guha (Guha, 1991; McCarthy, 1993; Guha et al., 2004), Local Model Semantics / Multi Context System (LMS/MCS) (Giunchiglia and Ghidini, 1998; Ghidini and Giunchiglia, 2001), and Propositional Logic of Context (PLC) (Buvac and Mason, 1993; Buvac and Kameyama, 1998). Please refer to a survey paper (Serafini and Bouquet, 2004) for more details. Recently, some of those work is extended to the semantic web setting (Bouquet et al., 2003; Guha and McCarthy, 2003; Bontas, 2004; Stoermer, 2006; Stoermer et al., 2006). These efforts have led to the realization that, due to the unavoidable contextuality of web ontologies, it is unrealistic to have a single ontology; instead, web ontologies are necessarily contextualized, distributed and modular.

3.1.2.2 Semantic Partial Reuse

Syntactical partial reuse of ontologies addresses the need to reuse knowledge with the pre-defined organizational structure of ontologies. However, such a syntactical solution is not

enough for "dynamic" reuse of ontologies, where no single pre-defined organizational structure can provide efficient reuse of the ontology in all possible reuse scenarios. For example, suppose one wants to develop a pet chicken ontology and reuse the animal ontology in Fig. 3.2. Even if the ontology has pre-defined modules which are relevant (e.g. about Pet and Poultry), it may not be necessary to reuse *all* the contents in those modules for this particular reuse scenario, e.g., knowledge about dogs in the Pet module is not relevant and does not need to be reused.

A semantic solution to partial reuse may be realized by providing methods for automatic discovery of semantically "relevant" modules for arbitrary reuse problems (Pan et al., 2006; Grau et al., 2007; Alani et al., 2006). Hence, not all knowledge in the reused ontologies (or ontology modules), but only the relevant part of it, will be propagated to the target ontology (or ontology module). However, several problems that related to semantic partial reuse are still remained open, including principled ways to support semantic partial reuse in ontology language, and algorithms to extract *minimal* relevant modules from an ontology.

3.1.2.3 Semantic Encapsulation

Analogous to software engineering where the "the separation of concerns" is a highly desirable feature of programs, breaking an ontology into semantically distinct modules that overlap in functionality as little as possible is desirable in knowledge engineering. (Loebe, 2006) argues that, resembling to software engineering, the basic notion of modules in ontologies may also be the separation of the interface (which defines services of a module) and the implementational body. An interface of an ontology module can be either a query language or subset of the given vocabulary. In this work, we refer such a feature as the *semantic encapsulation* of ontology modules.

To obtain the intuition of semantic encapsulation, we can consider the following example.

Example 3.1 : Suppose Alice and Bob create ontologies for their pets. These ontologies might be queried by their pet doctors. For example, a query against the two ontologies is that "if a *pet* y *eats grass*". This query can be denoted by $?(\exists \mathsf{eats}.\mathsf{Grass})(y)$ in description logic. There is no requirement that both Alice and Bob maintain their pet ontologies in the same language. For instance, Alice's agent can use a TBox of the ontology language $\mathcal{ALCO}$ (in which x, y are

nominals):

$$\begin{aligned} x &\sqsubseteq \mathsf{Grass} \\ y &\sqsubseteq \mathsf{Dog} \\ y &\sqsubseteq \exists \mathsf{eats}.x \end{aligned}$$

while Bob's agent can use an ontology written in $\mathcal{ALC}$ with an ABox:

$$\{\mathsf{Grass}(x), \mathsf{Dog}(y), \mathsf{eats}(y, x)\}$$

Both approaches guarantee that the concept membership query $?(\exists \mathsf{eats}.\mathsf{Grass})(y)$ has the same answer, although the underlying representations are different. Since implementation details are of no interest to users who query the ontologies, they can be easily hidden from such users.

Based on the discussion above and (Loebe, 2006), we believe semantic encapsulation in ontology modules means the following features:

- **The Separation of Interface and Content**. The communication between ontology modules is necessarily controlled and is only possible through the interfaces of those modules. It is also possible that one ontology has multiple interfaces (Bao and Honavar, 2005a).

- **Information Hiding**. Internal information of an module can be hidden from other modules or agents that querying the module. Hence, it may not be necessary for the whole vocabulary or axiom set of an ontology to be visible (i.e., in the reuse interface) to other modules or users.

- **Independence**. Ontology modules, while may not be completely isolated from each other, should have focused local knowledge domains such that an module can still answer queries about a local domain when used independently. Hence, interaction between ontology modules should not result in unintended changes of the inherent semantic structure of those modules. In other words, modules should be semantically loosely coupled and highly cohesive. Such a feature is very useful on the semantic web since ontologies, like web pages, are very likely to be dynamically published or vanished, and a "bad link" to a module should not result in other modules in the ontology being completely useless.

- **Substitutability**. Ideally, a semantically well-behaved module can be replaced by another module of with the same query interface. That feature would be useful since software agents on the semantic web, which produce ontologies/data and query ontologies/data of other agents, could be the products of many different enterprises.

Note that two modules that are loosely connected in syntax may be strongly interact with each other in semantics. For example, consider two ontologies:

$O_1 = \{A_1 \sqsubseteq A_2,\ A_2 \sqsubseteq A_3,\ \ldots\ ,\ A_{n-1} \sqsubseteq A_n\}$

$O_2 = \{\top \sqsubseteq A_1\}$

The two modules only share a single symbol A_1; however, when used together under the first-order semantics, the semantic structure of O_1 is strongly affected: all A_i $(i = 1, ..., n)$ become equivalent concepts.

It is also possible that two modules are semantically modular but have strong syntactic connectedness. For example, consider:

$O_1 = \{A_1 \sqsubseteq B,\ \ldots\ ,\ A_n \sqsubseteq B\}$

$O_2 = \{C \sqsubseteq A_1,\ \ldots\ ,\ C \sqsubseteq A_n\}$

O_1 and O_2 share every symbol except B in O_1 and C in O_2. However, the two ontologies do not influence the semantic structure of each other. Hence, they are only loosely coupled in semantics.

3.1.3 Other Modularity Considerations

In addition to syntactic and semantic modularity concerns, other modularity considerations include engineering benefits and scalability in reasoning and ontology building.

3.1.3.1 Engineering Benefits

Modularization brings benefits to software engineering "as a mechanism for improving the flexibility and comprehensibility of a system while allowing the shortening of its development time" (Parnas, 1972). Practice in ontology engineering suggests that modularity in ontologies may bring similar benefits for the construction and management of large ontologies. More precisely, the expected engineering benefits of ontology modularity include the following (in analog to their software engineering counterparts):

- **Collaboration**. An ontology with modular structure can be more easily constructed collaboratively with separate groups working on different modules of the ontology. Editing conflicts and semantic inconsistencies can be more easily detected and reconciled. Information hiding can help ontology designers to focus on the modules that fit best for their expertise.

- **Flexibility**. A modular structure allows ease composition or decomposition of an ontology; modules in existing ontologies can be more easily extracted and used in new ontologies; an ill-designed or obsolete module can be replaced with a new module with controlled impact on other modules.

- **Comprehensibility**. Building or using an ontology often requires human users to understand the content of the ontology. An ontology with coherent and compact modules can significantly improve its understandability. Modularity may also improve query performance.

- **Debugging**. Debugging an ontology is the process to detect and repair "errors" (e.g., inconsistency or unsatisfiable concepts) in the ontology (Kalyanpur et al., 2006). Well-defined semantic modularity can help developers to quickly locate and eliminate problems in ontologies.

Many of the above issues will be further addressed in detail in Chapter 7 (Collaborative Building of Modular Ontologies).

3.1.3.2 Scalability

Many ontology tools, e.g., reasoners, editors and query engines, are known to perform well on small-scale ontologies, but drastically degrade in performance when the size of the ontology increases. Modularity can help to improve the *practical* scalability of such tools to handle large-scale ontologies, including the following aspects.

- **Reasoning**. The memory and space cost of the reasoning process may dramatically increase when ontologies become large. Theoretical time complexity of consistency checking of $\mathcal{ALC}$ with TBox is already in exponential time and tableau-based algorithms used by popular reasoners run in 2NExpTime. Scalability studies (Gardiner et al., 2006) have

revealed that popular description logic reasoners, despite being highly optimized, may fail on reasoning tasks on ontologies with tens of thousands of concepts. Ontologies with large instance sets can have millions of triples (Guo et al., 2005). Hence, it is often unrealistic to preform reasoning over a large ontology in the centralized fashion. By decomposing a large ontology into multiple semantically coherent modules, it may be possible to substantially speedup reasoners. (Spaccapietra (Coordnator), 2005).

- **Communication**. Ontology modules are likely to be physically distributed. A centralized reasoning would require transferring ontologies to a single location. Partial reuse of ontology can reduce the time and network overhead required: instead of the entire ontology, only the relevant modules of the ontology are need to be transmitted.

- **Editing and Visualization**. Memory limits in semantic web terminals (especially for mobile devices (Wahlster, 2006)) present a major bottle-neck in handling large ontologies. Utilizing modularity, the memory required in editing and visualizing ontologies can be reduced since it is possible to load only a selected subset of modules into memory.

- **Ontology Evolution**. Modularization of an ontology makes it possible to localize the impact of changes to an ontology. For example, to track updates of an ontology, only the changes to relevant modules or axioms need to be archived (Bao et al., 2006g). Ontology evolution can also be realized by "patching" an ontology with new modules instead of re-publishing the whole ontology.

3.2 Modular Ontology Formalisms: Required Features

The previous section lists some application desiderata for ontology modularity. This section will further informally describe the desirable features of a "good" *modular ontology formalism* in its semantics and expressivity to achieve those application desiderata.

3.2.1 Semantic Requirements

Based on the desiderata discussed in the previous section, we first list a set of minimal semantic requirements for modular ontology languages (Bao et al., 2006b) on the semantic web as the basis for our design and comparison of modular ontology formalisms.

3.2.1.1 Localized Semantics

A modular ontology should not only be *syntactically modular* (e.g., by storing ontologies in separated XML name spaces), but also *semantically modular*. That is, the existence of a *global model* should not be a requirement for integration of ontology modules. Otherwise, it is difficult to support contextuality and reasoning with only local knowledge.

3.2.1.2 Exact Reasoning

The answer to a reasoning problem over a collection of ontology modules should be *semantically equivalent* to that obtained by reasoning over an ontology resulting from a selective*integration* of the relevant ontology modules. For example (see Fig. 3.3), suppose an ontology O contains

$$\{A \sqsubseteq B, B \sqsubseteq C, C \sqsubseteq D\}$$

and a modularized version of O has two modules $M_1 = \{A \sqsubseteq B\}$, $M_2 = \{C \sqsubseteq D\}$ and a semantic connection $B \xrightarrow{\sqsubseteq} C$, which represents the modularized version of $B \sqsubseteq C$. The answer to any reasoning problem (e.g., if $A \sqsubseteq D$) obtained by the integration of M_1, M_2 and $B \xrightarrow{\sqsubseteq} C$ should be the same as that obtained by using a sound and complete reasoner on O.

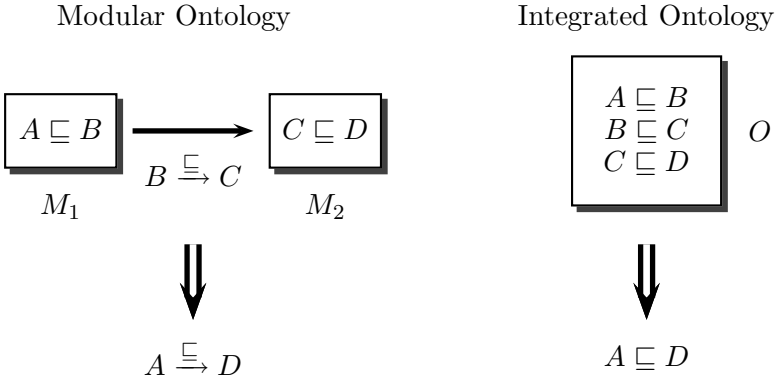


Figure 3.3 Reasoning Exactness

We call such a property the *reasoning exactness* of module ontologies. The description given above is only informal; more precise description depends on answers to the following questions:

- How are modules integrated? Are the modular ontology and the integrated ontology indeed equivalent representations of the same *intended* modeling scenario? For example, if we use $B \xrightarrow{\sqsubseteq} C$ to represent *concept inclusion* between B and C in the modular ontology,

its correspondence in the integrated ontology should be $B \sqsubseteq C$, and we expect "$\stackrel{\sqsubseteq}{\Longrightarrow}$" in the modular ontology to have the similar properties as that of "$\sqsubseteq$" in the integrated ontology (e.g., transitivity).

- From which *context* do we say that the reasoning is exact? Is the integrated ontology *unique* from points of view of different modules? Is the global consistency of *all* modules a prerequisite for integration?

- What reasoning problems should be answered identically by the modular ontology and its integrated counterpart? For example, suppose we have "$\stackrel{\sqsubseteq}{\Longrightarrow}$" to represent "$\sqsubseteq$" and "$\stackrel{\sqsupseteq}{\Longrightarrow}$" to represent "$\sqsupseteq$" in the modular ontology, the modular ontology entails $A \stackrel{\sqsubseteq}{\Longrightarrow} D$ when the integrated ontology entails $A \sqsubseteq D$, should it also be required that the modular ontology entails $\neg A \stackrel{\sqsupseteq}{\Longrightarrow} \neg D$ when the integrated ontology entails $\neg A \sqsupseteq \neg D$?

A related property named "compositionality" is stated in (Loebe, 2006):

> *"If certain logical properties are proved for modules, there should be general means to derive these properties for the overall system. For example, the consistency of a set of modules may immediately result in the consistency of their combination, i.e., due to the combination operation defined for modules"*

We regard reasoning exactness as a special case of compositionality. However, compositionality is a stronger requirement which may not hold in general. For example, even when all modules are locally consistent, their combination may not necessarily be consistent.

3.2.1.3 Directionality

A modular ontology framework must support *directional semantic relations* from a *source* module to a *target* module. A directional semantic relation affects only the reasoning within the target module but not the source module. This requirement is motivated by the fact that ontology reuse is typically an asymmetric process such that the "new" (target) module should not have information "backflow" into the reused (source) module. A similar requirement has been presented in (Borgida and Serafini, 2003).

For example, in the Fig. 3.4, module M_1 reuses module M_2; however, the new knowledge $B \sqsubseteq C$ should not be back propagated into M_1. Hence, M_1 should not entail that $A \sqsubseteq D$.

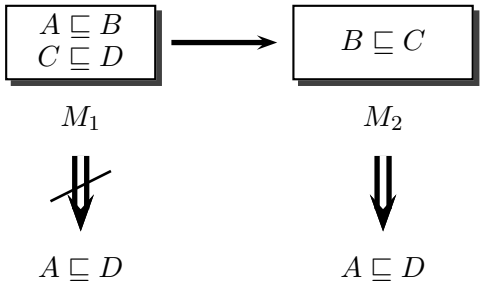


Figure 3.4 Directionality of Modular Ontologies

3.2.1.4 Monotonicity and Transitive Reusability

Classic knowledge reuse in description logics is *monotonic* in the sense that new knowledge does not alter logical consequences of existing knowledge. That is, if a statement α is provable in a knowledge base K, then α is also provable in $K \cup K'$ where K' is a new knowledge base. We wish that ontology reuse in a modular fashion will also have the monotonicity property, hence extending a knowledge base in the modular way will have the same effect as that of extending it in the classic way.

A special case of monotonicity is *transitive reusability*, i.e., knowledge contained in ontology modules should be both directly and indirectly reusable. That is, if a module M_1 reuses module M_2, and module M_2 reuses module M_3, then effectively, module M_1 reuses module M_3 (i.e., "relevant" axioms in M_3). For example, in the Fig. 3.5, knowledge in M_1 ($A \sqsubseteq B$) may be indirectly reused by M_3 to obtain the conclusion that $A \stackrel{\sqsubseteq}{\longrightarrow} D$.

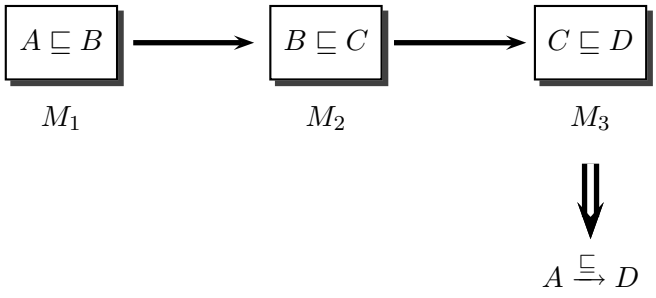


Figure 3.5 Knowledge Transitive Reusability

The monotonicity requirement is also different from monotonicity in classic knowledge reuse on the following aspects:

- Different modules may draw conclusions from different contexts. For example, suppose a

People ontology (with the implicit context of people) asserts that "every individual is a human" ($\top \sqsubseteq \mathsf{Human}$), and a Pet ontology reuse the People ontology, if the pet ontology is required to also assert the same conclusion in the pet context, it will have to assert that "every pet is a human" ($\mathsf{Pet} \sqsubseteq \mathsf{Human}$). Hence, contextuality of knowledge should be preserved when axioms "propagate" to other modules.

- An ontology module O may only be partially reused by another module O', some axioms in O may not be "relevant" to the reused part, hence it is not necessary to propagate such axioms to O'. For example, assuming O contains $\{A \sqsubseteq B, C \sqsubseteq D\}$, O' reuses O by importing the symbols A, B but not C, D; $C \sqsubseteq D$ may not be required from the point of view of O' since it is "irrelevant" to the knowledge about A, B.

3.2.1.5 Decidability

Decidability is required to develop sound and complete reasoners for modular ontologies. In practice, it is usually easier to ensure local decidability of component ontology modules as opposed to decidability of the entire ontology. Hence in designing a modular ontology formalism, we will focus on the "decidability transfer" property of modular ontologies, i.e., ensure that the whole ontology is decidable if every component module is locally decidable.

It is known that unrestricted combination of ontology modules can result in undecidability even when every component module is locally decidable. For example (from (Baader et al., 2000)), the union of $\mathcal{ALCF}$ ($\mathcal{ALC}$ extended with functional roles and the same-as constructor on chains of roles) and $\mathcal{ALC}^{+,\circ,\sqcup}$ ($\mathcal{ALC}$ extended with transitive closure, composition and union of roles), both known to be decidable, results in the undecidable logic $\mathcal{ALCF}^{+,\circ,\sqcup}$.

3.2.1.6 Discussion of Other Desiderata

We believe that the desiderata listed above are among the most critical ones for a modular ontology to be semantically sound and practically usable. Other desiderata that have been considered in the literature include: the ability to cope with local inconsistency or global inconsistency (Borgida and Serafini, 2003) and local completeness (Grau et al., 2006d) (i.e., a module has the complete knowledge about its own vocabulary). We do not consider them here since we believe they address disparate aspects from the desiderata required for the design of a *modular ontology language*, hence are best handled by other machineries.

Inconsistency handling requires techniques to come up with *meaningful* results from inconsistent modular ontologies, such as removing locally inconsistent modules from the reasoning process (Borgida and Serafini, 2003) (by giving empty interpretations to those modules), selecting a consistent subset of an inconsistent ontology (Huang et al., 2005), debugging "problems" that lead to the inconsistency (Parsia et al., 2005; Kalyanpur et al., 2006; Meilicke et al., 2007), and defeasible extensions to ontology languages (Heymans and Vermeir, 2002; Bassiliades et al., 2004). However, such techniques are motivated by a very different set of desiderata in the more general context of semantic web and are not special to modular ontologies. It is best to be separated from the modular ontology language specification and be preformed as extra service over the language layer.

Local completeness is a property that is best achieved by certain *design pattern of ontology modules* instead of as a property of the ontology language. Furthermore, requiring local completeness may preclude many useful modeling scenarios (e.g., refining the class hierarchy of existing symbols with new knowledge).

3.2.2 Expressivity Requirements

The second group of requirements that we consider is aimed at evaluating the *language expressivity*. The expressivity of modular ontology formalisms can be measured in two dimensions: the expressivity of component ontology modules and the expressivity of semantic connections between component ontologies.

In semantic web applications, we expect each ontology module to be expressed using the standard ontology language OWL, or can be translated into OWL by some mediators. Hence, in this work we assume each component ontology module is expressed in a subset of the description logic $\mathcal{SHOIQ}(D)$[1], which roughly corresponds to OWL-DL.

Semantic connections are relations between vocabularies in different modules. We illustrate such a need using an example of an ontology with two modules: a people module and a pet module. Typical semantic connections include the following:

- **Concept Subsumption** (and its special case, **Concept Equivalency**) between modules is probably the most urgently needed feature. We may need to assert that a concept in

[1]For the purpose of conciseness, we will omit the concrete domain (D) from discussions hereafter. However, the result we obtained can be easily extended to the case that with concrete domain.

one module is more general than (i.e., a generalization of) or less general than (i.e., a specialization of) another concept in another module. For example, we may need to say "DogOwner in the pet module is less general than Human in the people module" (corresponds to pet:DogOwner $\sqsubseteq$ people:Human in standard DL) and "Animal in the pet module is more general than Human in the people module" (corresponds to people:Human $\sqsubseteq$ pet:Animal in standard DL).

- **Boolean Concept Constructors**. They allow us to build complex concepts based on concepts from different modules, using boolean operators such as negation ($\neg$) , conjunction ($\sqcap$) and disjunction ($\sqcup$), e.g., "*not* Pets" (e.g., in people:Human $\sqsubseteq \neg$pet:Pet), "Male DogOwner" (people:Male$\sqcap$pet:DogOwner), "Baby or Puppy" (e.g., in people:Baby$\sqcup$ pet:Puppy $\sqsubseteq$ pet:MilkFeeding)

- **Restrictions**. If R is a role and C is a concept, the language may include existential restrictions ($\exists R.C$), universal restrictions ($\forall R.C$) and qualified number restrictions (e.g., $\leq 2R.C$). R or C may be a foreign symbol, or both are foreign symbols. These features can be further divided into tree groups 1) with local role name and foreign concept name; 2) with foreign role name and foreign concept name; 3) with foreign role name and local concept name.

- **Role Inclusion** (and its special case, **Role Equivalency**) between modules. For example, we may need to say " registering (a pet) implies ownership" (corresponds to pet:registerPet $\sqsubseteq$ people:owns in standard DL).

- **Role Inversion** between modules. For example, the inverse of pet:ownedBy is human:owns (corresponds to pet:ownedBy$^-$ $\sqsubseteq$ people:owns in standard DL).

- **Transitive Role**, which allows the use of a foreign transitive role, e.g., the pet module reuses a transitive role olderThan in the people module.

- **Nominal Correspondence**. For example, the pet module declares that golden (a nominal used to describe color) is the same as fair (a nominal to describe human hair color) in the human module.

Table 3.1 summarizes the features described above. They may be extended with other useful constructors, such as role construction (role complement ($\neg R$) , conjunction ($R \sqcap Q$) and

Table 3.1 Semantic Connection Expressivity of Modular Ontology Languages

Feature	**Notation**	**Corresponding Standard DL Features**
Concept Subsumption	$\sqsubseteq$	$C \sqsubseteq D$
Nominal Correspondence	$\mathcal{O}$	$a = b$
Role Inclusion	$\mathcal{H}$	$R \sqsubseteq Q$
Concept Negation	$\mathcal{C}$	$\neg C$
Concept Conjunction	$\sqcap$	$C \sqcap D$
Concept Disjunction	$\sqcup$	$C \sqcup D$
Universal Restriction	$\forall$	$\forall R.C$
Existential Restriction	$\exists$	$\exists R.C$
Attributive Modular Language	$\mathcal{ALC}_{=\sqcap\sqcup\forall\exists\mathcal{C}}$	
Number Restriction	$\mathcal{Q}$	$\leq nS.C, \geq nS.C$
Role Inversion	$\mathcal{I}$	R^-
Transitive Role	$R+$	$\mathsf{Trans}(R)$
	$\mathcal{S}_{=\mathcal{ALC}_{R+}}$	

C, D are concept names, a, b are nominal names, R, S, Q are role names and S is a simple role, n is a non-negative integer. For each formula, at least one name in it is a shared name (i.e., occurring in multiple modules).

disjunction $(R \sqcup Q))$[2], transitive closure (R^*) and role chain (e.g., $R \circ S \sqsubseteq R$). We do not consider them here since they have no correspondences in OWL-DL.

Based on different intended application scenarios, a specific modular ontology language may only contain a subset of the possible expressivity features. For example, Distritbuted Description Logics (DDL) (Borgida and Serafini, 2003; Ghidini and Serafini, 2006a) covers concept subsumption and nominal correspondence, and $\mathcal{E}$-connections (Grau et al., 2004b) addresses only concept and role construction with a special type of roles called "links".

3.3 A General Framework for Modular Ontologies

The goal of this section is to formalize many notions (informally) described in the previous section and present a general semantic framework for modular ontology languages. We will discuss an Abstract Modular Ontology (AMO) language, extending the Local Model Semantics framework of (Ghidini and Giunchiglia, 2001) with multiple domain relations, for the formal

[2]Limited role construction has been provided in $\mathcal{E}$-Connections (Grau, 2005).

specifications of semantic requirements and expressivity requirements of modular ontologies.

3.3.1 Semantics of Modular Ontologies at a Glance

"Ontology is the science of being"(Aristotle, *Metaphysics*). In a general sense, a modular ontology is a set of individual descriptions of the same domain that represent correlated, but not identical points of view of multiple observers, or agents. Thus, each ontology module can be seen as describing a point of view (or epistemic state) held by an agent with respect to the entities (objects) and their relations in the domain. We call the collection of entities that are observed by an agent the *local domain* of the agent's ontology.

For example, considering *Alice* and *Bob* who have different descriptions of the the same global domain (people in the world) due to contextual differences and limited knowledge. For example, *Alice* may believe "*Bob* is my best friend" while *Bob* believes "I'm not *Alice*'s best friend". The local domain of *Alice* will be the set of people she knows, which may be different from *Bob*'s local domain (the set of people he knows).

However, such local descriptions are not always independent of each other. For example, if *Alice* observes that "John is the father of Joe", it should not be the case that *Bob* observes that "Joe is older than John". We say that *semantic relations* reflect the ability of agents to build relations between their local points of view. For example, suppose agent *Bob* believes that

- " 'Joe' mentioned by *Alice* is the same as the individual 'Joe' I mentioned, 'John' mentioned by *Alice* is the same as the individual 'John' I mentioned " and
- "if an individual x is the father of y, then y is not older than x",

then the apparent inconsistency between their beliefs can be avoided. In this setting, semantic relations, just as ontologies, are also subjective *beliefs* rather than objective descriptions. For example, it is possible that *Alice* believes "the 'Joe' I mentioned is the same individual 'Joseph' mentioned by *Bob*" while *Bob* believes " 'Joseph' and 'Joe' I mentioned are two different persons"; hence the individual correspondences in *Alice* and *Bob*'s beliefs are different. In general, the scenario described above can be extended to a setting with multiple interacting agents. Due to the subjectivity and contextuality of their beliefs, those agents' local knowledge and semantic relations between each other may only be partially compatible.

Ontology modules can be viewed as epistemic descriptions of the physical world held by different agents, and semantic relations between ontology modules establish the (possibly partial) compatibility relation among those epistemic descriptions. In what follows, we will describe the **Abstract Modular Ontology** (AMO), a special type of Distributed First-Order Logics (DFOL) (Ghidini and Serafini, 1998), to provide the necessary framework for the evaluation of modular ontology languages. In particular, we will investigate the following problems:

- How can we formally describe the semantic requirements and expressivity requirements we presented in the last section?
- Under what circumstances can a reasoning process in a modular ontology language be said to be sound and complete?
- What are the sources of possible semantic difficulties in some modular ontology languages? How can such difficulties be avoided?
- How the compatibility relations between subjective beliefs are interpreted?

3.3.2 Abstract Modular Ontology

Distributed First-Order Logics (DFOL) was first introduced in (Ghidini and Serafini, 1998). A DFOL knowledge base (KB) includes a family of first order languages $\{L_i\}_{i\in I}$, defined over a finite set of indices I. We will use L_i to refer to the i-th module of the KB, which represents the description of agent i on its observed (partial) domain. An (i-)variable x or (i-)formula ϕ occurring in module L_i is denoted as $i : x$ or $i : \phi$ (we drop the prefix when there is no confusion). The signature (the set of all names) of L_i are i-terms. A DFOL interpretation constraint is in the form of $i : \phi(x_1, ..., x_n) \rightarrow j : \psi(y_1, ..., y_n)$, where ϕ, ψ are n-ary predicates and $\langle x_i, y_i\rangle$ is connected by a domain relation r_{ij}.

An **Abstract Modular Ontology** (AMO) (Bao et al., 2006b) KB $\langle\{L_i\}, \{\mathfrak{R}_{ij}\}_{i\neq j}\rangle$ is a variation of DFOL such that each component language L_i is a subset of description logics (DL)[3], and $\mathfrak{R}_{ij}$ represents the set of semantic relation rules between agent j's own knowledge and another agents i's ($j \neq i$) knowledge, which extend DFOL interpretation constraints (details

[3]In general, not all description logics are subset of the first order logic, such as the case for Description Logics that allow the transitive closure of roles or fixpoints, which can only be translated into predicate logic beyond first order.

in the follows). We restrict ourselves to the setting where each L_i is a subset of the expressive DL $\mathcal{SHOIQ}$. We also assume that semantic relation rules are binary, hence we do not consider n-ary relations $(n > 2)$[4], such as "if $i : \phi$ and $j : \rho$ then $k : \varphi$".

Instead of a single relation between each pair of agents, we assume that each agent may need to interact with another in different *roles* in different contexts. For example, a company can both buy and sell products from and to another company. Consequently, there may be multiple semantic relation roles between ontologies held by a pair of agents. We denote the set of roles of semantic relations between L_i an L_j as rol_{ij}.

Example of semantic relation rules in concrete modular ontology languages includes bridge rules in DDL (Borgida and Serafini, 2003), $\mathcal{E}$-connection (Grau, 2005), and concept importings in P-DL (Bao et al., 2006c) between two ontology modules. (Serafini et al., 2005b) have noted that such rules can be mapped to DFOL interpretation constraints. Note that DL concepts are unary FOL predicates and DL roles are binary predicates. Consequently, a semantic relation rule in AMO is an axiom in the form of:

$$i : C(x) \xrightarrow{\{r\}, r \in \mathsf{rol}_{ij}} j : D(x) \tag{3.1}$$

$$i : R(x_1, x_2) \xrightarrow{\{r\}, r \in \mathsf{rol}_{ij}} j : S(x_1, x_2) \tag{3.2}$$

where $C(D)$ is a DL concept formula and $R(S)$ is a DL role formula constructed using symbols from $L_i(L_j)$, $\{r\}$ is a set of semantic relation roles via which the rule is enforced. Note that free variables (e.g.,"x") have different reading at the lhs and rhs of the equations: on the lhs they denote variables in the domain of L_i, while on the rhs they denote variables in the domain of L_j.

A model of AMO includes a set of *local models* and *domain relations*. For each L_i, there exists a local interpretation domain Δ_i. Let M_i be the set of all DL models of L_i on Δ_i. We call each $\mathcal{I} \in M_i$ a *local model* of L_i. For a model of L_i, we have the usual DL interpretation for its concept, role and nominal names. For each semantic relation role $R \in \mathsf{rol}_{ij}$, the *domain relation* r_{ij}^{R}, where $i \neq j$, is a subset of $\Delta_i \times \Delta_j$ that represents the capability of the module j to map the objects of Δ_i into Δ_j via R.

Hence, the general form of semantic relation rules in AMO can be interpreted as (in the

[4]Examples of proposals with n-ary semantic relations include (Franconi et al., 2003) (motivated for peer-to-peer database applications) and (Serafini et al., 2005b).

FOL form):

$$i : C(x) \xrightarrow{\{\langle x,x'\rangle \in r_{ij}^R\}, R \in \mathsf{rol}_{ij}} j : D(x') \tag{3.3}$$

$$i : R(x_1, x_2) \xrightarrow{\{\langle x_p,x'_p\rangle \in r_{ij}^R\}, p\in\{1,2\}, R \in \mathsf{rol}_{ij}} j : S(x'_1, x'_2) \tag{3.4}$$

Equation 3.3 should be read as "if $i : C(x)$ and $\langle x, x'\rangle \in r_{ij}^R$, then $j : D(x')$". The reading of the equation 3.4 is similar.

For a domain relation r_{ij}^R of a semantic relation $R \in \mathsf{rol}_{ij}$, we use $\langle d, d'\rangle \in r_{ij}^R$ to denote that from the point of view of j, the object d in Δ_i is mapped to the object d' in Δ_j, via relation R. Finally, $r_{ij}^R(d)$ denotes the set $\{d' \in \Delta_j | \langle d, d'\rangle \in r_{ij}^R\}$. For a subset $D \subseteq \Delta_i$, $r_{ij}^R(D)$ denotes $\cup_{d\in D} r_{ij}^R(d)$.

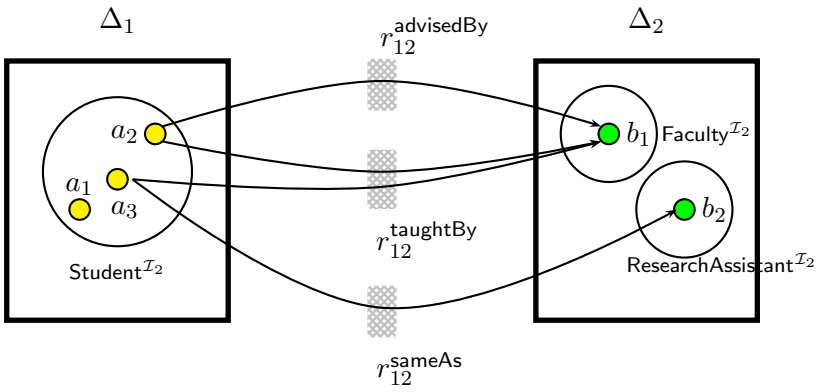


Figure 3.6 Domain Relation

Example 3.2 : An ontology contains two modules $L_{\{1,2\}}$. L_1 contains knowledge about students, such as $\mathsf{Graduate} \sqsubseteq \mathsf{Student}$ (an 1-formula). L_2 contains knowledge about faculties, such as $\mathsf{Professor} \sqsubseteq \forall \mathsf{hasAssistant}.\mathsf{ResearchAssistant}$.

The semantic relation roles rol_{21} is empty, while rol_{12} contains three relation roles advisedBy (to indicate advisorship between students and faculties), taughtBy (to indicate the student-instructor relationship) and sameAs (to indicate individual correspondence between the two domains). The semantic relation $\mathfrak{R}_{12}$ contains the rule "a ResearchAssistant must be Student", formally described as

$$\forall x \in \Delta_2, \mathsf{ResearchAssistant}(x) \xrightarrow{\{\langle x,y\rangle \in r_{12}^{\mathsf{sameAs}}\}} \exists y \in \Delta_1, \mathsf{Student}(y)$$

A model of the ontology is shown in Fig. 3.6. The local domain Δ_1 has Student objects a_1, a_2, a_3, and local domain Δ_2 has Professor object b_1 and ResearchAssistant object $\{b_2\}$. The

domain relations r_{12}^{sameAs} is $\{\langle 1 : a_3, 2 : b_2\rangle\}$, $r_{12}^{\mathsf{advisedBy}}$ is $\{\langle 1 : a_2, 2 : b_1\rangle\}$, and $r_{12}^{\mathsf{taughtBy}}$ is $\{\langle 1 : a_2, 2 : b_1\rangle\}, \langle 1 : a_3, 2 : b_1\rangle\}$. Note that the same pair of individuals $(1 : a_2, 2 : b_1)$ can be related by two different relations ($\mathsf{advisedBy}$ and $\mathsf{taughtBy}$). $r_{12}^{\mathsf{sameAs}}(1 : a_3) = \{2 : b_2\}$ indicates that (from the point of view of L_2) b_2 is the same individual as a_3.

3.3.3 Expressivity of AMO

The AMO framework given above does not address the ability of the language to express semantic relations between ontology modules. For example, can two modules share parts of their signatures? If some symbols are shared, does such a syntactical correspondence indeed ensure shared symbols in the two modules are "consistently" interpreted? How is the meaning of a symbol in one module related to the meaning other symbols in another module? How can the expressivity requirements we outlined in the previous section be precisely defined?

To answer those questions, we first examine how local domains of different ontology modules, as the epistemic descriptions of the physical world from multiple agents, are related.

3.3.3.1 Individual Image Relations

First, the same entity in the physical world may be observed by different agents (possibly from different points of view), hence are represented in the local domains of those agents' ontologies. For example (Fig. 3.7), suppose two agents i and j both know the person "Robert"; the agent i identifies the person as *bob* and the agent j identifies the person as *bobby*. The agent j may further observe that whenever the agent i mentions "*bobby*", it refers to the same person identified as "*bob*" by j. Such a correspondence can be represented as a special "image" domain relation from the domain of i to the domain of j (denoted by $r_{ij}^{\rightarrow}$), such that $\langle d, d'\rangle \in r_{ij}^{\rightarrow}$, implies that the object d' in the j's point of view denotes the *same* physical entity as the object d in the i's point of view; d' is called an *image* of d and d is a *pre-image* of d'. Hence, in the above example we have $\langle bob, bobby\rangle \in r_{ij}^{\rightarrow}$.

Note that the image domain relations, in general, are *not necessarily* one-to-one. For example, agent i may have two different individuals $MorningStar$ and $EveningStar$ in its local domain Δ_i, while another agent j maps both of them to the individual $Venus$ in its local domain Δ_j. In general, image domain relations are not necessarily symmetrical, either, due to the subjective ("directed") nature of domain relations, i.e., it may not always the case that

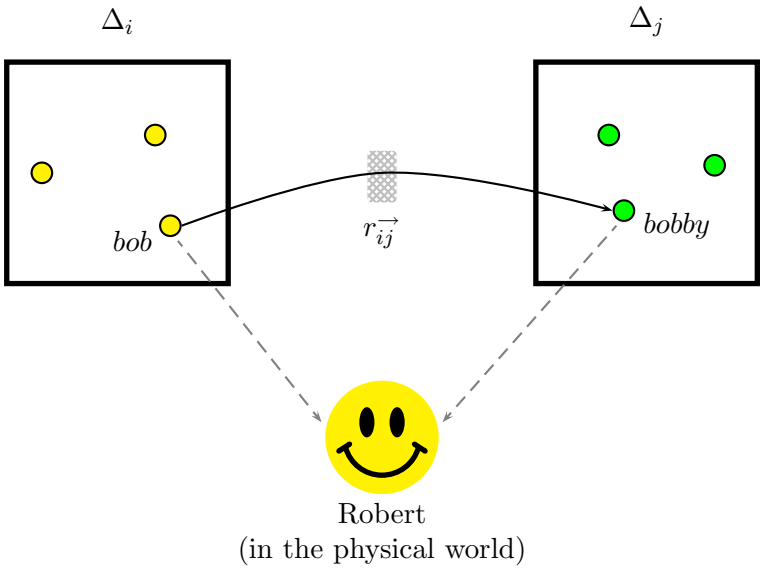


Figure 3.7 Image Domain Relation

$r_{ij}^{\rightarrow} = (r_{ji}^{\rightarrow})^{-}$ (for any $i \neq j$). However, we will show in the following text that arbitrary image domain relations can lead to semantic imprecision and reasoning difficulties, and principled approaches to avoid such problems.

3.3.3.2 Concept Images

Suppose agent i interprets a concept Student in its local domain Δ_i as $\mathsf{Student}^{\mathcal{I}_i}$ (a subset of Δ_i), then another agent j will regard $r_{ij}^{\rightarrow}(\mathsf{Student}^{\mathcal{I}_i})$ as "these objects in my local domain correspond to the concept Student from agent i's point of view". In other words, $r_{ij}^{\rightarrow}(\mathsf{Student}^{\mathcal{I}_i})$ indicates the reading of agent j on the interpretation of Student by agent i. We call $r_{ij}^{\rightarrow}(C^{\mathcal{I}_i})$ as the *concept image* of $C^{\mathcal{I}_i}$, and denote it as $C^{i \rightarrow j}$.

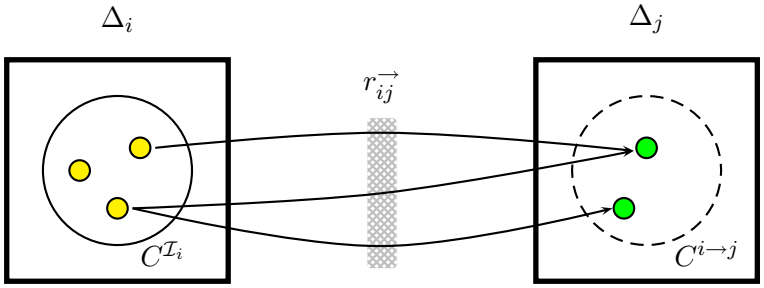


Figure 3.8 Concept Image

In general, a concept image $C^{i\to j}$ does not necessarily correspond to a named concept in ontology module j, or even be an empty set (indicating that no physical objects identified as C by i is observed by j). However, we may assert, in a local domain, the relation between a concept image and the interpretation of a named concept, or between a concept image and another concept image. The mechanisms to specify such relations varies in different concrete modular ontology languages.

3.3.3.3 Role Images

Using the image domain relation, an agent may also create "images" of *relations* of individuals in another agent's local domain. For example, suppose the local domain of an agent i has one Planet individual $Venus$, one Star individual Sun, and the binary relation circles$(Venus, Sun)$; the domain relations are as that shown in Fig. 3.9; the pair $\langle Venus, Sun\rangle$ hence can be mapped as pairs:

$$\langle MorningStar, Sol\rangle \quad , \quad \langle EveningStar, Sol\rangle$$
$$\langle MorningStar, Helios\rangle \quad , \quad \langle EveningStar, Helios\rangle$$

which indicate "the pairs of individuals in j's domain that correspond to the instances of role circles in i's domain".

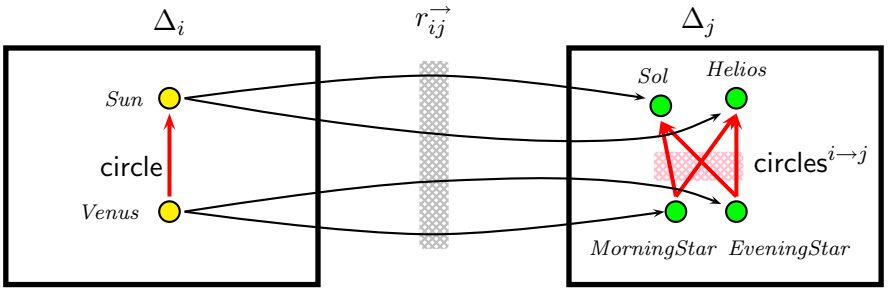


Figure 3.9 Role Image

In general, agent j can map role instances in i's domain to its local point of view: for any relation instance $\langle x_1, x_2\rangle$ in $\Delta_i \times \Delta_i$, j will regard $r_{ij}^{\rightarrow}(x_1) \times r_{ij}^{\rightarrow}(x_2)$ as a proper image of the relation. It should also be kept in mind that r_{ij} is always a relation viewed from j's point of view. For example, the fact that a person x thinks "y is my best friend" doesn't necessarily mean that y thinks "I'm x's best friend".

In summary, the *image* of the interpretation of a concept C or a role P from i to j is:

- $C^{i\to j}$: $r_{ij}^{\to}(C^{\mathcal{I}_i})$
- $P^{i\to j}$: $\bigcup_{\langle x,y\rangle\in P^{\mathcal{I}_i}} r_{ij}^{\to}(x)\times r_{ij}^{\to}(y)$

Similarly, *pre-image* of a concept D or a role R from j to i is defined as

- $D^{i\leftarrow j}$: $(r_{ij}^{\to})^{-}(D^{\mathcal{I}_j})$
- $R^{i\leftarrow j}$: $\bigcup_{\langle x,y\rangle\in R^{\mathcal{I}_j}} r_{ij}^{\to -}(x)\times r_{ij}^{\to -}(y)$

We may also extend such notations for concept image $C^{i\xrightarrow{R}j}$ or pre-image $D^{i\xleftarrow{R}j}$ via domain relation r_{ij}^{R}, instead of $r_{ij}^{\to}$.

3.3.3.4 Contextualized Constructors

Since each language L_i only describes a local domain and will be interpreted in the local domain Δ_i, the set of concept constructors (e.g., $\neg, \sqcap, \sqcup$) in AMO have only contextualized meaning. To see this, consider a special case of concept image, the images of negated concepts. For example, if agent i constructs a concept $\neg\mathsf{Student}$ (those that are not students) in its ontology about people, it refers to the set of individuals in Δ_i but not in $\mathsf{Student}^{\mathcal{I}_i}$, i.e.,

$$(\neg\mathsf{Student})^{\mathcal{I}_i} = \Delta_i \backslash \mathsf{Student}^{\mathcal{I}_i} \tag{3.5}$$

Another agent j may have a different local domain Δ_j (not necessarily disjoint from Δ_i) about jobs. We let C_{ij} be the (syntactical) place holder for the concept corresponding to the image of C, i.e., $(C_{ij})^{\mathcal{I}_j} = r_{ij}^{\to}(C^{\mathcal{I}_i}) = C^{i\to j}$. We have that

$$(\neg\mathsf{Student}_{ij})^{\mathcal{I}_j} = \Delta_j \backslash (\mathsf{Student}_{ij})^{\mathcal{I}_j} = \Delta_j \backslash \mathsf{Student}^{i\to j} \tag{3.6}$$

i.e., those objects in j's mind that do *not correspond to "Student"* in i's mind. It does not necessarily be the same as $r_{ij}^{\to}((\neg\mathsf{Student})^{\mathcal{I}_i})$ (those objects in j's mind that *correspond to "not Student"* in i's mind); for instance, $\Delta_j \backslash \mathsf{Student}^{i\to j}$ may contains objects that identify physical entities that are not people, e.g., a company, which is not in $r_{ij}^{\to}(\Delta_i)$ (hence not in $r_{ij}^{\to}(\Delta_i \backslash \mathsf{Student}^{\mathcal{I}_i})$). Hence, negation here has only local meaning: the "$\neg$" in (4.4) is interpreted in the domain Δ_i and the "$\neg$" in (4.5) is interpreted in the domain Δ_j.

In general, other constructors also should be contextualized. To make the difference explicit, we may attach subscripts "$_i$" to the constructors of L_i to obtain the following:

- Contextualized negation $\neg_i$: $(\neg_i C)^{\mathcal{I}_i} = \Delta_i \backslash C^{\mathcal{I}_i}$
- Contextualized conjunction $\sqcap_i$: $(C \sqcap_i D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i} \subseteq \Delta_i$
- Contextualized disjunction $\sqcup_i$: $(C \sqcup_i D)^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i} \subseteq \Delta_i$
- Contextualized universal restriction $\forall_i$: $(\forall_i R.C)^{\mathcal{I}_i} = \{x \in \Delta_i | \forall y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i} \rightarrow y \in C^{\mathcal{I}_i}\}$
- Contextualized existential restriction $\exists_i$: $(\exists_i R.C)^{\mathcal{I}_i} = \{x \in \Delta_i | \exists y \in \Delta_i, (x,y) \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\}$
- Contextualized number restriction $\leq_i$: $(\leq_i nR.C)^{\mathcal{I}_i} = \{x \in \Delta_i | \#(\{y \in \Delta_i | (x,y) \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\}) \leq n\}$ (similar for the $\geq_i$ case)

where C, D are i-concepts, R is an i-role.

One may ask the following questions: Can two local languages share a subset of their signatures? Can a constructor of the language L_i be used in the language L_j $(i \neq j)$? If a concept name C is shared by two languages L_i and L_j, how $\neg_i C$ and $\neg_j C$ are related? Is it possible to reduce contextualized constructors to "normal" DL constructors under certain conditions? In chapter 4, we will show that different modular ontology formalisms, motivated by different application scenarios, make different assumptions to answer those questions.

3.3.3.5 Defining Semantic Relation Rules

In a more general setting, a modular ontology language may also create *third party constraints*. For example, module j may reuse i-concept $\mathsf{Student}$ and k-concept People, and locally declare $i : \mathsf{Student} \sqsubseteq k : \mathsf{People}$. However, such a third party constraint can be avoided by an "alias" (place holder) such that $\mathsf{Student}^{i \rightarrow j}$ and $\mathsf{People}^{k \rightarrow j}$ are given local alias $\mathsf{Student}_{ij}$ and People_{ij}, respectively, thus transforming the concept inclusion to the one that connects only j-concepts.

A concept may be a complex concept constructed using a foreign role and/or a foreign concept, such as universal restriction (e.g., $\forall R.C$) or existential restriction (e.g., $\exists R.C$), as shown in the Table 3.2. It is also possible to construct concepts using semantic relations (see Table 3.2); however, since a semantic relation $Q \in \mathsf{rol}_{ij}$ corresponds to a subset of $\Delta_i \times \Delta_j$, we

Table 3.2 Possible AMO Expressivity Features

	Syntax	**Semantics**
`Concept Constructors`		
Concept Negation	$\neg_i C$	$r_{ij}^{\rightarrow}(\Delta_i \backslash C^{\mathcal{I}_i})$
Concept Conjunction	$C \sqcap_j D$	$C^{i\rightarrow j} \cap D^{\mathcal{I}_j}$
Concept Disjunction	$C \sqcup_j D$	$C^{i\rightarrow j} \cup D^{\mathcal{I}_j}$
Universal Restriction	$\forall_j R.C$	$\{x \in \Delta_j \vert \forall y \in \Delta_j, (x,y) \in R^{\mathcal{I}_j} \rightarrow y \in C^{i\rightarrow j}\}$
	$\forall_j Q^-.C$	$\{x \in \Delta_j \vert \forall y \in \Delta_i, (y,x) \in r_{ij}^{Q} \rightarrow y \in C^{\mathcal{I}_i}\}$
	$\forall_j P.D$	$\{x \in \Delta_j \vert \forall y \in \Delta_j, (x,y) \in P^{i\rightarrow j} \rightarrow y \in D^{\mathcal{I}_j}\}$
	$\forall_j P.E$	$\{x \in \Delta_j \vert \forall y \in \Delta_j, (x,y) \in P^{i\rightarrow j} \rightarrow y \in E^{k\rightarrow j}\}$
Existential Restriction	$\exists_j R.C$	$\{x \in \Delta_j \vert \exists y \in \Delta_j, (x,y) \in R^{\mathcal{I}_j}, y \in C^{i\rightarrow j}\}$
	$\exists_j Q^-.C$	$\{x \in \Delta_j \vert \exists y \in \Delta_i, (y,x) \in r_{ij}^{Q} \wedge y \in C^{\mathcal{I}_i}\}$
	$\exists_j P.D$	$\{x \in \Delta_j \vert \exists y \in \Delta_j, (x,y) \in P^{i\rightarrow j} \wedge y \in D^{\mathcal{I}_j}\}$
	$\exists_j P.E$	$\{x \in \Delta_j \vert \exists y \in \Delta_j, (x,y) \in P^{i\rightarrow j} \wedge y \in E^{i\rightarrow j}\}$
Number Restriction[1]	$\leq_j nR.C$	$\{x \in \Delta_j \vert \#(\{y \in \Delta_j \vert (x,y) \in R^{\mathcal{I}_j} \wedge y \in C^{i\rightarrow j}\}) \leq n\}$
	$\leq_j nQ^-.C$	$\{x \in \Delta_j \vert \#(\{y \in \Delta_i \vert (y,x) \in r_{ij}^{Q} \wedge y \in C^{\mathcal{I}_i}\}) \leq n\}$
	$\leq_j nP.D$	$\{x \in \Delta_j \vert \#(\{y \in \Delta_j \vert (x,y) \in P^{i\rightarrow j} \wedge y \in D^{\mathcal{I}_j}\}) \leq n\}$
	$\leq_j nP.E$	$\{x \in \Delta_j \vert \#(\{y \in \Delta_j \vert (x,y) \in P^{i\rightarrow j} \wedge y \in E^{i\rightarrow j}\}) \leq n\}$

`Role Constructors`		
Role Inverse	P^-	$\{(y,x) \in \Delta_j \times \Delta_j \vert (x,y) \in P^{i\rightarrow j}\}$
Transitive Role	$\mathsf{Trans}(P)$	$(P^{i\rightarrow j})^+ = P^{i\rightarrow j}$

`Semantic Relation Rules`		
Concept Subsumption	$C \sqsubseteq D$	$C^{i\rightarrow j} \subseteq D^{\mathcal{I}_j}$
	$C \sqsupseteq D$	$C^{i\rightarrow j} \supseteq D^{\mathcal{I}_j}$
Role Inclusion	$P \sqsubseteq R$	$P^{i\rightarrow j} \subseteq R^{\mathcal{I}_j}$
	$P \sqsupseteq R$	$P^{i\rightarrow j} \supseteq R^{\mathcal{I}_j}$
Nominal Correspondence	$\{x\} \rightarrow \{y\}$	$y \in r_{ij}^{\rightarrow}(x)$

[1] $\geq$ case is similar.

C is an i-concept, D is a j-concept, E is a k-concept; P is an i-role, R is a j-role, $Q \in \mathsf{rol}_{ij}$; x is an i-individual, y is a j-individual; $i \neq j$, $j \neq k$, i may be or may not be k. All formulae represent module j's point of view and constructed concepts (roles) are j-formulae.

need to use its inverse (Q^-) to construct a concept in L_j by universal, existential or number restrictions.

Note that arbitrary combination of the *possible* expressivity features in AMO may even lead to undecidability, since the union of multiple decidable logics may be undecidable (Baader et al., 2000). The design of a practical modular ontology language has to be a tradeoff between the expressivity and reasoning complexity.

3.3.4 Semantic Requirements of AMO

To precisely specify the semantic requirements of AMO, we need to answer several questions. First, *what are the logical consequences in an AMO*? How can local constraints in agents' local points of view influence each other? For example (fig. 3.10), if agent i thinks "a is b's best friend", and agent j thinks $i:a$ is x and $i:b$ is y in j's mind, can j infer the constraint that "x is y's best friend"? or a non-identical but compatible constraint such as "x knows y"? In other words, can knowledge in agent i's ontology *propagate* to agent j's ontology?

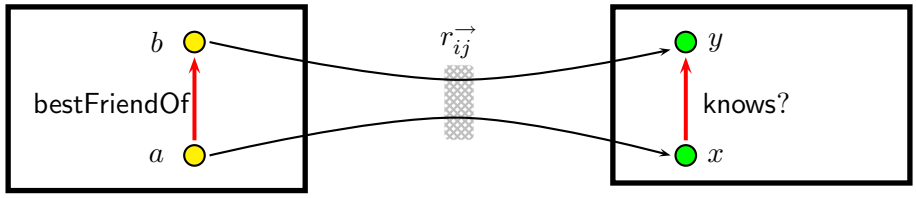
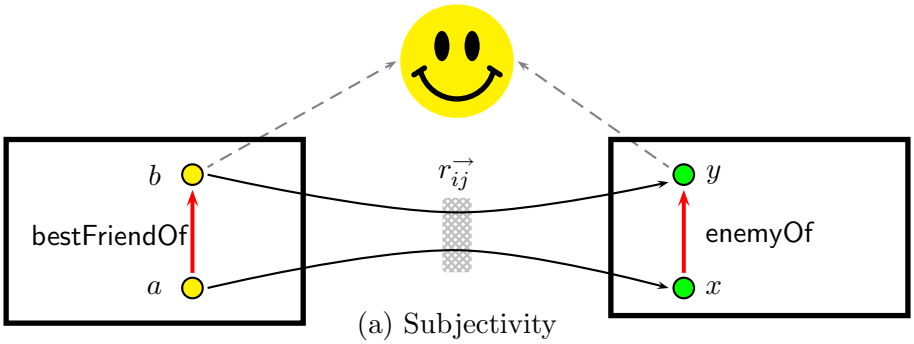


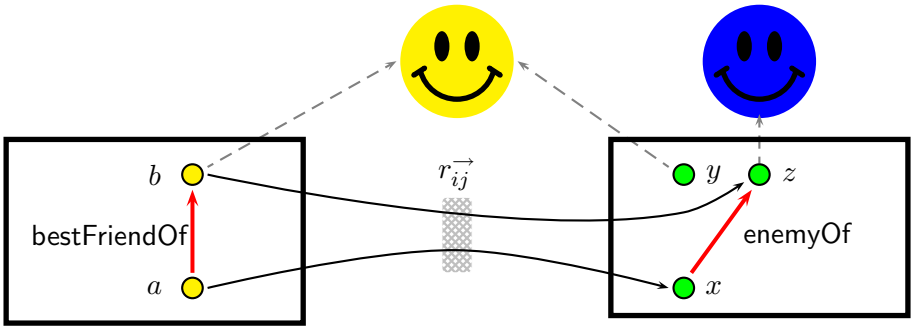
Figure 3.10 Propagation of Knowledge Among Agents

Second, *if there are inconsistencies in the points of view of two agents, what are the possible causes of such inconsistencies*? For example, if agent j holds the belief that "x is y's enemy", possible causes can be either i and j hold incompatible points of view while the image domain relations ($a \rightarrow x, b \rightarrow y$) are sound (fig. 3.11 (a)), or i and j actually hold compatible points of view but the image domain relations are wrong (e.g., j has mistaken z as y and names b as z locally, while z is x's enemy) (fig. 3.11 (b)). While the first type of inconsistency is hard to eliminate (subjectivity), are there principled ways to avoid the second type of inconsistency (miscommunication)?

Third, *if beliefs of agents are compatible, what is an "objective" way to integrate their knowledge*? How can we "restore" a description of the physical world that reflects the consensus



(a) Subjectivity



(b) Wrong Domain Relation (Miscommunication)

Figure 3.11 Two Types of Inconsistencies between Agents

among the agents, such that logical consequences are consistent in the *integrated description* and each local descriptions? For example (Fig. 3.12), if a person *Alice* (identified as $i : a$ and $j : x$) is observed as the best friend of another person *Bob* (identified as $i : b$ and $j : y$) by both i and j, how can we construct an "integrated" description that is acceptable by both i and j, such that if $i(j)$ asserts a conclusion (e.g. x is y's best friend), the "integrated" description can also confirm the conclusion?

Addressing such problems is critical in identifying and solving several semantic difficulties that arise in modular ontology languages. Next, we introduce some definitions that are useful in precisely stating the problems we informally outlined above.

Definition 3.1 (AMO Satisfiability) *Let* $M = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$ *be a model for an AMO* $O = \langle\{L_i\}, \{\mathfrak{R}_{ij}\}\rangle$*, where* $\mathcal{I}_i = \langle\Delta_i, (.)_i\rangle$ *is the local interpretation of* i *(*Δ_i *is the local domain of* i*,* $(.)_i$ *is the interpretation function of* i*) and* r_{ij} *denotes all domain relations between* $\mathcal{I}_i$ *and* $\mathcal{I}_j$*, including "image* $(\rightarrow)$*". We say that* M *satisfies* O*, denoted as* $M \models O$*, iff* $\mathcal{I}_i \models_c L_i$*, for all* i*, and* $M \models_c \mathfrak{R}_{ij}$*, for all* i *and* j*, where* $\models_c$ *stands for classic satisfiability. A concept* C *is*

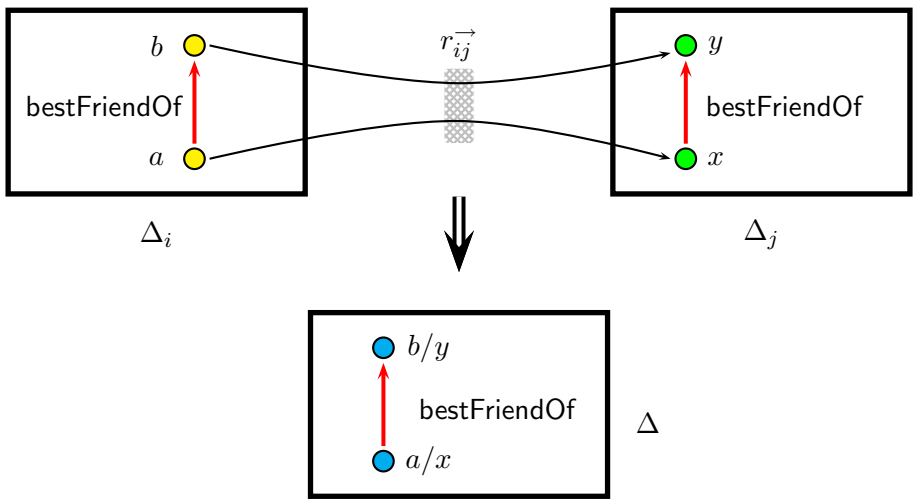


Figure 3.12 Agent Consensus

satisfiable in O witnessed by L_j if there is a model M of O such that $C^{\mathcal{I}_j} \neq \emptyset$.

Definition 3.2 (AMO Entailment) *An AMO O entails $C \sqsubseteq D$ witnessed by L_j, iff for any model $M = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$ of O, $\mathcal{I}_j \models_c C \sqsubseteq D$.*

Although the above definition only addresses intra-module subsumption, it can be easily extended to inter-module subsumption with a simple syntactical rewriting. If C is an i-concept, we can always create a j-concept C' interpreted as $C^{i \rightarrow j}$, and then $i : C \sqsubseteq j : D$ can be transformed as $j : C' \sqsubseteq j : D$.

We now precisely define the semantic requirements for modular ontologies given in section 3.2 with the following definitions:

Definition 3.3 (Localized and Globalized Semantics) *An AMO $O = \langle\{L_i\}, \{\mathfrak{R}_{ij}\}\rangle$ has only globalized semantics, iff for any model $M = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$ of O, $M \models O$, $\mathcal{I}_i = \langle\Delta_i, (.)_i\rangle$, local domains $\{\Delta_i\}$ of $\{L_i\}$ must be identical. Otherwise, it has localized semantics.*

Definition 3.4 (Decidability) *An AMO $O = \langle\{L_i\}, \{\mathfrak{R}_{ij}\}\rangle$ is decidable if for every concept satisfiability problem (therefore also entailment problem) C, there exists an algorithm that is capable of deciding in a finite number of steps whether there exists a model $M = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$, $M \models O$, such that C is satisfiable in $\mathcal{I}_i$.*

Directional Semantic Relations can be understood in the example: if j believe $i : a \rightarrow j : x, i : b \rightarrow j : y$ and x is y's best friend, it is not required i believes a is b's best friend. Formally, we have:

Definition 3.5 (Directionality) *Domain relations in an AMO are directional, iff for any model $M = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$ of O, for any $i \neq j$, the fact that $C^{i\rightarrow j} \subseteq D^{i\rightarrow j}$ is true in every $\mathcal{I}_j$ does not imply that $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$ must be true in every $\mathcal{I}_i$.*

Transitive reusability means that an agent can infer local constraints based on observing constraints in other agents' beliefs. For example, if i believes "a is b's best friend", and j believes domain relation $i : a \rightarrow j : x, i : b \rightarrow j : y$, then j may reuse i's knowledge and infer that "x is y's best friend". Furthermore, if another agent k who is confident in j's judgement, and believes $j : x \rightarrow k : p, j : y \rightarrow k : q$, then k also believes "p is q's best friend".

Definition 3.6 (Monotonicity and Transitive Reusability) *For an AMO $O = \langle\{L_i\}, \{\mathfrak{R}_{ij}\}\rangle$, L_i is said to be* reusable *by L_j $(j \neq i)$ if for any concepts C, D in L_i, such that $L_i \models C \sqsubseteq D$, we have that for any $M = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$ of O, $C^{i\rightarrow j} \subseteq D^{i\rightarrow j}$ must be true in $\mathcal{I}_j$. M is said to be monotone if for every $i \neq j$, L_i is reusable by L_j. L_i is said to be* transitively reusable *if for any j, k $(i \neq j \neq k)$, if L_i is reusable by L_j, and L_j is reusable by L_k, then we must have L_i is reusable by L_k.*

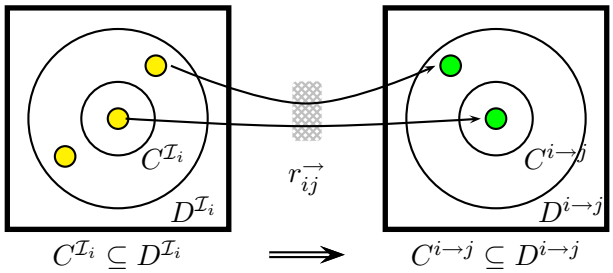


Figure 3.13 Reusability

The definition of reusability is illustrated in Fig. 3.13. Note that to ensure transitive reusability it is critical for an agent to have confidence in other agent's judgement and the sound communication (image domain relation) between agents. This requirement may enforce strong restrictions on possible image domain relations. We we will show in the next chapter that it is not always satisfiable in some modular ontology languages.

Exact reasoning means that the partial descriptions of all agents is consistent with the physical world in the sense that their combination is equivalent to the complete description of an agent with access to the combined knowledge base. Hence, compatible beliefs of agents may be combined such that the "merged belief" will be the consensus of individual agents. For example, if i believes x is the identifier of a person *Alice*, j believes a is the identifier of *Alice* and $i : x \rightarrow j : a$, then the merged state of the two agents will "believe" $i : x$ and $j : a$ are both identifiers of *Alice*. Thus, semantic relation rules, in their AMO form $i : \phi(x_1, ..., x_n) \rightarrow j : \psi(y_1, ..., y_n)$ (n=1 or 2), where x_i, y_i are connected by $r_{ij}^{\rightarrow}$, will be reduced to a normal FOL formula $\phi(x_1, ..., x_n) \rightarrow \psi(x_1, ..., x_n)$.

Definition 3.7 (Exact Reasoning) *Reasoning in an AMO* $O = \langle \{L_i\}, \{\mathfrak{R}_{ij}\} \rangle$ *is exact, iff for any model* $M = \langle \{\mathcal{I}_i\}, \{r_{ij}\} \rangle$ *of* O, $M \models \phi \Leftrightarrow M' \models_c \phi$ *for all concept formula* ϕ, *where* $M' = \langle \Delta^{\mathcal{I}}, (.)^{\mathcal{I}} \rangle$ *is a classical model obtained from* M *as the follows:*

- $\Delta^{\mathcal{I}} = \cup_i \Delta_i$
- *The assignment function* $(.)^{\mathcal{I}}$ *is defined as: for every (concept, role or nominal) name* $i : T$, $T^{\mathcal{I}} = T^{\mathcal{I}_i}$;
- *for every image domain relation, if* $\langle i : x, j : y \rangle \in r_{ij}^{\rightarrow}$, *add* $i : x = j : y$.
- *for every other domain relation* R, *if* $\langle i : x, j : y \rangle \in r_{ij}^{R}$, *add* $\langle x, y \rangle$ *to* $R^{\mathcal{I}}$.

3.4 Discussion and Related Work

There is a large body of literature on modularity in AI and knowledge representation (KR) in general, such as in intelligent agents (Bryson and Stein, 2001; Mukerjee and Mali, 2002), theorem proving (Plaisted and Yahya, 2003; Amir and McIlraith, 2000), logic programming (Bugliesi et al., 1994; Antoniou and Sperschneider, 1992; Antoniou, 1992), modal Logic (Herzig and Varzinczak, 2004), and relevant logic (Meghini and Straccia, 1996; Garson, 1989). We only focus here on the discussion of the *general framework of the semantics* of modular ontologies, instead of the *detailed specification* of module ontology languages (which will be discussed in Chapter 4).

3.4.1 Local Model Semantics and Distributed First Order Logic

The AMO framework presented here is strongly influenced by the Local Model Semantics (LMS) (Ghidini and Giunchiglia, 2001) framework and Distributed First Order Logic (DFOL) (Ghidini and Serafini, 1998) . The LMS framework is based on two principles:

- The principle of **locality**: an agent performs its local reasoning procedure only using locally available knowledge.
- The principle of **compatibility**: there is compatibility among the kinds of reasoning preformed in different agents.

More precisely, let $\{L_i\}_{i=1,...,n}$ be a family of languages, and let M_i to be the class of all the models of L_i, we call each $m \in M_i$ a *local model* of L_i. A compatibility sequence $\mathbf{c} = \langle \mathbf{c}_0, ..., \mathbf{c}_n \rangle$ is a subset of $M_1 \times ... \times M_n$. Hence, a compatibility sequence eliminates arbitrary co-existence of local models that are not compatible to each other.

In DFOL, compatibility sequences are syntactically represented as interschema constraints (or bridge rules) in the form of $i : \phi \rightarrow j : \psi$[5]. AMO semantic relation rules can be seen as restricted cases of DFOL interschema constraints. The difference between AMO semantic relation rules and general DFOL interschema constraints are as follows:

- Instead of assuming a single domain relation in DFOL interschema constraints, AMO semantic relation rules allow two ontologies to be connected by multiple domain relations.
- AMO semantic relation rules only allow unary or binary formulae on both sides of the rules, due to the fact that DL concepts and roles can be mapped to unary and binary predicates in FOL. Such a restriction is not required in general DFOL interschema constraints.
- AMO semantic relation rules are further obligated to satisfy semantic requirements of modular ontologies, such as decidability and knowledge reuse monotonicity, which are not required for general DFOL interschema constraints.

[5]There is also an extended version of interschema constraints of the form $i_1 : \phi_1, ..., i_n : \phi_n \rightarrow j : \psi$ to represent constraints among multiple languages (Serafini et al., 2005b).

3.4.2 Propositional Logic of Context

The Propositional Logic of Context (PLC) (Buvac and Mason, 1993; Buvac et al., 1995) is motivated to provide contextual description of knowledge. In PLC, a formula is always stated in a context and is represented by formulae of the form $ist(\kappa, \phi)$, where κ is a context and ϕ is a formula. Intuitively, it means that ϕ is true in the context of κ. This is interpreted using *possible world semantics*: $ist(\kappa, \phi)$ means that ϕ is true in every possible situation (i.e., truth assignment) of the context κ. Hence, PLC is in fact a multi modal language which extends propositional logic with the modality ist. A context κ_1 may be dependent on another context κ_2, which can be denoted by a context sequence $\kappa_1\kappa_2$. For example, if κ_1 represents Alice's belief and κ_2 represents Bob's belief, then $\kappa_1\kappa_2$ will represent that "Alice believes that Bob believes...".

It has been shown (Serafini and Bouquet, 2004) that PLC is multi-modal logic K restricted to a vocabulary (i.e., each context sequence is associated with a subset of atomic propositions) and extended with an axiom Δ:

$$ist(\kappa_1, ist(\kappa_2, \phi) \vee \psi) \rightarrow ist(\kappa_1, ist(\kappa_2, \phi)) \vee ist(\kappa_1, \psi)$$

Serafini and Bouquet (2004) have shown that PLC can be reduced to a multi-context system MPLC within the LMS framework. More precisely, for each context sequence $\overline{\kappa}$, the language $L_{\overline{\kappa}}$ is the smallest propositional language that contains all propositions in all context sequences and the atomic formula ϕ^{κ} (corresponding to $ist(\kappa, \phi)$) for any formula $\phi \in L_{\overline{\kappa}\kappa}$. The interschema constraints between $\{L_i\}$ (i stands for a context sequence) are:

$$\overline{\kappa} : \phi^{\kappa} \rightarrow \overline{\kappa}\kappa : \phi$$

$$\overline{\kappa}\kappa : \phi \rightarrow \overline{\kappa} : \phi^{\kappa}$$

$$\overline{\kappa}\kappa : \phi^{\kappa'} \rightarrow \overline{\kappa} : (\phi^{\kappa'})^{\kappa}$$

Bouquet and Serafini (2003) have shown that LMS with bridge rules can not be embedded in PLC. Thus, the LMS framework is more expressive than PLC.

It is easy to further reduce MPLC to the AMO framework presented above, such that propositions are concept names and interschema constraints are translated as semantic relation rules connected by the image domain relation. Note that the notion of $\overline{\kappa} : \phi^{\kappa}$ in AMO corresponds to the image of the concept ϕ in $L_{\overline{\kappa}\kappa}$, i.e., $\overline{\kappa} : \phi^{\kappa}$ will be interpreted as $\phi^{\overline{\kappa}\kappa \rightarrow \overline{\kappa}}$ in any model

of the knowledge base. It confirms the intuition of context sequences such that $\overline{\kappa} : \phi^{\kappa}$ is the observation of agent $\overline{\kappa}$ on the belief of ϕ of agent $\overline{\kappa}\kappa$. Hence, the AMO framework we introduced is adequate to capture the contextual properties of ontologies that are supported by PLC.

3.4.3 Epistemic Semantics for Peer-to-Peer Databases

A peer-to-peer (p2p) database system (Calvanese et al., 2003) contains a federation of autonomous local (peer) databases that are connected by a set of mapping rules. Each local database has a relational schema (possibly a mediated view over the real data). A p2p mapping rule is in the form of

$$q_1 \rightsquigarrow q_2$$

where q_1 (the tail of the rule) is a FOL formula over the *union of the alphabets of the peers* in the system, q_2 (the head of the rule) is a FOL formula over the *alphabet of a single peer*, and q_1 and q_2 are of the same arity.

The p2p data integration problem is closely related to the modular ontology problem if we regard the schema of each peer database as a local logic theory and the mapping rules between peers as the semantic relation rules between local theories.

The p2p mapping rules can be interpreted in the FOL semantics, as been studied in (Catarci and Lenzerini, 1993; Halevy et al., 2003; Koch, 2002) as :

$$\forall \mathbf{x}, (q_1(\mathbf{x}) \rightarrow q_2(\mathbf{x})) \tag{3.7}$$

where $\mathbf{x}$ is a tuple of free variables.

Calvanese et al. (2004) have argued that, instead of interpreting p2p mapping rules in the FOL semantics, it is better using epistemic semantics to interpret those mapping rules to ensure desirable properties of p2p systems, such as modularity, generality and decidability. The difference between the FOL semantics and the epistemic semantics is illustrated using the following example:

Example 3.3 : Suppose there are three knowledge bases: O_1 about student, O_2 about faculty, O_3 about payroll. The semantic relation rules ($\mathfrak{R}_1$) between the three knowledge bases are:

- Any Graduate in O_1 must be either a Research Assistant or a Teaching Assistant in O_2;
- Any Research Assistant in O_2 must have a Salary in O_3;

- Any Teaching Assistant in O_2 must have a Salary in O_3.

Suppose O_1 does not contain any information that whether a Graduate is a Research Assistant or Teaching Assistant in O_2. But, we may infer (in the classical first order semantics) from the given semantic relation rules that "any Graduate in O_1 must have a Salary in O_3".

However, suppose we have a different set of semantic relation rules ($\mathfrak{R}_2$):

- Anyone that is *known* to be a Graduate in O_1 must be either a Research Assistant or a Teaching Assistant in O_2;
- Anyone that is *known* to be a Research Assistant in O_2 must have a Salary in O_3;
- Anyone that is *known* to be a Teaching Assistant in O_2 must have a Salary in O_3.

The only difference between $\mathfrak{R}_1$ and $\mathfrak{R}_2$ is on that the latter is based on *epistemic* descriptions. Hence, with ($\mathfrak{R}_2$) as the semantic relation rules among $O_{\{1,2,3\}}$, we are not be able to infer whether an person is *known* to be a Research Assistant or Teaching Assistant, therefore can not infer that "anyone that is *known* to be Graduate in O_1 must have a Salary in O_3".

Formally, the epistemic interpretation of p2p mapping rules $q_1 \rightsquigarrow q_2$ is (Calvanese et al., 2003, 2004):

$$\forall \mathbf{x}, (\mathbf{K}q_1(\mathbf{x}) \rightarrow q_2(\mathbf{x}))$$

where $\mathbf{K}$ is the epistemic operator (Donini et al., 1992, 1998) that can be read as "it is known that...". $\mathbf{K}\alpha$ means the objects that are *known* to satisfy α, i.e., that satisfy α in all possible FOL models.

The formal semantics of the "$\mathbf{K}$" operator can be given in a simplified view of a Kripke structure (Kriple, 1962) of an S5 modal system (such that accessibility relations between possible worlds (models) are equivalence relations): an epistemic interpretation is a pair $(\mathcal{I}, \mathcal{W})$, where $\mathcal{W}$ is a set of FOL interpretations and $\mathcal{I} \in \mathcal{W}$; the models in $\mathcal{W}$ hold the *Common Domain Assumption* such that they share the same interpretation domain of individuals Δ. "$\mathbf{K}\alpha$" actually represents the *minimal knowledge* about α, such that $\mathbf{K}\alpha$ is satisfied in $(\mathcal{I}, \mathcal{W})$ by a tuple $\mathbf{x}$ iff $\alpha(\mathbf{x})$ is satisfied in *all* pairs $(\mathcal{J}, \mathcal{W})$ such that $\mathcal{J} \in \mathcal{W}$. We define the interpretation of $\mathbf{K}\alpha$ as

$$(\mathbf{K}\alpha)^{(\mathcal{I},\mathcal{W})} = \bigcap_{\mathcal{J}\in\mathcal{W}} \alpha^{(\mathcal{J},\mathcal{W})} \tag{3.8}$$

The main difference between the FOL semantics (3.7) and the epistemic semantics (3.8) of a mapping rules $q_1 \rightsquigarrow q_2$ is that: in epistemic semantics, not *all* tuples that satisfies q_1 will satisfy q_2, but only a *subset* of such tuples that are *known* to be q_1 will satisfy q_2. Hence, the epistemic semantics can be seen as a well-behaved approximation of the FOL semantics (Calvanese et al., 2004). For example, it can prevent the undecidability of query answering under the FOL semantics due to the interaction of inclusion dependencies, functional dependencies and cyclic mappings (Calvanese et al., 2004).

As shown in (Franconi et al., 2003), the epistemic semantics is equivalent to the context-based semantics of DFOL (Ghidini and Serafini, 1998). Indeed, different from the FOL semantics approach, in the context-based approach the overall system will not be reduced to a single flat theory (which may lead to the problems described in (Calvanese et al., 2004)), but to a set of local theories connected by interpretation constraints. The DFOL translation for the mapping rule $q_1 \rightsquigarrow q_2$ will be:

$$\forall \mathbf{x}, (q_1(\mathbf{x}) \xrightarrow{\{\langle x_i, y_i \rangle \in r_{12}^{\rightarrow}\}} q_2(\mathbf{y})) \tag{3.9}$$

Hence, not *all* tuples in the domain of L_1 that satisfies q_1 will be "mapped" (by the image domain relation) as q_2, but only a subset of them that has "images" in the domain of L_2 (i.e., the ones that agent 2 "knows") will be mapped. In that sense, an interpretation constraint (or a semantic relation rule in AMO) represents the *epistemic knowledge* about the "source" agent by the "target" agent. This is clearly demonstrated using the following example

Example 3.4 : Suppose we have an AMO with three modules and following semantic relation rules between them:

$$\begin{aligned} \forall x, \mathsf{Graduate}(x) &\xrightarrow{\{r_{12}^{\rightarrow}\}} \mathsf{ResearchAssistant}(x) \vee \mathsf{TeachingAssistant}(x) \\ \forall x, \mathsf{ResearchAssistant}(x) &\xrightarrow{\{r_{23}^{\rightarrow}\}} \mathsf{HasSalary}(x) \\ \forall x, \mathsf{TeachingAssistant}(x) &\xrightarrow{\{r_{23}^{\rightarrow}\}} \mathsf{HasSalary}(x) \end{aligned}$$

However, in resemblance to the epistemic semantics approach, we cannot infer from those rules that

$$\forall x, \mathsf{Graduate}(x) \xrightarrow{\{r_{13}^{\rightarrow}\}} \mathsf{HasSalary}(x)$$

A closer look at the interpretation constraints in DFOL and semantic relation rules in AMO reveals that they have strong resemblance to *rules*. In general, we may compare an AMO to

a hybrid systems of multiple DL theories connected by rules, which may not necessarily be reducible to a single description logic system. It is interesting to explore that whether modular ontologies in their general forms can be represented as an integration of description logics and rule-based systems, e.g., Horn Rules (Levy and Rousset, 1998), logic programming (Motik and Rosati, 2007; Motik et al., 2006) or MKNF (Minimal Knowledge, Negation as Failure) (Lifschitz, 1991). We believe that some of the results obtained in the study of combining DL and rules can be useful in the study of modular ontologies, e.g., the decidability and complexity of reasoning and query answering in modular ontologies.

3.4.4 Modular Ontologies based on Conservative Extensions

An alternative approach to modular ontologies is based on the notion of **conservative extensions** (Ghilardi et al., 2006; Grau et al., 2007, 2006a; Grau and Kutz, 2007) which allow ontology modules to be interpreted using the classical FOL semantics by requiring that they share the same global interpretation domain. The main intuition behind conservative extensions is to ensure *local completeness* of ontology modules, such that the combination of ontology modules will not alter the knowledge in any component module (Grau et al., 2006d). In particular, combining ontology modules cannot induce a new concept inclusion relation between existing concepts in any module. Hence, interaction between modules are controlled by requiring axioms in each module to have only "local" effects.

More precisely, if O is the union of a set of ontology modules $\{O_1, ..., O_n\}$, then we say O is a (deductive) conservative extension[6] of O_i if $O \models \alpha_i \Leftrightarrow O_i \models \alpha_i$ for any α_i of the form $C_1 \sqsubseteq C_2$, where C_1, C_2 are concepts in the language of O_i.

In general, to judge whether an extension is a conservative extension is a hard problem even for the DL $\mathcal{ALC}$ (decidable only in 2ExpTime-complete) (Grau et al., 2006a). It quickly becomes undecidable for the logic $\mathcal{ALCQIO}$ (Lutz et al., 2007). Nevertheless, there are practical methods to ensure conservativity of ontology extensions for a particular class of *local ontologies*. (Grau et al., 2007) have shown that if K_1 and K_2 are local ontologies, then $K_1 \cup K_2$ must be the conservative extension of K_i $(i = 1, 2)$, and the checking of ontology locality for $\mathcal{SHOIQ}$ can be

[6]There is also a related notion of model conservative extension (Lutz et al., 2007), such that O is a model conservative extension of O_1 if and only if every model of O_1 can be extended to a model of O. Model conservativity is strictly stronger than deductive conservativity. In what follows in this section, by default we only address deductive conservative extensions.

carried out in NExpTime time (Grau et al., 2007). There is also an syntactical approximation of locality which can be checked in polynomial time (Grau et al., 2007), which forbids the use of any axiom that is not "local" (e.g., $\top \sqsubseteq C$).

Both the conservative extension approach and our approach offers solutions to controlling unintended interactions among different ontology modules. However, the two approaches differ on whether keeping the conventional FOL semantics. The conservative extension approach forces a single global FOL interpretation domain for all ontology modules. However, it also prevents the different modules from interpreting axioms within their own local contexts since it in fact results in a single flat theory for all agents. Thus, the designers of different ontology modules have to play it safe by accounting for all possible contexts in which knowledge from a specific module might be reused. Locality of knowledge is achieved by precluding several otherwise useful modelling scenarios, such as the refining of relations between existing concepts in an ontology module and the reuse of nominals (Lutz et al., 2007). Furthermore, due to the requirement of a global model, reasoning with conservative extension based modular ontologies cannot benefit from the use of only local knowledge, which is one of the major intended advantage to having modular ontologies.

On the contrast, the contextualized semantics approach adopted by DFOL and AMO localizes the interpretation of reused knowledge. Locality of axioms in ontology modules is obtained "for free" by the *contextualized semantics*, thereby freeing ontology engineers from the burden of ensuring the reusability of an ontology module in contexts that are hard to foresee at the time of ontology construction. Thus, it is not necessary to explicitly restrict axioms in ontology modules for having only local effects.

CHAPTER 4. Package-based Description Logics

4.1 Overview

As noted by (Bouquet et al., 2003), ontologies on the semantic web need to satisfy two apparently conflicting objectives: *Sharing* or *r*euse of knowledge across autonomously developed ontologies, and accommodation of the *local points of view* or *contextuality* of knowledge, as we have illustrated by the examples in the previous chapter.

This chapter explores a formalism that can support *context-aware reuse* from multiple ontology modules. The resulting modular ontology language, package based description logics, allows the representation of ontology modules using components called packages. Each package consists of a set of highly related terms and relationships between them, with clearly defined access interfaces. In particular, it

- Allows each ontology module to use a subset of $\mathcal{SHOIQ}$, i.e., $\mathcal{ALC}$ augmented with transitive roles, role inclusion, role inversion, qualified number restriction and nominal concepts, hence covers a significant fragment of OWL-DL.

- Supports more flexible modeling scenarios than those supported by existing approaches, using a mechanism of *semantic importing* of names (including concept, role and nominal names) across ontology modules.

- Contextualizes the interpretation of reused knowledge. Locality of axioms in ontology modules is obtained "for free" by its *contextualized semantics*. A natural consequence of contextualized interpretation is that inferences that are drawn are always *from the point of view* of a witness package.

- Ensures that the result of reasoning is always the same as that obtained from a standard reasoner over an integrated ontology resulting from combining the relevant knowledge in

a context-specific manner. This ensures the *monotonicity* of inference in the distributed setting.

- Avoids many of the known reasoning difficulties of the existing approaches.

Part of this chapter was previously published in (Bao et al., 2006f,b,c; Bao and Honavar, 2006b,a; Bao et al., 2007e,c).

4.2 Package-based Description Logics

This section will introduce the basic features of the proposed language, package-based description logics (P-DL) $\mathcal{SHOIQP}$.

4.2.1 Syntax of P-DL

4.2.1.1 Packages

Informally, a package in $\mathcal{SHOIQP}$ can be viewed as a $\mathcal{SHOIQ}$ TBox and RBox. We define the *signature* $\mathsf{Sig}(P_i)$ of a package P_i as the set of names used in P_i. $\mathsf{Sig}(P_i)$ is the disjoint union of the set of concept names NC_i, the set of role names NR_i and the set of nominal names NI_i used in package P_i. The set of roles in P_i is defined as $\overline{\mathsf{NR}}_i = \mathsf{NR}_i \cup \{R^- | R \in \mathsf{NR}_i\}$ where R^- is the *inverse* of the role name R.

The signature $\mathsf{Sig}(P_i)$ of package P_i is divided into two disjoint parts: its local signature $\mathsf{Loc}(P_i)$ and its external signature $\mathsf{Ext}(P_i)$. For all $t \in \mathsf{Loc}(P_i)$, P_i is the *home package* of t, denoted by $P_i = \mathsf{Home}(t)$, and t is called an *i-name* (more specifically, an *i-concept name*, an *i-role name*, or an *i-nominal name*). We use *i-role* to refer to an i-role name or its *inverse*.

A role name $R \in \mathsf{NR}_i$ may be declared to be *transitive* in P_i using an axiom $\mathsf{Trans}_i(R)$. If R is declared transitive, R^- is also said to be *transitive*. We use $\mathsf{Tr}_i(R)$ to denote a role R being transitive in P_i.

A *role inclusion* axiom in P_i is an expression of the form $R \sqsubseteq S$, where R and S are i-roles. The *role hierarchy* for P_i is the set of all role inclusion axioms in P_i. The RBox $\mathcal{R}_i$ consists of the role hierarchy for P_i and the set of role transitivity declarations $\mathsf{Trans}_i(R)$. For a role hierarchy $\mathbf{R}$, if $R \sqsubseteq S \in \mathbf{R}$, then R is called a *sub-role* of S and S is called a *super-role* of R *w.r.t.* $\mathbf{R}$. An i-role is called *locally simple* if it not transitive nor has any transitive sub-role in P_i.

The set of $\mathcal{SHOIQP}$ concepts in P_i is defined inductively by the following grammar:

$$C := A|o|\neg_k C|C \sqcap C|C \sqcup C|\forall R.C|\exists R.C|(\leq nS.C)|(\geq nS.C)$$

where $A \in \mathsf{NC}_i$, $o \in \mathsf{NI}_i$, n is a finite non-negative integer, $R \in \overline{\mathsf{NR}}_i$, and $S \in \overline{\mathsf{NR}}_i$ is a locally simple role; $\neg_k C$ denotes the *contextualized negation* of concept C *w.r.t.* P_k. For any k and k-concept name C, $\top_k = \neg_k C \sqcup C$, and $\bot = \neg_k C \sqcap C$. Thus, there is no universal top ($\top$) concept or global negation ($\neg$). Instead, we have for each package P_k, a contextualized top $\top_k$ and a contextualized negation $\neg_k$. This allows a logical formula in P-DL (including $\mathcal{SHOIQP}$) to be interpreted within the context of a specific package.

A *general concept inclusion* (GCI) *axiom* in P_i is an expression of the form $C \sqsubseteq D$, where C, D are concepts in P_i. The TBox $\mathcal{T}_i$ of P_i is the set of all GCIs in P_i. Thus, formally, a *package* P_i is a pair $P_i := \langle \mathcal{T}_i, \mathcal{R}_i \rangle$. A $\mathcal{SHOIQP}$ ontology Σ is a set of packages $\{P_i\}$. We assume that every name used in a $\mathcal{SHOIQP}$ ontology Σ has a home package in Σ.

4.2.1.2 Semantic Importing between Packages

If a concept, role or nominal name $t \in \mathsf{Loc}(P_j) \cap \mathsf{Ext}(P_i)$, $i \neq j$, we say that P_i *imports* t and denote it as $P_j \xrightarrow{t} P_i$. We require that transitivity of roles be preserved under importing. Thus, if $P_j \xrightarrow{R} P_i$ where R is a j-role name, then $\mathsf{Trans}_i(R)$ iff $\mathsf{Trans}_j(R)$. If any local name of P_j is imported into P_i, we say that P_i imports P_j and denote it by $P_j \mapsto P_i$.

The *importing transitive closure* of a package P_i, denoted by P_i^+, is the set of all packages that are directly or indirectly imported by P_i. Let $P_i^* = \{P_i\} \cup P_i^+$. A $\mathcal{SHOIQP}$ ontology $\Sigma = \{P_i\}$ has an *acyclic importing relation* if, for all $i \neq j$, $P_j \in P_i^+ \rightarrow P_i \notin P_j^+$; otherwise, it has a *cyclic importing relation*.

A concept C is *understandable* by a package P_i if each concept and role name occurring in C has a home package in P_i^*.

We denote a Package-based Description Logic (P-DL) by adding the letter $\mathcal{P}$ to the notation for the corresponding DL. For example, $\mathcal{ALCP}$ is the package extension of the DL $\mathcal{ALC}$. We denote by $\mathcal{P}_\mathcal{C}$ a restricted type of P-DL that only allows importing of concept names. $\mathcal{P}^-$ denotes a P-DL with acyclic importing. In particular, $\mathcal{ALCP}_\mathcal{C}^-$ was studied in (Bao et al., 2006a), $\mathcal{ALCP}_\mathcal{C}$ was studied in (Bao et al., 2006e) and $\mathcal{SHOIQP}$ was studied in (Bao et al., 2007e).

4.2.1.3 Syntax Restrictions on Semantic Importing

Restrictions on Negations. We require that $\neg_k C$ (hence also $\top_k$) can appear in P_i, $i \neq k$, only if $P_k \mapsto P_i$ and all names in C are in $\mathsf{Sig}(P_k)$. Intuitively, this means that k-negation can only be used before a formula that is composed using names in P_k, and it can appear only in P_k or any package that directly imports P_k. Note that this also implies that $\mathsf{Sig}(C) \subseteq \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_k)$.

Restrictions on Imported Role Names. We require that *an imported role should not be used in role inclusion axioms*. This restriction is imposed because of two reasons. First, decidability requires that a role that is used in number restrictions be "globally" simple, i.e., that it has no transitive sub-role across any importing chain[1] (Horrocks et al., 1999). In practice, it is useful to restrict the use of imported roles in such a way that a role is globally simple iff it is locally simple. Second, a reduction of $\mathcal{SHOIQP}$ without such a restriction to an integrated ontology may require some features that are beyond the expressivity of $\mathcal{SHOIQ}$, such as role intersection. The decidability of $\mathcal{SHOIQP}$ with unrestricted use of imported role names still remains an open problem.[2]

4.2.2 Semantics

4.2.2.1 Local Semantics and Interpretation Constraints

A $\mathcal{SHOIQP}$ ontology has *localized semantics* in the sense that each package has its own local interpretation domain. Formally, for a $\mathcal{SHOIQP}$ ontology $\Sigma = \{P_i\}$, a *distributed interpretation* is a tuple $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^+} \rangle$, where $\mathcal{I}_i$ is a *local interpretation* of package P_i, with domain $\Delta^{\mathcal{I}_i}$, $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ is the *(image) domain relation* for the interpretation of the direct or indirect importing relation from P_i to P_j. For convenience, we use $r_{ii} = \mathrm{id}_{\Delta^{\mathcal{I}_i}} := \{(x,x) | x \in \Delta^{\mathcal{I}_i}\}$ to denote the identity mapping in the local domain $\Delta^{\mathcal{I}_i}$. Taking this convention into account, the distributed interpretation $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^+} \rangle$ may also be denoted by $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^*} \rangle$.

[1] This follows from the reduction from $\mathcal{SHOIQP}$ to $\mathcal{SHOIQ}$ given in Section 4.2.4.

[2] For some subsets of $\mathcal{SHOIQP}$, this restriction may be relaxed. For example, $\mathcal{ALCIOP}$ can be reduced to the DL $\mathcal{ALBO}$ (Schmidt and Tishkovsky, 2007) (extending $\mathcal{ALCO}$ with boolean role operators, inverse of roles and domain and range restriction operators), which is known to be decidable.

To facilitate our further discussion on interpretations, the following notational conventions will be used throughout. Given i, j, such that $P_i \in P_j^*$, for every $x \in \Delta^{\mathcal{I}_i}$, $A \subseteq \Delta^{\mathcal{I}_i}$ and $S \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$, define:

$$\begin{aligned} r_{ij}(A) &= \{y \in \Delta^{\mathcal{I}_j} | \exists x \in A, (x,y) \in r_{ij}\}, \\ r_{ij}(S) &= r_{ij} \circ S \circ r_{ij}^- \\ &= \{(z,w) \in \Delta^{\mathcal{I}_j} \times \Delta^{\mathcal{I}_j} | \exists (x,y) \in S, (x,z) \in r_{ij} \wedge (y,w) \in r_{ij}\}, \\ S(x) &= \{y \in \Delta^{\mathcal{I}_i} | (x,y) \in S\}. \end{aligned}$$

Moreover, let ρ be the equivalence relation on $\bigcup_i \Delta^{\mathcal{I}_i}$ generated by the collection of ordered pairs $\bigcup_{P_i \in P_j^*} r_{ij}$. This is the symmetric and transitive closure of the set $\bigcup_{P_i \in P_j^*} r_{ij}$. Define, for every i, j, $\rho_{ij} = \rho \cap (\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j})$.

Each of the local interpretations $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i} \rangle$ consists of a domain $\Delta^{\mathcal{I}_i}$, and the interpretation function $\cdot^{\mathcal{I}_i}$, which maps every concept name to a subset of $\Delta^{\mathcal{I}_i}$, every role name to a subset of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$ and every nominal name to an element in $\Delta^{\mathcal{I}_i}$, such that the following equations are satisfied, where R is a j-role, S is a locally simple j-role, C, D are concepts:

$$\begin{aligned} R^{\mathcal{I}_i} &= (R^{\mathcal{I}_i})^+, \text{ if } \mathsf{Trans}_i(R) \in \mathcal{R}_i \\ (R^-)^{\mathcal{I}_i} &= \{(x,y) | (y,x) \in R^{\mathcal{I}_i}\} \\ (C \sqcap D)^{\mathcal{I}_i} &= C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i} \\ (C \sqcup D)^{\mathcal{I}_i} &= C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i} \\ (\neg_j C)^{\mathcal{I}_i} &= r_{ji}(\Delta^{\mathcal{I}_j}) \backslash C^{\mathcal{I}_i}, \text{ for } \mathsf{Sig}(C) \subseteq \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_j) \\ (\exists R.C)^{\mathcal{I}_i} &= \{x \in r_{ji}(\Delta^{\mathcal{I}_j}) | \exists y \in \Delta^{\mathcal{I}_i}, (x,y) \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\} \\ (\forall R.C)^{\mathcal{I}_i} &= \{x \in r_{ji}(\Delta^{\mathcal{I}_j}) | \forall y \in \Delta^{\mathcal{I}_i}, (x,y) \in R^{\mathcal{I}_i} \rightarrow y \in C^{\mathcal{I}_i}\} \\ (\geqslant nS.C)^{\mathcal{I}_i} &= \{x \in r_{ji}(\Delta^{\mathcal{I}_j}) | \ |\{y \in \Delta^{\mathcal{I}_i} | (x,y) \in S^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\}| \geqslant n\} \\ (\leqslant nS.C)^{\mathcal{I}_i} &= \{x \in r_{ji}(\Delta^{\mathcal{I}_j}) | \ |\{y \in \Delta^{\mathcal{I}_i} | (x,y) \in S^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\}| \leqslant n\} \end{aligned}$$

Note that, when $i = j$, since $r_{ii} = \mathrm{id}_{\Delta^{\mathcal{I}_i}}$, $(\neg_j C)^{\mathcal{I}_i}$ reduces to the usual negation $(\neg_i C)^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \backslash C^{\mathcal{I}_i}$. Similarly, the other semantic definitions also reduce to the usual DL semantic definitions.

A local interpretation $\mathcal{I}_i$ *satisfies* a role inclusion axiom $R_1 \sqsubseteq R_2$ iff $R_1^{\mathcal{I}_i} \subseteq R_2^{\mathcal{I}_i}$ and a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. $\mathcal{I}_i$ is a *model* of P_i, denoted by $\mathcal{I}_i \vDash P_i$, if it satisfies all axioms in P_i.

The proposed semantics of $\mathcal{SHOIQP}$ is motivated by the need to overcome some of the limitations of existing approaches that can be traced back to the arbitrary construction of domain relations and the lack of support for contextualized interpretation. Specifically, we seek a semantics that satisfies the following desiderata:

- **Preservation of concept unsatisfiability**. The intuition is that an unsatisfiable concept expression can never be reused so as to be interpreted as a satisfiable concept. Formally, we say that a domain relation r_{ij} *preserves the unsatisfiability* of a concept C if whenever $C^{\mathcal{I}_i} = \emptyset$, it is necessarily the case that $C^{\mathcal{I}_j} = \emptyset$.

- **Transitive reusability of knowledge**. The intuition is that the consequences of some of the axioms defined in one module can be propagated in a transitive fashion to other ontology modules. For example, if a package P_i asserts that $C \sqsubseteq D$, and P_j directly or indirectly imports that axiom from P_i, then it should be the case that $C \sqsubseteq D$ is also valid from the *point of view* of P_j.

- **Contextualized interpretation of knowledge**. The intuition is that the interpretation of assertions in each ontology module is constrained by their context. When knowledge, e.g., axioms, in that module is reused by other modules, the interpretation of the reused knowledge should be constrained by the context in which the knowledge is being reused.

- **Improved expressivity**. In particular, the language should support

  1. both inter-module concept inclusion and concept construction using foreign concepts, roles and nominals;
  2. more general reuse of roles and of nominals than allowed by existing approaches.

A major goal of this chapter is to explore the constraints that need to be imposed on local interpretations so that the resulting semantics for $\mathcal{SHOIQP}$ satisfies the desiderata enumerated above. These constraints are presented in the following:

Definition 4.1 *An interpretation* $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^*} \rangle$ *is a* model *of a* $\mathcal{SHOIQP}$ *KB* $\Sigma = \{P_i\}$*, denoted as* $\mathcal{I} \models \Sigma$*, if the following conditions are satisfied.*

1. *For all* i, j, r_{ij} *is one-to-one, i.e., it is an injective partial function.*

2. Compositional Consistency*: For all i, j, k s.t. $i \neq j$, $P_i \in P_k^+$ and $P_k \in P_j^+$, we have* $\rho_{ij} = r_{ij} = r_{kj} \circ r_{ik}$.

3. *For every i-concept name C that appears in P_j, we have* $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$.

4. *For every i-role R that appears in P_j, we have* $R^{\mathcal{I}_j} = r_{ij}(R^{\mathcal{I}_i})$.

5. Cardinality Preservation for Roles*: For every i-role R that appears in P_j and every* $(x, x') \in r_{ij}$, $y \in R^{\mathcal{I}_i}(x)$ *iff* $r_{ij}(y) \in R^{\mathcal{I}_j}(x')$.

6. *For every i-nominal o that appears in P_j,* $(o^{\mathcal{I}_i}, o^{\mathcal{I}_j}) \in r_{ij}$.

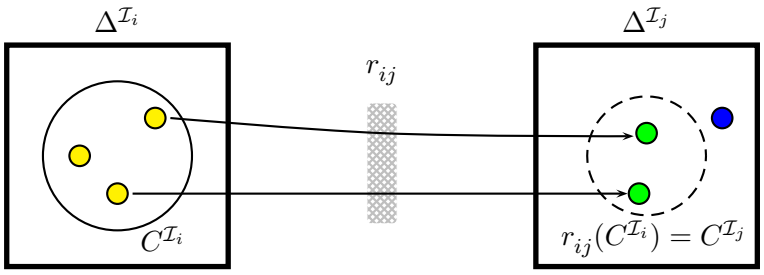
7. $\mathcal{I}_i \vDash P_i$, *for every i.*

The proposed semantics for $\mathcal{SHOIQP}$ is an extension of the semantics for $\mathcal{ALCP}_{\mathcal{C}}$ (Bao et al., 2006e), which uses Conditions 1,2,3 and 7 above, and of the semantics for Semantic Importing (Pan et al., 2006), which introduced Condition 5 above.

Intuitively, one-to-oneness (Condition 1, see Figure 4.1) and compositional consistency (Condition 2, Figure 4.2) ensure that the parts of local domains connected by domain relations match perfectly. Conditions 3 and 4 ensure consistency between the interpretations of concepts and of roles in their home package and the interpretations in the packages that import them. Condition 5 (Figure 4.3) ensures that r_{ij} is a total bijection from $R^{\mathcal{I}_i}(x)$ to $R^{\mathcal{I}_j}(r_{ij}(x))$; in particular, the sizes of $|R^{\mathcal{I}_i}(x)|$ and $|R^{\mathcal{I}_j}(r_{ij}(x))|$ are always the same in different local domains. Condition 6 ensures the uniqueness of nominals. In Section 4, we will show that Conditions 1-7 are minimally sufficient to guarantee that the desiderata for the semantics of $\mathcal{SHOIQP}$ as outlined above are indeed satisfied.

Note that Condition 2 implies that if P_i and P_j mutually (possibly indirectly) import one another, then $r_{ij} = \rho_{ij} = \rho_{ji}^- = r_{ji}^-$. However, if $P_j \notin P_i^*$, r_{ji} may not exist even if r_{ij} exists. Also note that r_{ij} need not be a total function.

Definition 4.2 *An ontology Σ is* consistent as witnessed by a package P_w *of Σ if P_w^* has a model $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^+} \rangle$, such that $\Delta^{\mathcal{I}_w} \neq \emptyset$. A concept C is* satisfiable as witnessed by P_w *if there is a model $\mathcal{I}$ of P_w^*, such that $C^{\mathcal{I}_w} \neq \emptyset$. A concept subsumption $C \sqsubseteq D$* is valid as witnessed by P_w*, denoted by $C \sqsubseteq_w D$, if for every model $\mathcal{I}$ of P_w^*, $C^{\mathcal{I}_w} \subseteq D^{\mathcal{I}_w}$.*



An image domain relation in P-DL is one-to-one, i.e., it is a partial injective function. It is not necessarily total, i.e., some individuals of $C^{\mathcal{I}_i}$ may not be mapped to $\Delta^{\mathcal{I}_j}$.

Figure 4.1 One-to-One Domain Relation

Hence, in $\mathcal{SHOIQP}$, the questions of consistency, satisfiability and subsumption problems are always answered from the local point of view of a *witness package*, and it is possible that different packages draw different conclusions from their own points of view.

As immediate consequences of the proposed semantics for the P-DL $\mathcal{SHOIQP}$, we have the following useful semantical equivalences, which extend various versions of the De Morgan's Law to $\mathcal{SHOIQP}$ (Proof is in the appendix).

Lemma 4.1 *Let $P_i \mapsto P_j$, C, D be concepts, R a k-role, such that $\mathsf{Sig}(C) \cup \mathsf{Sig}(D) \cup \{R\} \subseteq \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_j)$. Then, the following equalities hold from the point of view of P_j:*

1. $\neg_i C = \top_i \sqcap \neg_j C$;
2. $\neg_i (C \sqcap D) = \neg_i C \sqcup \neg_i D$;
3. $\neg_i (C \sqcup D) = \neg_i C \sqcap \neg_i D$;
4. $\neg_i (\exists R.C) = \neg_i \top_k \sqcup \forall R.\neg_j C$;
5. $\neg_i (\forall R.C) = \neg_i \top_k \sqcup \exists R.\neg_j C$;
6. $\neg_i (\leq nR.C) = \neg_i \top_k \sqcup \geq (n+1)R.C$;
7. $\neg_i (\geq (n+1)R.C) = \neg_i \top_k \sqcup \leq nR.C$.

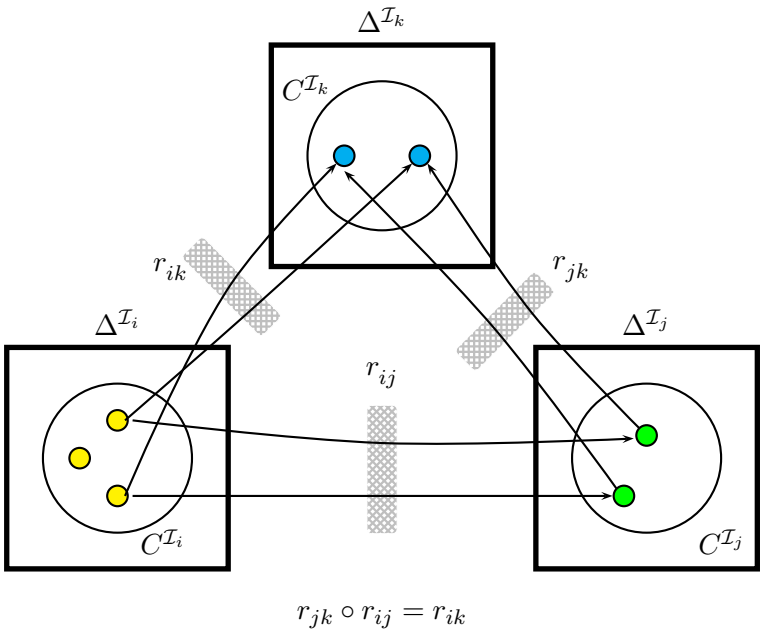


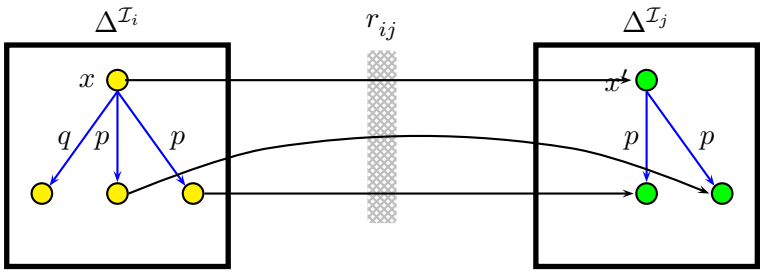
Figure 4.2 Compositionally Consistent Domain Relation

Note that when $i = j = k$, the equations in Lemma 4.1 reduce to the ordinary versions of De Morgan's Law in DL. These equations are helpful in simplifying proofs of other properties of $\mathcal{SHOIQP}$. Also note that, under the same hypotheses as those in Lemma 4.1,

$$\begin{aligned}
(\exists R.C)^{\mathcal{I}_j} &= \{x \in r_{kj}(\Delta^{\mathcal{I}_k}) | \exists y \in \Delta^{\mathcal{I}_j}, (x,y) \in R^{\mathcal{I}_j} \wedge y \in C^{\mathcal{I}_j}\} \\
&= \{x \in r_{kj}(\Delta^{\mathcal{I}_k}) |\ |\{y \in \Delta^{\mathcal{I}_j} | (x,y) \in R^{\mathcal{I}_j} \wedge y \in C^{\mathcal{I}_j}\}| \geq 1\} \\
&= (\geq 1 R.C)^{\mathcal{I}_j} \\
(\forall R.C)^{\mathcal{I}_j} &= \{x \in r_{kj}(\Delta^{\mathcal{I}_k}) | \forall y \in \Delta^{\mathcal{I}_j}, (x,y) \in R^{\mathcal{I}_j} \rightarrow y \in C^{\mathcal{I}_j}\} \\
&= \{x \in r_{kj}(\Delta^{\mathcal{I}_k}) |\ |\{y \in \Delta^{\mathcal{I}_j} | (x,y) \in R^{\mathcal{I}_j} \wedge y \notin C^{\mathcal{I}_j}\}| \leq 0\} \\
&= (\leq 0 R.\neg_j C)^{\mathcal{I}_j}
\end{aligned}$$

Hence, proofs involving existential restriction and value restriction may be reduced to those involving the corresponding number restrictions[3]. In what follows, we will only consider negation, conjunction and at-most number restriction as concept constructors since, as we have just pointed out, arguments for other constructors can be reduced to them.

[3]Note that R may not be a locally simple role in which case it cannot be used in number restrictions. However, the formulas above still allow us in practice to rephrase arguments involving existential restriction or universal restriction into corresponding arguments on number restrictions (for $n = 1$ or $n = 0$) regardless of the simplicity of R.



If an i-role p is imported by P_j, then every pair of p instances must have a "preimage" pair in Δ_i. The Cardinality Preservation for Roles, illustrated in this picture, requires that, if an individual x in $\Delta^{\mathcal{I}_i}$ has an image individual x' in $\Delta^{\mathcal{I}_j}$, then each of its p-neighbors must have an image in $\Delta^{\mathcal{I}_j}$ which is a p-neighbor of x'.

Figure 4.3 Cardinality Preservation for Roles

Next, we show that Condition 3 of Definition 5.1 holds not only for concept names, but, in fact, for arbitrary concepts (Proof is in the appendix). Lemma 4.2 will be used in Section 4 to show that the package description logic $\mathcal{SHOIQP}$ supports monotonicity of reasoning and transitive reusability of modules .

Lemma 4.2 *Let Σ be a $\mathcal{SHOIQP}$ ontology, P_i, P_j two packages in Σ such that $P_i \in P_j^+$, C a concept such that $\mathsf{Sig}(C) \subseteq \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_j)$, and R a role name such that $R \in \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_j)$. If $\mathcal{I} = \langle \{\mathcal{I}_u\}, \{r_{uv}\}_{P_u \in P_v^+} \rangle$ is a model of Σ, then $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$ and $r_{ij}(R^{\mathcal{I}_i}) = R^{\mathcal{I}_j}$.*

4.2.2.2 P-DL Semantics based on Partially-Overlapped Local Domains

A simplified version of the P-DL semantics may be given for the special case in which domain relations are interpreted as identity relations, i.e., $(x, y) \in r_{ij}$ iff $x = y \in \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j}$. Note that, in this case, local domains of ontology modules are in fact partially overlapping. The one-to-one and compositional consistency requirements are automatically satisfied by identity relations. Conditions 4 and 5 of Definition 5.1 may be unified so that the semantics in Definition 5.1 can be simplified as follows:

Definition 4.3 *An overlapping domain interpretation $\mathcal{I} = \{\mathcal{I}_i\}$ is a* model *of a $\mathcal{SHOIQP}$ ontology $\Sigma = \{P_i\}$ if the following conditions are satisfied.*

1. *For every i-concept name C that appears in P_j, $C^{\mathcal{I}_j} = C^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j}$.*

2. *For every i-role R that appears in* P_j, $R^{\mathcal{I}_j} = R^{\mathcal{I}_i} \cap (\Delta^{\mathcal{I}_j} \times \Delta^{\mathcal{I}_i}) = R^{\mathcal{I}_i} \cap (\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j})$.

3. *For every i-nominal name o that appears in* P_j, $o^{\mathcal{I}_i} = o^{\mathcal{I}_j}$.

4. $\mathcal{I}_i \models P_i$, *for every i.*

A similar semantics for terminology mappings between ontology modules based on partially overlapping domains is proposed in (Catarci and Lenzerini, 1993). In that framework, there is a global interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ and local domains $\Delta^{\mathcal{I}_i}$, such that each local domain $\Delta^{\mathcal{I}_i}$ is a subset of $\Delta^{\mathcal{I}}$ and two local domains may be partially overlapping. Mappings between concepts are of the following two forms:

$i : C \sqsubseteq_{ext} j : D$ (extensional inclusion), semantics: $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

$i : C \sqsubseteq_{int} j : D$ (intensional inclusion), semantics: $C^{\mathcal{I}} \cap \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} \subseteq D^{\mathcal{I}} \cap \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j}$

Intensional inclusions in (Catarci and Lenzerini, 1993) can be reduced to P-DL concept inclusions $C \sqsubseteq D \sqcap \top_i$ in package P_j, based on the following equations:

$$\begin{aligned} C^{\mathcal{I}} &= C^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i} \\ D^{\mathcal{I}} &= D^{\mathcal{I}_j} \subseteq \Delta^{\mathcal{I}_j} \\ C^{\mathcal{I}_j} &= C^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} = C^{\mathcal{I}} \cap \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} \\ D^{\mathcal{I}_j} \cap \Delta^{\mathcal{I}_i} &= D^{\mathcal{I}} \cap \Delta^{\mathcal{I}_j} \cap \Delta^{\mathcal{I}_i} \end{aligned}$$

Extensional inclusions, however, have no equivalent translation into P-DL. On the other hand, our approach is different from the approach in (Catarci and Lenzerini, 1993) in that extensional inclusions and intensional inclusions are not directional (under a global semantics), while semantic importing in P-DL is directional. Hence, the terminology mappings of (Catarci and Lenzerini, 1993) cannot, in general, be reduced to P-DL.

4.2.3 $\mathcal{SHOIQP}$ Examples

The semantic importing approach described here can model a broad range of scenarios that can also be modeled using existing approaches.

Example 4.1 :*Inter-module concept and role inclusions.* Suppose we have a *people* ontology P_1:

$$\begin{aligned} \top_1 &\sqsubseteq 1:\mathsf{Man} \sqcup 1:\mathsf{Woman} \\ 1:\mathsf{Boy} \sqcup 1:\mathsf{Girl} &\sqsubseteq 1:\mathsf{Child} \\ 1:\mathsf{Husband} &\sqsubseteq 1:\mathsf{Man} \sqcap \exists 1:\mathsf{marriedTo}.1:\mathsf{Woman} \end{aligned}$$

Suppose the *Work* ontology P_2 imports some of the knowledge from the *people* ontology:

$$2:\mathsf{FemaleEmployee} \sqsubseteq 2:\mathsf{Employee} \tag{4.1}$$

$$2:\mathsf{MaleEmployee} \sqsubseteq 2:\mathsf{Employee} \tag{4.2}$$

$$2:\mathsf{MaleEmployee} \sqsubseteq 1:\mathsf{Man} \tag{4.3}$$

$$2:\mathsf{FemaleEmployee} \sqsubseteq 1:\mathsf{Woman} \tag{4.4}$$

$$1:\mathsf{Child} \sqsubseteq \neg_2 2:\mathsf{Employee} \tag{4.5}$$

Axioms (4.3) and (4.4) model inter-module concept inclusions. This example also illustrates that the semantic importing approach can realize concept specialization ((4.3) and (4.4)) and generalization (4.5).

Example 4.2 :*Use of foreign roles or foreign concepts to construct local concepts.* Suppose a *marriage* ontology P_3 reuses the *people* ontology:

$$(= 1\ (1:\mathsf{marriedTo}).(1:\mathsf{Woman})) \sqsubseteq 3:\mathsf{Monogamist} \tag{4.6}$$

$$3:\mathsf{MarriedPerson} \sqsubseteq \forall(1:\mathsf{marriedTo}).(3:\mathsf{MarriedPerson}) \tag{4.7}$$

$$3:\mathsf{NuclearFamily} \sqsubseteq \forall(3:\mathsf{hasMember}).(1:\mathsf{Child}) \tag{4.8}$$

A complex concept in P_3 may be constructed using an imported role (4.7), an imported concept (4.8), or both an imported role and an imported concept (4.6).

Example 4.3 :*The use of nominals.* Suppose the *work* ontology P_2, defined above, is augmented with additional knowledge from a *calendar* ontology P_4, to obtain an augmented *work* ontology. Suppose P_4 contains the following axiom:

$$4{:}\mathsf{WeekDay} = \{4{:}Mon, 4{:}Tue, 4{:}Wed, 4{:}Thu, 4{:}Fri\},$$

where the nominals are shown in *italic* font. Suppose the new version of P_2 contains the following additional axioms:

$$\begin{aligned} 4 : Fri &\sqsubseteq \exists(2 : \mathsf{hasDressingCode}).(2 : \mathsf{CasualDress}) \\ \top_2 &\sqsubseteq \exists(2 : \mathsf{hasDressingCode}^{-}).(4 : \mathsf{WeekDay}) \end{aligned}$$

4.2.4 Reduction to Ordinary DL

A reduction $\Re$ from a $\mathcal{SHOIQP}$ KB $\Sigma_d = \{P_i\}$ to a $\mathcal{SHOIQ}$ KB Σ can be obtained as follows: the signature of Σ is the union of the local signatures of the component packages together with a global top $\top$, a global bottom $\bot$ and local top concepts $\top_i$, for all i, i.e., $\mathsf{Sig}(\Sigma) = \bigcup_i \mathsf{Loc}(P_i) \cup \{\top_i\} \cup \{\top, \bot\}$, and

a) For all i, j, k such that $i \neq j$, $P_i \in P_k^+$, $P_k \in P_j^+$, $\top_i \sqcap \top_j \sqsubseteq \top_k$ is added to Σ.

b) For each GCI $X \sqsubseteq Y$ in P_j, $\#_j(X) \sqsubseteq \#_j(Y)$ is added to Σ. The mapping $\#_j()$ is defined below.

c) For each role inclusion $X \sqsubseteq Y$ in P_j, $X \sqsubseteq Y$ is added to Σ.

d) For each i-concept name or i-nominal name C in P_i, $i : C \sqsubseteq \top_i$ is added to Σ.

e) For each i-role name R in P_i, $\top_i$ is stipulated to be its domain and range, i.e., $\top \sqsubseteq \forall R^{-}.\top_i$ and $\top \sqsubseteq \forall R.\top_i$ are added to Σ.

f) For each i-role name R in P_j, the following axioms are added to Σ:

   – $\exists R.\top_j \sqsubseteq \top_j$ (local domain);

   – $\exists R^{-}.\top_j \sqsubseteq \top_j$ (local range).

g) For each i-role name, add $\mathsf{Trans}(R)$ to Σ if $\mathsf{Trans}_i(R)$.

The mapping $\#_j()$ is adapted from a similar one for DDL (Borgida and Serafini, 2003) with modifications to facilitate context preservation whenever name importing occurs. For a formula X used in P_j, $\#_j(X)$ is:

- X, for a j-concept name or a j-nominal name.
- $X \sqcap \top_j$, for an i-concept name or an i-nominal name X.

- $\neg\#_j(Y) \sqcap \top_i \sqcap \top_j$, for $X = \neg_i Y$, where Y is a concept.
- $(\#_j(X_1) \oplus \#_j(X_2)) \sqcap \top_j$, for a concept $X = X_1 \oplus X_2$, where $\oplus = \sqcap$ or $\oplus = \sqcup$.
- $(\otimes R.\#_j(X')) \sqcap \top_i \sqcap \top_j$, for a concept $X = (\otimes R.X')$, where $\otimes \in \{\exists, \forall, \leq n, \geq n\}$ and R is an i-role.

For example, if C, D are concept names and R a role name,

$$\begin{aligned}
\#_j(j : (\neg_i i : C)) &= \neg(C \sqcap \top_j) \sqcap \top_i \sqcap \top_j \\
\#_j(j : (j : D \sqcup i : C)) &= (D \sqcup (C \sqcap \top_j)) \sqcap \top_j \\
\#_j(j : \forall(j : R).(i : C)) &= \forall R.(C \sqcap \top_j) \sqcap \top_j \\
\#_j(j : \exists(i : R).(i : C)) &= \exists R.(C \sqcap \top_j) \sqcap \top_i \sqcap \top_j
\end{aligned}$$

It should be noted that $\#_j()$ is *contextualized* so as to allow a given formula to have different interpretations when it appears in different packages.

4.2.5 Properties of Semantic Importing

In this section, we further justify the proposed semantics for $\mathcal{SHOIQP}$. More specifically, we present the main results showing that $\mathcal{SHOIQP}$ satisfies the desiderata given earlier (proofs are in appendix A.1).

The first main theorem shows that the consistency problem of a $\mathcal{SHOIQP}$ ontology can be reduced to a satisfiability problem of a $\mathcal{SHOIQ}$ concept w.r.t. the ontology obtained from the integration of all packages.

Theorem 4.1 *A $\mathcal{SHOIQP}$ KB Σ is consistent as witnessed by a package P_w if and only if $\top_w$ is satisfiable with respect to $\Re(P_w^*)$.*

Using Theorem 4.1 and the fact that concept satisfiability in $\mathcal{SHOIQ}$ is NExpTime-complete (Tobies, 2000, 2001), we can prove the following important consequence.

Theorem 4.2 *The concept satisfiability, concept subsumption and consistency problems in $\mathcal{SHOIQP}$ are* NExpTime*-complete.*

The next theorem shows that concept subsumption problems in $\mathcal{SHOIQP}$ can be reduced to concept subsumption problems in $\mathcal{SHOIQ}$.

Theorem 4.3 (Reasoning Exactness) *For a $\mathcal{SHOIQP}$ KB $\Sigma = \{P_i\}$, $C \sqsubseteq_j D$ iff $\Re(P_j^*) \models \#_j(C) \sqsubseteq \#_j(D)$.*

Proof: As usual, we reduce subsumption to (un)satisfiability. It follows directly from Theorem 4.1 that P_j^* and $C \sqcap \neg_i D$ have a common model if and only if $\Re(P_j^*)$ and $\#_j(C) \sqcap \neg \#_j(D)$ have a common model. Hence Theorem 4.3 holds. Q.E.D.

Discussion of Desiderata. To show that the package description logic $\mathcal{SHOIQP}$ supports transitive reusability and preservation of unsatisfiability, we prove the monotonicity of reasoning in $\mathcal{SHOIQP}$.

Theorem 4.4 (Monotonicity and Transitive Reusability) *For a $\mathcal{SHOIQP}$ KB $\Sigma = \{P_i\}$, if $P_i \in P_j^+$ and $C \sqsubseteq_i D$, then $C \sqsubseteq_j D$, where $\mathsf{Sig}(C) \cup \mathsf{Sig}(D) \subseteq \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_j)$.*

Proof: Suppose that $C \sqsubseteq_i D$. Thus, for every model $\mathcal{I}$ of P_i^*, $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. Now consider a model $\mathcal{I}$ of P_j^*. Since $P_i \in P_j^*$, $\mathcal{I}$ is also a model of P_i^*. Therefore, we have that $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. Hence, we obtain that $r_{ij}(C^{\mathcal{I}_i}) \subseteq r_{ij}(D^{\mathcal{I}_i})$, whence, by Lemma 4.2, $C^{\mathcal{I}_j} \subseteq D^{\mathcal{I}_j}$. This proves that $C \sqsubseteq_j D$. Q.E.D.

Theorem 4.4 ensures that when some part of an ontology module is reused, the restrictions asserted by it, e.g., domain restrictions on roles, will not be relaxed in a way that prohibits the reuse of imported knowledge. Theorem 4.4 also ensures that consequences of imported knowledge can be transitively propagated across importing chains.

In the special case where $D = \bot$, we obtain the following corollary:

Corollary 4.1 (Preservation of Unsatisfiability) *For a $\mathcal{SHOIQP}$ knowledge base $\Sigma = \{P_i\}$ and $P_i \in P_j^+$, if $C \sqsubseteq_i \bot$ then $C \sqsubseteq_j \bot$.*

Finally, the semantics of $\mathcal{SHOIQP}$ ensures that the interpretation of an axiom in an ontology module is constrained by its *context*, as seen from the reduction to a corresponding integrated ontology: $C \sqsubseteq D$ in P_j is mapped to $\#_j(C) \sqsubseteq \#_j(D)$, where $\#_j(C)$ and $\#_j(D)$ are now relativized to the corresponding local domain of P_j.

When a package P_i is directly or indirectly reused by another package P_j through name importing, some axioms in P_i may be effectively "propagated" to module P_j (i.e., may influence

inference from the point of view of P_j). P-DL semantics ensures that such axiom propagation will only affect the "overlapping" domain $r_{ij}(\Delta^{\mathcal{I}_i}) \cap \Delta^{\mathcal{I}_j}$, and not the entire domain $\Delta^{\mathcal{I}_j}$. Suppose package P_i contains an axiom $\neg_i \mathsf{Male} \sqsubseteq \mathsf{Female}$ and package P_j imports P_i. In this case, it is not required in P_j that $\top_j \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}$, since $r_{ij}(\Delta^{\mathcal{I}_i}) \subseteq \Delta^{\mathcal{I}_j}$, $\Delta^{\mathcal{I}_i} \backslash \mathsf{Male}^{\mathcal{I}_i} \subseteq \mathsf{Female}^{\mathcal{I}_i}$, i.e., $\Delta^{\mathcal{I}_i} = \mathsf{Male}^{\mathcal{I}_i} \cup \mathsf{Female}^{\mathcal{I}_i}$ does not necessarily imply $\Delta^{\mathcal{I}_j} = \mathsf{Male}^{\mathcal{I}_j} \cup \mathsf{Female}^{\mathcal{I}_j}$.

Hence, the effect of an axiom is always contextualized within its original designated context. Therefore, it is not necessary to explicitly restrict the use of the ontology language to ensure locality of axioms, as is required, for instance, by conservative extensions (Grau et al., 2007). Instead, the locality of axioms follows directly from the semantics of $\mathcal{SHOIQP}$.

4.2.6 Discussion on the P-DL Semantics

4.2.6.1 Necessity of P-DL Constraints on Domain Relations

The constraints on domain relations in the semantics of $\mathcal{SHOIQP}$ are minimal in the sense that if we drop any of them, we can no longer satisfy the desiderata summarized in Section 4.2.2.

Dropping Condition 1 of Definition 5.1 (one-to-one domain relation) leads to difficulties in preservation of concept unsatisfiability. For example, if the domain relations are not injective, then $C_1 \sqsubseteq_i \neg_i C_2$, i.e., $C_1 \sqcap C_2 \sqsubseteq_i \bot$, does not ensure $C_1 \sqcap C_2 \sqsubseteq_j \bot$. If the domain relations are not partial functions, multiple individuals in $\Delta^{\mathcal{I}_j}$ may be images of the same individual in $\Delta^{\mathcal{I}_i}$ via r_{ij}, whence unsatisfiability of a complex concept can no longer be preserved when both number restriction and role importing are allowed. For example, in such a case, if R is an i-role name and C is an i-concept name, $\geq 2R.C \sqsubseteq_i \bot$ does not imply $\geq 2R.C \sqsubseteq_j \bot$.

Dropping Condition 2 of Definition 5.1 (compositional consistency of domain relations) would result in violation of the transitive reusability requirement, in particular, and monotonicity of inference based on imported knowledge, in general. In the absence of compositional consistency of domain relations, the importing relations would be like bridge rules in DDL, in that they are localized w.r.t. the connected pairs of modules without supporting compositionality (Zimmermann and Euzenat, 2006).

In the absence of Conditions 3 and 4 of Definition 5.1, the reuse of concept and role names would only be syntactical, i.e., the local interpretations of shared concepts and role names would be independently determined resulting in no shared meaning.

Dropping Condition 5 (cardinality preservation of role instances), it would not be possible to ensure the consistency of local interpretations of complex concepts that use number restrictions. This consistency requirement ensures that the numbers of R-successors and R-predecessors of an individual are always the same as those in the interpretation corresponding to R's home package.

Condition 6 of Definition 5.1 is needed to ensure that nominals can only have one instance (which may be "copied" by multiple local interpretations associated by domain relations). Condition 7 is quite natural.

4.2.6.2 Contextualized Negation

Contextualized negation has been studied in logic programming (Polleres, 2006; Polleres et al., 2006). Existing modular ontology languages DDL and $\mathcal{E}$-Connections do not explicitly support contextualized negation in their respective syntax. In fact, in those formalisms, a negation is always interpreted with respect to the local domain of the module in which the negation occurs, not the union of all local domains. Thus, both DDL and $\mathcal{E}$-Connections in fact implicitly support the use of i-negation in the module i.

The P-DL syntax and semantics, proposed in this work, supports a more general use of contextualized negation so that a package can use the negations of its imported packages[4]. Hence, a statement can always be explicitly asserted in a specific context with the same semantic effect, regardless of where it is syntactically located. This allows some flexibility in refining existing modules by new modules that import them and prevents possible ambiguities in the contextual meaning of axioms.

4.2.6.3 Directionality of Importing

It has been claimed (Grau and Kutz, 2007) that the importing relation in P-DL is non-directional and that, therefore, it can be mapped to an imports-free semantics. More precisely, it has been claimed that a P-DL model $\mathcal{I}$ satisfies $r_{ij}(s^{\mathcal{I}_i}) = s^{\mathcal{I}_j}$ if only if it satisfies $r_{ji}(s^{\mathcal{I}_j}) = s^{\mathcal{I}_i}$, for any symbol s such that $P_i \xrightarrow{s} P_j$ (Definition 18 and Proposition 19 in (Grau and Kutz, 2007)). Hence, the claim goes, a P-DL ontology can be equivalently reduced to an imports-free ontology where a shared symbol s of P_i and P_j always has the same interpretation from the point of

[4]We thank Jeff Pan for discussions on this issue.

view of both P_i and P_j, i.e., $s^{\mathcal{I}_i} = s^{\mathcal{I}_j}$. However, such an observation is based on an incorrect understanding of the P-DL semantics. This deviation from the original P-DL semantics, as presented in (Bao et al., 2006b,c), is due to an erroneous interpretation of the following three points:

- The P-DL semantics does not require the existence of both r_{ij} and r_{ji}. Their joint existence is only required when P_i and P_j mutually import one another. Hence, even if $r_{ij}(s^{\mathcal{I}_i}) = s^{\mathcal{I}_j}$, it is possible that r_{ji} does not even exist, in which case the expression $r_{ji}(s^{\mathcal{I}_j})$ is meaningless.

- Domain relations are not total functions, but only partial functions. Hence, it is not required that *every* individual of $s^{\mathcal{I}_i}$ be mapped (by the one-to-one domain relation r_{ij}) to an individual of $s^{\mathcal{I}_j}$.

- Satisfiability and consistency have only contextualized meaning in P-DL. If P_j is not in P_i^*, the transitive importing closure of P_i, then models of P_i^* are independent from P_j. This is made clear in Definition 5.2 where satisfiability and consistency are always considered against a witness package.

4.2.6.4 P-DL Consistency and TBox Consistency

It has been argued that P-DL consistency can be reduced to TBox consistency in classic DL (Grau and Kutz, 2007)[5]. This is, indeed, possible by applying the reduction from P-DL to DL as presented in Section 4.2.4 and Theorem 4.1. Therefore, the argument goes, P-DL, similarly with the other two major modular ontology languages, DDL and $\mathcal{E}$-Connections, does not increase expressivity as compared to conventional DL languages. However, the reduction from P-DL to DL based on S-Compatibility, as proposed in (Grau and Kutz, 2007), is incorrect. Given two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ in the same DL and S, the shared signature of them:

> "...that they ($\mathcal{T}_1$ and $\mathcal{T}_2$) are S-Compatible if we can find an interpretation for the symbols in S that can be extended to both a model to $\mathcal{T}_1$ and a model to $\mathcal{T}_2$ by interpreting the additional predicates and possibly adding new elements to the interpretation domain." (Grau and Kutz, 2007)

[5] In (Grau and Kutz, 2007) the consistency of a P-DL ontology is called the satisfiability of the ontology, i.e., the condition that the ontology has a model.

Formally S-Compatibility is defined in (Grau and Kutz, 2007) as follows:

Definition 4.4 (Expansion) *An S-interpretation $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ is an expansion of an S'-interpretation $\mathcal{J}' = (\Delta^{\mathcal{J}'}, \cdot^{\mathcal{J}'})$ if 1) $S' \subseteq S$, 2) $\Delta^{\mathcal{J}'} \subseteq \Delta^{\mathcal{J}}$, and 3) $s^{\mathcal{J}} = s^{\mathcal{J}'}$ for every $s \in S'$.*

Definition 4.5 (S-compatibility) *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes expressed in a description logic $\mathcal{L}$, and let S be the shared signature of them. We say that $\mathcal{T}_1$ and $\mathcal{T}_2$ are S-compatible if there exists a model $\mathcal{J}_S$ of the symbols in S that can be expanded to a model $\mathcal{J}_1$ of $\mathcal{T}_1$ and to a model $\mathcal{J}_2$ of $\mathcal{T}_2$.*

It is claimed that for any TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ in $\mathcal{SHIQ}$, an imports-free ontology $O = \{\mathcal{T}_1, \mathcal{T}_2\}$ is consistent (i.e., has a model) iff $\mathcal{T}_1$ and $\mathcal{T}_2$ are S-compatible for the shared signature S (Corollary 29 (Grau and Kutz, 2007)). However, since a P-DL ontology is not always reducible to an imports-free ontology as the simple union of all modules (packages), a P-DL ontology consistency problem may not be reducible to an S-compatibility problem, as shown by the following example.

Example 4.4 : Let $\mathcal{T}_1 = \{D \sqcup \neg D \sqsubseteq C\}$, $\mathcal{T}_2 = \{C \sqsubseteq \bot\}$. The shared signature S is $\{C\}$ and $\mathcal{T}_1$ and $\mathcal{T}_2$ are not S-compatible. However, suppose we have a P-DL ontology such that $\mathcal{T}_1 \xrightarrow{C} \mathcal{T}_2$ and negation in $\mathcal{T}_1$ becomes contextualized negation $\neg_1$, then we have a model:

$$\begin{aligned} \Delta_1 &= C^{\mathcal{I}_1} = D^{\mathcal{I}_1} = \{x\} \\ \Delta_2 &= \{y\} \\ r_{12} &= r_{21} = \emptyset \end{aligned}$$

On the other hand, a P-DL ontology such that $\mathcal{T}_2 \xrightarrow{C} \mathcal{T}_1$ only has models with empty Δ_1. This example also demonstrates that P-DL importing is directional.

The next example shows that when we have nominals, we cannot reduce P-DL consistency problem[6] to an imports-free ontology as the simple union of all packages without preserving the contextuality of P-DL axioms.

Example 4.5 Use of Nominals: Let us consider the following TBoxes:

$$\begin{aligned} \mathcal{T}_1 &= \{\top \sqsubseteq i \sqcup j, \quad i \sqcap j \sqsubseteq \bot\} \\ \mathcal{T}_2 &= \{\top \sqsubseteq i\} \end{aligned}$$

[6]It is decidable with nominal importing according to Theorem 4.2.

where i, j are nominals, with the shared signature $S = \{i\}$. $\mathcal{T}_1$ and $\mathcal{T}_2$ are S-compatible but $\mathcal{T}_1 \cup \mathcal{T}_2$ is not consistent. Suppose we have a P-DL ontology with $\mathcal{T}_1 \xrightarrow{i} \mathcal{T}_2$. Since "$\top$" only has contextualized meaning in P-DL, these TBoxes in fact should be represented as

$$\begin{aligned} \mathcal{T}_1 &= \{\top_1 \sqsubseteq i \sqcup j, \quad i \sqcap j \sqsubseteq \bot\} \\ \mathcal{T}_2 &= \{\top_2 \sqsubseteq i\} \end{aligned}$$

Then, there exists a model for this P-DL ontology:

$$\begin{aligned} \Delta_1 &= \{x, y\}, i^{\mathcal{I}_1} = \{x\}, j^{\mathcal{I}_1} = \{y\} \\ \Delta_2 &= \{x'\}, i^{\mathcal{I}_2} = \{x'\} \\ r_{12} &= \{\langle x, x' \rangle\} \end{aligned}$$

In general, the reduction from P-DL modules to imports-free TBoxes with shared signatures, as suggested by (Grau and Kutz, 2007), does not preserve the semantics of P-DL. There is a fundamental difference between the two formalisms: in P-DL there is no universal top concept and, as a result, axioms only have localized effects while, in the imports-free TBoxes, concepts are not contextualized in their definitive domains and axioms in a module may interact freely. This is made clear in the reduction presented in Section 4.2.4.

In (Grau and Kutz, 2007) an alternative reasoning strategy for P-DL is suggested: reasoning about a P-DL ontology with modules $\{\mathcal{T}_i\}$ can be reduced to standard DL reasoning over the union of all ontology modules $\mathcal{T} = \mathcal{T}_1 \cup ... \cup \mathcal{T}_n$. Such a reduction is in general not correct because of the difference between the P-DL contextualized semantics and the conventional semantics of reasoning in the union of all modules.

Nevertheless, in the previous section we have shown that such a reduction from P-DL reasoning to DL reasoning is possible when we adopt the correct translation from P-DL to DL (i.e., the reduction process from P-DL modules to an ordinary DL). However, in general, it is not desirable to perform reasoning with P-DL using such a reduction, since this process requires the integration of all ontology modules, which is typically costly, and in many cases (e.g., in peer-to-peer applications) practically impossible. In fact, the suggested integration-based reasoning counteracts many of the motivating reasons for introducing modular ontologies, such as scalability and reasoning with local knowledge. Hence, instead of using standard DL reasoners, major modular ontology languages typically resort to specialized reasoning algorithms, e.g., for

$\mathcal{E}$-connections see (Grau et al., 2004b,a), for DDL (Serafini and Tamilin, 2004; Serafini et al., 2005a) and for P-DL (Bao et al., 2006e).

4.3 Adopt OWL as the Syntax for P-DL

4.3.1 Limitations of OWL Importing

Using name importing to connect multiple ontologies has been explored in the literature. Ontolingua (Fikes et al., 1997) is one of the first available tools that support reusing existing ontology modules[7] when designing new modules. It allows an ontology module to "include" another ontology module. Such an inclusion has the following features:

- It imports *both vocabulary and axioms*. When an ontology module A is included by another module B, not only terms defined in A, but also axioms of A, are translated into B.
- It is *transitive*. If a module A includes a module B, and B includes C, then A indirectly includes C.
- It reuses an ontology in its *entirety*. If a module A is reused, then all of its terms and axioms will be reused, even if the referring modules refers only a single symbol in A. Thus, there lacks "a more refined rule could include only those axioms that could affect the possible interpretations of the symbol"(Fikes et al., 1997).

The Ontolingua ontology inclusion mechanism will result in a union of all ontology modules in the inclusion transitive closure. Thus, if an assertion is satisfied by a module A, it will also be satisfied by any other module that directly or indirectly includes A.

OWL (Schreiber and Dean, 2004) provides a similar mechanism, the `owl:imports` construct, that allows multiple OWL ontologies being connected into a larger ontology. OWL Semantics and Abstract Syntax (Patel-Schneider et al., 2003) specifies that:

- "If an ontology imports another ontology, the axioms in the imported ontology (and any ontologies it imports, and so on) can be used for these purposes." (Section 2. Abstract Syntax)

[7] In what follows, we do not distinguish the use of "ontology" and "ontology module" since a single ontology can also be used as a module to construct another ontology.

- For any OWL ontology O and an abstract OWL interpretation I of O, "I satisfies each ontology mentioned in an `owl:imports` annotation directive of O" (Section 3. Direct Model-Theoretic Semantics)

Thus, OWL importing, similar to Ontolingua inclusions, provides a *syntactical* solution to connect and integrate ontologies staying in different OWL files into one ontology. However, such an approach is unsatisfactory in many aspects for modeling and reasoning with modular ontologies. Indeed, `owl:imports` may be one of the most debated features of OWL during and ever since the specification of the language[8]. Controversies about `owl:imports` are mainly focused on the follows problems:

- It does not provide *localized semantics*. (Bouquet et al., 2003) argued that OWL only provides a *global semantics* for an ontology module that satisfies *all* axioms and facts in *all* (directly and indirectly) imported modules of that module. Thus, from the logical point of view, if a module A imports module B, it is equivalent to copying all statements of B into A. On the contrast, a localized semantics (Bouquet et al., 2003; Bao et al., 2007e) will allow different ontology modules to have loosely coupled local interpretations so that statements in a module can be retained in their context, thus it will not require a complete consensus among involved ontology modules and can support reasoning without a forced integration of all ontology modules (while current OWL semantics implies such an integration for reasoning).

- It does not support *partial ontology reuse*. OWL provides no support for organizational granularity inside an ontology module, thus an ontology designer are faced with the all-or-nothing choice when reuses a module. That may cause problems when a user wants to customize an existing ontology module for partial reuse, or when the imported ontology module is very large. Therefore, it is heavily relied on developer's discipline to avoid unmanageable blow up of ontology size (Grau et al., 2004b).

Consequently, there are growing interests in extending OWL to support modular ontologies, including several *syntactical* extensions to OWL, e.g., dOWL (Avery and Yearwood, 2003), C-OWL (Bouquet et al., 2003) and $\mathcal{E}$-Connections (Grau et al., 2004b). However, existing

[8]Many of those discussions are archived in the email list of the W3C Web-Ont Working Group (`http://lists.w3.org/Archives/Public/www-webont-wg/`), 2002-2003

approaches are limited in expressivity and may present reasoning difficulties (which will be discussed in details in the next section).

Against this background, we argue for a *semantic* extension to OWL, i.e., a new interpretation of OWL grounded in *modular semantics* that allows the connected ontology modules to have contextualized local interpretations, which may overcome some of the limitations of existing approaches to linking OWL ontology modules. We also argue for restrictions on the use of OWL syntax for supporting syntactical partial reuse.

4.3.2 A Modular Semantics for OWL

Instead of introducing syntactic extensions to OWL, we propose a new semantics for `owl:imports` that provides direct support for modular ontologies. The basic intuition is that, instead of treating `owl:imports` connected OWL ontology modules as a single flat DL theory, we may treat them as a P-DL theory with each module as a package. Hence, `owl:imports` will correspond to the semantic importing in P-DL $\mathcal{SHOINP}(D)$[9].

Formally, a modular semantics for OWL is specified as:

Definition 4.6 (Modular OWL Interpretation) . *A modular OWL interpretation for a set of OWL ontology modules $O = \{O_i\}$ is a interpretation of a P-DL O' obtained from O in the following way: 1)* `owl:Thing` *in O_i is translated as $\top_i$; 2)* `owl:complementOf` *in O_i is translated as $\neg_i$; 3) $O_i \xrightarrow{t} O_j$ if O_j imports O_i and refers t where t is a term in the name space of O_i; 4) Other OWL constructs are translated into P-DL formulae accordingly.*

Hence, a modular OWL Interpretation is a pair $\mathcal{I} = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$, where each $\mathcal{I}_i = \langle\Delta_i, (.)^{\mathcal{I}_i}\rangle$ is the local interpretation of O_i. According to Definition 5.1, for any term importing relation $O_i \xrightarrow{t} O_j$, we have $r_{ij}(t^{\mathcal{I}_i}) = t^{\mathcal{I}_j}$.

The resulting semantics differs from the current OWL semantics in that it does not require the complete overlapping of local domains, i.e., $\Delta_i = \Delta_j$, for any i, j. Thus, it offers a *selective* importing mechanism that allows the parts of an imported ontology module (that are selected for reuse by another module) to "share" their interpretations with that module, whereas the other parts of the ontology (i.e., those that are not selected for reuse) may retain their local

[9]Note that the discussion on $\mathcal{SHOIQP}$ in the last section can be easily extended to the case with datatype support, i.e., $\mathcal{SHOIQP}(D)$. $\mathcal{SHOINP}(D)$ is obtained from $\mathcal{SHOIQP}(D)$ by allowing number restriction but not qualified number restriction.

interpretations. As we have shown in the last section, such a semantics allows knowledge sharing with localized semantics and the preservation of knowledge context.

Our proposal differs from existing approaches in several respects (which will be discussed in details in the next section):

1. Improved expressivity. For example, it supports both inter-module class subsumptions (e.g. $i : C \sqsubseteq j : D$) and inter-module role relations (e.g. $i : C \sqsubseteq \exists i(: r).(j : D)$).

2. The avoidance of associated reasoning difficulties in existing approaches.

Example 4.6 : Consider two ontology modules capturing knowledge about wine and food, respectively. Suppose the food module contains the following terms and axioms:

$$\mathsf{food : Apple} \sqsubseteq \mathsf{food : Fruit} \tag{4.9}$$

$$\mathsf{food : Grape} \sqsubseteq \mathsf{food : Fruit} \tag{4.10}$$

Suppose the wine module imports the food module and contains axioms:

$$\mathsf{wine : WineGrape} \sqsubseteq \mathsf{food : Grape} \tag{4.11}$$

$$\mathsf{wine : Wine} \sqsubseteq \exists \mathsf{wine : madeFrom.(food : Grape)} \tag{4.12}$$

An interpretation of the ontology contains two local interpretations:

- $\mathcal{I}_1$: $\mathsf{food : Fruit}^{\mathcal{I}_1} = \{\mathsf{x}_1, \mathsf{x}_2\}, \mathsf{food : Apple}^{\mathcal{I}_1} = \{\mathsf{x}_1\}, \mathsf{food : Grape}^{\mathcal{I}_1} = \{\mathsf{x}_2\}$.
- $\mathcal{I}_2$: $\mathsf{food : Grape}^{\mathcal{I}_2} = \mathsf{wine : WineGrape}^{\mathcal{I}_2} = \{\mathsf{y}_1\}$, $\mathsf{wine : Wine}^{\mathcal{I}_2} = \{\mathsf{y}_2\}$, $\mathsf{wine : madeFrom}^{\mathcal{I}_2} = \{\langle \mathsf{y}_2, \mathsf{y}_1 \rangle\}$.
- $r_{12} = \{\langle x_2, y_1 \rangle\}$.

Hence, the importing relation from food to wine is *not* complete, but *partial* in that only the terms selected for reuse (e.g., food:Grape) are interpreted in the shared part of local domains ($\{x_2\}$ in the example), whereas the terms that are *not* selected for reuse (e.g., food:Apple) retain their *local* interpretations.

The example also shows that the semantic importing strategy supports both inter-module class subsumptions (e.g., axiom 4.11) and inter-module role relations (e.g., axiom 4.12).

Hence, OWL, based on the proposed modular semantics, can be adapted as the syntax for P-DL. Because such an approach requires no new syntactic extensions to OWL, it ensures backward compatibility of the resulting modular OWL ontologies to many of the existing tools, for instance, ontology editors, parsers, visualizers and database storage. Such a backward compatibility may significantly reduce the cost in adopting modular ontologies. In addition, as we will discuss in the next chapter, it also supports distributed reasoning on OWL ontologies.

4.3.3 A Modular Syntax for OWL

OWL lacks the support for partial ontology reuse partly due to no syntactical modularity inside an ontology. For example, one might state the following OWL axioms in an ontology O (in abstract syntax)

$$\mathsf{Class}(\mathsf{Cat\ partial\ intersectionOf}(\mathsf{Animal}, \mathsf{restriction}(\mathsf{someValuesFrom}(\mathsf{loves\ Fish})))) \quad (4.13)$$

$$\mathsf{Class}(\mathsf{Fish\ partial\ Animal}) \quad (4.14)$$

which may have the RDF/XML syntax as[10]:

```
<owl:Class rdf:ID="Cat">
  <rdfs:subClassOf>
       <owl:Class rdf:ID="Animal"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
       <owl:Restriction>
         <owl:onProperty><owl:ObjectProperty rdf:ID="loves"/></owl:onProperty>
         <owl:someValuesFrom>
            <owl:Class rdf:ID="Fish">
               <rdfs:subClassOf rdf:resource="#Animal"/>
            </owl:Class>
         </owl:someValuesFrom>
       </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

[10]Validated by the WonderWeb OWL Ontology Validator (http://www.mygrid.org.uk/OWL/Validator)

Suppose there is another OWL ontology that reuses the class Fish and the fact that Fish $\sqsubseteq$ Animal, but does not need knowledge about Cat . However, since the "definition" of Fish and Animal are syntactical components of the "definition" of Cat, extracting such information from the reused ontology requires either manual processing or semantic analysis of the ontology. When the reused ontology is large, it is usually a time consuming task.

Hence, in addition to the lack of semantic modularity, current version of OWL also lacks the support for syntactical modularity, including the following problems:

- The RDF/XML syntax of OWL (will be called the "OWL syntax" by default in the follows) does not require explicit *declaration* of a class, a property or an individual. A class (property, individual) can be freely "defined" in any axiom, which makes partial reuse of ontologies being diffcult[11].

- The OWL syntax allows *arbitrary nesting* of a (grammatically legal) syntactical RDF/XML block inside another block, even if they are not parts of the same axiom in semantics. For example, the axiom 4.14 is represented as a syntactical part of axiom 4.13 in the OWL syntax. In general, OWL syntax does not require the correspondence between the semantic structure and the syntactical structure of an ontology.

- There is *no organizational structure inside* an OWL ontology. It is not possible for an OWL ontology to import a selective subset of axioms from another OWL ontology.

We argue that such problems can be solved without introducing new constructs to the OWL syntax. Syntactical modularity can be realized in OWL by disciplined usage of OWL syntax and extra-logical annotations. The proposal includes the following restrictions on the OWL syntax:

Required declaration. A term should always be "declared" before it is used in other semantic construction block. For example, Fish in the example given above should be first declared as

```
<owl:Class rdf:ID="Fish"/>
```

[11](Motik and Horrocks, 2006) shows that the lack of declaration in OWL syntax may lead to some other problems, e.g., ambiguous interpretation of certain syntactically well-formed OWL ontologies, and mismatches between OWL RDF and abstract syntax.

before it can be referred. Such declarations should not be nested in any other RDF/XML block nor has any nested subblock. To simplify the design of parsers, such declarations may be organized together at the beginning of the OWL file (after the file header)[12].

Disallowing nested axioms. A top-level RDF/XML block should corresponds to an axiom in the abstract syntax. Hence, no axiom in the abstract syntax will be nested inside another axiom in the RDF/XML syntax.

Assigning axioms into organizational units. Each top-level RDF/XML block (corresponds to an axiom) has an annotation property inUnit, which identifies the organizational unit of which the axiom belongs to.

Importing by units. Instead of requiring the range of `owl:imports` to be `owl:Ontology`, we may allow it to be URIs of "units" of ontologies, which are associated with selected subsets of axioms in the ontology. Thus, an ontology can be partially reused (imported) for a selected subset of its units by other ontologies.

For instance, the example ontology O given above can be rewritten as

```
<owl:AnnotationProperty rdf:ID="inUnit"/>
<owl:Class rdf:ID="Animal"/>
<owl:Class rdf:ID="Cat"/>
<owl:Class rdf:ID="Fish"/>
<owl:ObjectProperty rdf:ID="loves"/>

<owl:Class rdf:resource="#Cat">
  <inUnit>Unit_Mammals</inUnit>
  <rdfs:subClassOf>
    <owl:intersectionOf rdf:parseType="Collection">
       <owl:Class rdf:resource="#Animal"/>
       <owl:Restriction>
         <owl:onProperty rdf:resource="#loves"/>
         <owl:someValuesFrom><owl:Class rdf:resource="#Fish"/></owl:someValuesFrom>
       </owl:Restriction>
    </owl:intersectionOf>
```

[12]For other solutions for declaration in OWL, please refer (Motik and Horrocks, 2006).

```
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:resource="#Fish">
   <inUnit>Unit_Fishes</inUnit>
   <rdfs:subClassOf rdf:resource="#Animal"/>
</owl:Class>
```

In this syntax, each class or property is declared before it is used in any axiom (except for the declaration axiom itself). The two non-declaration axioms (possibly among other axioms of O) belong to two units: `Unit_Mammals` and `Unit_Fishes`. When another OWL ontology O' reuses the ontology O, O' may selectively reuse the axioms in `Unit_Mammals` or `Unit_Fishes`, or both of them.

The proposed modular syntax for OWL has the following advantages:

- It is backward compatible to existing OWL syntax and ontology tools. Ontologies in the new syntax may still be processed using existing parsers, editors and visualizers. On the other hand, extending existing tools to support syntactical modularity is relatively easy (compared to several other syntactical extensions to OWL).

- It ensures direct correspondence between the semantic structure and the syntactical structure of an ontology. Hence, an ontology in the modular syntax will have better human readability and is easier to be reused or maintained. The proposed syntax conforms to the principle of "separation of concern", which has been proven successful in software engineering.

- It allows syntactical partial reuse of an ontology. By selective importing of units, only axioms belonging to the imported units will be propagated to the referring ontology.

However, a unit in the modular syntax should not be considered as a semantic module, e.g., a package. While a package has its own local interpretation domain, units in the *same* OWL file (which itself may be considered as a package) share the same interpretation domain. In addition, a package has its own local signature (the set of names belonging to the namespace

of the package), while units in the same OWL ontology share the same local signature of the ontology; hence there is no importing relation between units in the same OWL ontology.

4.3.4 Summary and Discussion

The proposed modular semantics and syntax of OWL offers a promising approach to supporting localized semantics (hence the preservation of knowledge context) and partial reuse of ontologies. Because it requires no new syntactic extensions to OWL, it ensures backward compatibility of the resulting modular OWL ontologies processable using existing tools (in settings that do not require support for modularity).

The proposed syntax can serve as the syntax for the P-DL $\mathcal{SHOIN}(D)$. Hence, the creation, storage, parsing and visualization of P-DL ontologies are readily supported by many existing ontology tools. However, such an approach is also limited in several ways, including the following:

- The proposed semantics and syntax do not allow the contextualized negation ($\neg_i$) and contextualized "top" ($\top_i$) being used in ontology modules other than the module i.
- Relaxing the range of `owl:imports` to including objects that are not `owl:Ontology` objects will make `owl:imports` not being an instances of `owl:OntologyProperty`. This may lead to potential problems when the argument of `owl:imports` is not a valid ontology or ontology unit.
- The proposed semantics and syntax only partially support semantic partial reuse. It allows partial interpretation of terms in the imported module, such that terms not being explicitly imported may not be interpreted in the referring module. However, it does not address the problem of automatic identification of "axioms that could affect the possible interpretations of the (imported) symbol"(Fikes et al., 1997). A promising solution to such a problem is recently explored in (Grau et al., 2007).

An alternative syntax, P-OWL, that supports the full modeling ability offered by P-DL is proposed in (Bao and Honavar, 2004c). On the other hand, P-OWL is not backward compatible to existing ontology tools.

4.4 Related Work

4.4.1 Other Modular Ontology Languages

4.4.1.1 Early Development

Some of the modular ontology ideas can be traced to studies of knowledge engineering over the past few decades. The Cyc project (Lenat, 1995) divides the comprehensive Cyc knowledge base (expressed in the CycL language) into many *microtheories* - collections of concepts and facts pertaining to particular knowledge domains. Partition-based Logics (Amir and McIlraith, 2000) provides an approach to automatically decompose propositional and first-order logic (FOL) into *partitions* and an algorithm for reasoning with such partitions using message passing. These efforts provided the important initial experiences on building and reasoning with modular knowledge bases.

Both CycL and Partition-based Logics do not provide *localized semantics* for knowledge modules or principled ways of connecting ontology modules, Neither do they support partial reuse or mechanisms for controlling semantic heterogeneity among knowledge base modules. Indeed, even reusing a small portion of Cyc, OpenCyc [13], because of the lack of modularity, requires the entire OpenCyc ontology to be loaded although only a small part of it may be of interest. The OWL scaffolding version of OpenCyc v0.7.8b (>700MB), containing assertions for over 60,000 Cyc constants, "takes approximately 9 hours to load into Protege" (from OpenCyc homepage, 2004/06/04 announcement).

4.4.1.2 DFOL, DDL and C-OWL

Recent work on modular ontology languages is heavily influenced by contextual logics, and in particular, the **Local Models Semantics** (LMS) (Giunchiglia and Ghidini, 1998). LMS theory allows a family of logic languages to have *local models* that represent the local semantic points of view of each of the languages. A formula in one language may be the logical consequence of a formula in another language. Thus, LMS provides a practical tradeoff between locality and compatibility in multi-context knowledge bases.

A **Distributed First Order Logics** (DFOL) knowledge base (KB) (Ghidini and Serafini, 1998) (and hence, a DFOL ontology) includes a family of first order languages, each represents

[13] http://www.opencyc.org/

a piece of the global knowledge. DFOL semantics includes a set of *local models* (first order interpretations) for each of the language, and a set of *domain relations* between objects in those local models. Inference with DFOL is enabled by a calculus as an extension of natural deduction that allows theorem exporting among different languages, i.e., a theorem can be the logical consequence of another theorem in a different language.

Based on DFOL, **Distributed Description Logics** (DDL) (Borgida and Serafini, 2002) allows directional relations among multiple description logic modules where each module is in a subset of $\mathcal{SHIQ}$. The initial proposal provides *bridge rules* between concepts and individuals in different ontologies in forms of:

$i : C \xrightarrow{\sqsubseteq} j : D$ (INTO)

$i : C \xrightarrow{\sqsupseteq} j : D$ (ONTO)

$i : x \mapsto j : y$ (partial individual correspondence)

$i : x \xmapsto{=} j : \{y_1, y_2, ...\}$ (complete individual correspondence)

where C, D are concepts, x, y are individuals, i, j indicate indexes of ontologies. $\xrightarrow{\equiv}$ is used as the shorthand for both $\xrightarrow{\sqsubseteq}$ and $\xrightarrow{\sqsupseteq}$. INTO and ONTO rules are meant to simulate concept subsumptions across ontologies. For example, there may be bridge rules as

$$i : \mathsf{Dog} \xrightarrow{\sqsubseteq} j : \mathsf{Pet}$$
$$i : \mathsf{Animal} \xrightarrow{\sqsupseteq} j : \mathsf{Pet}$$

to indicate that $i : \mathsf{Dog}$ is less general than $j : \mathsf{Pet}$, and $i : \mathsf{Animal}$ is more general than $j : \mathsf{Pet}$.

Individual correspondences in DDL allow one-to-one or one-to-many mappings between individuals across ontologies, such as

$$i : \mathsf{US} \mapsto j : \mathsf{UnitedStates}$$
$$i : \mathsf{NYC} \xmapsto{=} j : \{\mathsf{Bronx}, \mathsf{Manhattan}, \mathsf{Queens}, \mathsf{Brooklyn}, \mathsf{StatenIsland}\}$$

Syntax of DDL includes **CTXML** (ConTeXt Markup Language) (Bouquet et al., 2002) and **C-OWL** language (Bouquet et al., 2003).

Subsequent extensions to DDL include heterogenous mapping between roles and concepts (Ghidini and Serafini, 2006b; Ghidini et al., 2007). For example, marriage relation can be represented by a concept Marraige in one ontology but by a role marriesTo in other ontology; a concept/role

bridge rule can be declared as

$$\mathsf{Marriage} \xrightarrow{\sqsubseteq} \mathsf{marriesTo} \quad \text{and}$$
$$\mathsf{Marriage} \xrightarrow{\sqsupseteq} \mathsf{marriesTo}$$

to indicate that Marriage instances are always linked to certain pairs of individuals (as marriesTo instances) of the other ontology.

4.4.1.3 View-based Modular Ontologies

A notable variation of DDL is the approach adopted by (Stuckenschmidt and Klein, 2003a). Influenced by DDL semantics, they adopt a view-based information integration approach to express relationships between ontology modules. In particular, in this approach ontology modules are connected by correspondences between conjunctive queries. This way of connecting modules provides a tradeoff between the simplicity of one-to-one mappings between concept names and the unrestricted use of logical languages to connect different modules.

(Stuckenschmidt and Klein, 2003a) defines an ontology module as a triple $M = (\mathcal{C}, \mathcal{R}, \mathcal{O})$, where $\mathcal{C}$ is a set of concept definitions, $\mathcal{R}$ is a set of relation definitions and $\mathcal{O}$ is a set of object definitions. A conjunctive query Q over an ontology module $M = (\mathcal{C}, \mathcal{R}, \mathcal{O})$ is defined as an expression of the form $q_1 \wedge ... \wedge q_m$, where q_i is a query term of the form $C(x)$, $R(x, y)$ or $x = y$, C and R are concept and role names, respectively, and x and y are either variables or object names.

Concepts in one module can be defined in terms of a conjunctive query over another module. Thus, the set of concept definitions $\mathcal{C}$ is divided into two disjoint sets of internally and externally defined concepts $\mathcal{C}_I$ and $\mathcal{C}_E$, respectively. An *internal concept* definition is specified using regular description logics based concept expressions in the form of $C \sqsubseteq D$ or $C \equiv D$, where C and D are atomic and complex concepts, respectively. An *external concept* definition is an axiom of the form $C \equiv M : Q$, where M is a module and Q is a conjunctive query with one free variable over M. Similarly, we can define *external relations* (i.e., roles) using conjunctive queries with two free variables. A modular ontology is then defined as a set of modules that are connected by external concept or relation definitions.

The semantics of these modules is defined in the same fashion as in DDL. In fact, an external concept definition $C \equiv M : Q_C$ can be translated into a DDL bridge rule between concepts:

$M : Q_C \xrightarrow{\equiv} C$. However, an external relation definition $P \equiv M : Q_P$ cannot be translated into a DDL bridge rule between roles (Serafini et al., 2005b), since in (Stuckenschmidt and Klein, 2003a) external relations are interpreted using on quarternary domain relations for roles, which is different from the binary domain relations for concepts. The semantics correspondence between the two types of domain relations is not specified by (Stuckenschmidt and Klein, 2003a).

(Stuckenschmidt and Klein, 2003b) extends this approach with caching previous query results of external modules and resolving inconsistencies between the cached results and the external ontology when ontology modules may evolve.

As far as when only concept reasoning is concerned, the approach of (Stuckenschmidt and Klein, 2003a) can be seen as a restricted form of DDLs. Hence, we will not discuss it as a sperate approach in what follows.

4.4.1.4 $\mathcal{E}$-Connections

DDL is limited in connecting modules with roles. For example, roles defined in other ontology modules (i.e., foreign roles) cannot be used to construct new concepts, or to construct new roles from foreign roles. On the contrast, **$\mathcal{E}$-Connections** (Kutz et al., 2002) focus on offering inter-module role connections. Some of the ideas incorporated into $\mathcal{E}$-connections can be traced back to the fusion of abstract description systems (ADS) (Baader et al., 2000), in which atomic roles are partitioned into disjoint sets that each can only be used in the constructors of the language of a single module. $\mathcal{E}$-connections between DLs (Kutz et al., 2003; Grau et al., 2004b, 2006c) restrict the concept languages of the component modules to be disjoint, and each of the $\mathcal{E}$-connected ontology modules is interpreted in a local domain (therefore ensure localized semantics). Roles are divided into disjoint sets of *local roles* (connecting concepts in one module) and *links* (connecting concepts in different modules). For example, two concepts $i : \mathsf{PetOwner}$ and $j : \mathsf{Pet}$ can be connected with a link owns such that:

$$i : \mathsf{PetOwner} \sqsubseteq \exists \mathsf{owns}.(j : \mathsf{Pet})$$

Such a division of links and local roles ensures the decidability transfer property: if all ontologies connected by E-connections (the set of links) are locally decidable, then their union is also decidable (Kutz et al., 2003).

An XML Syntax of E-Connections is first provided in (Grau et al., 2004b) for the e-connected version of OWL-Lite, denoted as $C^{\mathcal{E}}(\mathcal{SHIF}(D))$. More expressive extensions are reported in (Grau, 2005) as $C^{\mathcal{E}}_{\mathcal{IHN}^+{}_s}(\mathcal{SHOIQ}(D))$ which allow each connected modules to be a subset of OWL-DL, including $\mathcal{SHIQ}, \mathcal{SHOQ}$, and $\mathcal{SHIO}$. Subscripts ${}_{\mathcal{IHN}^+{}_s}$ stand for several link constructors of the form:

${}_{\mathcal{I}}$: $\mathsf{owns} = \mathsf{ownedBy}^-$ (link inverse)

${}_{\mathcal{H}}$: $\mathsf{sonOf} \sqsubseteq \mathsf{childOf}$ (link inclusion)

${}_{\mathcal{N}}$: $\mathsf{PetMania} \sqsubseteq (\geq 5\ \mathsf{owns}.\top_j)$ (link number restriction)

${}^{+}$: $\mathsf{Trans}(\mathsf{largerThan})$ (transitive link)

${}_{s}$: $\mathsf{Symmetric}(\mathsf{brotherOf})$ (symmetric link)

An extension of transitive link, called generalized link, is reported in (Parsia and Grau, 2005), which can control transitivity of links among modules.

The evolution of modular ontology language proposals is summarized in Figure 4.4.

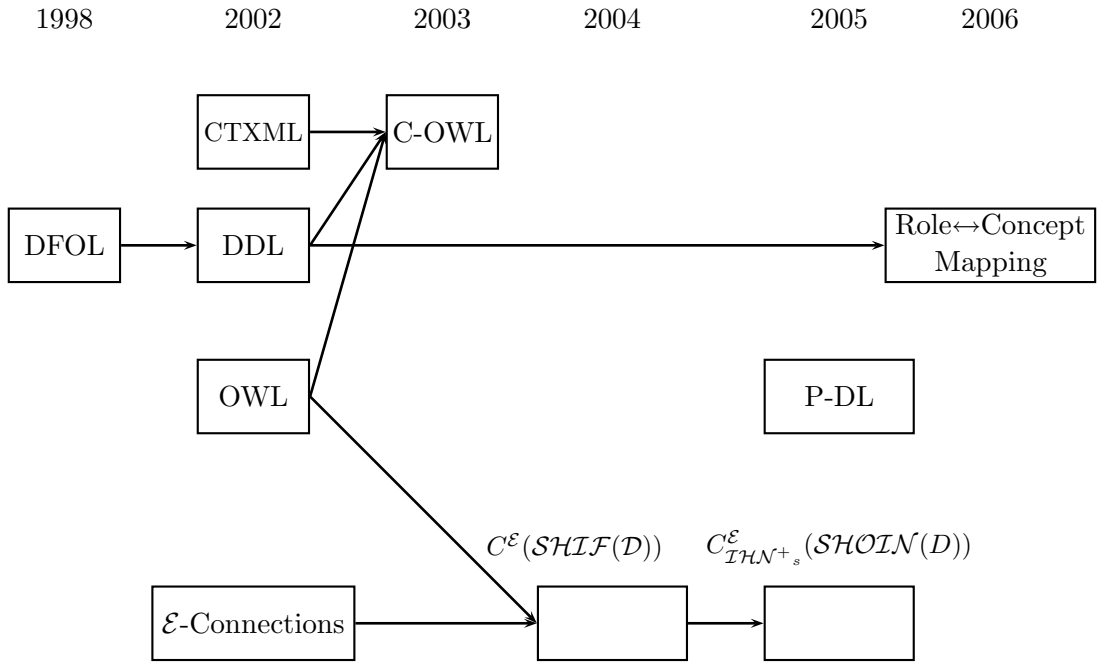


Figure 4.4 Evolution of Modular Ontology Languages

4.4.2 Semantics of Modular Ontology Languages

Although various modular ontology language proposals differ from each other in terms of language features, they share one feature in common in contrast with traditional ontology languages: all of the modular ontology language proposals support localized semantics. In other words, a (global) model for a modular ontology would contain a set of local models as well as a set of relations between those local models. In contrast, a traditional ontology (as well as in the fusion of logics (Baader et al., 2000) and in the approach based on conservative extensions (Grau et al., 2007)) always requires a single model that satisfies all restrictions in all modules of that ontology.

(Serafini et al., 2005b) compared the semantics of different modular ontology language proposals in the light of DFOL semantics. We will follow their approach and further investigate some important properties of modular ontology semantics (e.g., local domain disjointness) in the abstract modular ontology (AMO) framework we introduced in the last chapter.

To make this chapter self-contained, we briefly repeat the AMO framework. Formally, an abstract modular ontology (AMO) is a DFOL knowledge base consisting of a family of component languages $\{L_i\}$ (each called a module) and semantic relation rules $\{\mathfrak{R}_{ij}\}(i \neq j)$. Each L_i is in a subset of description logics (DL). In this paper, we restrict ourself to the setting where each L_i is a subset of the expressive DL $\mathcal{SHOIQ}(D)$. Modular ontology language proposals differ mainly with respect to how to define and interpret semantic relation rules $\{\mathfrak{R}_{ij}\}$.

A model of AMO includes a set of local models $\{\mathcal{I}_i\}$ and domain relations $\{\mathbf{r}_{ij}\}$. For each L_i, there exists a local model $\mathcal{I}_i = \langle \Delta_i, (.)_i \rangle$, where Δ_i is the local interpretation domain, $(.)_i$ is the assignment function for concept, role and individual terms in L_i. Each domain relation $r_{ij} \in \mathbf{r}_{ij}$ is a subset of $\Delta_i \times \Delta_j$. Note that it is possible to have multiple domain relations between a pair of local models. In particular, the image domain relation, denoted by $r_{ij}^{\rightarrow}$, indicating (from the point of view the module j) object correspondences between Δ_i and Δ_j, such that $\langle x, y \rangle \in r_{ij}^{\rightarrow}$ if x, y identify the same physical object. Finally, $r_{ij}(d)$ denotes the set $\{d' \in \Delta_j | \langle d, d' \rangle \in r_{ij}\}$. For a subset $D \subseteq \Delta_i$, $r_{ij}(D)$ denotes $\cup_{d \in D} r_{ij}(d)$.

4.4.2.1 Semantics of DDL

Semantics: DDL *Homogenous bridge rules* are in the form of:

Semantic Mapping	**Syntax**	**Semantics**
DDL		
Homogenous INTO	$i : \phi \xrightarrow{\sqsubseteq} j : \psi$	$r_{ij}(\phi^{\mathcal{I}_i}) \subseteq \psi^{\mathcal{I}_j}$
Homogenous ONTO	$i : \phi \xrightarrow{\sqsupseteq} j : \psi$	$r_{ij}(\phi^{\mathcal{I}_i}) \supseteq \psi^{\mathcal{I}_j}$
Heterogenous concept/role INTO	$i : C \xrightarrow{\sqsubseteq} j : R$	$cr_{ij}(C^{\mathcal{I}_i}) \subseteq R^{\mathcal{I}_j}$
Heterogenous concept/role ONTO	$i : C \xrightarrow{\sqsupseteq} j : R$	$cr_{ij}(C^{\mathcal{I}_i}) \supseteq R^{\mathcal{I}_j}$
Heterogenous role/concept INTO	$i : P \xrightarrow{\sqsubseteq} j : D$	$rc_{ij}(P^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$
Heterogenous role/concept ONTO	$i : P \xrightarrow{\sqsupseteq} j : D$	$rc_{ij}(P^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_j}$
Partial individual correspondence	$i : x \mapsto j : y$	$y^{\mathcal{I}_j} \in r_{ij}(x^{\mathcal{I}_i})$
Complete individual correspondence	$i : x \overset{=}{\mapsto} j : \{y_1, y_2, ...\}$	$r_{ij}(x^{\mathcal{I}_i}) = \{y_1^{\mathcal{I}_j}, y_2^{\mathcal{I}_j}, ...\}$
$\mathcal{E}$-Connections		
Existential Link Restriction	$i : (\exists E.(j : D))$	$r_E^-(D^{\mathcal{I}_j})$
Universal Link Restriction	$i : (\forall E.(j : D))$	$\Delta_i \backslash r_E^-(\Delta_j \backslash D^{\mathcal{I}_j})$
Number Link Restriction	$i : (\geqslant nE.D)$	$\{x \mid \lvert r_E(x) \cap D^{\mathcal{I}_j}\rvert_{\neq} \geqslant n\}$
Link Inverse	$E = F^-$	$r_E = r_F^-$
Link Inclusion	$E_1 \sqsubseteq E_2$	$r_{E_1} \subseteq r_{E_2}$
Transitive Link	$Trans(G; I)$	$(x, y) \in r_{G^{(ij)}} \wedge (y, z) \in r_{G^{(jk)}} \rightarrow (x, z) \in r_G$, for $i, j, k \in I$
Symmetric Link	$Symmetric(G)$	$r_{G^{(ij)}} = r_{G^{(ji)}}^-$
P-DL		
Importing	$i \xrightarrow{\phi} j$	$r_{ij}(\phi^{\mathcal{I}_i}) = \phi^{\mathcal{I}_j}$

Notations:

- DDL: ϕ is an i-concept(or role), ψ is an j-concept(or role); C is an i-concept, D is a j-concept, P is an i-role, R is a j-role; x is an i-individual, y_i is a j-individual; $r_{ij} \subseteq \Delta_i \times \Delta_j$ is the domain relation from i to j; $rc_{ij} \subseteq \Delta_i \times \Delta_i \times \Delta_j$ is the role to concept domain relation, $cr_{ij} \subseteq \Delta_i \times \Delta_j \times \Delta_j$ is the concept to role domain relation.

- E-Connections: E, E_1, E_2 are $\mathcal{E}$-connections from i to j, F is an $\mathcal{E}$-connection from j to i; D is a j-concept; G is a generalized link; I is a set of module indices; r_E is the domain relation for E, r_E^- is the inverse of r_E; $|S|_{\neq}$ stands for all-different cardinality of a set S, i.e. the number of elements in S if equivalent elements only counted as one element.

- P-DL: ϕ is a concept, role or individual name. Complete semantics is presented in section 4.2.2.

Table 4.1 Semantics of Modular Ontology Languages

INTO rule: $i : \phi \xrightarrow{\sqsubseteq} j : \psi$ (semantics: $r_{ij}(\phi^{\mathcal{I}_i}) \subseteq \psi^{\mathcal{I}_j}$)

ONTO rule: $i : \phi \xrightarrow{\sqsupseteq} j : \psi$ (semantics: $r_{ij}(\phi^{\mathcal{I}_i}) \supseteq \psi^{\mathcal{I}_j}$)

where ϕ is an i-concept and ψ is a j-concept (defined in (Borgida and Serafini, 2002)), or ϕ is an i-role and ψ is a j-role (defined in (Ghidini and Serafini, 2006a)). $r_{ij} \subseteq \Delta_i \times \Delta_j$ is the interpretation of $\mathcal{B}_{ij}$. We use $r_{ij}(x)$ to denote $\{y \in \Delta_j | (x, y) \in r_{ij}\}$; $r_{ij}(\phi^{\mathcal{I}_i})$ is defined as

- $\bigcup_{x \in \phi^{\mathcal{I}_i}} r_{ij}(x)$, when ϕ is a concept
- $\bigcup_{(x,y) \in \phi^{\mathcal{I}_i}} r_{ij}(x) \times r_{ij}(y)$, when ϕ is a role name

Similarly, heterogenous rules (defined in (Ghidini and Serafini, 2006b)) are interpreted using special types of domain relations rc_{ij} (for role to concept mapping) and cr_{ij} (concept to role mapping). Semantics of DDL is summarized in Table 4.1.

Translation into AMO: DDL homogenous bridge rules can be translated into semantic relation rules in AMO as:

$$
\begin{aligned}
i : C \xrightarrow{\sqsubseteq} j : D \quad &\Rightarrow \quad \forall x \in \Delta_i, C(x) \xrightarrow{\langle x,y \rangle \in r_{ij}^{\rightarrow}} D(y) \\
i : C \xrightarrow{\sqsupseteq} j : D \quad &\Rightarrow \quad \forall x \in \Delta_j, D(x) \xrightarrow{\langle y,x \rangle \in r_{ij}^{\rightarrow}} \exists y \in \Delta_i, C(y) \\
i : P \xrightarrow{\sqsubseteq} j : R \quad &\Rightarrow \quad \forall x_1, x_2 \in \Delta_i, P(x_1, x_2) \xrightarrow{\langle x_1,y_1 \rangle \in r_{ij}^{\rightarrow}, \langle x_2,y_2 \rangle \in r_{ij}^{\rightarrow}} R(y_1, y_2) \\
i : P \xrightarrow{\sqsupseteq} j : R \quad &\Rightarrow \quad \forall x_1, x_2 \in \Delta_j, R(x_1, x_2) \xrightarrow{\langle y_1,x_1 \rangle \in r_{ij}^{\rightarrow}, \langle y_2,x_2 \rangle \in r_{ij}^{\rightarrow}} \exists y_1, y_2 \in \Delta_i, P(y_1, y_2)
\end{aligned}
$$

Note that partial and complete individual correspondence can be seen as special cases of concept bridge rules.

4.4.2.2 Semantics of $\mathcal{E}$-Connections

Semantics: $\mathcal{E}$-connections allow construction of new concepts using links. Formally, given ontology modules $\{L_i\}$, an (one-way binary) link (more expressive $\mathcal{E}$-connections are beyond the scope of our discussion) $E \in \mathcal{E}_{ij}$, where $\mathcal{E}_{ij} (i \neq j)$ is the set of all links from the module i to the module j, can be used to construct a concept in module i, with the syntax and semantics specified as follows:

- $\langle E \rangle (j : C)$ or $\exists E.(j : C) : \{x \in \Delta_i | \exists y \in \Delta_j, (x, y) \in r_E, y \in C^{\mathcal{I}}\}$

- $\forall E.(j:C):\{x \in \Delta_i | \forall y \in \Delta_j, (x,y) \in r_E \rightarrow y \in C^{\mathcal{I}}\}$
- $\leq nE.(j:C):\{x \in \Delta_i | \#(\{y \in \Delta_j | (x,y) \in r_E, y \in C^{\mathcal{I}}\}) \leq n\}$
- $\geq nE.(j:C):\{x \in \Delta_i | \#(\{y \in \Delta_j | (x,y) \in r_E, y \in C^{\mathcal{I}}\}) \geq n\}$

where $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_E\}_{E \in \mathcal{E}_{ij}} \rangle$ is a model of the $\mathcal{E}$-connected ontology, $\mathcal{I}_i$ is the local model of L_i; C is a concept in L_j, with interpretation $C^{\mathcal{I}} = C^{\mathcal{I}_j}$; $r_E \subseteq \Delta_i \times \Delta_j$ is the interpretation of an $\mathcal{E}$-connection E.

Such an $\mathcal{I}$ can seen as a special case of a DFOL or AMO model $M_d = \langle \{\mathcal{I}_i\}, \{r_{ij}\} \rangle$ with each r_E ($E \in \mathcal{E}_{ij}$) acting as a domain relation r_{ij} (Serafini et al., 2005b). Extending the semantics of $\mathcal{E}$-connection axioms ((1) and (3) below) given in (Serafini et al., 2005b) so as to allow the use of constructed concepts ($\exists E.D$ and $\forall E.D$) on either side of subsumptions, we have:

1) $C \sqsubseteq \forall E.D : r_E(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$

2) $C \sqsupseteq \forall E.D : (\neg C)^{\mathcal{I}_i} \subseteq r_E^-((\neg D)^{\mathcal{I}_j})$, i.e., $r_E(C^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_j}$

3) $C \sqsubseteq \exists E.D : C^{\mathcal{I}_i} \subseteq r_E^-(D^{\mathcal{I}_j})$

4) $C \sqsupseteq \exists E.D : r_E((\neg C)^{\mathcal{I}_i}) \subseteq (\neg D)^{\mathcal{I}_j}$, i.e., $C^{\mathcal{I}_i} \supseteq r_E^-(D^{\mathcal{I}_j})$

where r_E^- is the inverse of r_E, C is an i-concept and D is a j-concept, C can be an atomic or complex concept. Note that case (2)(similarly also for (4)) can not be reduce to defining $C' \equiv \forall E.D$ and $C' \sqsubseteq C$ in i, since $\equiv$ is the short for $\sqsubseteq$ and $\sqsupseteq$.

Since interpretations of links are acting as domain relations, a distributed model of an $\mathcal{E}$-connected ontology may have multiple domain relations between two local models. Such a semantics for links is equivalent to allowing an i-role to have the range from and only from Δ_j. Thus, link constructors, like inversion or inclusion, are different from role constructors that bridge roles in *different* modules (which are illegal in $\mathcal{E}$-connections).

All concepts constructed using a link E from i to j are i-concepts. Thus they can be used in i as other local i-concepts. For example, the axiom

$$i : \mathsf{PetOwner} \sqsubseteq \exists \mathsf{owns}.(j : \mathsf{Pet})$$

would indicate a restriction in Δ_i such that:

$$\mathsf{PetOwner}^{\mathcal{I}_i} \subseteq r_{\mathsf{owns}}^-(\mathsf{Pet}^{\mathcal{I}_j}) \subseteq \Delta_i$$

Note that the semantics of $\mathcal{E}$-connections given in Table 4.1 is strictly equivalent to the forms given in (Kutz et al., 2003; Grau et al., 2004b) (introduced above) and (Serafini et al., 2005b). Using this representation, we may represent semantic relations other than concept subsumptions. For example, a concept intersection $\exists \mathsf{owns}.(j : \mathsf{Pet}) \sqcap i : \mathsf{Man}$ can be interpreted as:

$$r^{-}_{\mathsf{owns}}(\mathsf{Pet}^{\mathcal{I}_j}) \cap \mathsf{Man}^{\mathcal{I}_i}$$

and $i : (\forall E.(j : D))$ will be interpreted as $\Delta_i \backslash r^{-}_{E}(\Delta_j \backslash D^{\mathcal{I}_j})$ since $\forall E.D = \neg\exists E.(\neg D)$.

Transitive and symmetric links are *generalized links* (Parsia and Grau, 2005) based on the "punning" idea, i.e., allowing a link being interpreted in different contexts, and each of the interpretation is denoted with a superscript. For example, a link G may be used as $G^{(1)}$ from i to j and $G^{(2)}$ from j to k; G's interpretation will be the *union* of $r_{G^{(1)}} \subseteq \Delta_i \times \Delta_j$ and $r_{G^{(2)}} \subseteq \Delta_j \times \Delta_k$.

The semantics of $\mathcal{E}$-Connections when local domains of each ontology module are not required to be disjoint is called the *free-floating semantics* (Grau and Kutz, 2007). A different semantics called *separated semantics* that requires all local domains to be mutually disjoint has also been studied (Grau, 2005; Grau et al., 2006c). It has been claimed that the two semantics are equivalent in answering queries for all *legal assertions* in $\mathcal{E}$-Connections (Grau and Kutz, 2007).

Translation into AMO: Let $E \in \mathcal{E}_{ij}$, C be an i-concept and D be a j-concept, we may translate axioms in $\mathcal{E}$-Connections into AMO semantic relation rules:

$$C \sqsubseteq \forall E.D \Rightarrow \forall x \in \Delta_i, C(x) \xrightarrow{\langle x,y\rangle \in r_E} D(y)$$
$$C \sqsupseteq \forall E.D \Rightarrow \forall x \in \Delta_j, D(x) \xrightarrow{\langle x,y\rangle \in r^{-}_E} \exists y \in \Delta_i, C(y)$$
$$C \sqsubseteq \exists E.D \Rightarrow \forall x \in \Delta_i, C(x) \xrightarrow{\langle x,y\rangle \in r_E} \exists y \in \Delta_j, D(y)$$
$$C \sqsupseteq \exists E.D \Rightarrow \forall x \in \Delta_j, D(x) \xrightarrow{\langle x,y\rangle \in r^{-}_E} C(y)$$

4.4.2.3 Relation Between DDL and $\mathcal{E}$-Connections

Both DDL and $\mathcal{E}$-Connections conform to the "linking" approach such that concept languages of ontology modules are disjoint, and semantic relations between modules are only given by mappings like DDL bridge rules and $\mathcal{E}$-connection links. In fact, one-way binary $\mathcal{E}$-connections with inverse links is as expressive as DDL with bridge rules between concepts.

Reduction from DDL to $\mathcal{E}$-Connections: It has been argued that $\mathcal{E}$-connections are more expressive than DDL (Kutz et al., 2004; Grau, 2005; Grau and Kutz, 2007) because DDL can be reduced to $\mathcal{E}$-connections. However, the reliance of the reduction on the equivalence of $C \xrightarrow{\sqsubseteq} D$ (by default we denote C as an i-concept and D as a j-concept) to $\langle E\rangle.C \sqsubseteq D$ and $C \xrightarrow{\sqsupseteq} D$ to $\langle E\rangle.C \sqsupseteq D$ (Kutz et al., 2004; Grau, 2005), presents semantic difficulties with regard to DDL and $\mathcal{E}$-connections semantics in the DFOL framework (Serafini et al., 2005b). For example, ONTO($\xrightarrow{\sqsupseteq}$) rules in DDL is mapped to type d interpretation constraints in DFOL while $\langle E\rangle.C \sqsupseteq D$ is mapped to type b interpretation constraints in DFOL. It is also clear in their AMO translations:

$$\begin{aligned}
C \xrightarrow{\sqsupseteq} D \quad &\Rightarrow \quad \forall x \in \Delta_j, D(x) \xrightarrow{\langle x,y\rangle \in r_{ij}^{\rightarrow}} \exists y \in \Delta_i, C(y) \\
D \sqsubseteq \exists E.C \quad &\Rightarrow \quad \forall x \in \Delta_j, D(x) \xrightarrow{\langle x,y\rangle \in r_{E}^{-}} \exists y \in \Delta_i, C(y) \\
C \xrightarrow{\sqsubseteq} D \quad &\Rightarrow \quad \forall x \in \Delta_i, C(x) \xrightarrow{\langle x,y\rangle \in r_{ij}^{\rightarrow}} D(y) \\
D \sqsupseteq \exists E.C \quad &\Rightarrow \quad \forall x \in \Delta_i, C(x) \xrightarrow{\langle x,y\rangle \in r_{E}^{-}} D(y)
\end{aligned}$$

Such a result is because of a fundamental difference between the two formalisms on linking subjectivity. In DDL, a bridge rule B_{ij} is always the subjective point of view of j and to be used by reasoning in j. On the other hand, an $\mathcal{E}$-connection $\mathcal{E}_{ij}$ provides the module i the ability to construct new concepts, therefore it actually represents i's point of view of the domain relation, and is used by reasoning in i. It can be confirmed in the syntax proposal of $\mathcal{E}$-connections (Grau, 2005) where the domain of a link is the module in which it has been declared.

In fact, an ONTO bridge rule $C \xrightarrow{\sqsupseteq} D$ can be translated into $D \sqsubseteq \exists E^{-}.C$ (in module j) or $\forall E.D \sqsubseteq C$ (in module i), and an INTO bridge rule $C \xrightarrow{\sqsubseteq} D$ can be translated into $C \sqsubseteq \forall E.D$ (in module i) or $\exists E^{-}.C \sqsubseteq D$ (in module j), where E^{-} is the inversion of E.

Since DDL bridge rules are directional, we argue that a translation of a DDL bridge rule should be placed in the "target" module, not in the "source" module. Hence, $C \xrightarrow{\sqsupseteq} D$ should be translated into $D \sqsubseteq \exists E^{-}.C$ (in module j) and $C \xrightarrow{\sqsubseteq} D$ should be translated into $\exists E^{-}.C \sqsubseteq D$ (in module j). Therefore, *inverse links* being allowed is a necessary condition for DDL bridge rules to be translated into $\mathcal{E}$-connections axioms *with preserved directionality*.

Lemma 4.3 *One-way binary $\mathcal{E}$-connections, as presented in* (Kutz et al., 2004; Grau, 2005) *is at least as expressive as DDL with bridge rules between concepts as presented in* (Borgida and Serafini, 2002)*, when each module is in $\mathcal{SHIQ}$, only if inverse links are allowed.*

Thus, the language $\mathcal{C}^{\mathcal{E}}_{\mathcal{HI}}(\mathcal{SHIQ})$ is at least as expressive than DDL but $\mathcal{C}^{\mathcal{E}}_{\mathcal{HQ}}$ $(\mathcal{SHIQ})$ (allowing no inverse link) (Grau, 2005) is *not*.

Note that for $i : C \xrightarrow{\exists} j : D$, defining an $\mathcal{E}$-connection F from j to i, the onto rule still cannot be translated into $D \sqsubseteq \exists F.C$, since DDL semantics does not assume $r_{ij} = r_{ji}^{-}$, therefore $F \neq E^{-}$. To assert the inverse relation, we still need inverse link constructors in $\mathcal{E}$-connections.

Reduction from $\mathcal{E}$-Connections to DDL: On the other hand, it is also possible to reduce one-way binary $\mathcal{E}$-connections into DDL. It is because a link in $\mathcal{E}$-connections can be simulated as a normal role by giving a "foreign" concept in its range a local alias. Without loss of generality, we assume for every formula of the form $\forall E.D$, $\exists E.D$, $\leq nE.D$ or $\geq nE.D$ where E is a link, D must be a concept name, otherwise we can always replace D with a new concept name which is equivalent to D.

Definition 4.7 *A reduction π from a knowledge base $K = \langle \{L_i\}, \{\mathcal{E}_{ij}\} \rangle$ in one-way binary $\mathcal{E}$-connections to a knowledge base $\pi(K) = K' = \langle \{L'_i\}, \{\mathcal{B}_{ij}\} \rangle$ in DDL with bridge rules between concepts (where each module is in $\mathcal{SHIQ}$) is as the follows, where π_i is a mapping function from formulae in L_i to formulae in L'_i:*

- *For every i, let $L'_i = L_i$, define a concept $\top'_i$ in L_i, and let $\pi_i(\top_i) = \top'_i$;*
- *For every concept formula C occurring in L_i that only uses i-names, let $\pi_i(C) = C \sqcap \top'_i$, and add $C \sqsubseteq \top'_i$ to L'_i; for every i-role name P occurring in L_i, let $\pi_i(P) = P$, and add $\top'_i$ as its domain and range in L'_i; recursively (starting with atomic ones) replace every formula t with $\pi_i(t)$ in L'_i;*
- *For every $\mathcal{E}_{ij} \neq \emptyset$, define a concept $\top_{ji}$ in L'_i, add a bridge rule $\top'_j \xrightarrow{\equiv} \top_{ji}$ to $\mathcal{B}_{ji}$, and let $\pi_i(\top_j) = \top_{ji}$;*
- *For each $E \in \mathcal{E}_{ij}$, replace it with a new role E' in L'_i, with domain $\top'_i$ and range $\top_{ji}$, and let $\pi_i(E) = E'$;*
- *For every j-concept D occurring in L_i, replace D with a new concept D' in L'_i, add $D' \sqsubseteq \top_{ji}$ to L'_i, a bridge rule $D \xrightarrow{\equiv} D'$ to $\mathcal{B}_{ji}$, and let $\pi_i(D) = D'$.*

Lemma 4.4 *Let $K' = \pi(K)$ be a knowledge base in DDL with bridge rules between concepts obtained from a knowledge base K in one-way binary $\mathcal{E}$-connections where each module is in $\mathcal{SHIQ}$, C is a concept that is legal in the module i of K, then $K \models C \sqsubseteq \bot$ iff $K' \models \pi_i(C) \sqsubseteq \bot$.*

Proof: we show that K has a model with $C^{\mathcal{I}_i} \neq \emptyset$ iff K' has a model with $\pi_i(C)^{\mathcal{I}'_i} \neq \emptyset$. If K has a model $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_E\}_{E \in \mathcal{E}_{ij}} \rangle$, we can construct a model $\mathcal{I}' = \langle \{\mathcal{I}'_i\}, \{r_{ij}\} \rangle$ of K' in the following way:

- For every i, let $\Delta^{\mathcal{I}'_j} \leftarrow \Delta^{\mathcal{I}_j}$ and ${\top'_i}^{\mathcal{I}'_i} = \Delta^{\mathcal{I}_j}$;
- For every i and for every i-name t occurring in L_i, let $\pi_i(t)^{\mathcal{I}'_i} = t^{\mathcal{I}_i}$;
- For every $E \in \mathcal{E}_{ij}$ and every $(x, y) \in r_E$, 1) if $r_{ji}(y) = \emptyset$, then add a new individual y' in $\Delta^{\mathcal{I}'_i}$ and (y, y') to r_{ji}; 2) else add $(x, r_{ji}(y))$ to $\pi_i(E)^{\mathcal{I}'_i}$ (note that r_{ji} is one-to-one);
- For every j-concept D occurring in L_i, let $\pi_i(D)^{\mathcal{I}'_i} = r_{ji}(D^{\mathcal{I}_j})$;
- For every $\mathcal{E}_{ij} \neq \emptyset$, let ${\top_{ji}}^{\mathcal{I}'_i} = r_{ji}(\Delta^{\mathcal{I}'_j})$;
- If $C^{\mathcal{I}_i} \neq \emptyset$, then $\pi_i(C)^{\mathcal{I}'_i} = C^{\mathcal{I}'_i} \cap {\top'_i}^{\mathcal{I}'_i} = C^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_i} = C^{\mathcal{I}_i} \neq \emptyset$

On the other hand, if K' has a model $\mathcal{I}' = \langle \{\mathcal{I}'_i\}, \{r_{ij}\} \rangle$, we can construct a model $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_E\}_{E \in \mathcal{E}_{ij}} \rangle$ of K in the following way:

- For every i, let $\Delta^{\mathcal{I}_j} = {\top'_i}^{\mathcal{I}'_i}$;
- For every i and for every i-name t occurring in L_i, let $t^{\mathcal{I}_i} = \pi_i(t)^{\mathcal{I}'_i}$;
- For each $E \in \mathcal{E}_{ij}$ and every $(x, y) \in \pi_i(E)^{\mathcal{I}'_i}$, add $(x, r_{ji}^{-}(y))$ to r_E;
- If $\pi_i(C)^{\mathcal{I}'_i} \neq \emptyset$, it is easy to verify that $C^{\mathcal{I}_i} = \pi_i(C)^{\mathcal{I}'_i}$, thus $C^{\mathcal{I}_i} \neq \emptyset$. $\square$

Therefore, a concept satisfiability problem (hence a concept subsumption problem) in one-way binary $\mathcal{E}$-connections can be reduced to an equivalent problem in DDL with bridge rules between concepts. Together with the possible reduction from DDL to $\mathcal{E}$-connections, we have the theorem:

Theorem 4.5 *One-way binary $\mathcal{E}$-connections with inverse links, as presented in* (Kutz et al., 2004; Grau, 2005)*, is as expressive as DDL with bridge rules between concepts as presented in* (Borgida and Serafini, 2002)*, when each module is in $\mathcal{SHIQ}$.*

Discussion: Despite that the mutual reduction between DDL as in (Borgida and Serafini, 2002) and $\mathcal{E}$-connections is possible, there are notable differences between their intuitions. Bridge rules are *intended* to simulate general concept inclusions (GCI), hence should not be seen only as a special type of $\mathcal{E}$-connections. For example, empty $\mathcal{E}$-connection r_E is allowed, while GCIs between satisfiable concepts enforce restrictions on non-empty interpretations. Arbitrary "link" relations (when acting as bridge rules) may not preserve concept unsatisfiability among different modules which may result in some reasoning difficulties (Bao et al., 2006c). Furthermore, subset relations (between concept interpretations) is transitive, while links in general are not transitive[14]. Recent study on DDL reveals that arbitrary domain relations should be avoided, e.g., they should be one-to-one (Serafini et al., 2005a; Bao et al., 2006c) and non-empty (Stuckenschmidt et al., 2006). It is also not possible to translate DDL bridge rules between roles into $\mathcal{E}$-Connections axioms. Hence, we believe DDL (with its recent advance) and $\mathcal{E}$-Connections actually cover different application scenarios, and thus are complementary in their expressivity.

4.4.3 Limitations of Existing Approaches

4.4.3.1 DDL

Distributed concept correspondence between two modules in DDL covers some important scenarios of modular ontologies. However, DDL may present semantic difficulties in some situations. While DDL bridge rules are intended to simulate *concept inclusions* (Borgida and Serafini, 2002; Bouquet et al., 2003), arbitrary modelling with bridge rules may lead to undesired semantics, such as in the *Subsumption Propagation problem* and the *Inter-module Unsatisfiability problem*, as noted in (Grau et al., 2004b; Grau, 2005):

Example 4.7 Subsumption Propagation: A KB Σ_d includes modules $L_{\{1,2,3\}}$, each with an empty TBox; bridge rules $B_{12} = \{1 : \mathsf{Bird} \xrightarrow{\sqsupseteq} 2 : \mathsf{Fowl}\}$, $B_{23} = \{2 : \mathsf{Fowl} \xrightarrow{\sqsupseteq} 3 : \mathsf{Chicken}\}$. The relation $1 : \mathsf{Bird} \xrightarrow{\sqsupseteq} 3 : \mathsf{Chicken}$ is not inferred since bridge rules B_{13} is not specified, nor can be inferred.

[14]Transitive links (Parsia and Grau, 2005) are "punning" based, which is rather syntactical sugar. Also note that for $\mathcal{E}$-connections to simulate concept inclusions, the transitivity of links have to be explicitly declared between every pair of modules.

Note that bridge rules may be inferred between the *same* pair of modules. For example, if $1 : A \xrightarrow{\sqsubseteq} 2 : B$ and $2 : B \sqsubseteq 2 : C$, it can be inferred that $1 : A \xrightarrow{\sqsubseteq} 2 : C$. Intra-module subsumption may also be reused in some particular cases. For example, if $1 : A \sqsubseteq 1 : B$, $1 : A \xrightarrow{\sqsupseteq} 2 : C$ and $1 : B \xrightarrow{\sqsubseteq} 2 : D$, it can be inferred that $2 : C \sqsubseteq 2 : D$ (Serafini and Tamilin, 2005b). However, Example 4.7 shows that in general bridge rules in DDLs are not transitively reusable, thereby are restricted for many application scenarios, for instance, ontology alignment composition (Zimmermann and Euzenat, 2006).

Example 4.8 Inter-module Unsatisfiability(Grau et al., 2004b; Grau, 2005): DDLs may not detect unsatisfiability across ontology modules. A KB Σ_d includes modules $L_{\{1,2\}}$, $L_1 = \{1 : \mathsf{Bird} \sqsubseteq 1 : \mathsf{Fly}\}$, $L_2 = \{2 : \mathsf{Penguin} \sqsubseteq \top\}$, $B_{12} = \{1 : \mathsf{Bird} \xrightarrow{\sqsupseteq} 2 : \mathsf{Penguin}, 1 : \neg\mathsf{Fly} \xrightarrow{\sqsupseteq} 2 : \mathsf{Penguin}\}$. Penguin is still satisfiable in Σ_d since such a distributed model exists:

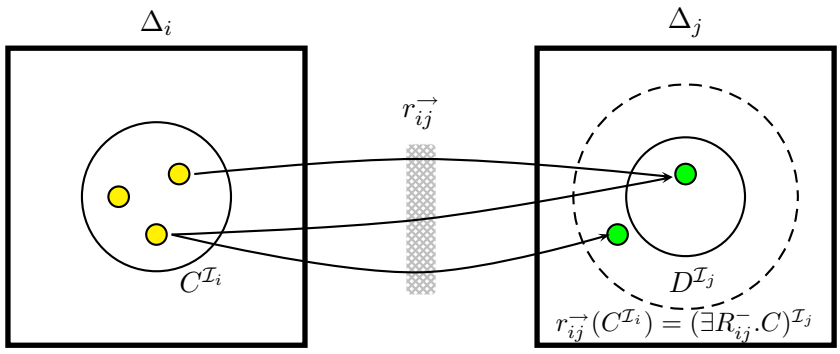
- m_1: $\Delta_1 = \{a, b\}$, $\mathsf{Bird}^{\mathcal{I}_1} = \mathsf{Fly}^{\mathcal{I}_1} = \{a\}, (\neg\mathsf{Fly})^{\mathcal{I}_1} = \{b\}$
- m_2: $\Delta_2 = \{x\}$, $\mathsf{Penguin}^{\mathcal{I}_2} = \{x\}$
- r_{12}: $\{\langle a, x\rangle, \langle b, x\rangle\}$

Such difficulties are rooted in the difference between DDL bridge rules and concept inclusion. DDL has implicit local domain disjointness assumption, i.e., individuals in each local domain are private to that domain, and DDL semantics does not take into account if individuals in different local domains may represent the same physical world object. Therefore, a bridge rule, while intended to simulate concept inclusion, cannot be read directly as concept inclusion, such as $i : A \sqsubseteq j : B$. Instead, it must be read as a classic DL axiom in the following way (Borgida and Serafini, 2002):

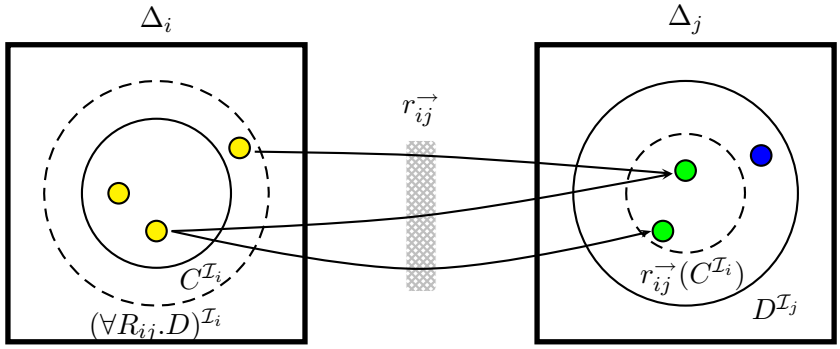
- $i : A \xrightarrow{\sqsubseteq} j : B \Rightarrow (i : A) \sqsubseteq \forall R_{ij}.(j : B)$
- $i : A \xrightarrow{\sqsupseteq} j : B \Rightarrow (j : B) \sqsubseteq \exists R_{ij}^{-}.(i : A)$

where R_{ij} is a new role representing correspondences B_{ij} between L_i and L_j. Such translations are best understood as shown in Figure 4.5.

Therefore, for the given subsumption propagation example, if $B_{13} = \emptyset$, entailment $\mathsf{Chicken} \sqsubseteq \exists R_{13}^{-}.\mathsf{Bird}$ is not always true. For the inter-module unsatisfiability problem, concept Penguin ($\sqsubseteq \exists R_{12}^{-}.(\mathsf{Fly}) \sqcap \exists R_{12}^{-}.(\neg\mathsf{Fly})$) is satisfiable.



(a) $C \stackrel{\sqsupseteq}{\Longrightarrow} D$



(b) $C \stackrel{\sqsubseteq}{\Longrightarrow} D$

Figure 4.5 Semantics of DDL Bridge Rules

Thus, the semantics of DDL are designed to simulate concept inclusion with a special type of roles, i.e., bridge rules. However, in the absence of a principled approach to avoid arbitrary domain relation interpretations for bridge rules, all semantic relations (bridge rules) between DDL modules are *localized* to pairs of modules that are bridged by the rules in question. Consequently, semantic relations between a pair of DDL modules cannot be safely reused by other modules, thereby precluding general subsumption propagation, and more generally, module transitive reusability. Note further that in order to enable distributed (not necessarily exact) reasoning in general, a DDL KB needs explicit declaration of domain relations between *each* pair of modules, leading to a blowup in the number of bridge rules, with the attendant inefficiency and increased risk of inconsistencies.

(Serafini et al., 2005a) has asserted that the inter-module unsatisfiability difficulty is the result of incomplete modelling. They have argued that it can be eliminated if extra information,

for example, $1 : \neg Bird \xrightarrow{\sqsubseteq} 2 : \neg Penguin$ and $1 : Fly \xrightarrow{\sqsubseteq} \neg 2 : Penguin$, is added to guarantee one-to-one domain relations. Our investigation reveals a more general result: one-to-one domain relations can guarantee that reasoning over DDL always yields the same result as that obtained from an integrated ontology when bridge rules are replaced with general concept inclusions (GCI). First, we have the definition:

Definition 4.8 *A domain relation r_{ij} for bridge rules B_{ij} is said to be* one-to-one *if it is an injective partial function, i.e., if $(x_1, y) \in r_{ij} \wedge (x_2, y) \in r_{ij} \rightarrow x_1 = x_2$, and $(x, y_1) \in r_{ij} \wedge (x, y_2) \in r_{ij} \rightarrow y_1 = y_2$.*

An integration process from a DDL ontology to an ordinary (global) DL ontology is given in (Borgida and Serafini, 2002). For a DDL ontology $\{L_i\}$, the global DL (GDL) ontology is defined as follows:

- There is a new top concept $\top_g$ and a new bottom concept $\bot_g$ in GDL.
- The primitive concepts of GDL consist of $i : A$ obtained from primitive concepts or constant concepts A (such as $\top_i$ and $\bot_i$) of L_i
- The primitive roles of GDL include $i : p$ obtained from primitive or constant roles p of L_i

The mapping $\#()$ from concepts/roles in L_i to concepts/roles in GDL is defined as follows: for atomic concepts, roles, and individuals $i : M$, $\#(i : M) = i : M$; for a complex concept constructor ρ with k arguments, $\#(i : \rho(X_1, ..., X_k)) = \top_i \sqcap \rho(\#(X_1), ..., \#(X_k))$. For example, $\#i : (\forall p.C) = \top_i \sqcap \forall(i : p).(\top_i \sqcap i : C)$.

Applying $\#()$ to a DDL knowledge base $\Sigma = \langle\{L_i\}, \{B_{ij}\}\rangle$, we get an integrated GDL (Borgida and Serafini, 2002) $\#(\Sigma)$ that contains:

- $\#(i : A) \sqsubseteq \#(i : B)$ for all $i : A \sqsubseteq B \in L_i$
- $\bot_i \sqsubseteq \bot_g$
- $\#(i : A) \sqsubseteq \top_i$ for each atomic concept A of L_i
- Axioms that ensure the domain and range of any i-role to be $\top_i$: $\top_i \sqsubseteq \forall(i : s).\top_i$, $\neg\top_i \sqsubseteq \forall(i : s).\bot_g$

However, in contrast to the approach taken in (Borgida and Serafini, 2002), we will translate bridge rules in DDL as GCIs in GDL. Hence, in addition to the above, $\#(\Sigma)$ will include:

- $\#(i:C) \sqsubseteq \#(j:D)$ for all $i:C \xrightarrow{\sqsubseteq} j:D \in B_{ij}$
- $\#(i:C) \sqsupseteq \#(j:D)$ for all $i:C \xrightarrow{\sqsupseteq} j:D \in B_{ij}$

Since the motivation of DDL bridge rules is to simulate concept subsumption as mentioned in DDL proposals (Borgida and Serafini, 2002; Bouquet et al., 2003; Serafini et al., 2005a), we believe that GCIs offer a more appropriate translation for bridge rules in comparing the result of reasoning in the distributed setting with that of the centralized setting. Note that the semantic difficulties of DDL under incomplete modelling is actually due to the semantic differences between concept subsumptions (i.e., GCIs) and bridge rules (as shown in the Examples 4.7 and 4.8). The following theorem reveals that the domain relations being one-to-one is a sufficient condition for exact reasoning in DDL if bridge rules are intended to represent inter-module concept inclusions (proof can be found in appendix).

Theorem 4.6 *Suppose* $\Sigma = \langle \{L_i\}, \{B_{ij}\} \rangle$ *is a DDL KB, where none of* L_i *uses role constants or role constructors, and all domain relations in all models of* Σ *are one-to-one, then*

- $\#(\Sigma) \models \#(i:X) \sqsubseteq \#(i:Y)$ *if and only if* $\Sigma \models_d i:X \sqsubseteq i:Y$
- $\#(\Sigma) \models \#(i:X) \sqsubseteq \#(j:Y)$ *if and only if* $\Sigma \models_d (i:X \xrightarrow{\sqsubseteq} j:Y \text{ or } (j:Y \xrightarrow{\sqsupseteq} i:X)$

At present, there is no principled approach in DDL to specify such domain relations. Adding $\neg C \xrightarrow{\sqsubseteq} \neg D$ for each $C \xrightarrow{\sqsupseteq} D$, as suggested in (Serafini et al., 2005a), does not necessarily result in injective (and hence, also not one-to-one) domain relations for every inter-module concept relations.

Example 4.9 :A KB Σ_d includes modules $L_{\{1,2\}}$, TBox of L_1 is $\{\mathsf{Woman} \equiv \neg\mathsf{Man}\}$, TBox of L_2 is $\{\mathsf{Girl} \equiv \neg\mathsf{Boy}\}$; bridge rules $B_{12} = \{1:\mathsf{Man} \xrightarrow{\sqsupseteq} 2:\mathsf{Boy}\}$. According to (Serafini et al., 2005a), we should also add $\neg 1:\mathsf{Man} \xrightarrow{\sqsubseteq} \neg 2:\mathsf{Boy}$ i.e. $1:\mathsf{Woman} \xrightarrow{\sqsubseteq} 2:\mathsf{Girl}$ to B_{12}. However, that does not rule out the possibility of a Girl object being both an image of a Man object and a Woman object, neither ensure one-to-one correspondence between Man objects and Boy objects.

Example 4.10 :(adopted from (Stuckenschmidt et al., 2006)) Module L_1 entails $\top \sqsubseteq 1 : \mathsf{Car}$, module L_2 entails $\mathsf{UsefulThing} \sqsubseteq \neg\mathsf{UselessThing}$, and there are bridge rules $1 : \mathsf{Car} \xrightarrow{\sqsubseteq} 2 : \mathsf{UsefulThing}$ and $1 : \mathsf{Car} \xrightarrow{\sqsubseteq} 2 : \mathsf{UselessThing}$. There is no required new bridge rules to be added according to (Serafini et al., 2005a). However, $1 : \mathsf{Car}$ is not unsatisfiable, since DDL semantics allows empty domain relations.

(Zimmermann, 2007) presents a solution for the subsumption propagation problem. It defines a *global domain of interpretation* Δ_ϵ which plays the role of a "blackboard" for all local domains. For a bridge rule in B_{ij}, instead of interpreting it using the domain relation r_{ij}, it is relying on two "equalizing functions" ϵ_i and ϵ_j from $\Delta^{\mathcal{I}_i}$ and $\Delta^{\mathcal{I}_j}$ (respectively) to Δ_ϵ, such that $i : C \xrightarrow{\sqsubseteq} j : D$ indicates $\epsilon_i(C^{\mathcal{I}_i}) \subseteq \epsilon_j(D^{\mathcal{I}_j})$ and $i : C \xrightarrow{\sqsupseteq} j : D$ indicates $\epsilon_i(C^{\mathcal{I}_i}) \supseteq \epsilon_j(D^{\mathcal{I}_j})$. However, such an approach cannot solve the inter-module unsatisfiability problem.

On the other hand, (Homola, 2007) presents a solution for the inter-module unsatisfiability problem. It defines a special type of bridge rules called *conjunctive onto-bridge rules* $\overset{\sqsupseteq}{\rightsquigarrow}$, such that for two such bridge rules $i : C \overset{\sqsupseteq}{\rightsquigarrow} j : G$ and $i : D \overset{\sqsupseteq}{\rightsquigarrow} j : H$, we must have $r_{ij}(C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}) \supseteq G^{\mathcal{I}_j} \cap H^{\mathcal{I}_j}$. However, this approach cannot solve the subsumption propagation problem.

Decidability of DDL with only (homogenous) bridge rules between concepts is obtained by its mutual reducibility with conventional DL as given in (Borgida and Serafini, 2002). However, it is remained open for the decidability of DDL with homogenous bridge rules between roles and heterogenous bridge rules.

DDL, as presented in (Borgida and Serafini, 2002), meets the *localized semantics*, *decidability* (for bridge rules between concepts) and *directional semantic relations* requirements, but *not* the *exact reasoning* and *monotonicity* (and its special case, *transitive reusability*) requirements. We have shown in section 4.2.5 that P-DL is able to solve those problems of DDL.

4.4.3.2 $\mathcal{E}$-Connections

$\mathcal{E}$-connections allow multiple links between modules and the construction of new concepts while DDL does not. Localized semantics is supported by $\mathcal{E}$-connections since local domains of modules are disjoint (Grau, 2005). Reasoning in $\mathcal{E}$-connections without generalized links is decidable and exact w.r.t a combined TBox of the $\mathcal{E}$-connected ontology, since a concept is satisfiable in the $\mathcal{E}$-connected ontology if and only if there is a combined model for the combined TBox and the concept (Grau et al., 2004b,a).

Directionality: Although links in $\mathcal{E}$-connections are directional, information may propagate through both directions of links, as shown in the following examples.

Example 4.11 : Let $L_1 = \{1 : C \sqsubseteq (\exists E.2 : G),\ (\exists E.2 : H) \sqsubseteq 1 : D\}$, $L_2 = \{2 : G \sqsubseteq 2 : F\}$, where $E \in \mathcal{E}_{12}$ is a link. $1 : C \sqsubseteq 1 : D$ is not entailed by L_1 alone, but is entailed from L_1 and L_2 together.

Example 4.12 : Let $L_1 = \{\top \sqsubseteq \{1 : o_1\} \sqcap (\exists E.2 : C) \sqcap (\exists E.2 : D) \sqcap (= 1E.\top)\}$, $L_2 = \{\top \sqsubseteq \{2 : o_2\}\}$ where o_1, o_2 are nominals, $E \in \mathcal{E}_{12}$ is a link. $2 : C \sqsubseteq 2 : D$ is not entailed by L_2 alone, but is entailed from L_1 and L_2 together.

Hence, $\mathcal{E}$-connections $C^{\mathcal{E}}_{\mathcal{IHN}^+{}_s}(\mathcal{SHOIQ}(D))$ as presented in (Grau, 2005) do not support directionality (as defined in Definition 3.5) in general. Note that it is not contradicting to Theorem 4.5 (the equivalency of DL and $\mathcal{E}$-connections) and the directionality of DDL, since Theorem 4.5 requires that each module to be in a subset of $\mathcal{SHIQ}$.

Transitive Reusability: Since inter-module concept inclusion is not supported by the $\mathcal{E}$-connections syntax, in general modules in an E-connected ontology is not transitively reusable. Although it has been claimed that $\mathcal{E}$-connections can simulate DDL bridge rules which is intended for modeling inter-module concept inclusion, since DDL itself does not support module transitive reusability, $\mathcal{E}$-connections do not support it either. It is even true if transitive links (Parsia and Grau, 2005) are allowed, as shown in the following example.

Example 4.13 : As we have shown earlier, a DDL into bridge rule $C \xrightarrow{\sqsubseteq} D$ can be translated into an axiom $\exists E^-.C \sqsubseteq D$ in $\mathcal{E}$-connections, where E is a link. Suppose we have an ontology O with three modules in $\mathcal{E}$-connections:

- $L_1 = \emptyset$
- $L_2 = \{(\exists E^-.1 : C) \sqsubseteq 2 : D\}$ (i.e., "$C \xrightarrow{\sqsubseteq} D$")
- $L_3 = \{(\exists E^-.2 : D) \sqsubseteq 3 : F\}$ (i.e., "$D \xrightarrow{\sqsubseteq} F$")
- $\mathsf{Trans}(E, (1,2), (2,3), (1,3))$, where E is a transitive link

However, we cannot infer that $(\exists E^-.1 : C) \sqsubseteq 3 : F$ (i.e., "$C \xrightarrow{\sqsubseteq} F$"), since the follow model of O exists:

- $\Delta^{\mathcal{I}_1} = C^{\mathcal{I}_1} = \{x\}$
- $\Delta^{\mathcal{I}_2} = D^{\mathcal{I}_2} = \{w\}$
- $\Delta^{\mathcal{I}_3} = \{y, z\}, F^{\mathcal{I}_3} = \{y\}$
- $r_E = \{(x, z)\}$

Thus, $(\exists E^-.1 : C)^{\mathcal{I}_3} = \{z\} \not\subseteq F^{\mathcal{I}_3}$

$\mathcal{E}$-connections, as presented in (Grau et al., 2004b; Grau, 2005), meets the *localized semantics*, *exact reasoning* and *decidability* requirements, but *not* the *directional semantic relations* and *monotonicity* (and its special case, *transitive reusability*) requirements.

Expressivity: The applicability of $\mathcal{E}$-connections in practice is also limited by its expressivity limitations (Grau, 2005):

- A concept cannot be declared as subclass of another concept in a foreign module thereby ruling out the possibility of asserting inter-module subsumption; a role cannot be declared as sub-role of a foreign role; neither foreign concepts nor foreign roles can be instantiated; cross-module concept conjunction or disjunction are also illegal. Note that such restrictions are presented under both the separated semantics and the free-floating semantics of $\mathcal{E}$-Connections, since the two semantics are equivalent.
- $\mathcal{E}$-connected ontologies have difficulties to be used with OWL importing mechanism, since importing may actually "decouple" the combination and result in inconsistency.
- $\mathcal{E}$-connected ontologies do not allow a name to be used as both a link name and a local role name, nor role inclusions between links and roles. The"punning" approach suggested by (Grau, 2005), where a name can have different interpretations, is rather a syntactical sugar and is limited for many applications[15].

We have shown earlier P-DL may overcome those limitations of $\mathcal{E}$-connections.

[15]For example, suppose E is used both as a local role in L_1 and a link in $\mathcal{E}_{12}$; however, it is unable to define a concept with at most n E-"neighbours", since $\leq nE(\top_1 \sqcup \top_2)$ is an illegal expression in $\mathcal{E}$-Connections.

Applicability: $\mathcal{E}$-connections requires strong domain separation among ontology modules, which may not be satisfied by many application scenarios, for instance, when the modular ontology makes use of an upper-ontology (Seidenberg and Rector, 2006). Several recent studies (Seidenberg and Rector, 2006; D'Aquin et al., 2007) reveal that the ontology segmentation technique based on the $\mathcal{E}$-connections notion (Grau et al., 2005) fails to generate useful modules for many representative ontologies, e.g., GALEN[16], ISWC[17], and TAP[18]. For many of those test cases, the segmentation based on $\mathcal{E}$-connections renders a single module that include most of the content in the original ontology (D'Aquin et al., 2007). Such results, together with the expressivity limitations, suggests that $\mathcal{E}$-connections may be too restrictive to be used as a general modular ontology formalism on the semantic web.

4.4.4 Relation between Other Formalisms and P-DL

Several other modular ontology formalisms can be reduced to P-DL we defined in this chapter.

Several other modular ontology formalisms can be simulated using the semantic importing approach we adopted in this paper.

4.4.4.1 Reduction of $\mathcal{E}$-Connections

In what follows, we will show how one-way $\mathcal{E}$-Connections KBs, as given in (Kutz et al., 2004; Grau et al., 2004b), can be reduced to $\mathcal{SHOIQP}$ KBs (proof is in the appendix).

Theorem 4.7 *One-way $\mathcal{E}$-Connections $C^{\mathcal{E}}_{\mathcal{HQ}}(\mathcal{SHOIQ})$ knowledge bases can be reduced to $\mathcal{SHOIQP}$ knowledge bases.*

It is easy in $\mathcal{SHOIQP}$ to reuse imported transitive roles or symmetric roles without the need for any specifically designed mechanism. $\mathcal{SHOIQP}$ also offers some additional modeling flexibility not provided by $\mathcal{E}$-Connections in its current form. For example, it is possible in $\mathcal{SHOIQP}$ to use foreign roles to define local concepts.

[16] http://www.co-ode.org/galen/
[17] http://annotation.semanticweb.org/iswc/iswc.owl
[18] http://athena.ics.forth.gr:9090/RDF/VRP/Examples/tap.rdf

4.4.4.2 Reduction of DDL

Theorem 4.8 DDL *with homogenous bridge rules between concepts can be reduced to* $\mathcal{SHOIQP}$.

Proof: It is due to the fact that DDL with homogenous bridge rules between concepts can be reduced one-way $\mathcal{E}$-connections which by Theorem 4.7 can be reduced to P-DL $\mathcal{SHOIQP}$. Hence, a reasoning task in DDL with bridge rules between concepts can be reduced to an equivalent reasoning task in $\mathcal{SHOIQP}$. Q.E.D.

Note that DDL with bridge rules between roles has no *equivalent* translation in $\mathcal{SHOIQP}$. In fact, given an i-role R and a j-role S, a translation of the into bridge rule $R \xrightarrow{\sqsubseteq} S$ in P-DL (in package P_i) is $R_{ij}^{-} \circ R \circ R_{ij} \sqsubseteq S$ and a translation of the onto bridge rule $R \xrightarrow{\sqsupseteq} S$ in P-DL is $R_{ij} \circ S \circ R_{ij}^{-} \sqsubseteq R$, which are both beyond the expressivity of $\mathcal{SHOIQP}$.

On the other hand, it is not possible, in general, to reduce $\mathcal{SHOIQP}$ to DDL either. Since DDL allows arbitrary domain relations, compositionally consistent domain relations, as required by $\mathcal{SHOIQP}$, are not realizable in DDL. Hence, there is no reduction from a $\mathcal{SHOIQP}$ knowledge base Σ to a DDL knowledge base $\mathcal{T}$ ensuring that every model of $\mathcal{T}$ can be mapped to a model of $\mathcal{SHOIQP}$. Syntactically, this is manifested by the bridge rule propagation problem in DDL (Bao et al., 2006c), e.g., $i : C \xrightarrow{\sqsubseteq} j : D$ and $j : D \xrightarrow{\sqsubseteq} k : E$ do not entail $i : C \xrightarrow{\sqsubseteq} k : E$. By contrast, Theorem 4.4 enables transitive reuse of subsumptions in $\mathcal{SHOIQP}$ ensuring that such inference difficulties can be avoided.

4.4.4.3 Relation to Semantic Importing of Pan et al.

The proposed P-DL $\mathcal{SHOIQP}$ improves the semantic importing introduced in (Pan et al., 2006) in several significant ways:

- **Increased expressivity**: The use of $\mathcal{SHOIQ}$ instead of the rather restricted $\mathcal{ALC}$ in individual modules and the support for concept, role and nominal importing contribute to this property.
- **Contextualized negation**: This ensures preservation of unsatisfiability.
- **Monotonicity**: This property is not guaranteed by the semantic importing approach of Pan et al. (Pan et al., 2006).

4.4.4.4 Summary

The expressivity of DDL, of $\mathcal{E}$-Connections and of $\mathcal{SHOIQP}$ are summarized in Table 4.4.4.4.

It is possible to simulate the one-to-one domain relations that are required in P-DL by the combination of DDL and E-connections. If we use bridge rules as a special type of $\mathcal{E}$-connections with "≤ 1" cardinality restriction in $\mathcal{E}$-connections, it effectively encodes the one-to-one domain relations. More precisely, for any pair of module i, j, if we denote E as the $\mathcal{E}$-connection for bridge rules from i to j, F as the $\mathcal{E}$-connection for bridge rules from j to i,, the following axioms can be added:

- In module i: $\top_i \sqsubseteq \leq 1E.\top_j$
- In module j: $\top_j \sqsubseteq \leq 1F.\top_i$
- $F = E^-$.

However, such a simulation does not always meet the compositional consistency requirement of P-DL. Therefore, such a combination of DDL and $\mathcal{E}$-connections, while it can solve the inter-module unsatisfiability problem, may fail on some problems that require module transitive reusability, such as the general subsumption propagation problem as outlined in Example 4.7.

Hence, $\mathcal{SHOIQP}$ is appealing in its expressivity power to model many representative scenarios supported by existing approaches. In addition, it also supports the general reuse of roles and nominals which is not supported by any of the existing approach.

The comparison of DDL, $\mathcal{E}$-Connections, and P-DL $\mathcal{SHOIQP}$ is summarized in Table 4.3 and 4.4. For convenience, we also include OWL-DL in Table 4.3.

4.4.5 Syntax Extensions to OWL

There are several efforts to support modular ontologies by extending the OWL syntax.

4.4.5.1 dOWL

dOWL (Avery and Yearwood, 2003) provides a "customization" mechanism to reuse an OWL ontology. dOWL replaces `owl:imports` with `dowl:importsAll` (imports all the elements of an ontology except elements explicitly mentioned that should be left out) and

`dowl:importsNone` (imports none of the elements of an ontology except elements explicitly mentioned that should be reused). In additional, dOWL provides a set of tuning assertions to modify an existing OWL ontology. For example, `dowl:removeClass` indicates removing of a class definition from the imported ontology, and `dowl:removeRange` means to remove the range restriction of a property from that ontology. Thus, dOWL may be used to "patch" an existing ontology, i.e., derive a new ontology by reusing an older version of the ontology. It may also support partial ontology reuse in a limited sense with explicit importing exceptions over `dowl:importsNone`. However, the major limitation of dOWL is the lack of a formal semantics. It is not clear whether the imported ontology and the importing ontology should share a global interpretation, or how to control possible inconsistencies due to the removal of statements from the imported ontology.

4.4.5.2 C-OWL

C-OWL (Bouquet et al., 2003), driven by the notion of DDL (Borgida and Serafini, 2002), provides a set of "bridge rules" to create class, property and individual correspondences between two OWL ontologies. C-OWL is extended from OWL and a previous syntax for DDL, CTXML (ConTeXt Markup Language) (Bouquet et al., 2002), an ontology mapping language across XML-based hierarchies.

Possible bridge rules include $\stackrel{\sqsupseteq}{\longrightarrow}$(onto, or more general than), $\stackrel{\sqsubseteq}{\longrightarrow}$(into, or less general than), $\stackrel{=}{\longrightarrow}$(equivalent), $\stackrel{*}{\longrightarrow}$(compatible) and $\stackrel{\perp}{\longrightarrow}$(disjoint). An XML syntax for C-OWL is given in (Bouquet et al., 2003), and an example for $\mathsf{wine} : \mathsf{ReadWine} \stackrel{\sqsupseteq}{\longrightarrow} \mathsf{vino} : \mathsf{VinoRosso}$ is shown as follows (adapted from (Bouquet et al., 2003)):

```
<cowl:mapping>
  <cowl:sourceOntology rdf:resource="http://example.org/wine.owl"/>
  <cowl:targetOntology rdf:resource="http://example.org/vino.owl"/>
  <cowl:bridgRule cowl:br-type="onto">
    <cowl:sourceConcept rdf:resource="http://example.org/wine.owl#RedWine"/>
    <cowl:targedConcept rdf:resource="http://example.org/vino.owl#VinoRosso"/>
  </cowl:bridgRule>
</cowl:mapping>
```

Instead of using `owl:imports` to connect ontology modules, C-OWL resorts to bridge rules.

The formal semantics of bridge rules is given in DDL interpretations as the follows((Bouquet et al., 2003), Definition 16):

- $i : x \xrightarrow{\sqsupseteq} j : y$: $r_{ij}(x^{\mathcal{I}_i}) \supseteq y^{\mathcal{I}_j}$
- $i : x \xrightarrow{\sqsubseteq} j : y$: $r_{ij}(x^{\mathcal{I}_i}) \subseteq y^{\mathcal{I}_j}$
- $i : x \xrightarrow{=} j : y$: $r_{ij}(x^{\mathcal{I}_i}) = y^{\mathcal{I}_j}$
- $i : x \xrightarrow{*} j : y$: $r_{ij}(x^{\mathcal{I}_i}) \cap y^{\mathcal{I}_j} = \emptyset$
- $i : x \xrightarrow{\perp} j : y$: $r_{ij}(x^{\mathcal{I}_i}) \cap y^{\mathcal{I}_j} \neq \emptyset$

4.4.5.3 $\mathcal{E}$-Connections Syntax

$\mathcal{E}$-Connections (Grau et al., 2004b) extension to OWL allows a property (called link property) to have range from classes of other modules, and such link properties can be used to construct local classes, as shown in the following example (adapted from (Grau, 2005)):

```
<owl:LinkProperty rdf:ID="providesAccommodation">
  <owl:foreignOntology rdf:resource="&acco;"/>
  <rdfs:domain rdf:resource="#Destination"/>
  <rdfs:range>
    <owl:ForeignClass rdf:about="&acco;#Accommodation">
    <owl:foreignOntology rdf:resource="&acco;"/>
  </owl:ForeignClass>
  </rdfs:range>
</owl:LinkProperty> <owl:Class rdf:ID="BudgetDestination">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#providesAccommodation"/>
      <owl:foreignOntology rdf:resource="&acco;"/>
      <owl:someValuesFrom>
        <owl:ForeignClass rdf:about="&acco;BudgetAccommodation"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

$\mathcal{E}$-Connections also allows each ontology module to have a local domain. Editing $\mathcal{E}$-Connected ontology is supported by Swoop (Kalyanpur et al., 2005).

C-OWL and $\mathcal{E}$-Connections both provide formal semantics for their syntactical extensions to OWL, thus are able to support inference, witnessed by DARGO (Serafini and Tamilin, 2005b)(for C-OWL) and Pellet (Sirin and Parsia, 2004)(for $\mathcal{E}$-Connections) implementations. However, such approaches are also limited in several ways:

- Each of them only covers some subset of expressivity that are needed for an expressive modular ontology language (Bao et al., 2006b). C-OWL does not support linking two classes in different modules with properties, and $\mathcal{E}$-Connections has no direct support for inter-module class subsumption.

- Those extensions may experience inference difficulties in certain setting as we analyzed in section 4.4.3.

- Their syntactical extensions are not compatible with existing OWL tools. With the introduction of a new set of syntax symbols on the top of OWL, those extensions are not directly supported by existing OWL tools such as ontology editors, parsers and reasoners.

On the contrast, the suggested modular semantics and syntax for OWL based on the notion of P-DL can overcome those limitations.

Table 4.2 Comparison of Expressivity

Modeling Scenario	**Syntax**	**DDL**	**$\mathcal{E}$-Connections**	$\mathcal{SHOIQP}$
Concept Subsumption	$C \sqsubseteq D$	$\surd$	$\times$	$\surd$
	$C \sqsupseteq D$	$\surd$	$\times$	$\surd$
Concept Negation	$\neg C$	$\times$[2]	$\times$	$\surd$
Concept Conjunction	$C \sqcap D$	$\times$[2]	$\times$	$\surd$
Concept Disjunction	$C \sqcup D$	$\times$[2]	$\times$	$\surd$
Universal Restriction	$\forall R.C$	$\times$[2]	$\surd$	$\surd$
	$\forall P.E$	$\times$[2]	$\times$	$\surd$
Existential Restriction	$\exists R.C$	$\times$[2]	$\surd$	$\surd$
	$\exists P.E$	$\times$[2]	$\times$	$\surd$
Number Restriction[1]	$\leq nR.C$	$\times$[2]	$\surd$	$\surd$
	$\leq nP.E$	$\times$[2]	$\times$	$\surd$
Role Inclusion	$P \sqsubseteq R$	$\surd$	$\times$	$\times$
	$P \sqsupseteq R$	$\surd$	$\times$	$\times$
Role Inverse	P^-	$\times$[2]	$\times$	$\surd$
Transitive Role	$\mathsf{Trans}(P)$	$\times$	$\surd$[3]	$\surd$
Nominal	$\{x\} \rightarrow \{y\}$	$\surd$	$\times$	$\surd$

[1] $\geq$ case is similar.

[2] While they are not directly supported by the DDL syntax, they can be realized in DDL in a limited sense by declaring "alias"'s for foreign names. For example, if C is an i-concept name, we may introduce C' in module j and require that $C \xrightarrow{\sqsubseteq} C'$ and $C \xrightarrow{\sqsupseteq} C'$, and, similarly, for an i-role name P. Thus, name importing can be simulated (in restricted ways) using DDL bridge rules.

[3] With generalized links (Parsia and Grau, 2005).

C is an i-concept, D is a j-concept, E is a k-concept; P is an i-role, R is a j-role, Q is a k-role; x is a i-individual, y is a j-individual; $i \neq j$, $j \neq k$, i, j may be or may not be k. All formulas represent module j's point of view and constructed concepts (roles) are j-terms.

	Localized Semantics	**Exact Reasoning**	**Directional Relation**	**Transitive Reusability**	**Decidability**
OWL-DL	No	Yes	No	Yes	Yes
DDL	Yes	No	Yes	No	Yes[1]
$\mathcal{E}$-Connections	Yes	Yes[2]	No	No	Yes
P-DL	Yes	Yes	Yes	Yes	Yes[3]

[1] yes for bridge rules between concepts, open for bridge rules between roles; [2] yes without generalized links; [3] yes for $\mathcal{SHOIQP}$.

Table 4.3 Comparison of Semantic Properties of Modular Ontology Languages

Modeling Scenario	**Syntax**	**DDL**	**$\mathcal{E}$-Connections**	$\mathcal{SHOIQP}$
Concept Subsumption	$C \sqsubseteq D$	$\surd^1$	$\times^3$	$\surd$
	$C \sqsupseteq D$	$\surd^1$	$\times^3$	$\surd$
Concept Negation	$\neg C$	$\times^2$	$\times^3$	$\surd$
Concept Conjunction	$C \sqcap D$	$\times^2$	$\times^3$	$\surd$
Concept Disjunction	$C \sqcup D$	$\times^2$	$\times^3$	$\surd$
Universal Restriction	$\forall R.C$	$\times^2$	$\surd$	$\surd$
	$\forall P.E$	$\times^2$	$\times^3$	$\surd$
Existential Restriction	$\exists R.C$	$\times^2$	$\surd$	$\surd$
	$\exists P.E$	$\times^2$	$\times^3$	$\surd$
Number Restriction*	$\leq nR.C$	$\times^2$	$\surd$	$\surd$
	$\leq nP.E$	$\times^2$	$\times^3$	$\surd$
Role Inclusion	$P \sqsubseteq R$	$\surd$	$\times$	$\surd$
	$P \sqsupseteq R$	$\surd$	$\times$	$\surd$
Role Inverse	P^-	$\times^2$	$\times$	$\surd$
Transitive Role	$\mathsf{Trans}(P)$	$\times$	$\surd^4$	$\surd$
Nominal	$\{x\} \rightarrow \{y\}$	$\surd$	$\times$	$\surd$

* $\geq$ case is similar.

[1] In restricted ways, presenting difficulties for the transitive reusability problem and inter-module unsatisfiability problem.

[2] While they are not directly supported by the DDL syntax, they can be simulated in DDL in limited sense by declaring "alias" for foreign names. For example, let C ba an i-concept name, we may declare C' in module j and require that $C \xrightarrow{\equiv} C'$, and similarly for an i-role name P. Thus, name importing can be simulated (in restricted ways) using DDL bridge rules.

[3] While they are not directly supported by the $\mathcal{E}$-connections syntax, they can be simulated in $\mathcal{E}$-connections in limited sense by using a special link acting as domain relations in DDL. Details in section 4.4.2.3.

[4] With generalized links (Parsia and Grau, 2005).

C is an i-concept, D is a j-concept, E is a k-concept; P is an i-role, R is a j-role, Q is a k-role; x is an i-individual, y is a j-individual; $i \neq j$, $j \neq k$, i, j may be or may not be k. All formulae are constructed in module j and represent j's point of view.

Table 4.4 Comparison of Expressivity of Modular Ontology Languages

CHAPTER 5. Distributed Reasoning with P-DL

Before P-DL can be used in practice, effective reasoning algorithms need to be developed. This chapter presents several reasoning algorithms for three P-DL languages with increasing expressivity:

- $\mathcal{ALCP}^{-}_{\mathcal{C}}$, i.e., $\mathcal{ALC}$ extended with acyclic concept importing;
- $\mathcal{ALCP}_{\mathcal{C}}$, i.e., $\mathcal{ALC}$ extended with (possibly cyclic) concept importing;
- $\mathcal{SHIQP}$, i.e., $\mathcal{SHIQ}$ extended with concept and role importing.

Part of this chapter was previously published in (Bao et al., 2006a,e, 2007b).

5.1 Overview

Effective use of web ontologies in practice requires support for inference across a loosely coupled federation of multiple, distributed, autonomous ontology modules, without having to combine the ontologies in one location. Current web ontology languages such as OWL (Schreiber and Dean, 2004) and the associated reasoners (e.g., FaCT++ (Tsarkov and Horrocks, 2004) and Pellet (Sirin and Parsia, 2004)) provide at best, very limited capabilities in such a setting. For example, an OWL ontology can "reuse" knowledge from another OWL ontology via the `owl:imports` construct. When one ontology imports another, the result is a union of the two ontologies with a single domain of interpretation. Inference in such a setting requires an *integration* of the relevant ontologies.

Because an OWL ontology can indirectly import knowledge from other OWL ontologies through arbitrarily deep importing chains (which collectively constitute its *importing transitive closure*), querying a small ontology might involve inference over a significant portion of the semantic web. This presents *scalability* challenges in terms of memory, time, and bandwidth

requirements: ontologies with more than a few tens of thousands of concepts are often beyond the capabilities of current reasoners (Gardiner et al., 2006).

The situation is further complicated in applications where *no global knowledge* of all ontology modules is available. For example, in a peer-to-peer setting that is not at all atypical of semantic web applications, each peer has access to only a subset of peers, namely, its local acquaintances (Bonifacio et al., 2004). In addition, many web applications require protection on private information in their ontologies; hence, those applications only provide limited query interfaces instead of exposing their ontologies explicitly (see Chapter 6 for more discussion). In both scenarios, integration of all ontologies is not possible.

Against this background, this chapter presents federated reasoning algorithms for several modular ontology languages in the P-DL family. The proposed reasoning algorithms can preform reasoning in a peer-to-peer fashion, such that each peer reasoner only requires local knowledge, and the overall reasoning process is enabled by messages exchanged between the federation of those peer reasoners. Hence, no integration of all ontology modules is required. In particular, we explore the following algorithms:

- In section 5.2, we present a synchronized, tableau-based reasoning algorithm for the relatively simple language $\mathcal{ALCP}_{\mathcal{C}}^{-}$, i.e., $\mathcal{ALC}$ extended with acyclic concept importing. This algorithm illustrates the basic features of P-DL reasoning algorithms.

- The acyclicity restriction is removed in section 5.3, hence leading to a reasoning algorithm for $\mathcal{ALCP}_{\mathcal{C}}$.

- Section 5.4 presents an asynchronous, parallel reasoning algorithm for $\mathcal{ALCP}_{\mathcal{C}}$ such that local reasoners can concurrently work on different subtasks in the reasoning process.

- Finally, in section 5.5 we explore an asynchronous reasoning algorithm for the P-DL $\mathcal{SHIQP}$, i.e., $\mathcal{SHIQ}$ extended with (possibly cyclic) concept and role importing.

Although our discussion focuses on the description of federated reasoners for several subsets of $\mathcal{SHIQP}$ for ease of exposition, we note that the approach can be extended in a rather straightforward way to more expressive P-DL logics, namely, $\mathcal{SHOIQP}(D)$ (an extension of $\mathcal{SHIQP}$ that allows the use of nominals and datatypes within individual modules and importing of nominals between modules) based on the techniques introduced for reasoning with

nominals and data types in the centralized setting for $\mathcal{SHOIQ}$ (Horrocks and Sattler, 2005) and $\mathcal{SHOQ}(D)$ (Horrocks and Sattler, 2001).

In this chapter, we focus on algorithmic design rather than on implementation details. The latter may include the communication protocols between the local reasoners and aspects of the process of synchronization and backtracking, such as, e.g., handshaking and acknowledgement protocols, remembering of previous choices, dependency between choices and the token passing protocol. We leave those details to the implementation of the algorithm, that is expected to be influenced by experimental studies for best performance. Some of those techniques have already been applied in popular DL reasoners, e.g., Pellet (Sirin and Parsia, 2004).

The basic idea of tableau algorithms has already been introduced in section 2.2, and P-DL $\mathcal{SHOIQP}$ has been introduced in section 4.2.

5.2 Reasoning in $\mathcal{ALCP}^{-}_{\mathcal{C}}$

One reason for which description logics enjoy good computational properties, e.g., being robustly decidable, is that they have the tree model property (Vardi, 1996; Grädel, 2001), i.e., if the ontology in question is consistent, it has at least one model which has a tree-shaped relational structure. Hence, a tableau algorithm for DL may decide the consistency of an ontology by searching for the existence of such a tree-shaped model, or a *completion graph*[1]. P-DL, as an extension of DL, still enjoys the tree-model property, but in a distributed fashion. If a P-DL ontology is consistent, it has a *distributed model* such that each local model (tableau) of it (for a component module of the ontology) is a forest, and all those local models can be seen as fragments of a conceptual, tree-shaped "global model". The P-DL tableau algorithm is motivated by the desire to discover such a model using a federation of local reasoners, each maintaining a local tableau, by message exchanging between those reasoners.

5.2.1 $\mathcal{ALCP}^{-}_{\mathcal{C}}$

To make this section self-contained, we first briefly introduce the P-DL $\mathcal{ALCP}^{-}_{\mathcal{C}}$, which is a subset of the P-DL $\mathcal{SHOIQP}$ we introduced in the last chapter. An $\mathcal{ALCP}^{-}_{\mathcal{C}}$ ontology has a set of packages each is an $\mathcal{ALC}$ TBox. For every package P_i, we replace the universal top

[1] In some expressive DLs, such as the ones with transitive roles, the completion graph is a tree-shaped skeleton of a model from which the model can be reconstructed. In DLs with nominals, the completion graph may not be a tree but a forest.

($\top$) and global negation ($\neg$) with their contextualized counterparts: contextualized top $\top_i$ and contextualized negation $\neg_i$. The set of names occurring in a package P_i is its *signature* $\mathsf{Sig}(P_i)$.

Every concept or role name t is associated with a *home package*, denoted as $\mathsf{Home}(t)$. A name $t \in \mathsf{Sig}(P_j)$, but has a home package P_i ($i \neq j$) is called an *imported name* in P_j. We say that P_j *imports* $i : t$ and denote it as $P_i \xrightarrow{t} P_j$. If any name with home package P_i is imported into P_j, we say that P_j imports P_i and denote it as $P_i \mapsto P_j$. $\neg_k C$ (hence also $\top_k$) can appear in P_i ($i \neq k$) only if $P_k \mapsto P_i$ and all names in C are in $\mathsf{Sig}(P_k)$.

The *importing closure* P_i^+ of a package P_i contains all packages that are directly or indirectly imported into P_i, such that:

- (direct importing) $P_j \mapsto P_i \Rightarrow P_j \in P_i^+$
- (indirect importing) $P_k \mapsto P_j$ and $P_j \in P_i^+ \Rightarrow P_k \in P_i^+$

An $\mathcal{ALCP}_{\mathcal{C}}^-$ ontology $\Sigma = \langle \{P_i\}, \{P_i \mapsto P_j\}_{i \neq j} \rangle$ has *acyclic importing* relation such that for any $i \neq j$, $P_j \in P_i^+ \rightarrow P_i \notin P_j^+$.

We use P_i^* to denote $P_i \cup P_i^+$. A concept C is *understandable* in a package P_i if the concept is constructed only using names in P_i and concept names in P_i^+.

An $\mathcal{ALCP}_{\mathcal{C}}^-$ ontology has *localized semantics* in the sense that each package has its own local interpretation domain. Formally, for an $\mathcal{ALCP}_{\mathcal{C}}^-$ ontology $\Sigma = \{P_i\}$, a *distributed interpretation* is a tuple $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^+} \rangle$, where $\mathcal{I}_i$ is the *local interpretation* of package P_i, with domain $\Delta^{\mathcal{I}_i}$, and $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ is the *(image) domain relation* for the interpretation of the direct or indirect importing relation from P_i to P_j. For convenience, we use $r_{ii} = \{(x, x) | x \in \Delta^{\mathcal{I}_i}\}$ to denote the identity mapping on the local domain $\Delta^{\mathcal{I}_i}$.

Given i, j, such that $P_i \in P_j^*$, define:

$$r_{ij}(A) \quad = \quad \{y \in \Delta^{\mathcal{I}_j} | \exists x \in A, (x, y) \in r_{ij}\}, \text{ for every } A \subseteq \Delta^{\mathcal{I}_i}.$$

Moreover, let ρ be the equivalence relation on $\bigcup_i \Delta^{\mathcal{I}_i}$ generated by the collection of all domain relations, i.e., the symmetric and transitive closure of the set $\bigcup_{P_i \in P_j^*} r_{ij}$. For every i, j such that $P_i \in P_j^*$, $\rho_{ij} = \rho \cap (\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j})$.

Each of the local interpretations $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}, \cdot^{\mathcal{I}_i} \rangle$ consists of a domain $\Delta^{\mathcal{I}_i}$ and an interpretation function $\cdot^{\mathcal{I}_i}$, which maps every concept name to a subset of $\Delta^{\mathcal{I}_i}$ and every role name to a subset of $\Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$, such that the following equations are satisfied, where R is an i-role name

and C, D are concepts:

$$\begin{aligned}
(C \sqcap D)^{\mathcal{I}_i} &= C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i} \\
(C \sqcup D)^{\mathcal{I}_i} &= C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i} \\
(\neg_j C)^{\mathcal{I}_i} &= r_{ji}(\Delta^{\mathcal{I}_j}) \backslash C^{\mathcal{I}_i} \\
(\exists R.C)^{\mathcal{I}_i} &= \{x \in \Delta^{\mathcal{I}_i} | (\exists y \in \Delta^{\mathcal{I}_i})((x,y) \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i})\} \\
(\forall R.C)^{\mathcal{I}_i} &= \{x \in \Delta^{\mathcal{I}_i} | (\forall y \in \Delta^{\mathcal{I}_i})((x,y) \in R^{\mathcal{I}_i} \rightarrow y \in C^{\mathcal{I}_i})\}
\end{aligned}$$

Note that, when $i = j$, $(\neg_j C)^{\mathcal{I}_i}$ reduces to the usual negation $(\neg_i C)^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \backslash C^{\mathcal{I}_i}$.

A local interpretation $\mathcal{I}_i$ is said to *satisfy* a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. $\mathcal{I}_i$ is called a *model* of P_i, denoted by $\mathcal{I}_i \models P_i$, if it satisfies all axioms in P_i.

Definition 5.1 *An interpretation $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^*} \rangle$ is a* model *of an $\mathcal{ALCP}_{\mathcal{C}}$ KB $\Sigma = \{P_i\}$, denoted by $\mathcal{I} \models \Sigma$, if the following conditions are satisfied.*

1. *For all i, j, r_{ij} is one-to-one, i.e., it is an injective partial function;*
2. Compositional Consistency*: For all i, j, k, $i \neq j$, s.t. $P_i \in P_k^*$ and $P_k \in P_j^*$, we have $\rho_{ij} = r_{ij} = r_{kj} \circ r_{ik}$;*
3. *For every i-concept name C that appears in P_j, we have $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$;*
4. *$\mathcal{I}_i \models P_i$, for every i.*

Note that if $P_j \notin P_i^*$, r_{ji} does not exist even if r_{ij} exists. Moreover, we have that $r_{ij} = r_{ji}^-$ if P_i and P_j mutually import one another. Also note that r_{ij} may not be a total function.

Definition 5.2 *An ontology Σ is* consistent as witnessed by a package *P_i of Σ if P_i^* has a model $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^+} \rangle$, such that $\Delta^{\mathcal{I}_i} \neq \emptyset$. A concept C is* satisfiable as witnessed by *P_i if there is a model of P_i^*, such that $C^{\mathcal{I}_i} \neq \emptyset$. A concept subsumption $C \sqsubseteq D$* is valid as witnessed by *P_i, denoted by $C \sqsubseteq_i D$, if for every model of P_i^*, $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. We use $C \equiv_j D$ as the abbreviation of $C \sqsubseteq_j D$ and $D \sqsubseteq_j D$.*

5.2.2 Distributed Tableaux for $\mathcal{ALCP}^{-}_{\mathcal{C}}$

Before the reasoning process starts, all concepts are converted into negation normal form (NNF), i.e., a form in which negation only occurs before concept names, including local "tops", and there are only j-negations in a package P_j. We use $\dot{\neg}_i C$ to denote the NNF of $\neg_i C$. We can transform formulae in P_j into NNF by applying the following rules:

$\neg_i(\neg_k D) \Rightarrow \top_i \sqcap (D \sqcup \neg_i \top_k)$ $\quad$ $\neg_i C \Rightarrow \top_i \sqcap \neg_j C$, where C is a concept name or a local top,

$\neg_i(C_1 \sqcap C_2) \Rightarrow \neg_i C_1 \sqcup \neg_i C_2$ $\quad$ $\neg_i(C_1 \sqcup C_2) \Rightarrow \neg_i C_1 \sqcap \neg_i C_2$

$\neg_i \exists R.D \Rightarrow \top_i \sqcap \forall R.\neg_j D$, $\quad$ $\neg_i \forall R.D \Rightarrow \top_i \sqcap \exists R.\neg_j D$

$\neg_i \bot \Rightarrow \top_i$ $\quad$ $\neg_i \top_i \Rightarrow \bot$

Lemma 5.1 *For any concept C in a package P_j and for any i such that $P_i \mapsto P_j$, $\dot{\neg}_i C \equiv_j \neg_i C$.*

Proof is in the appendix.

The main idea behind the $\mathcal{ALCP}^{-}_{\mathcal{C}}$ tableau algorithm is to construct multiple, federated local tableaux using only knowledge locally available to each module, instead of creating a single tableau using the integrated ontology resulting by combining all those modules. A set of messages will be exchanged between the local modules to connect the local tableaux by creating partial correspondences between them. Formally, we have:

Definition 5.3 *The set of subconcepts $sub(C)$ of an $\mathcal{ALCP}^{-}_{\mathcal{C}}$ concept C in NNF is inductively defined by:*

$$
\begin{aligned}
sub(A) &= \{A\}, \text{ for a concept name , including a local top concept, or its negation } A \\
sub(C \sqcap D) &= \{C \sqcap D\} \cup sub(C) \cup sub(D) \\
sub(C \sqcup D) &= \{C \sqcup D\} \cup sub(C) \cup sub(D) \\
sub(\exists R.C) &= \{\exists R.C\} \cup sub(C) \\
sub(\forall R.C) &= \{\forall R.C\} \cup sub(C)
\end{aligned}
$$

For every package P_i, we define $C_{\mathcal{T}_i} = \underset{(C \sqsubseteq D) \in \mathcal{T}_i}{\sqcap} (\dot{\neg}_i C \sqcup D)$.

Definition 5.4 *Let P_w be a witness package and D be an $\mathcal{ALCP}^{-}_{\mathcal{C}}$ -concept in NNF w.r.t. P_w, such that D is understandable by P_w. A* distributed tableau for D w.r.t. P_w *is a tuple*

$T = \langle \{T_i\}, \{t_{ij}\}_{P_i \in P_j^+} \rangle$, where each T_i is a local tableau, for $P_i \in P_w^*$, and t_{ij} is the tableau relation from a local tableau T_i to a local tableau T_j. Each local tableau is a tuple $T_i = (\boldsymbol{S}_i, \mathcal{L}_i, \mathcal{E}_i)$, where

- *$\boldsymbol{S}_i$ is a set of individuals,*
- *$\mathcal{L}_w : \boldsymbol{S}_w \rightarrow 2^{sub(D) \cup sub(C_{\mathcal{T}_w})}$ and $\mathcal{L}_i : \boldsymbol{S}_i \rightarrow 2^{sub(C_{\mathcal{T}_i})}$, $i \neq w$, map individuals to corresponding sets of concepts,*
- *$\mathcal{E}_i : \mathsf{NR}_i \rightarrow 2^{\boldsymbol{S}_i \times \boldsymbol{S}_i}$ maps roles to the corresponding sets of pairs of individuals.*

Each tableau relation t_{ij} is a subset of $\boldsymbol{S}_i \times \boldsymbol{S}_j$. Let ρ^t be the symmetric and transitive closure of the set $\bigcup_{P_i \in P_j^*} t_{ij}$. And, for all i, j, such that $P_i \in P_j^*$, set $\rho_{ij}^t = \rho^t \cap (\boldsymbol{S}_i \times \boldsymbol{S}_j)$.

The distributed tableau T should satisfy the following conditions:

(E) *there exists $x \in \boldsymbol{S}_w$, such that $D \in \mathcal{L}_w(x)$;*

(A0) *for every $x \in \boldsymbol{S}_i$, $C_{\mathcal{T}_i} \in \mathcal{L}_i(x)$;*

(A1) *if $C \in \mathcal{L}_i(x)$, then $\neg_i C \notin \mathcal{L}_i(x)$;*

(A2) *if $C_1 \sqcap C_2 \in \mathcal{L}_i(x)$, then $C_1 \in \mathcal{L}_i(x)$ and $C_2 \in \mathcal{L}_i(x)$;*

(A3) *if $C_1 \sqcup C_2 \in \mathcal{L}_i(x)$, then $C_1 \in \mathcal{L}_i(x)$ or $C_2 \in \mathcal{L}_i(x)$;*

(A4) *if $\forall R.C \in \mathcal{L}_i(x)$ and $\langle x, y \rangle \in \mathcal{E}_i(R)$, then $C \in \mathcal{L}_i(y)$;*

(A5) *if $\exists R.C \in \mathcal{L}_i(x)$, then, there exists $y \in \boldsymbol{S}_i$, such that $\langle x, y \rangle \in \mathcal{E}_i(R)$ and $C \in \mathcal{L}_i(y)$;*

(B1) *t_{ij} is a one-to-one partial function, for all i, j;*

(B2) *$\rho_{ij}^t = t_{ij} = t_{kj} \circ t_{ik}$ for all i, j, k, $i \neq j$, such that $P_i \in P_k^*$ and $P_k \in P_j^*$;*

(B3) *if C is an i-concept name, $P_i \xrightarrow{C} P_j, i \neq j$, then*

$$(\forall x' \in \boldsymbol{S}_j)((\exists x \in \boldsymbol{S}_i)(\langle x, x' \rangle \in t_{ij} \text{ and } C \in \mathcal{L}_i(x)) \text{ iff } C \in \mathcal{L}_j(x'));$$

Explanation: Conditions (A0)-(A5) are similar to the ones used in the tableau definition of $\mathcal{ALC}$ (Horrocks et al., 1999). Intuitively, Conditions (B1) and (B2) ensure that domain

relations are one-to-one and compositionally consistent. On the other hand, Condition (B3) ensures that $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$, for any concept name C.

It should be noted that the correspondence of individuals across multiple local tableaux is only *partial*. Some individuals in a local tableau may not be connected to any individuals in another local tableau. This conforms with the localized semantics of P-DL stipulating that each ontology module has its own interpretation domain.

The following lemma establishes the correspondence between concept satisfiability, hence, also between TBox consistency and concept subsumption, and the existence of a tableau for that concept in $\mathcal{ALCP}^{-}_{\mathcal{C}}$ (proof is in the appendix):

Lemma 5.2 *Let D be an $\mathcal{ALCP}^{-}_{\mathcal{C}}$ concept that is understandable by an $\mathcal{ALCP}^{-}_{\mathcal{C}}$ package P_w. Then D is satisfiable as witnessed by P_w iff D has a distributed tableau w.r.t. P_w.*

5.2.3 A Tableau Algorithm for $\mathcal{ALCP}^{-}_{\mathcal{C}}$

We now proceed to describe a sound and complete algorithm to determine the existence of a tableau for an $\mathcal{ALCP}^{-}_{\mathcal{C}}$ concept w.r.t. a witness package. The algorithm allows each local tableau to be created and maintained by a local reasoner. Thus, reasoning is carried out by a federation of reasoners that communicate with each other via messages instead of a single reasoner over an integrated ontology.

5.2.3.1 Distributed Completion Graph

The algorithm works on a *distributed completion graph*, which is a partial finite description of a tableau. A distributed completion graph is $G = \{G_i\}$, where $\{G_i\}$ is a set of *local completion graphs*. Each *local completion graph* $G_i = \langle V_i, E_i, \mathcal{L}_i \rangle$ consists of a finite set of finite trees, i.e., a forest, where V_i and E_i are the corresponding sets of nodes and edges respectively, and of a function $\mathcal{L}_i$, that assigns labels to nodes and edges in G_i. Each node x in V_i represents an individual in the corresponding tableau, denoted as $i : x$, and is labeled with $\mathcal{L}_i(x)$, a set of concepts of which x is a member. Each edge $\langle x, y \rangle \in E_i$ represents a set of role memberships in the tableau, and is labeled with $\mathcal{L}_i(\langle x, y \rangle)$, the corresponding set of role names.

If $R \in \mathcal{L}_i(\langle x, y \rangle)$, y is said to be a *local R-successor* of x and x is said to be a *local R-predecessor* of y. *Local ancestors* and *local descendants* of a node are defined in the usual manner.

Every node x has associated with it a node $origin(x)$, which, informally speaking, is the "original" node from which x is "copied". If $origin(i:x) = origin(j:y)$ and $P_i \in P_j^+$, we say that node y in G_j is an *image* of node x in G_i, denoted by $y = x^{i \rightarrow j}$, that node x is a *pre-image* of y, denoted by $x = y^{i \leftarrow j}$, and that there is a *graph relation* $\langle x, y \rangle$.[2]

A typical distributed completion graph is shown in Figure 5.1. Dotted edges in the graph represent graph relations. If we merge nodes of the same origin, all local graphs may, in fact, be merged into a tree-shaped global graph. Tree(s) in a local graph are fragments of the corresponding (virtual) global tree. In fact, the virtual global tree represents a conceptual model for the ontology resulting by integrating all modules.

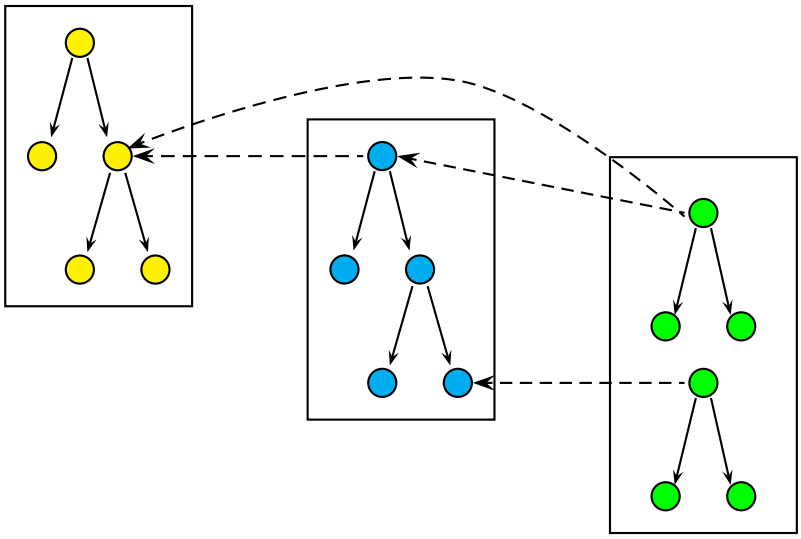


Figure 5.1 $\mathcal{ALCP}_{\mathcal{C}}^{-}$ Distributed Tableaux Example

5.2.3.2 Distributed Tableau Expansion

A distributed $\mathcal{ALCP}_{\mathcal{C}}^{-}$ completion graph is constructed by applying a set of tableau expansion rules and by exchanging messages between local reasoners. The $\mathcal{ALCP}_{\mathcal{C}}^{-}$ expansion rules are adapted from the $\mathcal{ALC}$ expansion rules (see Section 2.2.2) as follows: Each module is only locally internalized, instead of being globally internalized with respect to a combined TBox. A local completion graph can create "copies" of its local nodes in another local completion graph, as needed, during an expansion.

[2]Sometimes we use the same name with different prefixes for two nodes to indicate that they have the same origin, e.g., $i:x$ and $j:x$ means $origin(i:x) = origin(j:x)$. We may omit the prefix when it is clear from the context.

A *concept reporting message* propagates concept labels of a node to the corresponding image node or pre-image node. We use $S \mathrel{+}= X$ to denote the operation of adding the elements of the set X to a set S, i.e., the operation $S = S \cup X$. Using this notation, we have:

- A *forward concept reporting message* $r^{i \to j}(x, C)$ executes the following action: **if** there is a node $x' \in V_j$, such that $origin(x) = origin(x')$ and $C \notin \mathcal{L}_j(x')$, **then** $\mathcal{L}_j(x') \mathrel{+}= \{C\}$.

- A *backward concept reporting message* $r^{j \leftarrow i}(x, C)$ executes the following actions: **if** there is a node $x' \in V_j$, such that $origin(x) = origin(x')$, then do $\mathcal{L}_j(x') \mathrel{+}= \{C\}$ if $C \notin \mathcal{L}_j(x')$ and $C \neq \top_j$; **else** create a node x' in V_j with $origin(x') = origin(x)$, and do $\mathcal{L}_j(x') \mathrel{+}= \{C\}$ if $C \neq \top_j$.

Some nodes in the graph may be *blocked*, as will be explained later. The expansion rules are:

- CE-rule: **if** $C_{\mathcal{T}_i} \notin \mathcal{L}_i(x)$, **then** $\mathcal{L}_i(x) \mathrel{+}= C_{\mathcal{T}_i}$.

- $\sqcap$-rule: **if** $C_1 \sqcap C_2 \in \mathcal{L}_i(x)$, x is not blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}_i(x)$, **then** $\mathcal{L}_i(x) \mathrel{+}= \{C_1, C_2\}$.

- $\sqcup$-rule: **if** $C_1 \sqcup C_2 \in \mathcal{L}_i(x)$, x is not blocked, and $\{C_1, C_2\} \cap \mathcal{L}_i(x) = \emptyset$, **then** $\mathcal{L}_i(x) \mathrel{+}= \{C\}$ for some $C \in \{C_1, C_2\}$.

- $\exists$-rule: **if** $\exists R.C \in \mathcal{L}_i(x)$, x is not blocked and x has no local R-successor y of x in G_i with $C \in \mathcal{L}_i(y)$, **then** create a new node y with $origin(y) = y$, $\mathcal{L}_i(\langle x, y \rangle) = \{R\}$ and $\mathcal{L}_i(y) = \{C\}$.

- $\forall$-rule: **if** $\forall R.C \in \mathcal{L}_i(x)$, x is not blocked and there is a local R-successor y of x in G_i with $C \notin \mathcal{L}_i(y)$, **then** $\mathcal{L}_i(y) \mathrel{+}= \{C\}$.

- CPush-rule: **if** $C \in \mathcal{L}_i(x)$, where C is an i-concept name, with $P_i \xrightarrow{C} P_j$, x is not blocked and there exists an $x' = x^{i \to j} \in V_j$, such that x' is not blocked, with $C \notin \mathcal{L}_j(x')$, **then** transmit $r^{i \to j}(x, C)$.

- CReport-rule: **if** $C \in \mathcal{L}_i(x)$, where C is $\top_j$ or a j-concept name, x is not blocked and $x^{j \leftarrow i}$ does not exist or ($x' = x^{j \leftarrow i}$ exists, x' is not blocked and $C \notin \mathcal{L}_j(x')$), **then** transmit $r^{j \leftarrow i}(x, C)$.

- r-rule: **if** $origin(i : x) = origin(j : x')$, x, x' are not blocked, and there exists k such that $P_i \in P_k^+$, $P_k \in P_j^+$ and there is no $k : x''$ with $origin(j : x') = origin(k : x'')$, **then** transmit $r^{k \leftarrow j}(x', \top_k)$.

Explanation: The $\sqcap$-, $\sqcup$-, $\exists$-, $\forall$- and CE- rules are adaptations of the corresponding $\mathcal{ALC}$ expansion rules. The r-rule serves to ensure the compositional consistency of domain relations according to tableau Property (B2). The CPush- and CReport- rules are introduced to ensure $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$, for every i-concept name C, according to tableau Property (B3). The reader will get an even better feeling for the adoption of these rules while studying the soundness and completeness lemmas for the distributed algorithm, that will be presented later.

A distributed completion graph is *complete* if no $\mathcal{ALCP}_{\mathcal{C}}^-$ expansion rule can be applied to it, and it is *clash-free* if there is no x in any local completion graph G_i, such that both C and $\neg_i C$ are in $\mathcal{L}_i(x)$, for some concept C.

For a satisfiability query of a concept C as witnessed by a package P_w, where C is understandable by P_w, a local completion graph G_w, with an initial node x_0, such that $origin(x_0) = x_0$ and $\mathcal{L}_w(x_0) = \{C\}$, will be created first. The $\mathcal{ALCP}_{\mathcal{C}}^-$ tableau expansion rules will be applied until a complete and clash-free distributed completion graph is found or until all search efforts for such a distributed completion graph fail.

5.2.3.3 Blocking and Backtracking

When only acyclic importing among packages is considered, the termination and correctness of the algorithm can be obtained by using *subset blocking* and *token passing*.

Subset blocking has been applied in the $\mathcal{ALC}$ tableau algorithm. The motivation behind subset blocking is the detection of cycles in tableau expansions, as illustrated in the following example.

Example 5.1 Subset Blocking: Suppose we have two packages:

$P_1 : \{\top_1 \sqsubseteq 1 : A, \top_1 \sqsubseteq \exists(1 : R).(2 : B)\}$,

$P_2 : \{\top_2 \sqsubseteq \exists(2 : P).(2 : B)\}$;

The reasoning task is to check the consistency of P_1, therefore P_1 is the witness package. In Figure 5.2, G_1 and G_2 are local completion graphs for the two packages. In G_1, due to the $\exists$- and CE-rule, we may further expand w with an infinite chain of nodes, each with

labels $\{B, A, \exists R.B\}$. Such a cycle may be detected using the subset blocking technique: since $\mathcal{L}_1(w) \subseteq \mathcal{L}_1(y)$, it is not necessary to further expand $1 : w$ otherwise we will just repeat the expansion we did on $1 : y$. We say $1 : w$ is locally *blocked* by $1 : y$. Similarly, $2 : z$ is locally blocked by $2 : y'$. (End)

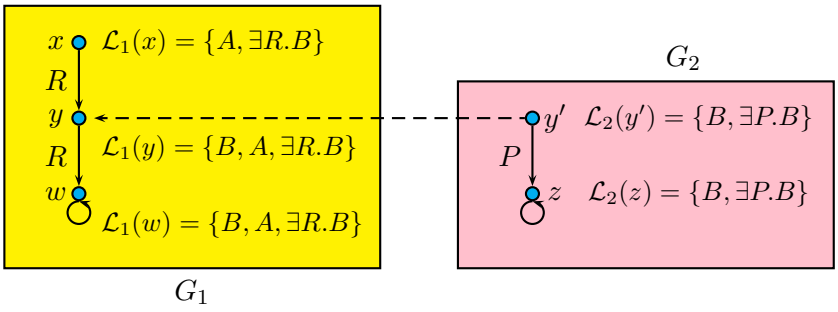


Figure 5.2 Subset Blocking Example in $\mathcal{ALCP}^{-}_{\mathcal{C}}$

Formally, we have:

Definition 5.5 (Subset Blocking) *For a distributed completion graph of an $\mathcal{ALCP}^{-}_{\mathcal{C}}$ ontology, a node x is* directly blocked *by a node y, if both x and y are in the same local completion graph G_i, for some i, y is a local ancestor of x, and $\mathcal{L}_i(x) \subseteq \mathcal{L}_i(y)$. Node x is* indirectly blocked *by a node y if one of x's local ancestors is directly blocked by y. Node x is* blocked *by y if it is directly or indirectly blocked by y.*

Subset blocking in $\mathcal{ALCP}^{-}_{\mathcal{C}}$ only depends on the *local* information in completion graphs, i.e., a local completion graph determines blocking regardless of whether a node has any image or preimage nodes in any other local completion graphs and irrespective of the labels of those nodes. Thus, a node is blocked only by its *local* ancestors. As we will show in Example 5.5, subset blocking is required to guarantee the correctness of reasoning.

Token passing is used to coordinate expansions in different local completion graphs, as illustrated by the following example.

Example 5.2 : Suppose we have two packages:

$P_1 : \{\top_1 \sqsubseteq (2 : D_3), \top_1 \sqsubseteq ((2 : D_1) \sqcap \exists(1 : R).(1 : C) \sqcap \forall(1 : R).(\neg_1(1 : C))) \sqcup \neg_1(2 : D_2)\}$

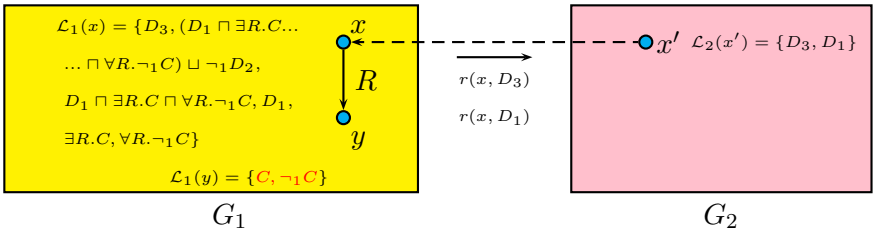
$P_2 : \{(2 : D_1) \sqsubseteq (2 : D_2)\}$

The reasoning task is to check the consistency of P_1. Figure 5.3 (a) and (b) show the running of $\mathcal{ALCP}^{-}_{\mathcal{C}}$ tableau expansions in one scenario, which will be referred to as Scenario 1. In (a),

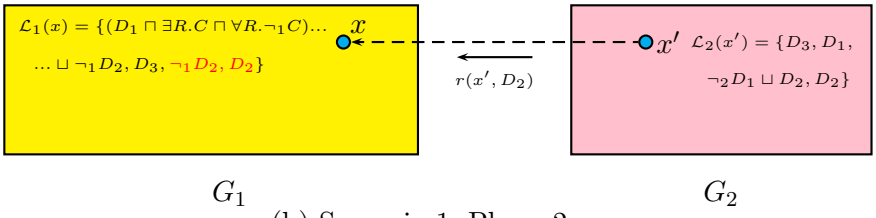
we first apply the CE-rule and $\sqcap$-rule, adding D_3 into $\mathcal{L}_1(x)$, which results in the firing of the CPeport-rule, the message $r^{2\leftarrow 1}(x, D_3)$ and the creation of x'. Next, due to the CE- and $\sqcup$-rule, we may choose adding $D_1 \sqcap \exists R.C \sqcap \forall R.\neg_1 C$ into $\mathcal{L}_1(x)$, which leads to a reporting message $r^{2\leftarrow 1}(x, D_1)$. Further applying expansion in G_1, we will generate node y and find a clash in $\mathcal{L}_1(y)$.

In Figure 5.3 (b), due to the clash, we will restore the status of node $1 : x$, as it had been before the choice in the $\sqcup$-rule was made, and try the next choice, i.e., adding $\neg_1 D_2$ into $\mathcal{L}_1(x)$. In the meantime, local completion graph G_2 may apply the CE- and $\sqcup$-rules and add D_2 into $\mathcal{L}_2(x')$. Since, by a domain relation that was established before the clash in Phase 1, x is an image node of x' and P_1 imports D_2, G_2 will apply the CPush-rule and send the message $r^{2\rightarrow 1}(x', D_2)$, which will lead to a clash in $\mathcal{L}_1(x)$. Hence, according to Scenario 1, we may assert that P_1 is not consistent since all choices in the tableau expansion of G_1 lead to clashes.

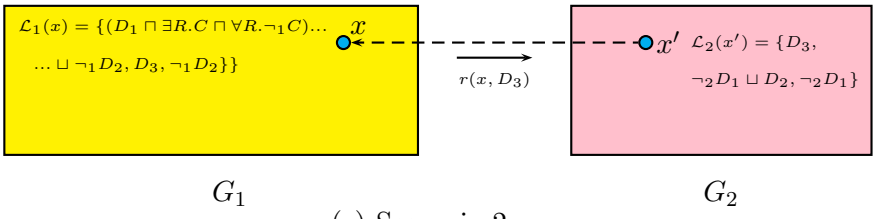
However, Figure 5.3 (c) shows another expansion scenario, which will be referred to as Scenario 2, that finds a consistent distributed completion graph. Hence, P_1 is actually consistent.



(a) Scenario 1, Phase 1



(b) Scenario 1, Phase 2



(c) Scenario 2

Figure 5.3 The Need for Token Blocking

The problem with Scenario 1 in Example 5.2 is caused by the asynchronous operation of the different local reasoners. The message $r^{2\rightarrow 1}(x', D_2)$ in Phase 2 is in fact a consequence of the choice of adding $D_1 \sqcap \exists R.C \sqcap \forall R.\neg_1 C$ into $\mathcal{L}_1(x)$ and the subsequent message $r^{2\leftarrow 1}(x, D_1)$ in Phase 1. However, since local reasoners run autonomously and communication between them may be delayed, as it relies on network conditions, when the message $r^{2\rightarrow 1}(x', D_2)$ arrives at G_1, the previous choice, in which $r^{2\leftarrow 1}(x, D_1)$ was sent, has already been abandoned. Thus, the clash arising in Phase 1 is a false one, since it mixes consequences of different choices during the tableau expansion.

To avoid such problems, we resort to the techniques of *token passing* and of *clocks* in order to synchronize the creation of the local completion graphs by the local reasoners. The basic idea is to coordinate the expansion of all local completion graphs in such a way that, at any time, all node and edge labels in all local completion graphs always belong to the same sequence of non-deterministic choices.

Definition 5.6 (Local Clocks) *Clocks of local completion graphs are maintained in the following way:*

- *Every local completion graph G_i has a local* clock K_i *of integer type, initialized to 0.*
- *For every concept label C in $\mathcal{L}_i(x)$ where x is a node in G_i, there is a* timestamp $t_i(x, C)$ *of integer type. Informally speaking, this time stamp records the clock value of the last choice in the application of the $\sqcup$-rule before C was added into $\mathcal{L}_i(x)$, possibly in another local completion graph.*
- *For every role label R in $\mathcal{L}_i(\langle x, y\rangle)$, where $\langle x, y\rangle$ is an edge in G_i, there is a* timestamp $t_i(x, \langle x, y\rangle)$ *of integer type.*
- *If a reporting message $r^{i\rightarrow j}(x, C)$ or $r^{j\leftarrow i}(x, C)$ is sent, $t_j(j : x, C)$ will be the same as $t_i(i : x, C)$ and $K_j = \max\{K_j, t_i(i : x, C)\}$;*
- *When a new label is added in G_i by an application of CE-, $\sqcap$-, $\forall$- or $\exists$- rules, its timestamp will be the value of the clock K_i;*
- *When a new concept label is added in G_i by an application of the $\sqcup$- rule, the clock K_i is increased by 1 and the label's timestamp is set to be the new value of the clock K_i.*

Definition 5.7 (Token and Backtracking) *A token* $\mathbb{T}$ *is passed between local completion graphs. It is originally assigned to the local completion graph of the witness package. Only the local completion graph that has* $\mathbb{T}$ *can apply the* $\sqcup$*-rule. A local completion graph whose clock value is no smaller than the clock value of any other local completion graph is a* token target*. We require that* 1) $\mathbb{T}$ *only stays at a token target;* 2) *if* G_i *has* $\mathbb{T}$ *and* G_i *is complete, then* $\mathbb{T}$ *is transferred to a token target that is not complete.*[3]

A node x *in* G_i *is said to* have a t-clash *if both* C *and* $\neg_i C$ *are in* $\mathcal{L}_i(x)$*, for some concept* C*, and* $t = \max\{t_i(x, C), t_i(x, \neg_i C)\}$.

A distributed completion graph is said to be synchronized *if* 1) *all concept report messages have arrived at targets; and* 2) *all local completion graphs have stopped expansion.*

A pruning operation $\mathsf{Prune}(t)$ *(where* t *is the timestamp parameter) in* G_i *does the following:*

- *Removes all concept and role labels in* G_i *with timestamp* $\geq t$.
- *Removes every node with empty label set and its incoming edges.*
- *Sets* K_i *to the largest timestamp of concept labels in* G_i *after the pruning, i.e.,* $K_i = \max\{t_i(x, C) : x \in V_i, C \in \mathcal{L}_i(x)\}$.

If a t*-clash occurs in* G_i*,* G_i *will broadcast a* t-clash message *to all other local completion graphs, such that the following steps will be executed in order:*

- *Stop all expansions at all local completion graphs, until the distributed completion graph is synchronized.*
- *Perform* $\mathsf{Prune}(t)$ *in all local completion graphs.*
- *Transfer* $\mathbb{T}$ *to a token target.*

The pruning operation is necessary to restore all local completion graphs to their status just before the choice which led to the clash, or to the initial status of the local tableau, if no choice at all had ever been made.

[3] *We do not require a particular token passing protocol (i.e., when and how* $\mathbb{T}$ *should be transferred) on purpose. We believe it is best that it be determined based on empirical results. In what follows, for the sake of concreteness, we adopt a strategy according to which* $\mathbb{T}$ *may be transferred immediately after a concept reporting message, if the message target becomes a token target after the message is sent. We emphasize that this is not the only strategy that can be adopted, nor do we claim that it is the most efficient one.*

Token passing ensures that all local completion graphs are synchronized and that there is only one local completion graph that can apply non-deterministic expansions at any time. Whenever a t-clash is detected, consequences (i.e. nodes and edge labels) dependent on the choice at time t in all local completion graphs will be purged before any other non-deterministic choice can be made. Hence, the handling of a t-clash ensures that different choices in the searching for a clash-free distributed completion graph are always being kept separate. In Example 5.2, Scenario 1, after G_1 detects the clash, it will send a clash message to G_2 and all local completion graphs will be synchronized. Hence, even if the message $r^{2\rightarrow 1}(x', D_2)$ has already been sent before the clash is detected, D_2 will be purged from $\mathcal{L}_1(x)$ during pruning, before G_1 tries other choices. Hence, problems like the one encountered in Phase 2 of Scenario 1 are avoided.

Note that local completion graphs may perform expansions on different reasoning subtasks concurrently. This improves the overall efficiency and scalability of the reasoning process. Further, note that with the introduction of messages, subset blocking in $\mathcal{ALCP}^{-}_{\mathcal{C}}$ is *dynamic*: it can be established, broken and re-established. Moreover, the completeness of a local completion graph is also dynamic. A complete local completion graph may become incomplete, i.e., some expansion rules may become applicable, when a new reporting message arrives.

5.2.3.4 $\mathcal{ALCP}^{-}_{\mathcal{C}}$ Expansion Examples

Example 5.3 Transitive Subsumption Propagation: Given three packages:

$$\begin{aligned} P_1 &: \{1 : A \sqsubseteq 1 : B\} \\ P_2 &: \{1 : B \sqsubseteq 2 : C\} \\ P_3 &: \{2 : C \sqsubseteq 3 : D\} \end{aligned}$$

The query is $1 : A \sqsubseteq 3 : D$ w.r.t. the witness package P_3. The expansion and message exchange between local completion graphs are shown in Figure 5.4. The following steps result from the execution of the algorithm:

1. G_3 is initialized with the token $\mathbb{T}$ and the node x with $\mathcal{L}_3(x) = \{A \sqcap \neg_3 D\}$; applying $\sqcap$- and CE- rules in G_3, $A, \neg_3 D$ and $\neg_3 C \sqcup D$ are added into $\mathcal{L}_3(x)$. $K_3 = 0$ and all concept labels in G_3 have the timestamp 0.

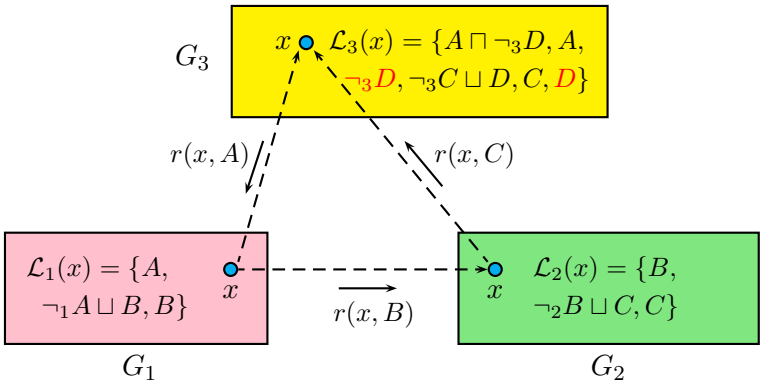


Figure 5.4 Transitive Subsumption Propagation in $\mathcal{ALCP}^{-}_{\mathcal{C}}$

2. Since A has home package P_1, a message $r^{1\leftarrow 3}(x, A)$ is sent and G_3 transfers $\mathbb{T}$ to G_1. G_1 is initialized with $\mathcal{L}_1(x) = \{A\}$. Applying $\sqcup$- and CE- rules in G_1, $\neg_1 A \sqcup B$ and B are added into $\mathcal{L}_1(x)$. $K_1 = 1$.

3. Since P_2 imports P_1, P_3 imports P_2 and $origin(1 : x) = origin(3 : x)$, we apply the r-rule, creating $2 : x$, with $origin(2 : x) = origin(3 : x)$.

4. Applying the CPush-rule, G_1 sends the message $r^{1\rightarrow 2}(x, B)$ and $\mathbb{T}$ to G_2. Applying the $\sqcup$- and CE- rules in G_2, $\neg_2 B \sqcup C$ and C are also added into $\mathcal{L}_2(x)$. $K_2 = 2$.

5. Applying the CPush-rule in G_2, C is added to $\mathcal{L}_3(x)$ and $\mathbb{T}$ is passed to G_3. Applying the $\sqcup$-rule in G_3, $K_3 = 3$ and D is added to $\mathcal{L}_3(x)$. The 3-clash $\{D, \neg_3 D\} \subseteq \mathcal{L}_3(x)$ is detected.

6. As a result, D is now removed from $\mathcal{L}_3(x)$ and clash messages with timestamp 3 are sent to G_1 and G_2, but nothing is removed from G_1 or G_2. K_3 is set to 0. $\mathbb{T}$ is transferred to G_2, since it has the largest clock value.

7. Similarly, all other choices in applying the $\sqcup$-rule lead to clashes. Hence, no clash-free and complete distributed tableau can be found for $A \sqcap \neg_3 D$. Therefore $A \sqsubseteq D$, as witnessed by P_3.

This example shows that P-DL offers a solution to the well-known problem of non-composability of ontology mappings, that is present in DDL Zimmermann and Euzenat (2006).

Example 5.4 Detect Inter-module Unsatisfiability: Given two packages $P_1 : \{1 : B \sqsubseteq 1 : F\}$, $P_2 : \{2 : P \sqsubseteq 1 : B, 2 : P \sqsubseteq \neg_2(1 : F)\}$, test the satisfiability of $2 : P$, as witnessed by P_2. The results shows $2 : P$ is unsatisfiable as witnessed by P_2 (Figure 5.5):

1. G_2 is initialized with $\mathbb{T}$ and the node x with $\mathcal{L}_2(x) = \{P\}$, $K_2 = 0$. Applying the $\sqcup$-rule and the CE-rule, $\neg_2 P \sqcup B$, $\neg_2 P \sqcup \neg_2 F$, B and $\neg_2 F$ are added into $\mathcal{L}_2(x)$ and $K_2 = 2$.

2. Since B's home package is P_1, we apply the CReport-rule, resulting in the creation of $1 : x$ and $\mathcal{L}_1(x) = \{B\}$. $\mathbb{T}$ is passed to G_1. $K_1 = 2$.

3. Applying the $\sqcup$- and CE- rules in G_1, $\neg_1 B \sqcup F$ and F are added into $\mathcal{L}_1(x)$. $K_1 = 3$.

4. Applying the CPush-rule, the message $r^{1\to 2}(x, F)$ is sent and F is added into $\mathcal{L}_2(x)$, resulting in a clash.

5. Since $t_2(x, \neg_2 F) = 2$ and $t_2(x, F) = 3$, G_2 has a 3-clash. F is removed from $\mathcal{L}_2(x)$ and a clash message is sent to G_1. $K_2 = 2$.

6. G_1 receives the clash message and removes F from $\mathcal{L}_1(x)$. However, the next choice, i.e., adding $\neg_1 B$, also leads to a clash.

7. Similarly, all other choices in G_1 lead to clashes.

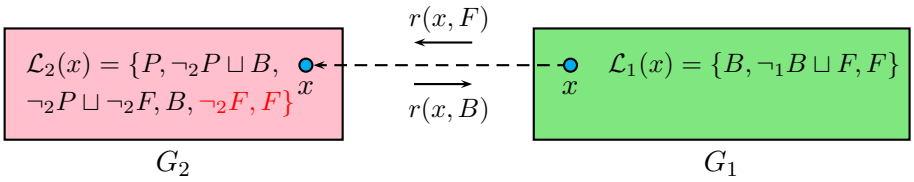


Figure 5.5 Detect Inter-module Unsatisfiability in $\mathcal{ALCP}^{-}_{\mathcal{C}}$

This example shows that P-DL can also solve the inter-module unsatisfiability problem, that is present in DDL Grau et al. (2004b).

Example 5.5 Reasoning from the Local Point of View: Given two packages

$$P_1 : \quad \{1 : A \sqsubseteq 1 : C\}$$

$$P_2 : \quad \{1 : A \sqsubseteq \exists(2 : R).(2 : B), 2 : B \sqsubseteq (1 : A) \sqcap \neg_2(1 : C)\}$$

We need to test the satisfiability of $1 : A$, as witnessed by P_1 and P_2, respectively. It is easy to see that A is satisfiable as witnessed by P_1, but unsatisfiable as witnessed by P_2. Figure 5.6 shows one possible execution when the witness package is P_2.

- G_2 is initialized with $\mathbb{T}$, $2 : x$ and $\mathcal{L}_2(x) = \{A\}$; applying the CE-, $\sqcap$- and $\sqcup$- rules, $\neg_2 A \sqcup \exists R.B$, $\neg_2 B \sqcup (A \sqcap \neg_2 C)$, $\exists R.B$ and $\neg_2 B$ are added to $\mathcal{L}_2(x)$. $K_2 = 2$.

- A has home package P_1, whence a message $r^{1 \leftarrow 2}(x, A)$ is sent. Consequently, $1 : x$ is created, with $\mathcal{L}_1(x) = \{A\}$. $\mathbb{T}$ is not transferred to G_1 because $K_1 = 0 < K_2$.

- Applying the CE-rule in G_1, $\neg_1 A \sqcup C$ is added to $\mathcal{L}_1(x)$. Now G_1 stops, since it does not have $\mathbb{T}$.

- *In the mean time*, G_2 applies the $\exists$-, CE-, $\sqcap$ and $\sqcup$-rules, creating the node $2 : z$ with $\mathcal{L}_2(z) = \{B, \neg_2 A \sqcup \exists R.B, \neg_2 B \sqcup (A \sqcap \neg_2 C), \exists R.B, A \sqcap \neg_2 C, A, \neg_2 C\}$. The message $r^{1 \leftarrow 2}(z, A)$ is sent. Node $1 : z$ is created with $\mathcal{L}_1(z) = \{A\}$. $K_1 = K_2 = 4$. $\mathbb{T}$ is transferred to G_1.

- Applying the CE- and $\sqcup$-rules in G_1, C is added to $\mathcal{L}_1(x)$ and $\mathcal{L}_1(z)$. Two messages $r^{1 \rightarrow 2}(x, C)$ and $r^{1 \rightarrow 2}(z, C)$ are sent. $K_1 = K_2 = 6$. $\mathbb{T}$ is transferred back to G_2.

- Since $\{C, \neg_2 C\} \subseteq \mathcal{L}_2(z)$, a 6-clash is detected in G_2. $\mathsf{Prune}(6)$ is preformed at G_1 and G_2.

- Similarly, all other choices lead to clashes.

This example shows that reasoning in P-DL always supports the local semantic point of view of the witness package. In this way, the same reasoning problem may have different answers from the points of view of different packages.

This example also illustrates the fact that subset blocking in $\mathcal{ALCP}^{-}_{\mathcal{C}}$ only depends on the local ancestorship rather than on the "global" ancestorship. One might have argued that, since $2 : x$ is the local ancestor of $2 : z$, $1 : x$ is a global ancestor of $1 : z$, whence it should have been allowed to block $1 : z$. However, that strategy would have resulted in incorrect blocking: after the message $r^{1 \leftarrow 2}(z, A)$ is sent, $\mathcal{L}_1(1 : z) = \{A\}$, which is a subset of $\mathcal{L}_1(1 : x)$. If this had resulted in blocking $1 : z$, adding C to $\mathcal{L}_1(1 : z)$ would have been prevented and this would have led to the erroneous discovery of a consistent distributed completion graph. On the other hand,

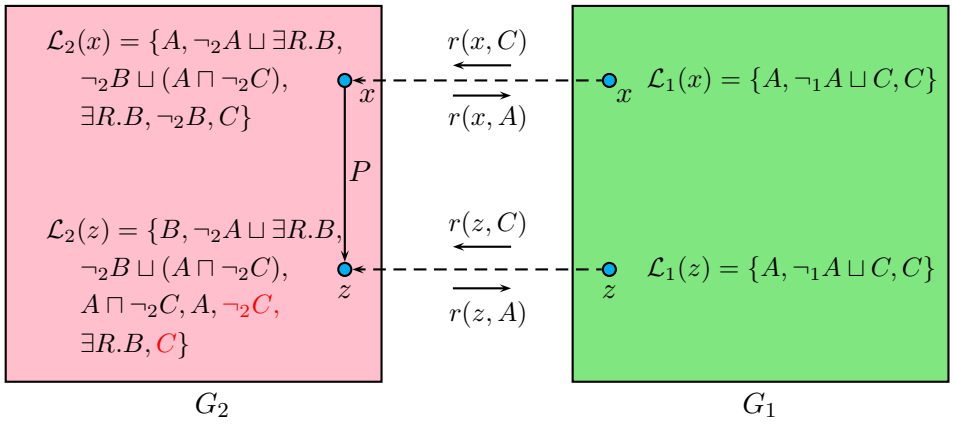


Figure 5.6 Reasoning from Local Point of View in $\mathcal{ALCP}^-_{\mathcal{C}}$

in the next section, it is shown that, with the presence of cyclic importing, global ancestorship is needed in blocking to ensure termination.

5.2.4 Soundness, Completeness, Termination and Complexity

In order to show that the algorithm is a decision procedure for concept satisfiability in $\mathcal{ALCP}^-_{\mathcal{C}}$, it is necessary to prove that the algorithm terminates, that the models that can be constructed from clash-free and complete distributed completion graphs, generated from the algorithm, are valid with respect to the semantics of the logic (soundness) and that the algorithm always finds a model if one exists (completeness). Proofs of lemmas and theorems can be found in the appendix.

Termination and complexity of the algorithm is obtained by proving that there is an upper bound for the total size of all local completion graphs.

Lemma 5.3 *Let Σ be an $\mathcal{ALCP}^-_{\mathcal{C}}$ ontology and D be an $\mathcal{ALCP}^-_{\mathcal{C}}$ concept that is understandable by a witness package P_w in Σ. The $\mathcal{ALCP}^-_{\mathcal{C}}$ tableau algorithm runs in worst case non-deterministic $O\left(2^m \times \prod_{P_j \in P^*_w} 2^{2^{n_j} \times \log n_j}\right)$ time, where $n_i = |C_{\mathcal{T}_i}|$, $i \neq w$, $n_w = |C_{\mathcal{T}_w}| + |D|$ and $m = |P^*_w|$.*

Lemma 5.4 (Termination and Complexity) *Let Σ be an $\mathcal{ALCP}^-_{\mathcal{C}}$ ontology and D be an $\mathcal{ALCP}^-_{\mathcal{C}}$ concept, that is understandable by a witness package P_w in Σ. The $\mathcal{ALCP}^-_{\mathcal{C}}$ tableau algorithm runs in worst case 2*NExpTime *w.r.t. the size of D and the size of the largest package in P^*_w.*

Proof: Let $n_k = \max\{|C_{\mathcal{T}_i}|\}$ be the size of the largest package in P_w^*, $n_D = |D|$ be the size of D, and $m = |P_w^*|$ be the number of packages in the importing closure of P_w. In general, $m \ll 2^{n_k \log n_k}$. By Lemma 5.3, it follows that the total size of all local completion graphs is bounded by

$$O\Big(2^m \times 2^{m \times 2^{(n_k+n_D)\times \log (n_k+n_D)}}\Big) < O\Big(2^{2^{(n_k+n_D)^2}}\Big) \qquad \text{Q.E.D.}$$

.

In fact, by the proof of Lemma 5.3, it follows that the complexity of the $\mathcal{ALCP}_{\mathcal{C}}^-$ algorithm is bounded by $\text{NTIME}\big(\prod_{P_j \in P_w^*} 2^{2^{n_j \log n_j}}\big) = \text{NTIME}\Big(2^{\sum\limits_{P_j \in P_w^*} 2^{n_j \log n_j}}\Big)$, where $n_i = |C_{\mathcal{T}_i}|$, for $i \neq w$, and $n_w = |C_{\mathcal{T}_w}| + |D|$. On the other hand, an equivalent reasoning task over the integrated ontology[4] using the $\mathcal{ALC}$ tableau algorithm of **?** will be bounded by $\text{NTIME}\big(2^{2^{n_\Sigma \log n_\Sigma}}\big)$, where $n_\Sigma = \sum\limits_{P_j \in P_w^*} n_j$ is the size of the integrated ontology. Since, ordinarily, $m \ll 2^{n_k \log n_k}$,

$$\sum_{P_j \in P_w^*} 2^{n_j \log n_j} \ll 2^{\sum\limits_{P_j \in P_w^*} n_j \log n_j} < 2^{n_\Sigma \log n_\Sigma}$$

The last inequality holds because, for every $x_1 \geq 1, x_2 \geq 1$, we have $x_1 \log x_1 + x_2 \log x_2 - (x_1 + x_2) \log (x_1 + x_2) = x_1(\log x_1 - \log (x_1 + x_2)) + x_2(\log x_2 - \log (x_1 + x_2)) < 0$. Thus, under the hypotheses that each module in the ontology is moderately sized and that the communication between local reasoners is reliable, it would be reasonable to expect the distributed $\mathcal{ALCP}_{\mathcal{C}}^-$ reasoning algorithm to terminate significantly faster than its classical counterpart applied on the integrated ontology.

In the following two lemmas, soundness and completeness of the $\mathcal{ALCP}_{\mathcal{C}}^-$ algorithm are proven.

Lemma 5.5 (Soundness) *If the $\mathcal{ALCP}_{\mathcal{C}}^-$ algorithm yields a complete and clash-free distributed completion graph for a concept D w.r.t. a witness package P_w, then D has a tableau w.r.t. P_w.*

Lemma 5.6 (Completeness) *If an $\mathcal{ALCP}_{\mathcal{C}}^-$ concept D has a distributed tableau w.r.t. a witness package P_w, then the $\mathcal{ALCP}_{\mathcal{C}}^-$ algorithm produces a complete and clash-free distributed completion graph for D w.r.t. P_w.*

[4] A reduction to an integrated ontology is described in Bao et al. (2007e).

By the proceeding lemmas, we obtain the following theorem:

Theorem 5.1 *Let Σ be an $\mathcal{ALCP}^-_{\mathcal{C}}$ ontology and D be an $\mathcal{ALCP}^-_{\mathcal{C}}$ concept, that is understandable by a witness package P_w in Σ. The $\mathcal{ALCP}^-_{\mathcal{C}}$ tableau algorithm is a sound, complete, and terminating decision procedure for satisfiability of D as witnessed by P_w. This decision procedure is in 2*NExpTime *w.r.t. the size of D and the size of the largest package in P^*_w.*

5.3 A Reasoning Algorithm for $\mathcal{ALCP}_{\mathcal{C}}$

5.3.1 Extended Subset Blocking

The reasoning algorithm for $\mathcal{ALCP}^-_{\mathcal{C}}$ may fail if we relax the acyclicity assumption, i.e., when applied to the P-DL $\mathcal{ALCP}_{\mathcal{C}}$, as illustrated by the following example.

Example 5.6 : Suppose we have two packages that mutually import one another:

$$P_1 : \{\top_1 \sqsubseteq \exists(1 : R).(2 : D)\}$$
$$P_2 : \{\top_2 \sqsubseteq \exists(2 : P).(1 : C)\}$$

The reasoning task is to check the consistency of the ontology as witnessed by P_1. If we employ the decision procedure for $\mathcal{ALCP}^-_{\mathcal{C}}$, the algorithm will not terminate, as shown in Figure 5.7. Since there is mutual importing, each of the local completion graphs G_1 and G_2 can send reporting messages and create new nodes in the other. Subset blocking, as given in the previous section, cannot prevent local completion graphs from exchanging messages in a cyclic fashion, which leads to non-termination.

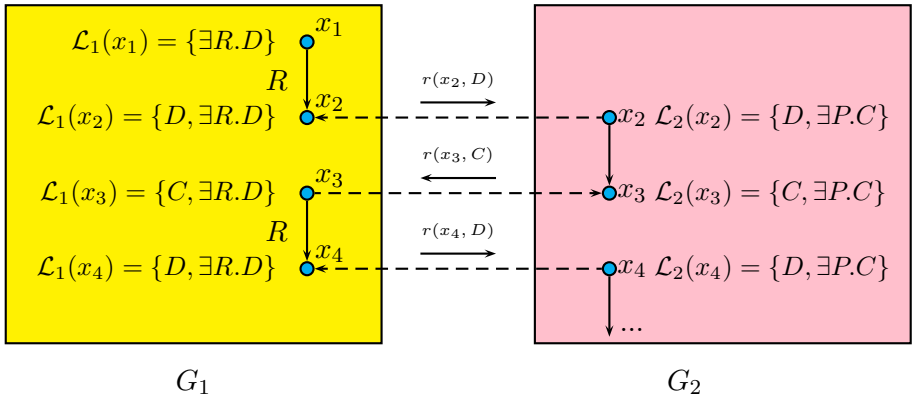


Figure 5.7 Non-termination Caused by Cyclic Importing

Extending the $\mathcal{ALCP}^{-}_{\mathcal{C}}$ algorithm to handle cyclic importing relations requires the detection and prevention of cyclic message exchanges as well as of cyclic local expansions. In fact, termination of the $\mathcal{ALCP}^{-}_{\mathcal{C}}$ algorithm is due to the fact that a local completion graph G_i can cause the creation of a local top node, i.e., a node without a local predecessor, in another local completion graph G_j by an application of the CReport- or the r-rule if and only if the package P_i directly or indirectly imports package P_j. With the presence of cyclic importing, the creation of infinitely many local top nodes in a local completion graph may not be avoided. Thus, subset blocking may fail, since it can only ensure that the number of local descendant nodes of a local top node is limited.

Termination with cyclic importing can be regained if we can ensure that, in any local completion graph, the number of local top nodes as well as the number of local descendant nodes of each local top node are limited. This goal will be realized by an appropriate extension of subset blocking.

Definition 5.8 (Extended Subset Blocking) *A node x is a* global ancestor *of another node y, which may be in a different local graph, if $origin(x) \neq origin(y)$ and there is a path from x to y on the graph $G' = \bigcup_i G_i \cup \{(u,v)|origin(u) = origin(v)\}$, i.e., a path using both local edges and edges in the symmetric closure of graph relations. For any i and any x, y, such that $x \in V_i$, x is a* least global ancestor *of y in G_i if x is a global ancestor of y and there is no other $z \in V_i$ such that z is a global ancestor of y and x is a global ancestor of z.*

For a distributed completion graph of an $\mathcal{ALCP}_{\mathcal{C}}$ ontology, a node x in G_i is directly blocked *by a y in G_i if* 1) *y is a global ancestor of x and $\mathcal{L}_i(x) \subseteq \mathcal{L}_i(y)$, and* 2) *for every $j \neq i$, if there are x', y' in G_j such that $origin(x') = origin(x)$ and $origin(y') = origin(y)$, then $\mathcal{L}_i(x') \subseteq \mathcal{L}_i(y')$. Node x is* indirectly blocked *by a node y if one of x's global ancestors is directly blocked by y. Finally, node x is* blocked *by y if it is directly or indirectly blocked by y.*

Explanation: With extended subset blocking, a node x, including a local top node, can be blocked by one of its global ancestors y, if every local "copy" of x contains no more information than the corresponding local "copy" of y. In this case, expansions at x are not needed, since corresponding expansions must have been preformed at y. Hence, the creation of infinitely many local top nodes as well as of infinitely large local trees under each local top node is avoided. A more detailed analysis of this point will be presented in the termination proof of the algorithm.

For instance, in Example 5.6, $1 : x_4$ will be blocked by $1 : x_2$. As a result, the backward concept reporting messages $r^{2\leftarrow 1}(x_4, D)$ will not be sent (a similar message $r^{2\leftarrow 1}(x_2, D)$ has been sent before) and the reasoning process will terminate. On the other hand, when applying extended subset blocking in Example 5.5, node $1 : z$ will not be blocked by node $1 : x$, whence the necessary forward reporting message $r^{1\rightarrow 2}(z, C)$ will not be undesirably blocked.

Labeling for Global Ancestorship: Since each local reasoner is autonomously maintained, the topology of a local completion graph may not be available to other reasoners. To keep track of the global ancestor relationship in the distributed setting, we may use a labeling schema for dynamic tree representation, since, by merging nodes of the same origin, all local completion graphs can be combined into a tree. The basic intuition is to assign localized, informative labels to each node in the distributed graph which will contain global topology information of the graph. Each node of the same origin will be assigned the same label. In this way, testing global ancestorship can be reduced to comparison of the labels of different nodes. Several labeling schemas for static and/or dynamic trees have been recently proposed Christophides et al. (2004); Chen et al. (2004); Korman et al. (2004); Cohen et al. (2005). The adoption of a particular labeling schema is to be decided during the implementation of the algorithm. It will partially depend on the communication protocol on which the algorithm will be based to achieve best performance.

5.3.2 Correctness and Complexity

The reasoning algorithm for $\mathcal{ALCP}_{\mathcal{C}}$ is a modified version of the $\mathcal{ALCP}_{\mathcal{C}}^{-}$ algorithm, resulting by replacing subset blocking by extended subset blocking and by adding the labeling technique of the various nodes, as described previously.

Theorem 5.2 *Let Σ be an $\mathcal{ALCP}_{\mathcal{C}}$ ontology and D an $\mathcal{ALCP}_{\mathcal{C}}$ concept, that is understandable by a witness package P_w in Σ. The $\mathcal{ALCP}_{\mathcal{C}}$ tableau algorithm is a sound, complete and terminating decision procedure for satisfiability of D as witnessed by P_w. This decision procedure is in 2*NExpTime *w.r.t. the size of D and the total size of packages in P_w^*.*

Proof: Proofs of the soundness and completeness are similar to the proofs of Lemmas 5.5 and 5.6, respectively. So we concentrate on termination and complexity.

Termination will be proven by showing that the "combined" completion graph G', resulting from the various local completion graphs by merging all nodes of the same origin into one node, is finite. For a local completion graph G_i, let $n_i = |C_{\mathcal{T}_i}|$, for $i \neq w$, $n_w = |C_{\mathcal{T}_w}| + |D|$, $n_\Sigma = \sum\limits_i n_i$ and $m = |P_w^*|$. Let x_0 be the initial node of G_w.

For every node in G_i, its out-degree is at most n_i, whence, for every node in G', its out-degree is bounded by n_Σ. Similarly, the size of the concept label set of each node in G_i is bounded by n_Σ. The depth of G' is a most 2^{n_Σ} due to the extended subset blocking. Hence, the total number of nodes in G' is bounded by

$$O\Big((n_\Sigma)^{2^{n_\Sigma}}\Big) = O\Big(2^{2^{n_\Sigma \log n_\Sigma}}\Big).$$

Therefore, the total number of nodes in the distributed completion graph is bounded by

$$O\Big(m \times 2^{2^{n_\Sigma \log n_\Sigma}}\Big). \qquad \text{Q.E.D.}$$

With the presence of cyclic importing, the *worst-case* time complexity of the $\mathcal{ALCP}_\mathcal{C}$ algorithm is bounded by the total size of all packages, while that of the $\mathcal{ALCP}_\mathcal{C}^-$ algorithm is only bounded by the size of the largest package involved in the reasoning task. This result indicates that avoiding cyclic importing between ontology modules will significantly improve reasoning performance.

The $\mathcal{ALCP}_\mathcal{C}$ algorithm has the same *worst-case* time complexity with the $\mathcal{ALC}$ tableau algorithm applied on the combined ontology from all modules. However, the analysis in Theorem 5.2 does not take into account the gain resulting from local reasoners concurrently exploring different reasoning sub-tasks. We believe that, with the proper design of communication protocols between local reasoners, the distributed $\mathcal{ALCP}_\mathcal{C}$ tableau algorithm has the potential of processing a reasoning task more efficiently than would a centralized reasoner.

5.4 Asynchronous Federated Reasoning for $\mathcal{ALCP}_\mathcal{C}$

The $\mathcal{ALCP}_\mathcal{C}^-$ and $\mathcal{ALCP}_\mathcal{C}$ algorithms we presented above utilize token passing to synchronize expansions in multiple local reasoners. However, whenever a clash occurs, other local reasoners have to stop for synchronization, thus reasoners cannot *concurrently* work together on different non-deterministic choices. To maximize usage of the computational resources in peer reasoners,

it is desirable to allow *asynchronous* reasoning strategy such that local reasoners can solve the reasoning task in parallel fashion.

Instead of using the token passing, we may adopt a thread-based labeling strategy to keep the local completion graphs in sync with regard to the logical consequences of different non-deterministic choices.

We assign a *thread* id for each label in a local completion graph, and each reporting message. Let λ be the set of all threads. Threads in λ have a tree-structure, such that every thread except the root thread has exactly one *parent*. We will use partial order ($\leq$) over λ to identify thread ancestry, i.e., $a \leq b$ denotes that thread a is an *ancestor* of thread b. Two threads are said to belong to the same *thread lineage* if one of them is an ancestor of the other. To record thread ancestry in the distributed setting, we can use a prefix numbering strategy, e.g., $a.b$ is the parent of $a.b.c$.

For every concept label C in $\mathcal{L}_i(x)$ and every role label R in $\mathcal{L}_i(\langle x, y\rangle)$, there is an associated thread $th(x, C)$, $th(\langle x, y\rangle, R)$ and $th(\langle x, x'\rangle)$, respectively. Labels not belonging to the same thread lineage are treated as different labels (graph relations).

For a local completion graph G_i and a thread t, $G_i(t)$ is a *projection* of G_i, obtained by including only the labels in G_i that are associated with the thread t or t's ancestors. A node x in G_i is said to be subset blocked for thread t if x is blocked in $G_i(t)$. $G(t) = \{G_i(t)\}$ is the projection of the global completion graph for thread t.

We have the following rules to assign threads:

- When the graph for the witness package is initialized, the label of its root node has the initial thread id t_0; labels resulting from CE-rule also have thread id t_0.

- When a deterministic rule (any rule except the $\sqcup$-rule) is applied in G_i, if its triggering ("if") condition is true in $G_i(t)$, then the resulting expansion will be based on $G(t)$, and any new label or messages resulting from the application of the rule will inherit the same thread id t.

- When a non-deterministic rule (the $\sqcup$-rule) is applied in G_i, if its triggering condition is true in $G_i(t)$, then the expansion will be based on $G(t)$, and any new label or messages resulting from its application will have a new thread id t' such that $t \leq t'$.

- If the triggering condition of a rule is true for two threads t_1 and t_2 of the same thread lineage in G_i and $t_1 \leq t_2$, we will trigger the expansion rule on $G_i(t_1)$.

A local completion graph sends a t-clash message to others local completion graphs when it detects a t-clash. Some details are provided below. A local completion graph G_i is said to have a clash on thread t if it receives a t-clash message, a clash is detected in $G_i(t)$, or every child thread of t has a clash and there is no more child thread of t can be tried in this local completion graph. When G_i has a clash on thread t, it will immediately: 1) remove all labels of thread t' such that $t \leq t'$; 2) remove all edges and nodes with empty label set caused by such deletion; 3) broadcast the clash to all other local completion graphs that have no clash at t; 4) try another choice (thread).

A thread represents a sequence of choices in applying non-deterministic rules. Note that the timestamp approach used in token blocking is a special case of thread, i.e., when we only allow a single active thread in all local completion graphs. With thread-based labeling, a local completion graph does not need a token to apply non-deterministic rules. Hence, multiple local reasoners can preform the reasoning task in asynchronous and parallel fashion.

Because each thread can be handled separately, the soundness, completeness, termination and complexity of the thread-based algorithm is similar to the the token-based algorithm for $\mathcal{ALCP}_{\mathcal{C}}^{-}$ and $\mathcal{ALCP}_{\mathcal{C}}$.

Lemma 5.7 *Let Σ be an $\mathcal{ALCP}_{\mathcal{C}}^{-}$ ontology and D be an $\mathcal{ALCP}_{\mathcal{C}}^{-}$ concept that is understandable in a witness package P_w in Σ. The thread-based $\mathcal{ALCP}_{\mathcal{C}}^{-}$ tableau algorithm with subset blocking is a sound, complete, and terminating decision procedure for satisfiability of D as witnessed by P_w. The decision procedure is in 2*NExpTime *w.r.t. the size of D and the size of the largest package in P_w^*.*

Lemma 5.8 *Let Σ be an $\mathcal{ALCP}_{\mathcal{C}}$ ontology and D be an $\mathcal{ALCP}_{\mathcal{C}}$ concept that is understandable in a witness package P_w in Σ. The thread-based $\mathcal{ALCP}_{\mathcal{C}}$ tableau algorithm with extended subset blocking is a sound, complete, and terminating decision procedure for satisfiability of D as witnessed by P_w. The decision procedure is in 2*NExpTime *w.r.t. the size of D and the total size of all packages in P_w^*.*

An alternative representation of the asynchronous tableau algorithm for $\mathcal{ALCP}_{\mathcal{C}}$ is the *ABox tree* representation (Bao et al., 2006e), wherein each label is represented as a *fact* and all facts

in a local completion graph are organized in a tree. Each branch in the tree corresponding to a thread, i.e., a sequence of non-deterministic choices in tableau expansions. It can be easily proven that the ABox tree and the completion graph based representations are equivalent.

5.5 Reasoning in $\mathcal{SHIQP}$

5.5.1 Overview

The P-DL $\mathcal{SHIQP}$ (i.e., $\mathcal{ALCP}_{\mathcal{C}}\mathcal{H}_{\mathcal{R}+}\mathcal{IQP}$) is extended from $\mathcal{ALCP}_{\mathcal{C}}$ with the following additional features:

- The (possibly cyclic) importing of role names ("$\mathcal{P}$");
- The use of transitive roles, inverse roles and role hierarchies ("$\mathcal{H}_{\mathcal{R}+}\mathcal{I}$");
- The use of qualified number restriction ("$\mathcal{Q}$").

Hence, a reasoning algorithm for $\mathcal{SHIQP}$ presents several challenges to handle those features:

- In which way role instances in multiple local tableaux correspond? What kind of messages should be sent to establish such correspondences?
- How can we extend the use of more complex blocking strategies (e.g., double blocking (Horrocks and Sattler, 2005)) required by the additional role constructors?
- How can we ensure that correct counting of neighbors of a node required by the number restrictions in the distributed setting?
- How can we adapt additional tableau expansion rules (e.g., the "shrinking" rule for $\leq$ number restriction) used for $\mathcal{SHIQ}$ to work in the distributed setting?

In this section, we will present an asynchronous, federated reasoning algorithm for $\mathcal{SHIQP}$ based on the asynchronous reasoning algorithm for $\mathcal{ALCP}_{\mathcal{C}}$ introduced in the last section and the tableau algorithm for $\mathcal{SHIQ}$ (Horrocks et al., 1999).

5.5.2 P-DL $\mathcal{SHIQP}$

We briefly remind the syntax and semantics of $\mathcal{SHIQP}$.

Syntax: A $\mathcal{SHIQP}$ ontology consists of a finite set of modules called *packages* and *importing relations* between packages.

Informally, a package in $\mathcal{SHIQP}$ can be seen as a $\mathcal{SHIQ}$ TBox and RBox. We define the *signature* $\mathsf{Sig}(P_i) = \mathsf{N}_i$ of a package P_i, as the set of names used in P_i. $\mathsf{Sig}(P_i)$ is the disjoint union of the set of concept names NC_i and the set of role names NR_i used in package P_i. We denote by NR_i^+ ($\mathsf{NR}_i^+ \subseteq \mathsf{NR}_i$), the set of of *transitive role* names in P_i. The set of roles in P_i is defined as $\overline{\mathsf{NR}}_i = \mathsf{NR}_i \cup \{R^- | R \in \mathsf{NR}_i\}$ where R^- is the inverse of role R. A role $R \in \overline{\mathsf{NR}}_i$ is said to be a *transitive in* P_i, denoted as $\mathsf{Trans}_i(R)$, iff $R \in \mathsf{NR}_i^+$ or $R^- \in \mathsf{NR}_i^+$.

For each package P_i, $\mathsf{Sig}(P_i)$ is divided into two disjoint parts: its local signature $\mathsf{Loc}(P_i)$ and its external signature $\mathsf{Ext}(P_i)$. $\forall t \in \mathsf{Loc}(P_i)$, P_i is the *home package* of t, and t is called an i-name (more specifically, an i-concept name or an i-role name). We use i-role to refer to an i-role name or its *inverse*.

A *role inclusion* axiom in P_i is an expression of the form $R \sqsubseteq S$, where R and S are i-roles in P_i. The *role hierarchy* for P_i is the set of all role inclusion axioms in P_i. The RBox $\mathcal{R}_i$ consists of the role hierarchy for P_i and the set of role transitivity declarations $\mathsf{Trans}_i(R)$.

The set of $\mathcal{SHIQP}$ concepts in P_i is defined inductively by the following grammar:

$$C := A | \neg_k C | C \sqcap C | C \sqcup C | \forall R.C | \exists R.C | (\leq nS.C) | (\geq nS.C)$$

where $A \in \mathsf{NC}_i$, n is a positive integer, $R \in \overline{\mathsf{NR}}_i$, and $S \in \overline{\mathsf{NR}}_i$ is a locally simple role (defined below); $\neg_k C$ denotes the *contextualized negation* of concept C w.r.t. P_k. For any k and k-concept name C, $\top_k = \neg_k C \sqcup C$, and $\bot = \neg_k C \sqcap C$. $\neg_k C$ and $\top_k$ can appear in P_i ($i \neq k$) only if $P_k \mapsto P_i$ and $\mathsf{Sig}(C) \subseteq \mathsf{Sig}(P_k)$

A general concept inclusion (GCI) axiom in P_i is an expression of the form $C \sqsubseteq D$, where C, D are concepts in P_i. The TBox $\mathcal{T}_i$ of P_i is the set of all GCIs in P_i.

If a concept or role name $t \in \mathsf{Loc}(P_j) \cap \mathsf{Ext}(P_i)$ ($i \neq j$), we say that P_i *imports* t and denote it as $P_j \xrightarrow{t} P_i$. We require that transitivity of roles is preserved under importing. Package importing relation ($\mapsto$) and importing transitive closure of a package are defined in the usual way.

For an RBox $\mathcal{R}_i$, we define $\overline{\mathcal{R}_i} = \{R^- \sqsubseteq S^- | R \sqsubseteq S \in \mathcal{R}_i\}$ and $\boxdot_i$, the transitive-reflexive closure of $\sqsubseteq$ over the $\mathcal{R}_i \cup \overline{\mathcal{R}_i}$.

For a role hierarchy $\mathbf{R}$, if $R \sqsubseteq S \in \mathbf{R}$, then R is called the *sub-role* of S and S is called the *super-role* of R w.r.t. $\mathbf{R}$. A role is said to be *locally simple* in P_i if it is neither transitive nor has any transitive sub-roles w.r.t. $\mathcal{R}_i^*$.

Semantics: For a $\mathcal{SHIQP}$ ontology $\Sigma = \langle \{P_i\}, \{P_i \mapsto P_j\}_{i \neq j} \rangle$, a *distributed interpretation* is a tuple $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{i \neq j} \rangle$, where $\mathcal{I}_i = \langle \Delta^{\mathcal{I}_i}, (.)^{\mathcal{I}_i} \rangle$ is a *local interpretation* of package P_i; $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$ is the *(image) domain relation* for the interpretation of importing relation from P_i to P_j. Each local interpretation $\mathcal{I}_i$ satisfies the following conditions (where R is a j-role):

$$
\begin{aligned}
R^{\mathcal{I}_i} &= (R^{\mathcal{I}_i})^+, \text{for transitive role } R \\
(R^-)^{\mathcal{I}_i} &= \{\langle x, y \rangle | \langle y, x \rangle \in R^{\mathcal{I}_i}\} \\
(C \sqcap D)^{\mathcal{I}_i} &= C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i} \\
(C \sqcup D)^{\mathcal{I}_i} &= C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i} \\
(\neg_j C)^{\mathcal{I}_i} &= r_{ji}(\Delta^{\mathcal{I}_j}) \backslash C^{\mathcal{I}_i}, \text{for } \mathsf{Sig}(C) \subseteq \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_j) \\
(\exists R.C)^{\mathcal{I}_i} &= \{x \in r_{ji}(\Delta^{\mathcal{I}_j}) |\ \exists y, \langle x, y \rangle \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\} \\
(\forall R.C)^{\mathcal{I}_i} &= \{x \in r_{ji}(\Delta^{\mathcal{I}_j}) |\ \forall y, \langle x, y \rangle \in R^{\mathcal{I}_i} \rightarrow y \in C^{\mathcal{I}_i}\} \\
(\geqslant nR.C)^{\mathcal{I}_i} &= \{x \in r_{ji}(\Delta^{\mathcal{I}_j}) |\ \#\{y | \langle x, y \rangle \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\} \geqslant n\} \\
(\leqslant nR.C)^{\mathcal{I}_i} &= \{x \in r_{ji}(\Delta^{\mathcal{I}_j}) |\ \#\{y | \langle x, y \rangle \in R^{\mathcal{I}_i} \wedge y \in C^{\mathcal{I}_i}\} \leqslant n\}
\end{aligned}
$$

A local interpretation $\mathcal{I}_i$ *satisfies* a role inclusion axiom $R_1 \sqsubseteq R_2$ iff $R_1^{\mathcal{I}_i} \subseteq R_2^{\mathcal{I}_i}$ and a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$. $\mathcal{I}_i$ is a *model* of P_i, denoted as $\mathcal{I}_i \vDash P_i$, if it satisfies all axioms in P_i. $\mathcal{I}$ is a *model* of Σ, denoted as $\mathcal{I} \vDash \Sigma$, if the following restrictions on domain relations are satisfied:

1. For all i, j, r_{ij} is one-to-one, i.e., it is an injective partial function.

2. *Compositional Consistency*: For all i, j, k s.t. $i \neq j$, $P_i \in P_k^+$ and $P_k \in P_j^+$, we have $\rho_{ij} = r_{ij} = r_{kj} \circ r_{ik}$, where ρ is the symmetric and transitive closure of the union of all domain relations.

3. For every i-concept name C that appears in P_j, we have $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$.

4. For every i-role R that appears in P_j, we have $R^{\mathcal{I}_j} = r_{ij}(R^{\mathcal{I}_i})$.

5. *Cardinality Preservation for Roles*: For every i-role R that appears in P_j and every $(x, x') \in r_{ij}$, $y \in R^{\mathcal{I}_i}(x)$ iff $r_{ij}(y) \in R^{\mathcal{I}_j}(x')$.

6. $\mathcal{I}_i \vDash P_i$, for every i.

Concept satisfiability, concept subsumption and ontology consistency are defined in the same fashion as that of $\mathcal{ALCP}^{-}_{\mathcal{C}}$.

5.5.3 A Tableau for $\mathcal{SHIQP}$

A $\mathcal{SHIQP}$ formula can be transformed in the Negation Normal Form (NNF) using the similar rules for $\mathcal{ALCP}^{-}_{\mathcal{C}}$ and the following additional rules:

$$\begin{aligned} \neg_i(\leqslant nR.C) &\Rightarrow \top_i \sqcap (\geqslant (n+1)R.C) \\ \neg_i(\geqslant (n+1)R.C) &\Rightarrow \top_i \sqcap (\leqslant nR.C) \end{aligned}$$

Let $clos_i(C)$ be the closure of "subconcepts" of a $\mathcal{SHIQP}$ concept C w.r.t. $\boxplus_i$ (similar to the Definition 6.20 of (Tobies, 2001)), defined as the smallest set X of $\mathcal{SHIQP}$-concepts that satisfies the following conditions:

- $C \in X$,
- X is closed under sub-concepts[5] and $\dot{\neg}_i$,
- if $\forall R.D \in X$, $T \boxplus_i R$, and $\mathsf{Trans}(T)$, then $\forall T.D \in X$.

Let D be a $\mathcal{SHIQP}$-concept in NNF and P_w be a witness package (D is understandable in P_w). A *distributed tableau* is a tuple $T = \langle \{T_i\}, \{t_{ij}\}_{P_i \in P_j^+} \rangle$ for D w.r.t. P_w, where each T_i is a local tableau for $P_i \in P_w^*$, and t_{ij} is the tableau relation between T_i and T_j.

Each local tableau is a tuple $T_i = (\mathbf{S}_i, \mathcal{L}_i, \mathcal{E}_i)$, where $\mathbf{S}_i$ is a set of individuals, $\mathcal{L}_w : \mathbf{S}_w \rightarrow 2^{clos_w(D) \cup clos_w(C_{\mathcal{T}_w})}$ and $\mathcal{L}_i : \mathbf{S}_i \rightarrow 2^{clos_i(C_{\mathcal{T}_i})}$ (for $i \neq w$) map individuals to corresponding sets of concepts, $\mathcal{E}_i : \overline{\mathsf{NR}}_i \rightarrow 2^{\mathbf{S}_i \times \mathbf{S}_i}$ map roles to the corresponding sets of pairs of individuals. There must be some individual $x \in S_w$ such that $D \in \mathcal{L}_w(x)$. We define $S_i^T(x, C) := \{y \in S_i | \langle x, y \rangle \in \mathcal{E}_i(S) \wedge C \in \mathcal{L}_i(y)\}$.

[5]Sub-concepts of a $\mathcal{SHIQP}$ concept C is defined by extending the Definition 5.3 with $sub(\leq R.C) = \{\leq R.C\} \cup sub(C)$ and $sub(\geq R.C) = \{\geq R.C\} \cup sub(C)$.

Each tableau relation t_{ij} is a subset of $\mathbf{S}_i \times \mathbf{S}_j$.

Furthermore, the following conditions should hold for T. The conditions **(A0)** – **(A11)** are similar to the ones used in the tableau definition for $\mathcal{SHIQ}$ (Horrocks et al., 1999):

(A0-A5) are the same as that of the $\mathcal{ALCP}^-_{\mathcal{C}}$ tableau algorithm (section 5.2.3).

(A6) if $\forall S.C \in \mathcal{L}_i(x)$ and $\langle x, y\rangle \in \mathcal{E}_i(R)$ for some $R \sqsubseteq_i S$ with $\mathsf{Trans}_i(R)$, then $\forall R.C \in \mathcal{L}_i(y)$,

(A7) $\langle x, y\rangle \in \mathcal{E}_i(R)$ iff $\langle y, x\rangle \in \mathcal{E}_i(R^-)$,

(A8) if $\langle x, y\rangle \in \mathcal{E}_i(R)$ and $R \sqsubseteq_i S$, then $\langle x, y\rangle \in \mathcal{E}_i(S)$,

(A9) if $\leq nS.C \in \mathcal{L}_i(x)$, then $|S_i^T(x, C)| \leq n$,

(A10) if $\geq nS.C \in \mathcal{L}_i(x)$, then $|S_i^T(x, C)| \geq n$,

(A11) if $\bowtie nS.C \in \mathcal{L}_i(x)$ and $\langle x, y\rangle \in \mathcal{E}_i(S)$, then $\{C, \dot{\neg}_i C\} \cap \mathcal{L}_i(x) \neq \emptyset$, where $\bowtie$ is $\leq$ or $\geq$.

The second set of requirements is required by the semantics of package extensions:

(B1-B4) are the same as that of the $\mathcal{ALCP}^-_{\mathcal{C}}$ tableau algorithm (section 5.2.3).

(B4) If R is an i-role , $P_i \xrightarrow{R} P_j$, $i \neq j$, then $\forall x', y' \in \mathbf{S}_j, ((\exists x, y \in \mathbf{S}_i, \langle x, x'\rangle \in t_{ij}, \langle y, y'\rangle \in t_{ij}$ and $\langle x, y\rangle \in \mathcal{E}_i(R))$ iff $\langle x', y'\rangle \in \mathcal{E}_j(R)$

(B5) If $\langle x, y\rangle \in \mathcal{E}_i(R)$, R is an i-role that appears in P_j $(i \neq j)$, and there are some $x' \in \mathbf{S}_j$ such that $\langle x, x'\rangle \in t_{ij}$, then there is some $y' \in \mathbf{S}_j$ such that $\langle y, y'\rangle \in t_{ij}$ and $\langle x', y'\rangle \in \mathcal{E}_j(R)$.

Explanation: Intuitively, B4 ensures $r_{ij}(R^{\mathcal{I}_i}) = R^{\mathcal{I}_j}$ for any role R; B5 ensures cardinality preservation of roles.

The following lemma establishes the correspondence between concept satisfiability and the existence of a tableau for that concept:

Lemma 5.9 *Let D be a $\mathcal{SHIQP}$ concept that is understandable in a $\mathcal{SHIQP}$ package P_w. Then D is satisfiable as witnessed by P_w iff D has a distributed tableau w.r.t. P_w.*

Proof sketch (full proof is in the appendix): The proof of this lemma is analogous to that for $\mathcal{SHIQ}$ (Horrocks et al., 1999). We can construct a distributed model $\mathcal{I}$ from a tableau by taking $\mathbf{S}_i$ as the local interpretation domain of P_i (for all i such that $P_i \in P_w^*$) and adding the missing role-successors for transitive roles. Domain relations in $\mathcal{I}$ correspond to tableau relations. This construction ensures that all restrictions on domain relations are satisfied; and

if $C \in \mathcal{L}_i(x)$ then $x \in C^{\mathcal{I}_i}$. Conversely, we can also transform a model of D and P_w^* into a corresponding distributed tableau.

5.5.4 An Asynchronous Tableau Algorithm for $\mathcal{SHIQP}$

We now proceed to describe a sound and complete algorithm to determine the existence of a tableau for a $\mathcal{SHIQP}$ concept w.r.t. a witness package.

5.5.4.1 Distributed Completion Graph

The algorithm works on a *distributed completion graph*, similar to that of $\mathcal{ALCP}_{\mathcal{C}}^{-}$. A distributed completion graph is $G = \{G_i\}$, where $\{G_i\}$ is a set of *local completion graphs*. $G_i = \langle V_i, E_i, \mathcal{L}_i \rangle$, origin of nodes, and image/preimage nodes are defined similar to that of $\mathcal{ALCP}_{\mathcal{C}}^{-}$.

If $S \in \mathcal{L}_i(\langle x, y \rangle)$ and $S \boxtimes_i R$, y is said to be a local *R-successor* of x and x is said to be a local *R-predecessor* of y. *Local ancestors* and *local descendants* of a node are defined in the usual manner. A node y is called a local *R-neighbor* of a node x in G_i if y is either an R-successor of x or an R^{-}-successor of y in T_i. Global ancestorship is defined similar to that of $\mathcal{ALCP}_{\mathcal{C}}$.

A node is called a *local top* node if it has no local ancestor. We define $S_i^G(x, C) := \{y \in V_i |$ y is an S-neighbor of x and $C \in \mathcal{L}_i(y)\}$.

Whenever necessary, we use $x \neq y$ to denote that the nodes x and y in a local completion graph G_i represent distinct individuals.

5.5.4.2 Distributed Tableau Expansion

A distributed $\mathcal{SHIQP}$ completion graph is constructed by applying a set of tableau expansion rules and by exchanging messages between local reasoners.

Blocking: The blocking strategy for $\mathcal{SHIQP}$ is similar to the *double blocking* used for $\mathcal{SHIQ}$ (Horrocks et al., 1999) to ensure the termination of the algorithm.

A node u is *directly blocked* by a v if for every two nodes x, y in any local completion graph G_i such that $origin(x) = origin(u)$ and $origin(y) = origin(v)$, iff none of x's global ancestors is blocked, and either

- x has no global predecessor in G_i, y is a global ancestor of x and $\mathcal{L}_i(x) = \mathcal{L}_i(y)$, or
- x has a global ancestors x', y and y' in G_i such that 1) x is a global successor of x' and y is a global successor of y', 2) $\mathcal{L}_i(x) = \mathcal{L}_i(y)$ and $\mathcal{L}_i(x') = \mathcal{L}_i(y')$, and 3) $\mathcal{L}_i(\langle x', x\rangle) = \mathcal{L}_i(\langle y', y\rangle) \neq \emptyset$.

A node y is indirectly globally blocked iff one of its local ancestor is blocked, or it is a successor of a node x and $\mathcal{L}_i(\langle x, y\rangle) = \emptyset$. A node is globally blocked if either it is directly blocked or it is indirectly blocked.

Intra-Tableau Expansions: The first set of "intra-tableau" expansion rules is similar to the ones used in $\mathcal{SHIQ}$.

- CE-rule: **if** $C_{\mathcal{T}_i} \notin \mathcal{L}_i(x)$ and x is not indirectly blocked, **then** $\mathcal{L}_i(x)$+= $\{C_{\mathcal{T}_i}\}$
- $\sqcap$-rule: **if** $C_1 \sqcap C_2 \in \mathcal{L}_i(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}_i(x)$, **then** $\mathcal{L}_i(x)$+= $\{C_1, C_2\}$
- $\sqcup$-rule: **if** $C_1 \sqcup C_2 \in \mathcal{L}_i(x)$, x is not indirectly blocked, and $\{C_1, C_2\} \cap \mathcal{L}_i(x) = \emptyset$, **then** $\mathcal{L}_i(x)$+= $\{C\}$ for some $C \in \{C_1, C_2\}$
- $\exists$-rule: **if** $\exists R.C \in \mathcal{L}_i(x)$, x is not blocked, and x has no local R-neighbor y in G_i with $C \in \mathcal{L}_i(y)$, **then** create a new node y with $\mathcal{L}_i(\langle x, y\rangle) = \{R\}$ and $\mathcal{L}_i(y) = \{C\}$
- $\forall$-rule: **if** $\forall R.C \in \mathcal{L}_i(x)$, x is not indirectly blocked, and there is an R-neighbor y of x in G_i with $C \notin \mathcal{L}_i(y)$, **then** $\mathcal{L}_i(y)$+= $\{C\}$
- $\forall_+$-rule: **if** $\forall S.C \in \mathcal{L}_i(x)$, x is not indirectly blocked, an there is some R with $\mathsf{Trans}_i(R)$, $R \boxtimes_i S$ and there is an R-neighbor y of x with $\forall R.C \notin \mathcal{L}_i(y)$, **then** $\mathcal{L}_i(y)$+= $\{\forall R.C\}$
- *choose*-rule: **if** $(\bowtie n S.C) \in \mathcal{L}_i(x)$, x is not indirectly blocked, and there is a local S-neighbor y of x in G_i with $\{C, \dot{\neg}_i C\} \cap \mathcal{L}_i(y) = \emptyset$, **then** $\mathcal{L}_i(y)$+= $\{E\}$ for some $E \in \{C, \dot{\neg}_i C\}$, where $\bowtie$ is $\leq$ or $\geq$.
- $\geq$-rule: **if** $(\geq n S.C) \in \mathcal{L}_i(x)$, x is not indirectly blocked, and there are no n local S-neighbors $y_1, ..., y_n$ of x with $C \in \mathcal{L}_i(y_k)$ and $y_k \neq y_j$ for each $0 \leq k < j \leq n$, **then** create n new nodes $y_1, ..., y_n$ with $\mathcal{L}_i(\langle x, y_k\rangle) = \{S\}$ and $\mathcal{L}_i(y_k) = \{C\}$ and $y_k \neq y_j$ for $0 \leq k < j \leq n$.

- $\leq$-rule: **if** $(\leq nS.C) \in \mathcal{L}_i(x)$, x is not indirectly blocked, $|S_i^G(x, C)| \geq n$ and there are two S-neighbors y, z of x with $C \in \mathcal{L}_i(y) \cap \mathcal{L}_i(z)$, y is not a local ancestor of x, and not $y \neq z$, **then** $\mathsf{Merge}(y, z)$.

The Merge operation merges a node with another node. $\mathsf{Merge}(y, z)$ *merges* y into z by letting z inherit all the predecessors of y, and blocking y and its sub-trees (Horrocks et al., 1999), with the following operations:

- $\mathcal{L}_i(z) \mathrel{+}= \mathcal{L}_i(y)$
- Let x be the local predecessor of y. if z is an ancestor of x, then $\mathcal{L}_i(\langle z, x\rangle) \mathrel{+}= \mathsf{Inv}(\mathcal{L}_i(\langle x, y\rangle))$ else $\mathcal{L}_i(\langle x, z\rangle) \mathrel{+}= \mathcal{L}_i(\langle x, y\rangle)$
- $\mathcal{L}_i(\langle x, y\rangle) \leftarrow \emptyset$
- set $u \neq z$ for all u with $u \neq y$

Messages Between Local Reasoners: In addition to concept reporting messages, local reasoners may also communicate with each other via **role reporting messages**. A role reporting message propagates role labels of an edge to the corresponding image or preimage edge.

- A *forward role reporting message* $r^{i \to j}(\langle x, y\rangle, R)$ where $x, y \in V_i$ $(i \neq j)$ and R is a role in P_i executes the following actions: **if** there is $x' = x^{i \to j} \in V_j$, **then**

  (a1) **if** there is no $y^{i \to j} \in V_j$, **then** create y' in V_j with $origin(y') = origin(y)$ **else** let $y' = y^{i \to j}$;

  (a2) **if** there is no edge $\langle x', y'\rangle \in E_j$, **then** create $\langle x', y'\rangle$ with $\mathcal{L}_j(\langle x', y'\rangle) = \{R\}$;

  (a3) **else if** $P \notin \mathcal{L}_j(\langle x', y'\rangle)$ **then** $\mathcal{L}_j(\langle x', y'\rangle) \mathrel{+}= \{R\}$.

- A *backward role reporting message* $r^{j \leftarrow i}(\langle x, y\rangle, R)$ executes the following actions:

  (b1) Transmit $r^{j \leftarrow i}(x, \top_j)$ and $r^{j \leftarrow i}(y, \top_j)$. As a result, $x' = x^{j \leftarrow i}$ and $y' = y^{j \leftarrow i}$.

  (b2) Do similar operations (a2)-(a3) as in $r^{i \to j}(\langle x, y\rangle, R)$.

Inter-Tableau Expansions: We now proceed to describe "inter-tableau" expansion rules. These rules essentially connect the local completion graphs (tableaux) together by establishing the correspondences between nodes (individuals) and edges (role instances) across the local completion graphs.

- CPush-rule, CReport-rule and r-rule are the same as we introduced before.
- RPush-rule: **if** y is an R-successor of x in G_i, x is not blocked, R is an i-role, $P_i \xrightarrow{R} P_j$, there are $x' = x^{i \rightarrow j}$, and $y^{i \rightarrow j}$ does not exist or it exits but is not an R-successor of x', **then** transmit $r^{i \rightarrow j}(\langle x, y \rangle, R)$.
- RReport-rule: **if** y is an R-successor of x in G_i, x is not blocked, R is a j-role ($i \neq j$), and there is no $x' = x^{j \leftarrow i}$ and $y' = y^{j \leftarrow i}$ in G_j such that y' is an R-successor of x', **then** transmit $r^{j \leftarrow i}(\langle x, y \rangle, R)$.
- $\neq$-rule: **if** $x, y \in V_i$ and $x \neq y$ and there is some j ($i \neq j$) such that there are x', y' in V_j such that $origin(x) = origin(x')$, $origin(y) = origin(y')$ and $x' \neq y'$ is not in G_j, **then** add $x' \neq y'$ to G_j
- *merge*-rule: **if** Merge(x, y) is preformed in P_i, **then** for all $j \neq i$ if there are $x, y \in V_j$ such that $origin(x) = origin(x')$, $origin(y) = origin(y')$, perform Merge(x', y').

We require that $\neq$-rule has higher priority than the *merge*-rule.

Maintenance of the origins of nodes ensure that the tableau relations mirror the one-to-one domain relations (tableau property B1). RPush-rule and RReport-rule ensure tableau property B5; *RPush*-rule also ensures tableau property B6 (cardinality closure of roles); $\neq$-rule ensures that nodes that represent distinct individuals are not merged.

The *merge*-rule propagates a Merge operation to the image and preimage nodes of the merged nodes. Note that it is guaranteed that if a local completion graph is a forest before the merge operation it is still a forest after the operation. This is because for every $x, y \in V_i$, y is a successor of x iff $y^{i \rightarrow j}$ is a successor of $x^{i \rightarrow j}$ (if they exist).

5.5.4.3 Backtracking and Parallel Reasoning

We say that a distributed completion graph contains a *clash* if

1. for some G_i, some concept name C and a node x of G_i, $\{C, \neg_i C\} \subseteq \mathcal{L}_i(x)$, or

2. for some G_i, some role S and node x of G_i, $\leq nS.C \in \mathcal{L}_i(x)$ and there exist $n+1$ S-neighbors $y_0, ..., y_n$ of x in G_i with $C \in \mathcal{L}_i(y_k)$ for each $0 \leq k \leq n$ and $y_k \neq y_j$ for each $0 \leq k < j \leq n$, or

A distributed completion graph is said to be *clash-free* if it contains no clash, and it is said to be *complete* if no $\mathcal{SHIQP}$ expansion rules can be applied to it.

Whenever a clash is detected, the algorithm should backtrack to the previous choice and try one of the other possible choices. Similar to the asynchronous algorithm for $\mathcal{ALCP}_\mathcal{C}$ introduced in the last section, we adopt a thread-based labeling strategy to keep the local reasoners in sync with regard to the logical consequences of different non-deterministic choices. The differences from the thread-based labeling strategy of $\mathcal{ALCP}_\mathcal{C}$ are:

- Non-deterministic rules also include *choose*- and $\leq$- rules, in addition to the $\sqcup$-rule;

- We assign a *thread* id also for each inequality between nodes, and each Merge operation, in addition to each node or edge label, each graph relation, and each reporting message.

To determine the satisfiability of a $\mathcal{SHIQP}$ concept D as witnessed by P_w, the tableau algorithm starts with a local completion graph G_w, $x \in V_w$ and $\mathcal{L}_w(x) = \{D\}$. Then the distributed tableaux is expanded according to the expansion rules until no rule can be applied or a clash is found. If this process results in a complete, clash-free distributed completion graph, then D is satisfiable as witnessed by P_w, otherwise D is unsatisfiable as witnessed by P_w.

The following theorems give the correctness and complexity of the algorithm:

Theorem 5.3 *Let Σ be a $\mathcal{SHIQP}$ ontology and D be a $\mathcal{SHIQP}$ concept that is understandable in a witness package P_w in Σ. The $\mathcal{SHIQP}$ tableau algorithm with double blocking is a sound, complete, and terminating decision procedure for satisfiability of D as witnessed by P_w. The decision procedure is in 2*NExpTime *w.r.t. the size of D and the toal size of all packages in P_w^*.*

Thus, the complexity of the distributed reasoning algorithm for $\mathcal{SHIQP}$ is no worse than that of the complexity of the corresponding DL $\mathcal{SHIQ}$ (Tobies, 2001). That result suggests limiting the size of ontology modules to be moderate will be helpful to ensure the scalability of

the reasoning process. Note that this analysis ignores the efficiency gains resulting from different local reasoners being able to concurrently process different threads during the distributed tableau expansion process.

5.5.5 Example

Example 5.7 :(Figure 5.8): There are two packages:

$P_1 : \mathcal{R}_1 = \{1 : R_2 \sqsubseteq 1 : R_1\}, \mathcal{T}_1 = \{\top_1 \sqsubseteq (\leq 2(1 : R_1).\top_1) \sqcap (\exists 1 : R_2.\top_1)\}$.

$P_2 : \mathcal{R}_2 = \emptyset, \mathcal{T}_2 = \{2 : C \sqsubseteq \top_2\}$.

and $\mathsf{Trans}_1(1 : R_2), \mathsf{Trans}_2(1 : R_2)$.

The reasoning task is to check satisfiability of $\forall R_2.C \sqcap \geq 2R_1.(\exists R_2.\neg_2 C)$ as witnessed by P_2 (answer is NO). x'_1 is initially created in G_2; after applying the $\geq$-rule, two R_1 successor (x'_2, x'_3) of x'_1 are created, and RReport-rule is triggered, thus nodes x_1, x_2, x_3 and edges (x_1, x_2) and (x_1, x_3) are created in G_1. Then, a R_2-successor x_4 of x_1 is generated in G_1 by $\exists$-rule; x'_4 is generated by CPush-rule in G_2; then x_4 is merged into x_3 by $\leq$-rule, and x'_4 is merged into x'_3 by *merge*-rule. Hence, $R_2 \in \mathcal{L}_2(\langle x'_1, x'_3\rangle)$, therefore $\forall R_2.C$ is added to $\mathcal{L}_2(x'_3)$ according to $\forall_+$ rule, which will finally result in a clash in x'_5 (created by $\exists$-rule). In this example, role reporting messages are critical to discover $R_2 \in \mathcal{L}_2(\langle x'_1, x'_3\rangle)$ so as to ensure the correct application of the $\forall_+$-rule.

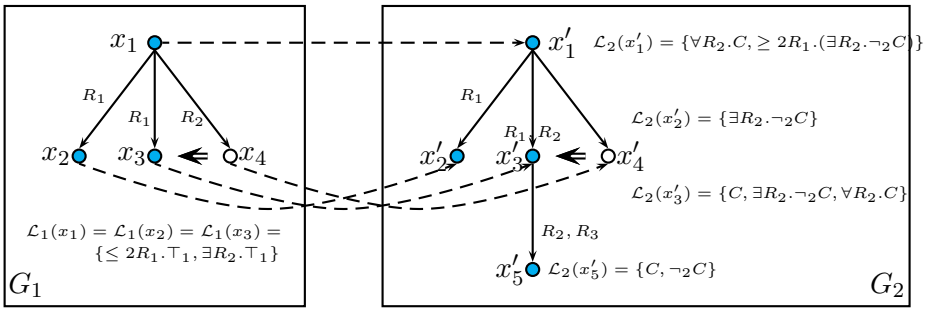


Figure 5.8 Example of $\mathcal{SHIQP}$ Tableau Expansion

5.6 Related Work

Partition-based Logics and SOMEWHERE: Several authors have recently investigated distributed reasoning algorithms for modular ontologies. Partition-based Logics (Amir and

McIlraith, 2000) provides an approach to automatically decompose propositional and first-order logic (FOL) into *partitions* and an algorithm for reasoning with such partitions using message passing. The SOMEWHERE peer-to-peer data management system (Adjiman et al., 2006) provides a distributed query answering algorithm in a "propositional" fragment of Description Logics. On the hand, our focus is on sound and complete distributed reasoning with description logics.

DDL: (Serafini and Tamilin, 2004; Serafini et al., 2005a) describe a tableau-based reasoning algorithm for Distributed Description Logics (DDL) with acyclic bridge rules between concepts. The algorithm divides a reasoning problem w.r.t. a DDL TBox into several local reasoning problems answered by local modules. The basic idea behind this algorithm is to infer concept subsumption in one module from subsumptions in another module and inter-module *bridge rules*. For example, consider ontology modules i and j in which the concepts A, B and G, H respectively are defined, given the bridge rules $i : A \xrightarrow{\sqsupseteq} j : G$, $i : B \xrightarrow{\sqsubseteq} j : H$ and module i entails $A \sqsubseteq B$, then it is possible for module j to infer that $G \sqsubseteq H$. Thus, an ontology module may submit a subsumption query (or unsatisfiability query) to another module to complete a local reasoning task.

The algorithm is implemented in the DRAGO system (Serafini et al., 2005a; Tamilin, 2007), which allows multiple reasoners communicate with each other (via TCP connections) to preform a reasoning task.

This approach is extended by (Serafini and Tamilin, 2005a) on distributed instance retrieval in DDL, by (Serafini et al., 2005a) on reasoning with cyclic bridge rules (with a fix-point semantics), and by (Ghidini and Serafini, 2006a) on reasoning with bridge rules between roles.

The DDL reasoning algorithm in fact builds a virtual tree shaped global completion graph (for the integrated ontology from all modules) by constructing multiple trees in local reasoners using local knowledge. That is in accord with the basic intuition of P-DL reasoning algorithms. However, the two approaches are different on how to decompose the virtual global completion graph. In the DDL approach (Figure 5.9), each local reasoner builds a "branch" of the global tree. On the other hand, in the P-DL approach (Figure 5.10), since the concept languages of modules are not disjoint, a local reasoner build a "projection" of the global tree, and some nodes may be "shared" by multiple local trees.

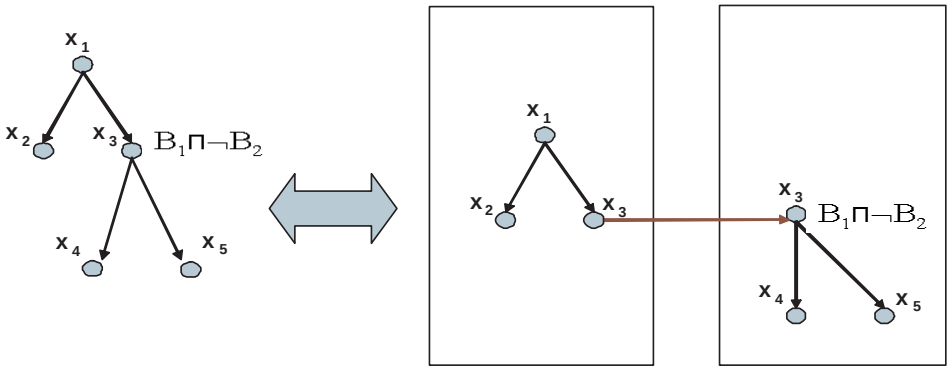


Figure 5.9 Completion Graph in the DDL Tableau Algorithm

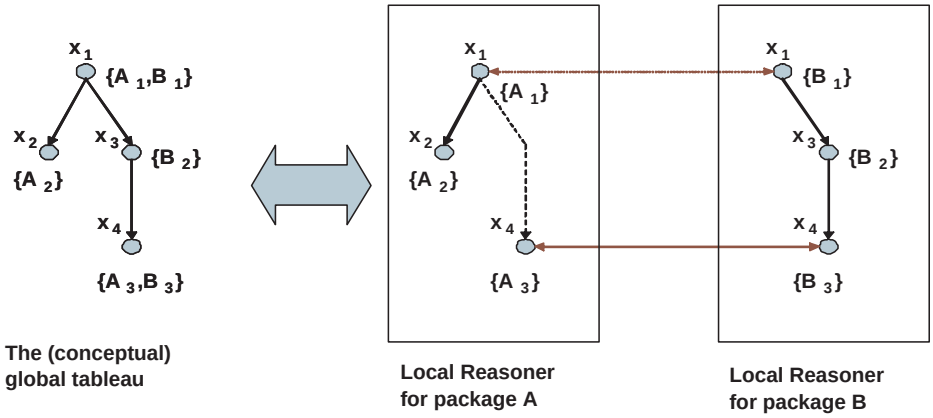


Figure 5.10 Completion Graph in P-DL Tableau Algorithms

The DDL reasoning algorithm is limited in several ways. Due to limitations of the DDL semantics as we discussed in the last chapter, transitive reuse of bridge rules is not supported in general. The algorithm also does not support inference of bridge rules. For instance, if $L_1 = \{1 : A \sqsubseteq 1 : B\}, L_2 = \emptyset, \mathcal{B}_{12} = \{1 : B \xrightarrow{\sqsubseteq} 2 : C\}$, the algorithm cannot infer that $1 : A \xrightarrow{\sqsubseteq} 2 : C$[6]. On the contrast, P-DL can solve such problems since 1) P-DL semantics ensures transitive reusability of ontology modules; 2) semantic importing in P-DL allows "inter-module" semantics relations (like DDL bridge rules) and "intra-module" semantics relations (like local concept subsumptions) to be treated in the same fashion, hence the P-DL reasoning algorithms can handle both the two scenarios.

$\mathcal{E}$-Connections: In (Grau et al., 2004b; Grau, 2005; Grau et al., 2006c) a tableau-based

[6]Reasoning about properties of bridge rules has been addressed in (Stuckenschmidt et al., 2006). However, (Stuckenschmidt et al., 2006) does not provide a decision procedure for the inference of bridge rules.

reasoning procedure for $\mathcal{E}$-Connections is presented. It generates a set of local completion graphs (typically trees) linked by $\mathcal{E}$-connection instances (cross-module role instances), as illustrated in Figure 5.11. Each local completion graph is associated with a color and the reasoning process is performed on the *combined completion forest* resulting by combining all those local completion graphs.

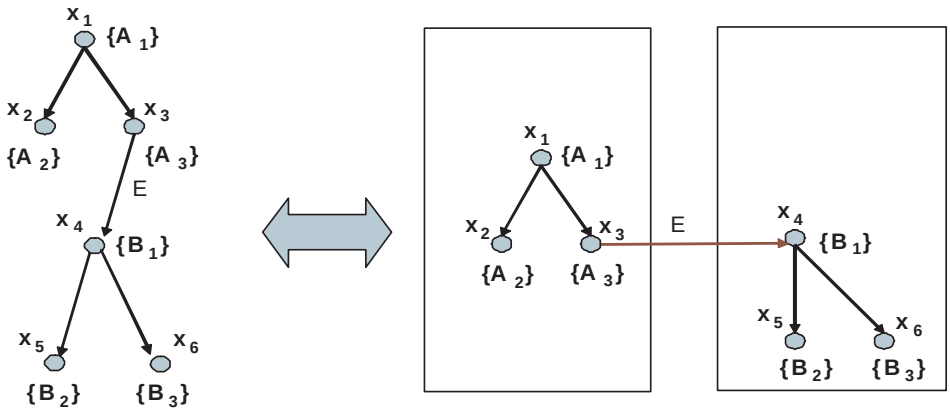


Figure 5.11 Completion Graph in $\mathcal{E}$-Connections Tableau Algorithms

The $\mathcal{E}$-connections algorithm adopts the approach of "coloring", but not of physically separating, local completion graphs. Hence, it is assumed in the algorithm that a local completion graph can freely access information of other local completion graphs, e.g., the node successor relationship and neighborhood, node and edge labels and blocking conditions. Therefore, no message passing or any other forms of communication between local completion graphs are required, nor are specially designed distributed backtracking strategies provided. Thus, the implementation of the algorithm in the Pellet reasoner (Sirin et al., 2007) utilizes a *single* reasoner to preform reasoning tasks for an $\mathcal{E}$-connected ontology.

However, such an approach implicitly assumes the availability of *global knowledge* in all ontology modules for the reasoning in a modular ontology to be possible. This counteracts many of the benefits of having a modular ontology; in particular, scalability and the preservation of module privacy. For example, as is implied by the CE-rule of its algorithm, the Pellet implementation requires that all ontology modules be loaded into the *same* memory space. Thus, this implementation implicitly requires the integration of all ontology modules. By contrast, P-DL, and also DDL, reasoning algorithms are genuinely distributed reasoning algorithms, not requiring, either implicitly or explicitly, the integration of ontology modules.

CHAPTER 6. Reasoning with Hidden Knowledge

In the previous chapters we explored selective knowledge reuse in modular ontologies. The focus of this chapter is on another aspect of selective knowledge reuse, i.e., reuse ontologies with partially hidden knowledge.

Part of this chapter was previously published in (Bao and Honavar, 2006c; Bao et al., 2007d).

6.1 Overview

The imperative for sharing information on the semantic web has to be balanced against copyright, privacy, security, or commercial concerns which require the participants to protect sensitive information e.g., data, knowledge, from other parties. Hence, there is a need for mechanisms that enable the participants to selectively share information with other parties without risking disclosure of sensitive information. Our focus in this chapter is on selective sharing of ontologies on the web: in particular, answering queries against an ontology, without disclosing *hidden knowledge*, i.e., knowledge that needs to be protected from disclosure.

Current proposals for *policy* languages (Tonti et al., 2003) for information hiding on the semantic web rely on complete denial of access to the hidden parts of an ontology when answering queries against the ontology. We argue that such approaches are overly restrictive in that they prohibit the use of hidden knowledge in answering queries even in scenarios where it is possible to do so without disclosing the hidden knowledge.

Against this background, this chapter examines the problem of answering queries against ontologies based on reasoning with hidden knowledge without risking its unintended disclosure. Specifically, we explore a framework for *privacy-preserving reasoning* on the semantic web. Unlike *access control* policies used in databases (Jajodia and Wijesekera, 2001) or their web counterparts (e.g., XACML (Godik and Moses, 2002)) that protect hidden knowledge on a

syntactic level, our approach protects hidden knowledge on the *semantic* level. Thus, queries against an ontology can be answered based on inference using hidden knowledge whenever it is possible to do so without disclosing the hidden knowledge.

The main contributions of the chapter are:

- A precise formulation of the problem of *privacy-preserving reasoning* on the semantic web.
- A general framework for privacy-preserving reasoning for semantic web ontologies that exploits the *indistinguishability* of hidden knowledge from *incomplete* knowledge under the *Open World Assumption* (OWA).
- A set of privacy-preserving reasoning strategies for *description logic* (DL) ontologies using the notion of *conservative extension* (Grau et al., 2006a).
- Privacy-preserving reasoning strategies for the important special case of *hierarchical ontologies* via a reduction of privacy-preserving reasoning to *graph reachability* analysis.
- Preliminary investigation for privacy-preserving reasoning with P-DL.

6.2 Motivating Examples

We start with some examples of applications to motivate the need for privacy-preserving reasoning on the semantic web.

Example 6.1 (Online Calendar): Consider Bob who uses an online calendar to manage his schedule and to coordinate his daily activities with others. Suppose the fact that Bob has a date with his girlfriend at noon is stored in Bob's calendar, along with additional knowledge, e.g., "date is a kind of activity" and "If person x has an activity at time t, person x is busy at time t" and so on. Suppose Bob does not wish to share with his colleagues that he has a date with his girlfriend at noon. However, it might be necessary for his colleagues to know that Bob is busy at noon. In such a scenario, a query to Bob's calendar as to whether Bob is busy at noon should be answered as "Yes" (which is inferred using both the sensitive knowledge and non-sensitive knowledge), whereas a query as to whether Bob has a date with his girlfriend at noon should be answered as "Unknown". However, if the use of hidden knowledge was forbidden, it would be impossible for the calender to inform Bob's colleagues that "Bob is busy at noon", although it is possible to do so, without revealing the details of Bob's noon-time activity.

Example 6.2 (Commercial Information Service): Consider a company, say *U-Travel* that provides travel information to online customers. Suppose *U-Travel* offers a query service that provides limited information to the public but more detailed information to subscribers who paid a fee. Suppose the *U-Travel*'s ontology contains the following knowledge: (a) Sun Lodge is a 2-star hotel (b) a 2-star hotel is a hotel. Suppose *U-Travel* is be willing to reveal that "Sun Lodge is a hotel" to the public, yet it wants hide the fact that "Sun Lodge is a 2-star hotel" from all but its subscribers. If *U-Travel* query service could not use hidden information i.e., that Sun Lodge is a 2-star hotel, it would not be able to inform a non-paying subscriber that Sun Lodge is a hotel, although it is possible to do so, without compromising the hidden knowledge.

Example 6.3 (Healthcare): (based on a similar example given in (Farkas et al., 2006)): Jane needs to take a certain preventive medicine which is intended for use by women who are believed to have a high risk of developing breast cancer. Jane does not want her pharmacy to supply the details of the prescription to her health insurance company, otherwise the insurance company may infer that she has a high risk of developing breast cancer and increase her health insurance premium. In such a setting, in order for Jane to be reimbursed by her insurance company, the pharmacy needs to be able to certify to the insurance company that Jane has indeed incurred a medical expense (without the full details) that is covered by her insurance policy.

One can easily imagine similar needs for selective sharing of inferences based on hidden knowledge in many other scenarios including, for example, business dealings between companies, interactions between different governmental agencies (e.g., intelligence, law enforcement, public policy), cooperation among independent nations on matters of global concern (e.g., counter-terrorism).

6.3 Privacy-Preserving Reasoning: General Framework

6.3.1 Partially Hidden Knowledge

We proceed to introduce the basic notion of hidden knowledge. A *knowledge base* (KB) K over a language L consists of a set of *axioms* $K = \{\alpha_1, ..., \alpha_n\}$. We assume that K *is consistent and every axiom in it is not a tautology.* We use $\mathsf{Sig}(\alpha_i)$ to denote the set of names occurring in an axiom α_i and $\mathsf{Sig}(K)$ to denote the *signature* of a KB K, $\mathsf{Sig}(K) = \cup_{i=1}^{n} \mathsf{Sig}(\alpha_i)$.

For a specific agent, the set of axioms in a KB K is divided into two mutually exclusive parts: a visible part K_v and a hidden part K_h, with the corresponding signatures $\mathsf{Sig}(K_v)$ and $\mathsf{Sig}(K_h)$. We call $\mathsf{Sig}(K_v)$ the *visible signature* and $\mathsf{Sig}(K_h) - \mathsf{Sig}(K_v)$ the *hidden signature*. In what follows, a wide hat (e.g., $\widehat{\mathsf{HiddenName}}$) is used to indicate that a name is hidden. We denote a KB K with a visible part K_v and a hidden part K_h by (K_v, K_h).

We write $K \vdash \gamma$ to mean that γ is *classically provable* from K. If every axiom in a KB K_2 is classically provable from another KB K_1, we say that K_1 *entails* K_2 and denote it as $K_1 \vdash K_2$.

In some scenarios, it is useful to tailor the hidden and the visible parts of a KB K with respect to different agents that might query K. We call the division of the visible and the hidden KB of K w.r.t. an agent a the *scope policy* of K for a. In principle, a KB may have different scope policies for different agents. In what follows, we will focus on "safe" query answering for one agent against a partially hidden KB.

An abstract scope policy can be specified by associating *scope limitation modifiers* (SLM) with names and axioms in a knowledge base, and in particular, for packages in a P-DL ontology (Bao et al., 2006f). An SLM controls the visibility of the corresponding name or axiom to agents[1].

Definition 6.1 (SLM) *The scope limitation modifier of a name or an axiom t in an ontology K is a boolean function $f(x,t)$, where x is an agent which can access t iff $f(x,t) = true$.*

For example, we can define SLMs as follows:

- $\forall x, public(x,t) := true$, means t is accessible everywhere.
- $\forall x, private(x,t) := (t \in x)$, means t is visible only locally (by the agent of its home package).

We can also define other types of SLMs as needed. For example, $\forall x, friend(x,t) := (x = P_1)$ will grant the access of t to a particular agent P_1. More complex SLMs may be realized using some concrete policy languages, e.g., KAoS (Uszok et al., 2003) or Rei (Kagal et al., 2003).

[1] WLOG, we assume that a KB or a module is always associated with an agent.

Example 6.4 : Consider an ontology $K = (K_v, K_h)$ of the *U-Travel* company. We use the *partial-order* relation $\leq$ to indicate concept inclusion. The hidden part K_h contains

$$\begin{aligned} \mathsf{SunLodge} &\leq \widehat{\mathsf{2StarHotel}} \\ \widehat{\mathsf{2StarHotel}} &\leq \mathsf{Inn} \end{aligned}$$

where the hidden signature is $\{\widehat{\mathsf{2StarHotel}}\}$. The visible part K_v contains

$$\begin{aligned} \mathsf{SunLodge} &\leq \mathsf{AAADiscountable} \\ \mathsf{Inn} &\leq \mathsf{Hotel} \end{aligned}$$

Hence, the visible signature is $\mathsf{Sig}(K_v) = \{\mathsf{SunLodge}, \mathsf{Inn}, \mathsf{AAADiscountable}, \mathsf{Hotel}\}$.

6.3.2 Privacy-Preserving Inference

Our basic approach to designing a privacy-preserving reasoner for a partially-hidden KB is to ensure that the answers to queries do not reveal hidden knowledge unintendedly. The central idea is to design a reasoner that exploits the *Open World Assumption* (OWA) of ontology languages, to make it impossible for the querying agent to distinguish between knowledge that is unknown to the reasoner (because of the incompleteness the KB) and the knowledge that is being protected by the reasoner. A query that cannot be safely answered without running the risk of disclosing hidden knowledge will be answered *as if* the reasoner lacks *the complete knowledge* to answer the query.

Unlike the Closed World Assumption (CWA) which is implicit in databases, OWA assumes that an ontology may be incomplete with regard to the knowledge of the world being modeled. Therefore, and failure to prove an assertion does not imply the validity of the negation of the assertion. For instance, in Example 6.1, when queried whether "Bob has a date with his girlfriend at noon", if the answer is "Unknown", the querying agent cannot conclude that "Bob does not have such a date" (the negation of the assertion). Consequently, the querying agent cannot determine if the relevant information (the details of Bob's noon-time activity) is not in the KB or if the information is in the KB but is protected.

Before we formalize the notion of privacy-preserving reasoning using hidden knowledge to answer queries against an ontology, we state some natural requirements that need to be met by a reasoner operating in the setting outlined above.

1. **Honesty**. The reasoner should not "lie". That is, answers produced by the reasoner should always be *consistent* with its KB.

2. **History Independence**. The reasoner should always respond to a given query q against a fixed KB K with the same answer regardless of the *history* of queries that have been posed against K.

3. **History Safety**. The reasoner must ensure that the answers it produces are *safe* in the sense that it is not possible for a querying agent to infer hidden knowledge based on the answers to past queries from the same reasoner and the visible part of KB.

The first requirement is desirable if the goal of the reasoner is to provide as much information as it can, without providing wrong information (i.e., information that is inconsistent with its KB). The last two requirements are natural because it is unrealistic to assume that any reasoner that is used on the semantic web can "memorize" all previous queries that it has answered or track the identity of every agent that has queried it.

We now proceed to define a reasoner and a privacy-preserving reasoner:

Definition 6.2 (Reasoner) *Let K be a KB over a language L, Q the query space (the set of possible assertions to be tested against K) over L, and A the answer space. A reasoner R for K is an algorithm that defines a function $R : K \times Q \to A$. For a specific KB K we define $R_K : Q \to A$ by setting $R_K(q) = R(K, q)$.*

An immediate consequence of this definition is that a reasoner R is "history independent" in the sense suggested by requirement 2 above.

R might employ an *inference engine* which can be viewed as a classical reasoner with answer space $A = \{\mathrm{Y}, \mathrm{N}\}$ such that $\forall q \in Q$, $R_K(q) = \mathrm{Y}$ iff $K \vdash q$ (thus, $R_K(q) = \mathrm{N}$ iff $K \nvdash q$). While an inference engine always responds in a truthful manner, a reasoner, in order to protect some parts of K, may have an incentive to pick an answering strategy which does not respond with the "whole truth". For example, a reasoner may answer "U" (Unknown) even if the correct answer (from the inference engine) is "Y" or "N". The answer to a query q may be "U" either because the reasoner has incomplete knowledge (i.e., $K \nvdash q$ and $K \nvdash \neg q$) under OWA, or because the "truthful" answer to q might risk disclosure of hidden knowledge. Under OWA,

because the querying agent can not distinguish between these two cases, the reasoner is able to answer queries based on inference using hidden knowledge without revealing it.

Definition 6.3 (Privacy-Preserving Reasoner) *Let $K = (K_v, K_h)$ be a KB over a language L, Q the query space in L, and $A = \{\mathrm{U}, \mathrm{Y}, \mathrm{N}\}$ the answer space, and R a reasoner for K. We define: $Q_Y = R_K^{-1}(Y), Q_N = R_K^{-1}(N), Q_U = R_K^{-1}(U)$. Clearly, $Q = Q_U \cup Q_Y \cup Q_N$.*
*(a) R is **strongly privacy-preserving** w.r.t. K if it satisfies the following two axioms:*

- *Honesty Axiom: $(q \in Q_Y \Rightarrow K \vdash q)$ and $(q \in Q_N \Rightarrow K \vdash \neg q)$.*
- *Strong Safety Axiom: $\forall \alpha$ that is not a tautology and $\mathsf{Sig}(\alpha) \subseteq \mathsf{Sig}(K_h)$, $K_h \vdash \alpha \Rightarrow (K_v \cup Q_Y \nvdash \alpha)$.*

*(b) R is **weakly privacy-preserving** w.r.t. K if it satisfies the Honesty Axiom and the following axiom:*

- *Weak Safety Axiom: $\forall \alpha$, $\alpha \in K_h \Rightarrow (K_v \cup Q_Y \nvdash \alpha)$*

The honesty axiom requires that reasoners provide answers that do not contradict the given KB (i.e., $K \cup Q_Y$ is consistent). The strong safety axiom requires that the answers provided by reasoners do not disclose any consequence that can be drawn from the hidden knowledge alone. The weak safety axiom requires the reasoner to protect only axioms that are explicitly mentioned in the hidden part of the KB (but not necessarily their consequences). It is easy to see if a reasoner is strongly privacy-preserving then it is also weakly privacy-preserving.

The distinction between "strong safety" and "weak safety" is useful since different applications need different degrees of privacy preservation. In the *U-Travel* example, if the ontology provider is willing to disclose consequences of the hidden knowledge, e.g., "$\mathsf{SunLodge} \leq \mathsf{Inn}$", it can utilize a weakly privacy-preserving reasoner. On the other hand, in the online calendar example, suppose we have an additional piece of hidden knowledge "Alice is Bob's girl friend". Now, if Bob wants to protect *any* conclusion that may follow from the hidden part of his KB, e.g., that "Bob has a date with Alice at noon", Bob will need a strongly privacy-preserving reasoner.

It can be shown that in a general setting, strong safety is a very restrictive requirement. For example, if there exist axioms $\beta \in K_h$ and $\gamma \in K_v$ (with $\mathsf{Sig}(\gamma) \subseteq \mathsf{Sig}(K_h)$) such that $\beta \vee \gamma$

is not a tautology, then there is no strongly privacy-preserving reasoner for $K = (K_v, K_h)$. On the other hand, weakly privacy-preserving reasoners exist for any KB that satisfies $\alpha \in K_h \Rightarrow K_v \nvdash \alpha$. Intuitively, this means that no hidden axiom is provable from the visible KB. However, as we shall see later, it is possible to design strongly privacy-preserving reasoners in special cases, for instance, hierarchical ontologies (e.g., the *U-Travel* example).

6.3.3 General Strategies

In this section, we discuss general strategies to designing privacy-preserving reasoners.

Definition 6.4 (Strategy) *Let L be a language, $\boldsymbol{K}_L$ the class of all knowledge bases over L, and $\boldsymbol{R}_L$ the class of all reasoners over $\boldsymbol{K}_L$. A strategy for L is a function $\mathfrak{R} : \boldsymbol{K}_L \rightarrow \boldsymbol{R}_L$ such that for every $K \in \boldsymbol{K}_L$, $R = \mathfrak{R}(K)$ is a reasoner for K. The strong/weak safety scope of a strategy $\mathfrak{R}$, $Scope(\mathfrak{R}) = \{K \in \boldsymbol{K}_L |\ \mathfrak{R}(K)$ is a strongly/weakly privacy-preserving reasoner for $K\}$.*

A strategy needs to compromise between two possibly conflicting goals:

1. *Generality*: An ideal strategy has the largest possible safety scope, i.e., is able to yield safe reasoners for the largest possible subclass of $\mathbf{K}_L$.

2. *Informativeness*: An ideal strategy is one that yields reasoners that provide as much information as possible in their answers to queries against their KBs, that is, reasoners that result in the smallest possible Q_U.

The following two strategies correspond to the "extreme" choices with respect to these two goals:

- *Dummy Strategy*, i.e., one that always generates a *dummy reasoner*, who answers "U" to every possible query against its KB. Obviously, a dummy reasoner is weakly privacy-preserving for any KB $K = (K_v, K_h)$ such that $\forall\alpha,\ \alpha \in K_h \Rightarrow K_v \nvdash \alpha$. *Note that this condition is the weakest condition for a KB to have privacy-preserving reasoners.* A dummy strategy is most general, but least informative. It has the largest scope, but answers given by reasoners that are based on it provide no information at all.

- *Naive Strategy*, i.e., one that generates a *naive reasoner* that reveals everything that follows from its knowledge base, including the hidden part of the KB. A naive reasoner is most informative, but is least general: It is privacy-preserving only for those KB that have no hidden knowledge at all (i.e., $K_h = \emptyset$).

In practice, we may need to make tradeoff between the conflicting requirements of generality and informativeness of strategies.

We now proceed to present a general approach for generating weakly privacy-preserving reasoners for semantic web ontologies based on the notion of *conservative extension* (Grau et al., 2006a). The basic idea behind this approach is as follows: Answers to previous queries may be used by the querying agent to *extend* the visible part of the KB. The safety of the strategy can be guaranteed if we can ensure that no conclusions compromising the hidden knowledge can be inferred from such an extension.

Definition 6.5 (Conservative Extension, (Grau et al., 2006a)) *Let K and K' be two knowledge bases. $K \cup K'$ is a conservative extension of K, written as $K \cup K' \rightleftarrows K$, if for every formula α such that $\mathsf{Sig}(\alpha) \subseteq \mathsf{Sig}(K)$, $K \cup K' \vdash \alpha$ iff $K \vdash \alpha$.*

Let $K_{vc} \subseteq K_v$ be the minimal set of visible axioms that contain names in $\mathsf{Sig}(K_h)$, i.e., $\mathsf{Sig}(K_v) \cap \mathsf{Sig}(K_h) \subseteq \mathsf{Sig}(K_{vc})$. Intuitively, because the querying agent does not know names that are not in $\mathsf{Sig}(K_v)$, the names in K_{vc} correspond to "critical signature", i.e., the subset of $\mathsf{Sig}(K_h)$ that is known to the querying agent. If we can ensure that answers to queries together with $K_v - K_{vc}$ do not reveal any knowledge about $\mathsf{Sig}(K_h)$ beyond those in K_{vc}, we can effectively protect every axiom in K_h. Therefore, if we can ensure that any extension of K_v with the results of previous queries is a conservative extension of the *critical* visible axioms K_{vc}, we can protect hidden knowledge. The following lemma captures this intuition more formally:

Lemma 6.1 *Let $K = \{K_v, K_h\}$ be a KB such that $\forall \alpha,\ \alpha \in K_h \Rightarrow K_v \nvdash \alpha$, R a reasoner for K. R is a weakly privacy-preserving reasoner for K if it satisfies the honesty axiom and $K_v \cup Q_Y \rightleftarrows K_{vc}$.*

Proof: We only need to show that the weak safety axiom holds under the stated conditions:

$$\begin{aligned} \alpha \in K_h &\Rightarrow K_v \not\models \alpha \\ &\Rightarrow K_{vc} \not\models \alpha \\ &\Rightarrow K_v \cup Q_Y \not\models \alpha \qquad \square \end{aligned}$$

6.4 Privacy-Preserving Reasoning with $\mathcal{SHIQ}$ Ontologies

In this section we present a "safe" reasoning strategy based on conservative extensions for the description logic $\mathcal{SHIQ}$, which covers a significant part of OWL. (Grau et al., 2007) have shown that in the special case of *semantically local* ontologies, it is possible to check whether an extension of a $\mathcal{SHIQ}$ ontology is a conservative extension in polynomial time.

Informally, an axiom is *semantically local* w.r.t. a signature S if it imposes no restrictions on the interpretation of names in S. A finite set of axioms is local w.r.t. S if every axiom in it is local w.r.t. S. Practical ways to ensure *semantic locality* of $\mathcal{SHIQ}$ ontologies have been elucidated by (Grau et al., 2007):

Definition 6.6 (Locality) *(*Definition 3 of (Grau et al., 2007)*) Let $\boldsymbol{S}$ be a $\mathcal{SHIQ}$-signature and let $\boldsymbol{E} \subseteq \boldsymbol{S}$. Positively local concepts C_E^+ and negatively local concepts C_E^- are defined as follows:*

$C_E^+ := A|(\neg C^-)|(C \sqcap C^+)|(\exists R^+.C)|(\exists R.C^+)|(\geqslant n\ R^+.C)|(\geqslant n\ R.C^+)$

$C_E^- := (\neg C^+)|(C_1^- \sqcap C_2^-)$

where A is a concept name from $\mathbf{S}\backslash\mathbf{E}$, $R \in \mathsf{Rol}(\mathbf{S})$, $C \in \mathsf{Con}(\mathbf{S})$, $C^+ \in C_E^+$, $C_{(i)}^- \in C_E^-, i = 1, 2$, and $R^+ \notin \mathsf{Rol}(\mathbf{E})$. A role inclusion axiom $R^+ \sqsubseteq R$ or transitivity axiom $\mathsf{Trans}(R^+)$ is local w.r.t. $\mathbf{E}$. A GCI is local w.r.t. $\mathbf{E}$ if it is either of the form $C^+ \sqsubseteq C$ or $C \sqsubseteq C^-$, where $C^+ \in C_E^+$, $C^- \in C_E^-$ and $C \in \mathsf{Con}(\mathbf{S})$. A $\mathcal{SHIQ}$ ontology K is local w.r.t. $\boldsymbol{E}$ if every axiom in it is local w.r.t. $\boldsymbol{E}$.

(Grau et al., 2007) established relationship between the notions of conservative extension and locality of ontologies which we summarize (adapted for simpler presentation in our setting) in the following lemma:

Lemma 6.2 *(*Definition 3 and Lemma 5 of (Grau et al., 2007)*) Suppose K_1 and K_2 are two*

$\mathcal{SHIQ}$ TBoxes such that K_1 is local w.r.t. $\mathsf{Sig}(K_2)$ and K_2 is local w.r.t. $\emptyset$. Then $K_1 \cup K_2$ is a conservative extension of K_2.

We can now define $\mathfrak{R}_{CE}$ (read *CE*-strategy), a reasoning strategy for $\mathcal{SHIQ}$ ontologies, based on the notion of conservative extension. Given a $\mathcal{SHIQ}$ TBox $K = \{K_v, K_h\}$ and subsumption query q, $\mathfrak{R}_{CE}$ specifies a reasoner for K that answers q as follows:

```
IF q is local w.r.t. Sig(K_vc) and Sig(q) ⊆ Sig(K_v)
   IF K ⊢ q, return Y
   ELSE IF K ⊢ ¬q, return N
          ELSE return U /*incomplete knowledge)*/
ELSE return U /*hidden knowledge*/
```

Note that if $q := C \sqsubseteq D$, $K \vdash q$ means $\forall \mathcal{I}$, $\mathcal{I} \vDash K \Rightarrow C^{\mathcal{I}} \backslash D^{\mathcal{I}} = \emptyset$, and $K \vdash \neg q$ means $\forall \mathcal{I}$, $\mathcal{I} \vDash K \Rightarrow C^{\mathcal{I}} \backslash D^{\mathcal{I}} \neq \emptyset$.

Lemma 6.3 *The weak safety scope of $\mathfrak{R}_{CE}$ includes all $\mathcal{SHIQ}$ TBoxes $K = \{K_v, K_h\}$ that satisfy the following properties:*

- *$K_v - K_{vc}$ is local w.r.t. $\mathsf{Sig}(K_{vc})$;*
- *K_{vc} is local w.r.t. $\emptyset$;*
- *$\forall \alpha \in K_h$, $K_v \nvdash \alpha$*

Proof: Let $R = \mathfrak{R}_{CE}(K)$, where K satisfies the given properties. Clearly, R satisfies the honesty axiom. By the definition of the algorithm, Q_Y is local w.r.t. $\mathsf{Sig}(K_{vc})$. Since $K_v - K_{vc}$ is local w.r.t. $\mathsf{Sig}(K_{vc})$, $Q_Y \cup (K_v - K_{vc})$ is local w.r.t. $\mathsf{Sig}(K_{vc})$. Since K_{vc} is local w.r.t. $\emptyset$, by Lemma 6.2, $(K_v - K_{vc}) \cup Q_Y \cup K_{vc} = K_v \cup Q_Y \rightleftarrows K_{vc}$. By Lemma 6.1, R is a weakly privacy-preserving reasoner for K. $\square$

It is worth noting that we do not require the hidden knowledge K_h also to be local, as long as $\alpha \in K_h \Rightarrow K_v \nvdash \alpha$. This affords considerable flexibility for ontology engineers in designing the KB.

An important advantage of the *CE*-strategy for $\mathcal{SHIQ}$ ontologies is that a weakly privacy-preserving reasoner can be built as a meta reasoner which calls inference service from a standard

DL reasoner for $\mathcal{SHIQ}$. Thus, implementing a weakly privacy-preserving reasoners for $\mathcal{SHIQ}$ ontologies is quite straightforward.

6.5 Privacy-Preserving Reasoning with Hierarchical Ontologies

Unlike in the case of general DL ontologies, it is possible to define a strongly safe strategy, and hence strongly safe privacy-preserving reasoners in the case of *hierarchical* ontologies e.g., tree or DAG-structured ontologies.

Formally, a hierarchical ontology K over a finite set of names S can be represented as a set of visible or hidden partial order axioms, denoted by $K = (S, \leq)$ (as illustrated in Example 6.4). Reasoning with K can be reduced to the graph reachability problem by defining a corresponding directed graph $G = (V, E)$, where V is the vertex set corresponding to elements of S, and E is the edge set corresponding to $\leq$ axioms. Let $\mathbf{G}$ be the set of all directed graphs. In the following, we will identify K with the corresponding G.

A vertex (or edge) is said to be a visible vertex (or edge) if it is mapped from a visible term (or axiom); otherwise it is said to be hidden. A hidden edge only connects hidden vertices. Let E_h be the set of all hidden edges, E_v be the set of all visible edges, and $E = E_h \cup E_v$. For any set edges $F \subseteq E$, let F^+ denote the transitive closure of F, and $F^{\leq m} = \cup_{k=1}^{m} F^k$.

For any two visible vertices x and y, $y \leq x$ if x is reachable from y in the graph G, i.e., there exists a path from y to x (which in general, may contain both visible and hidden edges). Note that because of the open world assumption, it is not necessarily the case that $y \leq x$ is false simply because there is no path from y to x in G.

An affirmative answer to a query about the reachability from y to x in G is equivalent to augmenting G by adding a new visible edge $\langle y, x \rangle$. Hence, in order to realize privacy-preserving reasoning, we should ensure that the initial graph G (derived from K) can be augmented with previous answers without revealing the existence of hidden edges.

First, it is easy to see that strong safety and weak safety properties can be reduced to each other in the case of hierarchical ontologies:

Lemma 6.4 *R is a strongly privacy-preserving reasoner for $G = (V, E_v \cup E_h)$ iff R is a weakly privacy-preserving reasoner for $G = (V, E_v \cup E_h^+)$.*

This lemma follows from the fact that E_h^+ contains all possible inference results that can be obtained by considering *only* the hidden edges E_h. Henceforth, we will focus on weak safety.

We now proceed to define several classes of graphs that have safe strategies with different degree of informativeness:

$$\mathbf{S}_{m,n} = \{G \in \mathbf{G} | (E^{\leq m} - E_h)^{\leq n} \cap E_h = \emptyset\},$$

where m, n can be "+" indicating transitive closure. Graphs in $\mathbf{S}_{m,n}$ are called (m,n)-safe. Intuitively, m represents the ability of the reasoner to detect possible safety hazards, and n represents the ability of the querying agent to discover knowledge from previous answers and the visible part of the graph. We are only interested in the case when n is + since it represents the case when the querying agent is the most powerful[2]. It is easy to verify that: $\mathbf{S}_{+,+} = \cap_{m=1}^{\infty} \mathbf{S}_{m,+}$.

Now we will consider several specific reasoning strategies for those classes of graphs. Since there is no negation, the resulting reasoners will answer only with "Y" or "U" (i.e., there are no "N" answers). Note that requirements for a weakly privacy-preserving reasoner in this context are:

- **Honesty Axiom**: $Q_Y \subseteq E^+$
- **Weak Safety Axiom**: $(E_v \cup Q_Y)^+ \cap E_h = \emptyset$.

The *dummy* reasoner: A dummy reasoner responds to every query with the answer "U" (i.e., $Q_Y = \emptyset$). It preserves the safety of precisely those graphs that satisfy $E_v^+ \cap E_h = \emptyset$. This is exactly the defining condition of the class of (1,+)-safe graphs $\mathbf{S}_{1,+}$. Obviously, this strategy has the widest safety scope, and not surprisingly, is also least informative.

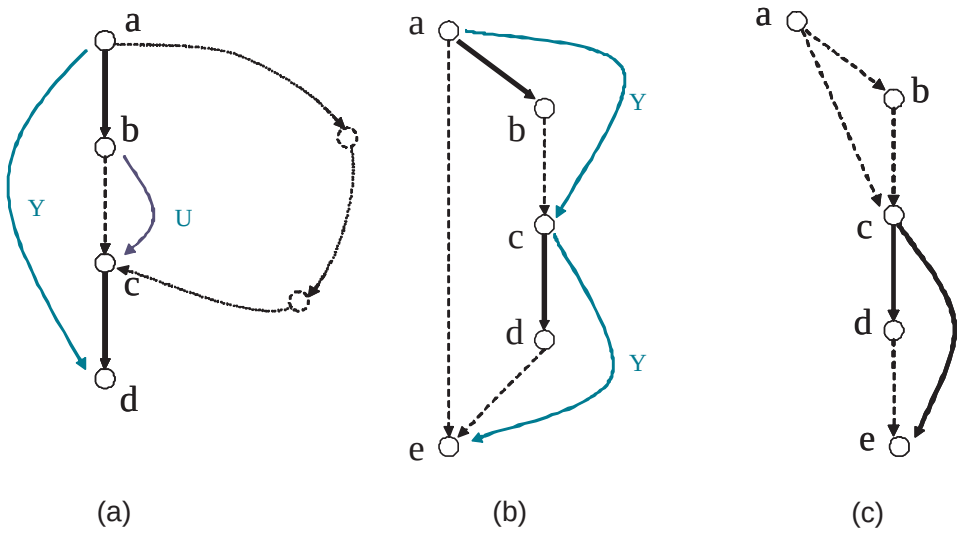
The *obvious* reasoner: An obvious reasoner responds with an answer "Y" to only those queries whose answers follow from E_v i.e. $Q_Y \subseteq E_v^+$. Its weak safety scope is the same as that of the dummy reasoner ($\mathbf{S}_{1,+}$).

The *safe* reasoner: let $Q_Y = E^+ - E_h$. Clearly, this reasoner satisfies the honesty axiom. To satisfy the weak safety axiom, we need $(E_v \cup (E^+ - E_h))^+ \cap E_h = \emptyset$, i.e., $(E^+ - E_h)^+ \cap E_h = \emptyset$. Hence, its weak safety scope is $\mathbf{S}_{+,+}$ (which is smaller than $\mathbf{S}_{1,+}$). Similarly, if $Q_Y = E^{\leq m} - E_h$, the corresponding weak safety scope is $\mathbf{S}_{m,+}$.

[2]We note that for every m, n, we can easily construct a reasoning strategy whose safety scope is $\mathbf{S}_{m,n}$.

The *naive* reasoner: this reasoner always gives away all the information it has, i.e., $Q_Y = E^+$. It is trivially honest, and its weak safety scope consists of only those graphs that do not have any hidden edges, i.e, $\{G|E_h = \emptyset\}$, which is clearly a subset of $\mathbf{S}_{+,+}$.

The above reasoning strategies for hierarchical ontologies clearly illustrate the tradeoff between generality and informativeness. Among these, the *safe* reasoner is of particular interest since it is able to generate informative answers for a fairly large class of graphs without compromising the hidden knowledge [3]. Given a query $x \leq y$, it will answer "Y" if $\langle x, y \rangle$ is in E^+ but *not* in E_h, and "U" otherwise.



Solid edges are visible, dashed edges are hidden, edges labeled with "Y" or "U" are queries with corresponding answers, dotted edges in (a) show some possible edges that are not present in the graph.

Figure 6.1 Safety of Hierarchical Ontologies

Some examples of hierarchical ontologies are shown in Figure 6.1 as graphs. The graph in (a) is $(+,+)$-safe. For instance, the query $a \leq d$ can be answered "Y" using both visible and hidden edges without revealing hidden edge $b \leq c$. This is because, due to OWA, there might exist unknown paths that connect a and d but not b and c. The graph shown in (b) is not $(+,+)$-safe: two "seemingly safe" queries $a \leq c$ and $c \leq e$ may be combined to reveal the hidden edge $a \leq e$. Note that the graph will become $(+,+)$-safe if we remove the hidden edge $a \leq e$.

[3] A simple Java implementation of the reasoner is available at http://www.cs.iastate.edu/~baojie/pub/wi2007

It is easy to verify that the graphs in Figure 6.1 (c) and Example 6.4 are also $(+,+)$-safe.

The *safe* reasoner only requires computation of transitive closure to test if a graph is safe, which takes $O(|V|^3)$ time and $O(|V|^2)$ space. More efficient algorithm exists for sparse, tree-like ontologies (Wang et al., 2006): reachability problem can be answered in constant time, with index time $O(|V|+|E|+t^3)$ and index size $O(|V|+t^2)$, where $t \ll |V|$ is the number of *non-tree* edges.

6.6 Discussion: Privacy-Preserving Reasoning in P-DL

6.6.1 Overview

In the previous sections we discussed privacy-preserving reasoning in a single ontology. It would be interesting to explore whether such approaches can be extended to the case of *privacy-preserving reasoning with modular ontologies*. The objective of this section is to explore several minimal requirements of a privacy-preserving reasoner for modular ontologies in P-DL languages. While the design of such a reasoner is still remained open, we believe discussion in this section will be helpful for the future investigation on this topic.

Privacy-preserving reasoning in modular ontologies is different from that in classic DLs (e.g., $\mathcal{SHIQ}$) in that knowledge is distributed in multiple modules and controlled by different agents. In such a setting, it is not possible to integrate all modules into a single ontology and use a single reasoner, which has the access to the *global knowledge* about all modules, to perform a reasoning task. Hence, it is necessary to preform the reasoning task in a *distributed* fashion, such that no agent of any module may infer hidden knowledge in any other module during the reasoning process.

In the last chapter, we described several distributed reasoning algorithms for P-DL, where the overall reasoning task is undertook by a federation of multiple local reasoners, each for an ontology module (i.e., a package in P-DL). Local reasoners may exchange messages, including concept and role reporting messages and clash messages, to construct a distributed tableau for the ontology in question. Extending such a strategy to working with partially hidden knowledge presents several challenges, including the following:

- What is the precise notion of privacy-preserving reasoning in the distributed setting?

- Can we ensure that, for any sequence of messages exchanged between any local reasoners, no local reasoner may *logically* infer hidden knowledge of any other package from those messages?

- For what types of P-DL ontologies there exists useful (e.g., reasonably informative) privacy-preserving reasoning strategies?

In what follows we will conduct some preliminary investigations on those problems. For the cause of simplicity, we only consider the P-DL $\mathcal{ALCP}_{\mathcal{C}}$, i.e., $\mathcal{ALC}$ extended by (possibly cyclic) concept importing between packages. However, similar results may be obtained for more expressive P-DLs, e.g., $\mathcal{SHIQP}$.

It should be noted that in this section we are restricted to only *logical safety* of reasoning, i.e., no hidden knowledge may be inferred by logical methods. We do not address the *statistical* aspect of privacy preserving reasoning or belief revising based on multiple queries, e.g., (Gray and Syverson, 1998). Combining the strength of the two approaches will be our future work.

6.6.2 Distributed Privacy-Preserving Reasoning: General Setting

6.6.2.1 Distributed Partially Hidden Knowledge

For an $\mathcal{ALCP}_{\mathcal{C}}$ ontology $\langle\{K_i\}, \{K_i \mapsto K_j\}_{i\neq j}\rangle$, each package K_i is the disjoint union of a hidden part K_{ih} and a visible part K_{iv}[4]. We require that imported names can not be in K_i's hidden signature (i.e., $\mathsf{Sig}(K_i) - \mathsf{Sig}(K_{iv})$) and no hidden name in K_i^+ can be imported by K_i.

We use $K_i \vdash \alpha$ to mean that α is witnessed by K_i regardless the division of visible and hidden knowledge.

For an $\mathcal{ALCP}_{\mathcal{C}}$ ontology Σ, we denote by Σ_v the visible part of all packages $\langle\{K_{iv}\}, \{K_{iv} \mapsto K_{jv}\}_{i\neq j}\rangle$.

We assume each package is controlled by an autonomous agent which may query other packages using a reasoner. An agent can access both visible and hidden knowledge in the package of its own, but only "see" the visible parts of packages of other agents. Hence, an agent may not know the complete imported signature of another package.

[4]It may also be possible that each package has different scope limitation policies toward different other packages. For the cause of simplicity, we do not consider privacy-preserving reasoning in such a setting and will investigate it in the future.

6.6.2.2 Message Safety: Informal Description

In the course of reasoning in an $\mathcal{ALCP}_{\mathcal{C}}$ ontology, the agent of each package may use a local reasoner[5] which may exchange messages with reasoners of other packages in the manner we described in the last chapter. Note that in the $\mathcal{ALCP}_{\mathcal{C}}$ tableau algorithm, there is only concept reporting messages but no role reporting messages.

There are two types of communication between a local reasoner and the outside world: the queries and answers it exchanged with an agent who submit those queries, and messages (i.e., concept labels of some individuals) it exchanged with other local reasoners. Accordingly, there are two general types of "unsafe" behavior of a local reasoner, i.e., unsafe answers to queries (which is similar to the single ontology setting), and unsafe messages exchanged with other local reasoners under certain circumstances, which are demonstrated by the following example.

Example 6.5 : Suppose we have a P-DL ontology with two packages:

$K_1 = K_{1h} = \{1 : C \sqcap 1 : D \sqsubseteq \bot\}$ and

$K_2 = K_{2v} = \{2 : E \sqsubseteq 1 : C \sqcap 1 : D\}$

Suppose $R_{1,2}$ are local reasoners for $K_{1,2}$, respectively, which will construct local completion graphs $G_{1,2}$. The reasoning task is to check the satisfiability of $2 : E$ as witnessed by K_2. G_2 has an initial node $2 : x$ such that $\mathcal{L}_2(2 : x) = \{E\}$ in the thread t_0. Applying $\mathcal{ALCP}_{\mathcal{C}}$ expansion rules (all operations belong to thread t_0):

1. $\mathcal{L}_2(2 : x) = \{E, C, D\}$;
2. two messages $r^{1\leftarrow 2}(x, C)$ and $r^{1\leftarrow 2}(x, D)$ are sent to G_1, hence a node $1 : x$ is created in G_1 with $\mathcal{L}_1(1 : x) = \{C, D\}$
3. G_1 sends back a clash message $\bot^{1\rightarrow 2}(t_0)$.

If there is *no* other local reasoner involved in the reasoning process, R_2 may infer that $C \sqcap D$ is unsatisfiable in K_1, hence the hidden knowledge of K_1 is compromised. (End)

We extend the (informal) requirements for privacy-preserving reasoners to the distributed setting. In addition to Honesty, History Independence and History Safety requirements, a local reasoner should also satisfy that

[5]We refer by "the local reasoner of a package" to mean the reasoner used by the agent of the package henceforth.

4 **Message Safety**. For any local reasoner, no other local reasoners can logically infer hidden knowledge in the package of this local reasoner through messages exchanged between them.

6.6.2.3 Message Safety: Formal Definition

Formally, we have the following definition:

Definition 6.7 *Let $K = \langle \{K_i\}, \{K_i \mapsto K_j\}_{i \neq j} \rangle$ be an $\mathcal{ALCP}_{\mathcal{C}}$ ontology. Let $\boldsymbol{R}$ be a set of reasoners for K, such that every $R_i \in \boldsymbol{R}$ a reasoner for K_i based on the $\mathcal{ALCP}_{\mathcal{C}}$ tableau algorithm. Let G_i be the local completion graph generated by R_i.*

For every concept reporting message exchanged among $\boldsymbol{R}$, let node $i : x$ be the sender and node $j : x$ be the receiver, we say $(i : x, j : x)$ is a message edge. A message path is a sequence of message edges, such that the receiver of a message (except the last one) in the path is the sender of the next message and there is no circle in the path.

For every $i \neq j$, every thread t, and every node $i : x$ (in R_i), a conversation $\boldsymbol{C}^{i \leftrightarrow j}(x, t)$ of t and x between R_i and R_j is a sequence of messages such that:

- *Each message is either a concept report message or a clash message exchanged between $i : x$ and its image or preimage nodes $j : x$ in R_j (i.e., $origin(i : x) = origin(j : x)$) in the same thread linage of t;*
- *There is at most one clash message in the conversation; if it exists, it must be the last one in the conversation and have thread argument t;*

Note that one clash message may exists in different conversations for different nodes.

We use $C^{i \rightarrow j}(x, t)$ to denote the conjunction of concepts (note that they are concept names or its negation in $\mathsf{Sig}(K_i) \cap \mathsf{Sig}(K_j)$) occurred in concept report messages from $i : x$ to $j : x$ in the conversation $\boldsymbol{C}^{i \leftrightarrow j}(x, t)$.

A conversation $C = \boldsymbol{C}^{i \leftrightarrow j}(x, t)$ is risky *for i against j if*

1) $i : x$ is created by a concept reporting message from $j : x$ (hence the first message in C is a backward concept reporting message from $j : x$);

2) Let S be the set of nodes exists before $i : x$ is created and have the same origin as $i : x$, then all message paths from a node in S to $i : x$ share the same last message edge $(j : x, i : x)$;

3) C is ending with a clash message $\perp^{i\rightarrow j}(t)$;

4) At least one message from j to i belongs to the thread t;

5) There is no other node $i:y$ such that $\boldsymbol{C}^{i\leftrightarrow j}(y,t)$ satisfies conditions 1-4.

Intuitively, the fact that a conversation $\mathbf{C}^{j\leftrightarrow k}(x,t)$ is risky for j against k means that expansions at $j:x$ are *only* influenced by $k:x$. Thus, R_k may potentially infer a concept inclusion witnessed by K_j, as shown in Example 6.5. Formally, we have the lemma:

Lemma 6.5 *Let K $\langle\{K_i\},\{K_i\mapsto K_j\}_{i\neq j}\rangle$ be an $\mathcal{ALCP}_\mathcal{C}$ ontology, R_i a reasoner for K_i based on the $\mathcal{ALCP}_\mathcal{C}$ tableau algorithm (for every i). Let $R_{j,k}$ be reasoners for packages $K_{j,k}$ in K, if there is a conversation $\boldsymbol{C}^{j\leftrightarrow k}(x,t)$ that is risky for j against k, for some node $j:x$ and thread t, then $C^{j\rightarrow k}(x,t)\sqsubseteq_j\perp$.*

Proof sketch: $j:x$ must be a local top node as a result of the $\mathcal{ALCP}_\mathcal{C}$ tableau expansion. Expansions at $j:x$ are the logical consequence of messages it received. Let S be the set of nodes exists before $j:x$ is created and have the same origin as $j:x$, it must be case that expansions at $j:x$ are the logical consequence of messages sent from $k:x$. Hence, if $\mathbf{C}^{j\leftrightarrow k}(x,t)$ is risky, it is equivalent to query the satisfiability of $C^{j\rightarrow k}(x,t)$ against K_j with a negative answer. Hence R_k may infer that $C^{j\rightarrow k}(x,t)\sqsubseteq_j\perp$.

Note that $j:x$ may send message to some nodes in S, or send/receive messages with some nodes not in S and not "reachable" by message paths from nodes in S except through $(k:x,j:x)$. $j:x$'s local descendants may also communicate with nodes in other reasoners. Those communications do not change the conclusion. (End)

Note that if any of the 5 conditions for risky conversation in Definition 6.7 is not satisfied, $C^{jk}(x,t)\sqsubseteq_j\perp$ can not be inferred, as shown by following examples.

Example 6.6 (Condition 1 Relaxed): $K_1=\{1:D\sqsubseteq\exists R.2:C\}$ $K_2=\{2:C\sqsubseteq 2:E\}$,. The reasoning task is to check satisfiability of $(1:D\sqcap(\forall 1:R.\neg_1 2:C)$ witnessed by K_1. A distributed completion graph may be constructed as follows:

$G_1:\{E,\exists R.C,\forall R.\neg_1 C\}\in\mathcal{L}_1(x),\{D,C,\neg_1 C\}\in\mathcal{L}_1(y),\mathcal{L}_1(\langle x,y\rangle)=\{R\},r_{21}=\langle 2:y,1:y\rangle$;

$G_2:\mathcal{L}_2(y)=\{C,D\}$.

The messages exchanged between G_1 and G_2 are $r^{2\leftarrow 1}(x,C)$, $r^{2\rightarrow 1}(x,D)$ and the clash message $\perp^{1\rightarrow 2}$ (thread argument omitted). However, D is satisfiable witnessed by K_1.

Example 6.7 (Condition 2 Relaxed): Suppose we have four $\mathcal{ALCP}_{\mathcal{C}}$ packages:

$K_1 : \{2 : D \sqsubseteq 3 : E\}$

$K_2 : \{4 : C \sqsubseteq 2 : D\}$

$K_3 : \{3 : E \sqsubseteq 4 : F\}$

$K_4 : \{4 : C \sqcup 4 : F \sqsubseteq \top_4\}$

The reasoning task is to check satisfiability of $C \sqcap \neg_4 F$ witnessed by K_1. Using $\mathcal{ALCP}_{\mathcal{C}}$ expansion rules, we can construct a distributed completion graph as follows:

$G_1 : \mathcal{L}_1(x) = \{C, \neg_4 F, D, E\}$, $r_{21} = \langle 2 : x, 1 : x\rangle$, $r_{31} = \langle 3 : x, 1 : x\rangle$, $r_{41} = \langle 4 : x, 1 : x\rangle$

$G_2 : \mathcal{L}_2(x) = \{C, D\}$, $r_{42} = \langle 4 : x, 2 : x\rangle$

$G_3 : \mathcal{L}_3(x) = \{C, E, F\}$, $r_{43} = \langle 4 : x, 3 : x\rangle$

$G_4 : \mathcal{L}_4(x) = \{C, \neg_4 F, F\}$

The messages exchanged between G_3 and G_4 are $r^{4\leftarrow 3}(x, F)$ and the clash message $\perp^{4\rightarrow 3}$ (thread argument omitted). However, F is satisfiable witnessed by K_4. The clash detected at $4 : x$ is the result of messages from $G_{1,2,3}$ together. (End)

Example 6.8 (Condition 3 Relaxed): Let $K_j = K_{jh} = \{C \sqsubseteq D\}$ where C, D are visible j-concept names. K_i import C, D from K_j. The local completion graph G_i sends a message $r^{j\leftarrow i}(x, C)$ to G_j. G_j may send back the message $r^{j\rightarrow i}(x, D)$. G_i can not infer from the answer that $C \sqsubseteq_j D$. For example, it is possible that $C \sqsubseteq D \sqcup E$ in K_j where E is a hidden concept name in K_j. Due to the non-deterministic nature of tableau expansion, G_i can not distinguish from the two scenarios based on the exchanged messages. (End)

Example 6.9 (Condition 4 Relaxed): Let $K_1 = K_{1h} = \{1 : C \sqsubseteq 1 : D \sqcup 1 : E, 1 : E \sqsubseteq \neg_1 1 : C\}, K_2 = \{1 : C \sqcap 1 : D \sqsubseteq \top_2\}$. The reasoning task is to check satisfiability of $1 : C$ witnessed by K_2. The local completion graph G_2 sends a message $r^{1\leftarrow 2}(x, C)$ of thread t_0 to G_1. G_1 may make a non-deterministic choice (hence generates a new thread t_1 as a child thread of t_0) and add $\{E, \neg_1 C\}$ to $\mathcal{L}_1(x)$ and send back a clash with argument t_1. However, G_2 cannot infer that C is unsatisfiable in K_1. (End)

Example 6.10 (Condition 5 Relaxed): $K_1 = \emptyset, K_2 = \{2 : C \sqsubseteq \perp\}$, K_1 imports $2 : C, 2 : D$ from K_2. The reasoning task is to check satisfiability of $(\exists 1 : R.2 : C) \sqcap (\exists 1 : R.2 : D)$ witnessed by K_1. A distributed completion graph may be constructed as follows:

$G_1 : \mathcal{L}_1(x) = \{(\exists 1 : R.2 : C) \sqcap (\exists 1 : R.2 : D)\}$, $\mathcal{L}_1(y_1) = \{C\}$, $\mathcal{L}_1(y_2) = \{D\}$ $\mathcal{L}_1(\langle x, y_1\rangle) = \mathcal{L}_1(\langle x, y_2\rangle) = \{R\}$, $r_{21} = \{\langle 2 : y_1, 1 : y_1\rangle, \langle 2 : y_2, 1 : y_2\rangle\}$;

$G_2 : \mathcal{L}_2(y_1) = \{C\}, \mathcal{L}_2(y_2) = \{D\}$.

The messages exchanged between G_1 and G_2 are $r^{2\leftarrow 1}(y_1, C)$, $r^{2\leftarrow 1}(y_2, D)$ and the clash message $\bot^{2\rightarrow 1}$ (thread argument omitted). However, G_1 cannot distinguish $C \sqsubseteq_2 \bot$ or $D \sqsubseteq_2 \bot$.

6.6.3 Requirement for Distributed Privacy-preserving Reasoners

We now proceed to describe requirements for *distributed privacy-preserving local reasoners*.

Definition 6.8 *Let $K = \langle\{K_i\}, \{K_i \mapsto K_j\}_{i\neq j}\rangle$ be an $\mathcal{ALCP}_\mathcal{C}$ ontology. Let $K_i = (k_{iv}, K_{ih})$ be a package in K, Q_i the set of concept subsumption queries for K_i, $A = \{Y, N, U\}$ the answer space, and R_i a reasoner for K_i. Q_{iY}, Q_{iN}, Q_{iU} are defined as usual.*

For all i, We use Q_{iM} to denote the set of all axioms of the form $C^{i\rightarrow j}(x, t) \sqsubseteq \bot$, for each conversation $\boldsymbol{C}^{i\leftrightarrow j}(x, t)$ that is risky for i against j, for every j, x, t;

*R_i is **distributed weakly privacy-preserving** w.r.t. K_i if it satisfies the Honesty Axiom, the Weak Safety Axiom (see Definition 6.3) and the following axiom:*

- *Message Safety Axiom: for all $\alpha \in K_{ih}$, we have $K_{iv} \cup Q_{iY} \cup Q_{iM} \nvdash \alpha$.*

However, extending the distributed tableau-based algorithms for P-DL as we presented in the last chapter to obtain a distributed privacy-preserving reasoning algorithm is still remained as an open problem. We give in the follows some discussions that might be useful for the investigation of such an algorithm in the future.

Local reasoners should only have local knowledge abut messages. We first argue that a local reasoner should know only messages exchanged between itself and its local acquaintances, but not messages exchanged between other local reasoners, the existence of such messages, nor the structure of local completion graphs in other local reasoners. That is because each local reasoner is autonomously created and maintained, thus it is neither practical nor desirable to share information about messages globally. Hence, typically a local reasoner may not know whether another reasoner can detect if a conversation between them is risky since it requires global knowledge about messages. Thus, a practical algorithm may need to avoid *potentially* risky using only local knowledge about messages.

The reasoner should be conversation independent. Note that each local reasoner should be history independent, hence for a given query, it should always return the same answer regardless what conversations have been exchanged in particular reasoning processes. (Note that it is possible that reasoner sends out different messages giving the same set of "incoming" messages due to the non-deterministic nature of tableau expansions.) That will guarantee the uniqueness of query answering.

Not all risky conversations are "unsafe". While a risky conversation may expose knowledge of a package, it is not necessary to avoid all risky conversations. For example, if the subsumption revealed by a risky conversation is entailed by the visible part of the package, no hidden axiom will be compromised from it. Thus, a privacy-preserving algorithm only needs to detect and avoid risky conversations that are indeed "unsafe".

Conservative extensions in the modular setting may be useful. Conservative extension is proven useful in privacy-preserving reasoning in the single ontology setting. However, existing work on conservative extension (Lutz et al., 2007; Grau et al., 2007) do not support localized semantics which is foundation of all major modular ontology languages. Extending such a notion to the modular ontology setting may be helpful to design distributed privacy-preserving reasoners.

6.7 Related Work

Problems of trust and privacy on the web in general, and the semantic web in particular, are topics of significant current interest.

6.7.1 Policy Languages

Access control policies have been widely studied in databases (Jajodia and Wijesekera, 2001). Knowledge scope limitation was introduced into knowledge bases in Ontolingua (Fikes et al., 1997), which restricts access, during ontology integration, to symbols that are designated as private. Early efforts aimed at developing formal languages for access control include Extensible Access Control Markup Language (XACML) (Godik and Moses, 2002; Kolovski et al., 2007) and Security Assertions Markup Language (SAML) (Hallam-Baker and Maler, 2002). Recently, there is growing body of work on trust and privacy on the semantic web. In particular, there are several policy language proposals (Tonti et al., 2003; Bonatti et al., 2006; Kagal

et al., 2004), such KAoS (Uszok et al., 2003), Rei (Kagal et al., 2003) and Ponder (Damianou et al., 2001), that can control the access to resources or operations from unauthorized users.

Research on encryption of sensitive information focuses on preventing unauthorized access to such information using cryptographic protocols. W3C XML Encryption[6] working group has proposed an XML syntax for encrypting or decrypting digital content in XML documents. (Giereth, 2005) has studied the hiding of a fragment of an RDF document by encrypting it while the rest of the document remains publicly readable.

These access control policies and encryption techniques either allow or prohibit access to information in a complete fashion. They do not allow the use of private knowledge to answer queries that can be safely answered using private knowledge without revealing it. In contrast, this chapter explores how to relax the restriction on access to private or hidden knowledge so that they can still be used in answering queries with guaranteed safety.

6.7.2 Preventing Unwanted Inference

Our work is closely related to the work of Farkas and other (Farkas, 2006; Farkas et al., 2006; Jain and Farkas, 2006) on preventing unwanted inferences in data repositories. (Farkas, 2006) outlines several important aspects of preventing unwanted inference on the semantic web. Their *privacy information flow model* (Farkas et al., 2006) includes a privacy mediator that prevents agents in the system from being able to (indirectly) infer the private data of other agents. The inference algorithm assumes a tree-like data model, selection-projection queries, and a domain knowledge base (consisting of assertions in the form of Horn clauses) with closed world semantics. In contrast, the approach proposed in this paper assumes open world semantics which is more natural in the semantic web setting.

(Jain and Farkas, 2006) have proposed an RDF authorization model that can selectively control access to stored RDF triples, assign security classification to inferred RDF triples and check for unauthorized inferences. This model assigns a highest-level security label to each (stored or inferred) RDF triple using a pre-specified set of syntactic rules. In contrast, the focus of our work is on mechanisms that allow the use of hidden knowledge to answer queries without revealing hidden knowledge.

[6] http://www.w3.org/Encryption/2001/

6.7.3 Epistemic Semantics

Logics with specially designed semantics have also been applied in information hiding. (Cuppens, 1990; Glasgow et al., 1992) describe a logic that includes epistemic operators for reasoning about knowledge and deontic operators for reasoning about permission and obligation. (Gray and Syverson, 1998) adopt a hybrid approach by connecting the information-theoretic formulations of security and logical formulations of knowledge and probability in distributed systems. Such methods may be extended to description logics if we add the "K" (epistemic) operator, e.g. $\mathcal{ALCK}$ (Donini et al., 1998). In contrast, our approach is based on the standard first-order semantics, thus is easier to implement based on existing standard reasoners.

CHAPTER 7. Collaborative Building of Modular Ontologies

Modular ontologies provide an attractive framework for the necessary compromise between the need for knowledge sharing and the need for knowledge hiding in collaborative design and use of ontologies. Structured organization of ontology entities in modules may bring to ontology design and reuse, the same benefits as those provided by modules in software design and reuse in software engineering. In this chapter we will present principled ways and software tools for collaborative ontology building (COB) driven by the notion of modular ontologies.

Part of this chapter was previously published in (Bao and Honavar, 2004a, 2005a; Bao et al., 2006g).

7.1 General Desiderate of Collaborative Ontology Building

7.1.1 Motivations

The process of constructing a small-scale ontology is typically *non-collaborative*, which means that it involves only a single user (see Fig. 7.1). Such an ontology is usually stored in one or several files. When editing the ontology with an available tool (e.g., Protege[1], OBO-Edit[2]), the curator needs to make a local copy of this ontology. After editing, the initial ontology is completely replaced with the new version that is the result of the editing process.

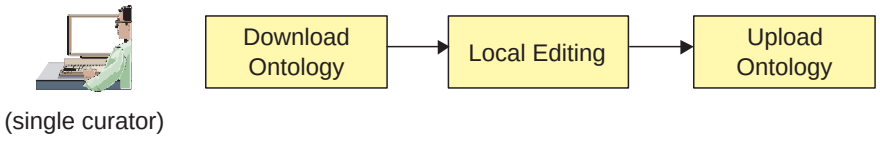


Figure 7.1 Non-collaborative Ontology Building

In contrast, the process of constructing large-scale ontologies is necessarily *collaborative*.

[1] http://protege.stanford.edu/
[2] http://oboedit.org/

Ontologies that are intended to be useful to specific communities often consist of thousands of terms. For example, the Gene Ontology contains 2×10^5 terms and the Gramineae Taxonomy contains 7×10^5 terms. Furthermore, such ontologies have to capture the collective knowledge and expertise of multiple experts and research groups. An unavoidable consequence of the increasing size and complexity of ontologies is the need for *collaboration* among multiple experts or research groups. Such collaboration can be either direct (as in collaborative creation of an ontology), or indirect (through the reuse of previously published, autonomously developed ontologies). In such a setting, a large ontology is built and curated by a community, with each of its members contributing only a small part of the ontology.

A typical large-scale ontology construction scenario is given in the following (Figure 7.2): The animal genomics community consists of several autonomous, geographically dispersed, research groups around the world. Animal Trait Ontology (ATO) (Hu et al., 2006) has been developed for a diverse set of species (e.g., for cross-species comparisons). No single research group possesses all of the expertise needed to construct the desired ATO. Consequently, it is necessary and natural for groups with different areas of expertise (e.g., species-specific expertise about horses, chicken, pigs, etc.) to work more or less independently to create ontology modules that can then be linked together as needed. Because multiple groups might hold different ontological commitments, terminological clashes or conceptual differences between the groups (and hence the ontology modules created by them) are simply unavoidable. Hence, there is a need for mechanisms for linking ontology modules so as to preserve the semantic locality while ensure a partial consensus on publicly shared knowledge. Furthermore, inherent inefficiencies (with regard to memory and processing time needs) in the use (e.g. editing, reasoning, communicating) of large ontologies can be minimized by taking advantage of the modular nature of the ontologies.

This argues for a modular approach to design and use of complex ontologies wherein ontologies such as ATO, instead of being treated as a monolithic entity, are organized into modules that reflect the organizational structure of knowledge in a domain of interest. For instance, it is natural to organize ATO (at a fairly high level), in terms of species-specific modules (e.g., those that focus on horse, cattle, chicken, etc).

Unfortunately, existing ontology editing tools such as the DAG-Edit (Day-Richter, 2004) and OBO-Edit (Mungall, 2005) offer, at best, limited support for such collaborative development.

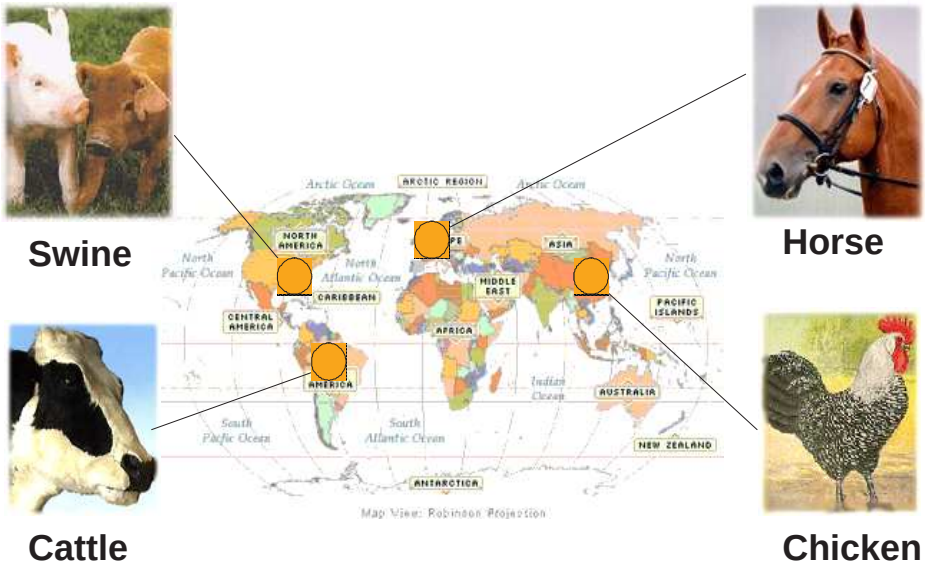


Figure 7.2 Collaborative construction of an Animal Trait Ontology

Consequently, there is an urgent need for tools for creating and processing increasingly large, collaboratively developed ontologies, with bounded time and space resources. To address this need, we have developed extensions to ontology languages and software tools for *collaborative ontology building* (COB). Our approach builds on recent advances in modular ontologies Bao et al. (2006f) to facilitate the creation of large ontologies within a distributed curator model.

7.1.2 Requirement of COB Environments

We draw on our experience with the construction of ATO to articulate some of the requirements of a COB environment. A COB environment needs to offer support for several tasks:

Knowledge Integration: The target ontology typically requires integration of ontology fragments contributed by multiple participants. For example, ATO needs to integrate contributions from individuals or research groups with expertise with regard to specific species.

Concurrent Management: Different curators need to be able to work on different parts of the ontology simultaneously. Suppose that curator A downloads the current version of ATO and performs local editing on "PigMeatQuality"; before A submits the modified version of ATO, another curator B may submit changes on "PigHealth". Therefore, COB environment

must ensure that when A submits his or her locally edited copy, the version updated by B is not inadvertently overwritten.

Consistency Maintenance: Components of the same ontology developed by different curators may be inconsistent since an ontology usually reflects the local point of view of each curator. For example, "PigLoinWeight" may be taken as a sub-trait of "PigMeatQuality" by one expert, but as a "PigProduction" trait by another expert. Inconsistencies can arise as a result of obsoleting ontology terms. For instance, while a user A is in the midst of defining a term "NumberOfTeatsOfSow" under "ExteriorTraits", another user B may be eliminating "ExteriorTraits" (e.g., by merging it with with "ReproductiveTraits") and submitting the change before user A does. Hence, ensuring the semantic consistency of the resulting ontology requires reconciliation of different points of view.

Privilege Management: In order to ensure the accuracy of the ontology, the COB curator community needs to include individuals with different levels of privileges, based on their expertise, authority, and responsibility. For instance, a curator A may be responsible for all pig traits (all terms under "SusScrofa"), while a curator B may be responsible only for pig health traits (all terms under "PigHealth").

History Maintenance: COB environments should have mechanisms to recover from wrong, unintended or even malicious changes to an ontology. Therefore, changes to an ontology must be recorded in order to be able to track the authorship of a change and to prevent loss of important information.

Scalability: Many ontologies consist of tens of thousands of terms. Consequently, the COB process has to be scalable to large ontologies. For example. if curator A only intends to edit the ontology component about pig meat quality, it is neither necessary, nor desirable to make A download and submit the parts of ATO that are not affected by his or her work.

7.2 CVS-based Collaboration and its Limitations

There is a growing awareness in the ontology community about the need for collaboration in the construction of large ontologies. Consequently, there is a growing interest in the development of COB environments and tools. The Gene Ontology consortium represents one of the most successful collaborative efforts aimed at creating large ontologies in the biological domain. The GO collaborative building process (Gene Ontology Consortium, 2005) is based on a concurrent

versions system (CVS), a request tracking system hosted on SourceForge, and natural language communications among GO users and curators (facilitated by several email lists). This process consists of the following major steps (Fig. 7.3):

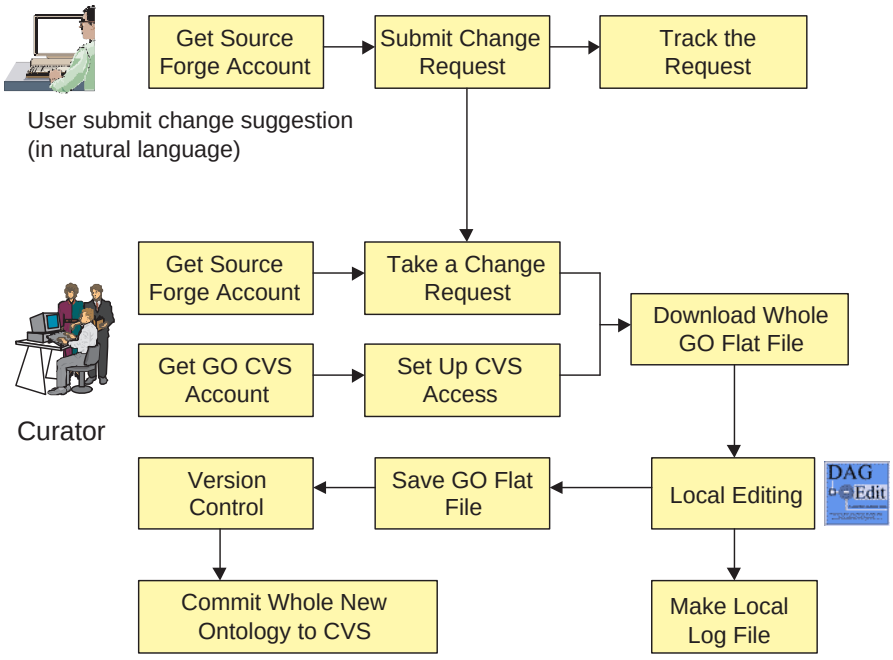


Figure 7.3 Collaborative Ontology Building with CVS: Gene Ontology

1. A user submits a change request on SourceForge.
2. One of the GO curators claims the request.
3. The curator downloads the GO flat file from CVS.
4. (optional) The curator declares "ownership" of the GO terms in a file (GO_numbers) in the CVS.
5. (optional) The curator sends emails to other curators to avoid conflicting.
6. The curator edits the flat file downloaded (e.g., with DAG-edit) and saves the modified file.
7. The curator compares the local flat file with the current version in the CVS repository and merges all changes made by other curators after his or her last download.

8. The curator uploads the modified ontology to CVS.

CVS and email list based collaboration approach, while useful, fails to satisfactorily address the key requirements of COB environments enumerated above because of the following drawbacks:

Unprincipled Authorization and Organization: There is no principled mechanism to ensure curator privilege assignments, nor clear organizational division of the whole ontology into smaller manageable units.

Risk of Inconsistency: There is no principled way to avoid unintended couplings and over-writing. The validity and consistency of the ontology are dependent almost entirely on the discipline exercised by the human curators (e.g., the habit of checking differences between versions before submission) and good community communications (e.g., via email lists). Hence it is not surprising that there are many inconsistencies in GO (Yeh et al., 2003).

Lack of Support for Editing or Reuse of Parts of the Ontology: A curator has to download the *entire* ontology, before editing, and submit the *entire* modified ontology, after editing, although a small part of the ontology may actually be affected by the changes. Similarly, a user cannot download and reuse only a selected subset of GO, e.g., the part of GO that is concerned with description of Kinases.

Expensive History Maintenance: In CVS version control, even a minor edit of the ontology, e.g., editing of a single term, relationship or property, causes the ontology file to be replicated in its entirety. In addition, tracing the changing history of a term requires processing the entire ontology text file for comparisons. As a consequence, retrieving the relevant information about editing history of a relevant ontology fragment is rather expensive.

Limited Participation: In the absence of a principled way to grant different levels of privileges to different types of users (e.g., core curators versus normal curators), and a handy tool to accept/deny/modify/revert *local* changes made by other curators, the curator community has to be limited to a small number of trusted curators. This limits the participation of the broader scientific community in the ontology building process and results in a bottleneck that slows down the rate at which ontology can change in response to community input.

Therefore, success of the CVS-based approach to COB relies heavily on the implicit community commitment and cooperation, and on the self-discipline of the involved participant as opposed to mechanisms that are designed specifically to support collaboration. Hence, there is

an urgent need for knowledge representation formalisms, as well as systems and and software for COB. In what follows, we describe an approach based on an organization of a complex ontology into ontology modules or packages as we described in previous chapters.

7.3 COB-Editor

Many of the drawbacks of current approaches to COB arise from lack of support for localizing the interactions among different parts of a large ontology. The primary source of this difficulty is the lack of an organizational structure which forces us to treat an ontology in its entirety. In other words, the current state of knowledge representation languages for ontologies is reminiscent of the early programming languages which lacked support for organizing programs into coherent units (e.g., subroutines).

7.3.1 Organizing Ontologies into Packages

We observe that an ontology that results from collaboration among multiple experts or research groups can be viewed as consisting of smaller modules, called **packages** (Bao et al., 2006f). Each such package encapsulates a closely related set of terms and relations between terms. Together, these terms and relations represent the ontological commitments of an individual expert or a research group (or a sub-community) regarding a small, coherent part of the universe of discourse (e.g., traits of interest to the livestock community).

The notion of package is closely related to Package-based Description Logics as we introduced in Chapter 4. For it to be useful in the COB setting, we extend the notion of package-based ontology to also support hierarchial organization structure and scope limitation. To make this chapter self-contained, we briefly introduce the notion of package:

Definition 7.1 (Package) *Let an ontology O be a set of axioms. A* package *P of O is a subset of axioms in O, $\mathsf{Sig}(P)$ is the signature of P; a subset of $\mathsf{Sig}(P)$, denoted by $\mathsf{Loc}(P)$, is called P's* local signature*; for every name t in $\mathsf{Loc}(P)$, P is called the* home package *of t, denoted by $P = \mathsf{Home}(t)$; $\mathsf{Ext}(P) = \mathsf{Sig}(P) - \mathsf{Loc}(P)$ is P's* external signature*. If $t \in \mathsf{Loc}(P_1) \cap \mathsf{Ext}(P_2)$, we say P_2 imports t from P_1.*

For example, in the ATO ontology, **Pig** is a package in ATO, "LoinEyeArea" and "MeatQuality" are names in **Pig**; subclassOf("LoinEyeArea", "MeatQuality") is an axiom in **Pig**; "LoinEyeArea"

is a member of **Pig** and **Pig** is the home package of "LoinEyeArea".

Note that the axioms in an ontology package specify its *semantic structure*. However, COB calls for recognition of the *organizational structure* of an ontology. The package-based ontology language offers language constructs that allow an ontology package to be declared as a *sub package* of another package. Such *organizational* relations among packages allow specification of the organizational structure of an ontology in terms of hierarchical nesting of packages.

Definition 7.2 (Package Nesting) *A package P_1 can be nested in one and only one other package P_2. This is denoted by $P_1 \in_N P_2$. P_1 is said to be a* sub package *of P_2 and P_2 is the* super package *of P_1. $\in_N^*$ is the transitive closure of $\in_N$. The collection of all package nesting relations in an ontology constitutes the* organizational hierarchy *of the ontology.*

For example, the package **Pig** can contain smaller packages, such as **PigMeatQuality** and **PigHealth**. **PigMeatQuality** (the sub package) is nested in Pig (the super package).

Package-extended ontologies also allow creators of packages to exert control over the visibility of each name within the package, thereby allowing selective sharing (or conversely, hiding) of ontological commitments captured by a package. It is realized using *scope limitation modifier* as we introduced in section 6.3.1.

We utilize three default SLMs in this study:

- $public(p, t) :=$ True, means name t is accessible everywhere.
- $protected(p, t) := (t \in p) \vee (p \in_N^* \mathsf{Home}(t))$, means name t is visible to its home package and all its descendant packages on the organizational hierarchy.
- $private(p, t) := (t \in p)$, means name t is visible only to its home package.

We require that if a package P imports a name t, t should be visible to P and cannot be in $\mathsf{Loc}(P)$.

Scope limitation helps in collaboration on specifying *term extensibility* and reducing unwanted coupling. Thus, a package may selectively allow or forbid names in its local signature from being imported, thus being changed in semantics, by other packages. For example, if the **PigMeatQuality** package contains a local name texture which is not intended to be further

used by other packages (e.g., adding new restrictions on texture), texture may be declared as a *private* name in **PigMeatQuality**. On the contrast, **PigMeatQuality** package may contain a name color, which is best further defined by other packages, color may be declared as a public name. Note that the released ontology may have a different scope limitation policy (or does not require scope limitation) from that in the design phase.

7.3.2 Benefits of Modular Organization for COB

Modular structure of ontologies can be exploited in COB in several ways:

Division of Labor: A package consists of a set of closely related terms and axioms in a smaller sub domain (e.g., pig traits) of a main domain. Therefore, a package can be assigned to curators or experts with the best knowledge of the relevant sub-domain. The package hierarchy helps organize and manage interactions among collaborating groups of sub-domain experts with different degrees and scopes of expertise.

Scalability: When the ontology has a modular structure, a curator is not required to work with the entire ontology. Instead, the curator can download and edit one or more packages that fall within his or her (sub) domain of expertise, while other packages are being edited by other curators. Because the effects of edits are *localized*, different curators can independently work on packages e.g., **PigMeatQuality** and **PigHealth**, that belong to different parts of the organizational hierarchy. This simplifies the task of propagating the effect of changes within individual packages through the rest of the ontology. The result is significant reduction in communication overhead, computational cost (e.g., parsing, consistency check), memory requirements, as well as the cost of history tracking in collaborative editing of very large ontologies.

Partial Reuse: Ontologies with modular structure can be partially reused. For example, a user interested in pig traits would only need to download packages about pig (e.g., **Pig** and **PigMeatQuality**), and avoid having to deal with ontology packages such as **Cattle** and **Horse** which are no interest to the user.

Broadened Participation: One of the reasons behind the success of the world-wide web is the network effect: a large number of users were able to contribute modules (web pages) that make up the world-wide web. The power of the network effect is confirmed by experience with collaborative projects like DMOZ and Wikipedia. Effective COB environments can enable broader participation in ontology building efforts without sacrificing the quality of the

resulting ontology. Specifically, a user with the appropriate privileges can make a change to the **PigMeatQuality** package (as opposed to just proposing such a change for consideration via an email list); this change could be approved or denied by a curator with higher privileges.

7.3.3 The COB-Editor

Based on the package-based ontology framework, we have developed COB Editor[3] (See Figure 7.4), a collaborative ontology editor prototype for building and deploying biological ontologies. The editor allows ontology developers to create a community-shared ontology server with remote database storage and support for concurrent browsing and editing of an ontology. Specifically, the current implementation of the COB Editor offers support for:

Support for Module: A large ontology is organized into packages with nested scope. This helps localize the effects of changes to an ontology in a setting where multiple experts collaborate in developing a large ontology.

Database Storage for Ontologies: The ontology is stored on a relational database server (e.g., Postgres); a user can connect to the server and check out one or more packages, edit them, and check them back in the database (when finished with editing). Database storage allows retrieval of only the relevant parts of an ontology (as opposed to the entire ontology).

Concurrent Editing: Multiple users can concurrently edit an ontology, through appropriate locking mechanisms, without a user inadvertently overwriting the work of others. Thus, different modules of an ontology can be developed by different authors. A user can edit an ontology by editing one or more packages. The packages being edited are locked for writing and unlocked when a the user submits the edited version(s) of the checked out package(s)

Change Tracking: When a user submits an updated version of an ontology module, the ontology server automatically creates the change log for affected terms and relations in the package. This feature can be used to track the history and if necessary, to revert a term or a relation to its previous version.

User Privilege Management: Authors of the ontology can have different levels of privileges (such as ontology administrator, and package administrator) over modules in an ontology.

[3]http://www.animalgenome.org/bioinfo/projects/ATO/. COB-Editor was developed from July 2005 to August 2006, in collaboration with Zhiliang Hu (requirement specification and database maintenance), LaRon Hughes (testing) and Peter Wong (testing and documentation). It is evolved from the ATO Editor (http://boole.cs.iastate.edu/indus/ato/ato.html), developed by a ISU group including Jie Bao, Swetha Gottimukkula, LaRon Hughes, Jialin Le and Jia Tao.

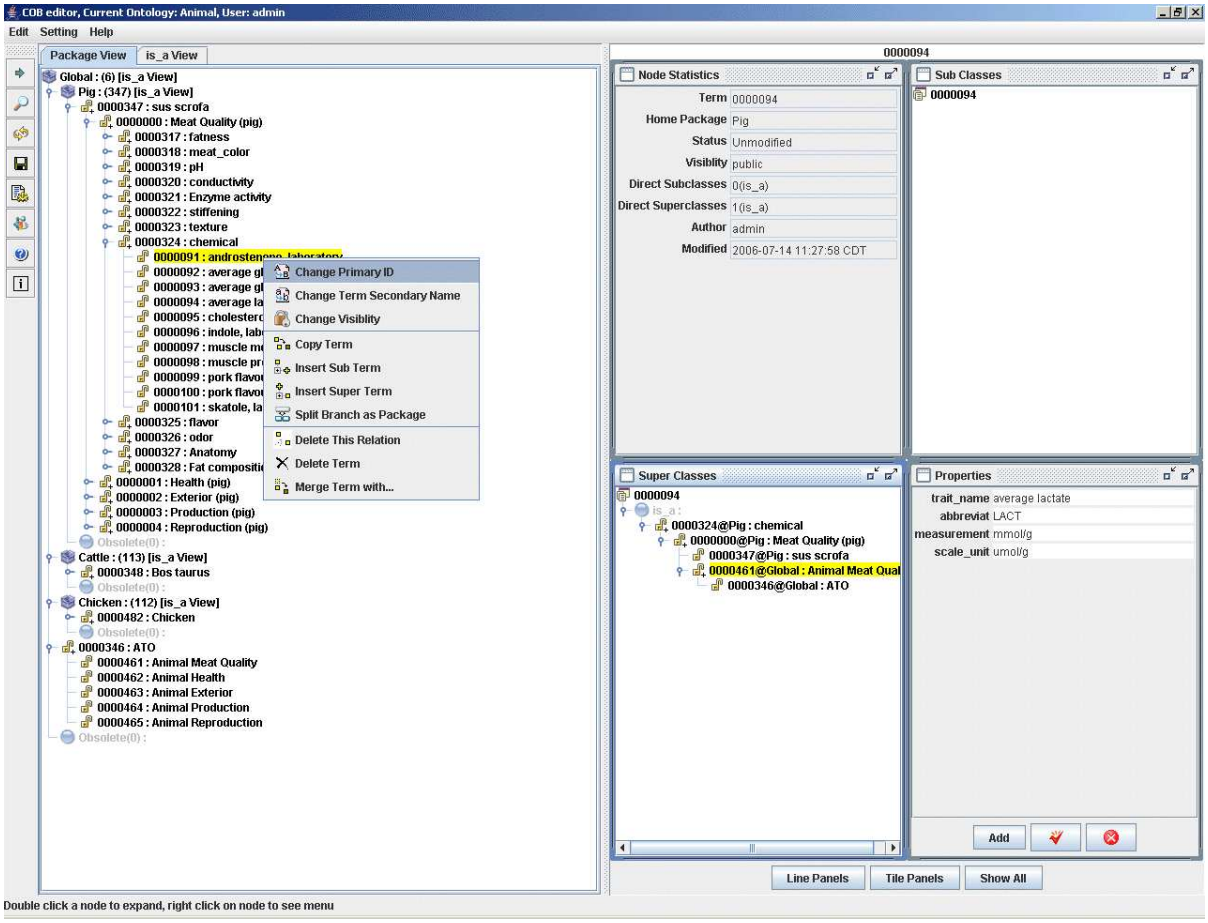


Figure 7.4 COB Editor

The author of a package can authorize other users the access to certain terms, therefore controlling the extensibility of that package.

Navigation: Users can browse and edit DAG-structured ontologies using a GUI. Contextual information, such as superclasses and subclasses are provide for fast identification of related term. The editors allows terms in an ontology being arranged and viewed in different hierarchial trees, e.g., both is-a and part-of trees. Different packages may be browsed using different settings.

User Communication The editor has a built-in instant messenger for the user community to coordinate editing actions.

Import and Export of Ontologies in Multiple Formats: Users can import and export ontologies in several standard formats, such as OBO (Open Biomedical Ontologies) and OWL into and out of the COB Editor.

The editing process in COB Editor is summarized in Fig. 7.5.

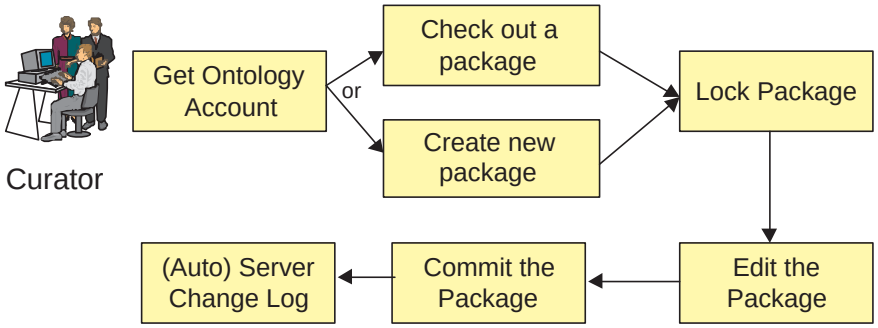


Figure 7.5 Collaborative Ontology Building with Package-extended Ontology

The editor is implemented in Java to ensure its universal portability across different hardware and operating system platforms.

7.4 WikiOnt: Wiki-based Modular Ontology Editor

7.4.1 Overview

WikiOnt is another collaborative ontology editor prototype driven by the modular ontology notion. WikiOnt is among the first efforts that are aimed to bring together collaboration technique from the social web (or "web 2.0") and the efforts on providing formal "meaning" of resources on the semantic web. It is designed as a light-weight, easily-accessible (e.g., from hand-held devices) ontology building tool with a wiki-like editing environment.

Wiki is typically used in collaborative documentation writing and website building. A typical wiki system (e.g. wikipedia.com) includes

- a script language (usually a simplified subset of HTML tags),
- a set of wiki pages written in the script language and shown in translated HTML pages,
- a RCS version control system to record modification of contents,
- a user profile and concurrent conflict management system to enable multiple users editing the same contents,
- a content navigation system such as showing link-in and link-out pages,

- and a simple-to-use, browser-based editing environment to generate or modify content on the fly.

Many of those features are also desirable in collaborative building of modular ontologies. WikiOnt turns a wiki system into an ontology building tools by using wiki script as the syntax of an ontology language and using wiki pages as the storage of ontology modules. By borrowing many of the mature features of wiki systems, WikiOnt is ideal for fast and collaborative development of OWL and P-DL based ontologies.

WikiOnt shares many features with COB-Editor, e.g., concurrent editing, partial editing of an ontology, change tracking and the support for module. We present in what follows, some unique features of WikiOnt.

7.4.2 Features

Figure 7.6 shows the architecture of WikiOnt. A user or software agent may interact with wiki storage, which can be implemented either in file-based system or in relation database. An in-memory model of the ontology may be created when it is necessary (e.g., for consistency checking) using the Jena[4] RDF parser and repository. WikiOnt supports the import and export between an existing OWL ontology and the constructed ontology in the wiki storage.

Wiki Engine. The core of the system is a wiki engine which is implemented in Java and JSP extending the JSPWiki engine[5]. The wiki engine:

- provides a web interface for users;
- translates the ontology markup script to HTML pages presented in the web browser;
- manages the storage of wiki pages, in plain file or in database (depending on user choice);
- provides version control: when a modification for an axiom is submitted, the previous version is stored and could be restored when the committed version is found incorrect or impropriate;
- provides transaction management to prevent editing conflicts (will be explained later);

[4]http://jena.sourceforge.net/
[5]http://www.jspwiki.org

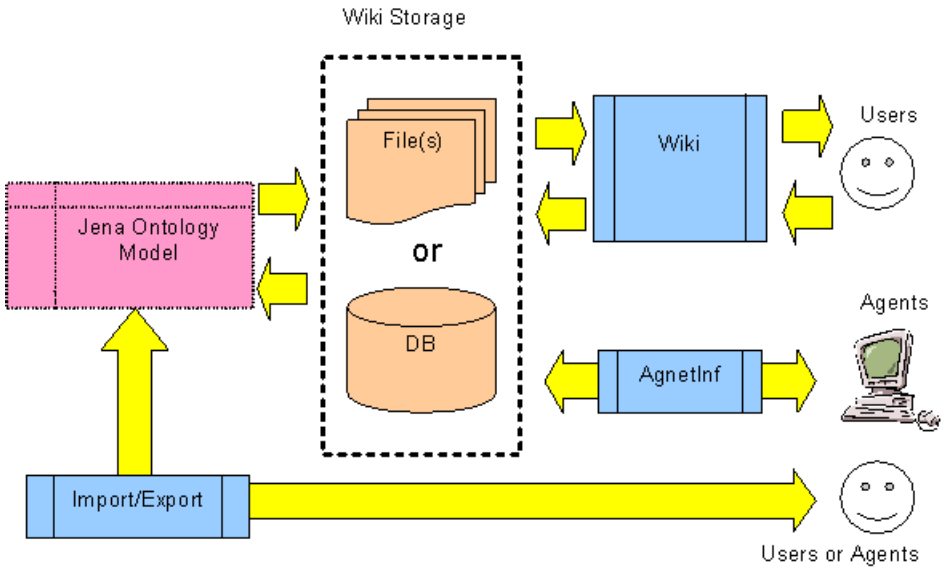


Figure 7.6 The Architecture of WikiOnt

- generates reference report for wiki pages: terms being used in a module, and other modules that referring the module in question, are listed for navigation purpose.

- generates a RSS feed for ontology repository updates.

User Management. Each participant in WikiOnt is considered as an agent. Agent is assigned with different privileges, such as ontology administrator and package manager. Agent could join the editing of any existing ontology module or create new modules. It is possible that an agent is a software agent which may collect knowledge from other sources (e.g., other published ontologies).

WikiOnt Script. An ontology is organized in multiple wiki pages, each is described using a markup script that corresponds to the a simplified syntax of the OWL language. When a wiki page is under editing, its wiki markup script is loaded and translated into user friendly representation, such as a HTML web page. The WikiOnt markup script is a human readable syntax equivalent to the N-Triple syntax of OWL, where each axiom in the ontology is represented by a set of triples (an example is shown in Figure 7.7). Closely related axioms are organized in the same wiki page that corresponds to a module in the ontology. A user can create a new page or modify the source script of an existing page. The editing action may be assisted by several wizards (such as class creating wizard) and a browser (eg. showing the subclasses and

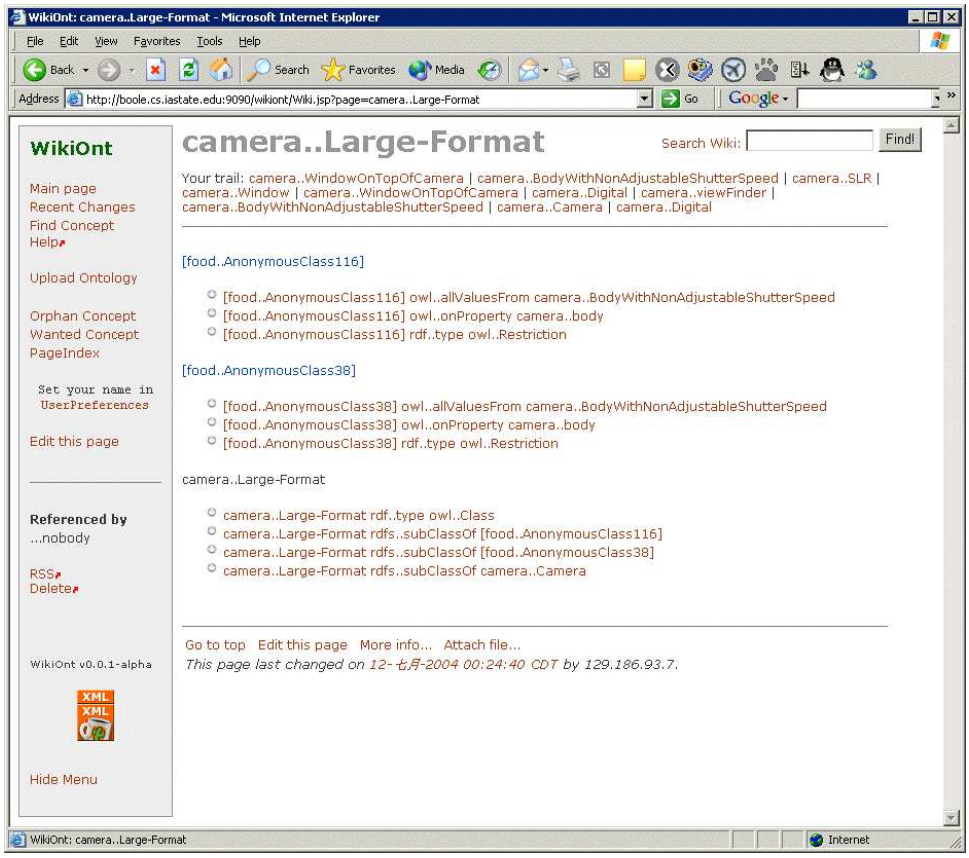


Figure 7.7 A Wiki Page in the WikiOnt System

superclasses of the concept in question).

Partial In-memory Model. While many popular ontology editors have an in-memory model for the whole ontology in editing, WikiOnt does not maintain in-memory model for each resident ontology for several reasons.

- An in-memory model limits the scalability of the system with respect to both the size of an ontology and the number of ontologies in the ontology repository.
- In-memory model implicitly assumes the existence of a global ontology during the ontology development process, which is not practical in general in a collaborative ontology building scenario.
- Even when the size of the ontology in question is huge, usually only a small fraction of its axioms are involved during an editing action.

Hence, we store the ontology as a set of separate, possibly distributed blocks in WikiOnt.

Each block is serialized to external storage when it's not being actively edited, and being loaded into the memory only if it's edited or referred. A partial ontology model will be dynamically loaded into memory only if it is needed. This is inspired by widely used techniques of database memory management where partial content of the database is dynamically loaded and unloaded to allow manipulation of of a much larger volume of data than can fit in limited memory.

Ontology exporting/importing. When a serialized ontology is needed, e.g., for reasoning, we export selected set of wiki pages as a single ontology file. User may also upload an ontology file in OWL format into Wiki ontology repository. We use the Jena toolkit to create the in-memory model and as parser/writer for ontology files.

Transaction Management. Transaction management is to ensure consistency of module and protect critical resource from multiple access. If a page (i.e., a module) is under the editing of an agent, WikiOnt will deny the write-access of other agents to the module and other related modules.

Current implementation of WikiOnt is limited in several ways. There is no graphical user interface that may assist users to visualize and edit an ontologies. The support for modularization is limited, e.g., no support for package hierarchy or scope limitation. A new version of WikiOnt is currently under development that will address those problems.

7.5 Related Work

With the growing need for ontologies, there are growing efforts on developing ontology editors (Gomez-Perez et al., 2002; Denny, 2002, 2004). While most widely-used ontology editors, such as Protege (before version 3.1 of 2006), SWOOP (Kalyanpur et al., 2005) and DAG-Edit (Day-Richter, 2004), work very well for developing a single ontology module, they do not lend themselves to collaborative ontology building. This is due to the lack of a built-in formalism to support modular ontology representation, and the lack of support for communication and cooperation among multiple individuals in editing a shared ontology consisting of multiple, independently developed modules.

Support for collaborative ontology development has begun to receive some attention, for example in systems such as CODE (Hayes et al., 2003), OntoEdit [6], Ontolingua (Farquhar et al., 1995), and WebODE (Vega et al., 2001). Most of them provide concurrent access con-

[6] http://www.ontoknowledge.org/tools/ontoedit.shtml

trol with transaction oriented locking, and in some cases, even rollback. However, to the best of our knowledge, none of the existing ontology editors offer support for a modular ontology representation - a feature that is critical for collaborative development of large ontologies by communities of users. Consequently, parsing, editing, consistency checking, and change tracking processes do not scale to large ontologies. In contrast, the COB tools presented in this chapter complement existing ontology development tools in areas where there is a need for active collaboration among experts in developing ontologies and the complete ontology can be naturally viewed as consisting of many loosely coupled modules.

Recent versions of Protege offers a multi-user mode[7] (released in April 2006), which allows multiple users to edit the same ontology concurrently. The ontology in question is resided on a ontology server, all the changes made by one user are seen immediately by other users. Protege 3.3 (released in June 2007) further supports several collaboration features including change annotation, discussion thread, proposal, voting and chat (Tudorache and Noy, 2007). However, current Protege implement does not support modular presentation of an ontology, hence there is no principled way to control conflicts in editing or the control of the propagation of inadvertent editing mistakes. Since an in-memory model of the whole ontology is required for each client, scalability of the tool may be a concern when the ontology in question is large. There also lacks automatic version control and reverting mechanism as provided by COB-Editor and WikiOnt.

TopBraid Composer (from version 1.2.0, August 2006) supports a multi-user mode by allowing all users to work on a shared Sesame[8] repository. An ontology may import other ontologies, which may be selectively loaded into local memory, thus it provides (limited) support for organizational separation of an ontology. User may also selectively lock a local file to make it read-only. Thus, TopBraid Composer might be used as a collaborative ontology building tool. Compared with TopBraid Composer, COB-Editor provides stronger support on hierarchial management of ontology modules and multi-level privilege management of users.

Using Wikis as collaboration tools on the semantic web has received considerable attentions in the past two years (Krorzsch et al., 2005; Oren et al., 2006; Muljadi et al., 2006; Fischer et al., 2006; Kawamoto et al., 2006; Schaffert, 2006; Buffa and Gandon, 2006; Auer et al., 2006; Backhaus et al., 2007), including several devoted workshops (Völkel et al., 2006b,a). However,

[7] http://protege.cim3.net/cgi-bin/wiki.pl?MultiUserTutorial
[8] http://www.openrdf.org/

to the best of our knowledge, no existing system supports the construction of modular ontologies as supported by COB-Editor and WikiOnt.

CHAPTER 8. Conclusion and Discussion

8.1 Contributions and Impacts

Ontologies with clearly defined modularity and selective knowledge hiding are needed in a wide range of applications. In this research, we propose a package-based description logics (P-DL) approach as a modular ontology language. Compared with existing proposals, the P-DL approach is distinguished by stronger expressivity and the avoidance of many known semantic problems in existing approaches. In particular, this work makes original contributions to the field of knowledge representation (KR) and semantic web on the following problems.

The formal investigation of semantics and expressivity requirements of modular ontologies and detailed comparison of existing approaches (Bao et al., 2006b,c; Bao and Honavar, 2006b; Wang et al., 2007) (Chapter 3 and Section 4.4). We investigated:

- modular ontology desiderata in terms of syntactical modularity, semantic modularity, managerial benefits and scalability, thus clarify many controversies regarding to the notion of "modularity" in ontologies;
- the comparison of several semantics of modular ontologies, including local model semantics, contextual semantics, epistemic semantics and first-order semantics based on conservative extensions; we show that local model semantics is capable of encoding the contextual semantics and episematic semantics;
- the specification of the Abstract Modular Ontology (AMO) framework, extended from distributed first-order logic (DFOL) (Ghidini and Serafini, 1998) based on the local model semantics, as the basis for comparing different modular ontology languages.
- the formal description of minimal semantic requirements (e.g., directionality and transitive reusability) and expressivity of modular ontologies using AMO.

- the comparison of Distributed Description Logics (DDL), $\mathcal{E}$-Connections and P-DL; we prove that DDL with bridge rules between concepts and one-way binary E-connections where each module is in $\mathcal{SHIQ}$ are equivalent, while it was believed that the latter is strictly more expressive than the former; we show that both DDL and $\mathcal{E}$-connections do not in general support module transitive reusability, which is supported by P-DL; we also show that DDL with homogenous bridge rules and one-way binary $\mathcal{E}$-Connections $C^{\mathcal{E}}_{\mathcal{IHQ}}(\mathcal{SHOIQ})$ can be reduced to P-DL $\mathcal{SHOIQP}$.

The syntax and semantics of Package-based Description Logics (P-DL) (Bao and Honavar, 2004b; Bao et al., 2006f,d; Bao and Honavar, 2006a; Bao et al., 2007e) (Chapter 4). We investigated:

- extending description logics with contextualized negations and package divisions to support the desiderata of modular ontologies; we present the syntax and semantics of P-DL $\mathcal{SHOIQP}$, which exhibits stronger expressivity and avoids many semantic problems of existing approaches; in particular, we show that P-DL is well-suited for establishing general inter-ontology concept and role subsumption relations which are only restrictively supported by existing approaches;

- the integration of P-DL into OWL and semantics web tools; we show that it is possible to adopt OWL as the syntax for P-DL (with minor restrictions) without introducing new language constructs, hence ensuring the maximal backward compatibility to existing tools.

Distributed reasoning algorithms for modular ontologies based on P-DL (Bao et al., 2006a,e, 2007b) (Chapter 5). We investigated:

- distributed tableau algorithms for several languages in the P-DL family, including $\mathcal{ALCP}^{-}_{\mathcal{C}}$, $\mathcal{ALCPC}$, and $\mathcal{SHIQP}$; the proposed algorithms is capable of preforming reasoning tasks in peer-to-peer fashion without the integrating of all ontology modules; we show that the proposed reasoning algorithms are significantly less complex than the corresponding classical reasoning algorithms over integrated ontologies;

- asynchronous reasoning algorithms for P-DLs such that local reasoners may concurrently working on different reasoning sub-task to improve performance.

Reasoning with hidden knowledge (Bao and Honavar, 2006c; Bao et al., 2007d) (Chapter 6). We investigated:

- the precise formulation of the problem of *privacy-preserving reasoning* on the semantic web, which may offer the necessary reasoning support on the top of syntactical access control provided by semantic web policy languages;
- the general framework for privacy-preserving reasoning exploits the indistinguishability of hidden knowledge from incomplete knowledge under the open world assumption;
- a general approach to design weakly privacy-preserving reasoners based on the notion of conservative extensions; in particular, we provide privacy-preserving reasoning strategies for description logics based on the notion of local ontologies from the modular ontology study (Grau et al., 2007);
- privacy-preserving reasoning strategies for the important special case of hierarchical ontologies via a reduction to graph reachability analysis;
- preliminary investigation on the notion of hidden knowledge and privacy-preserving reasoning in P-DL.

The development of collaborative ontology building tools based on the notion of modular ontologies (Bao and Honavar, 2004a; Bao et al., 2006g). (Chapter 7). We investigated:

- the requirement of collaborative ontology building (COB) environment and the limitation of existing approaches, including CVS-based collaboration and the shared server based approaches;
- the benefits of using modular ontologies COB in tools; motivated from software engineering experiences, we extend the package-based ontology notion with package hierarchies and scope limitation modifiers to support collaborative management of large ontologies;
- the development of research prototypes as the proof-of-concept tools for COB using the modular ontology notion, including COB-Editor (targeted for editing biological ontologies) and WikiOnt (browser-based, general purpose ontology editor).

Impacts: the work presented in this dissertation may have potential impacts in several fields in KR and semantic web, including modular ontologies, the combination of logic formalisms, partial ontology reuse, the specification of next generation web ontology language, distributed reasoning, and collaborative ontology building. Part of the work has been presented or will be presented in major international conferences including (among others):

- Twenty-Second Conference on Artificial Intelligence (AAAI-07), July 2007, Vancouver, Canada;
- The First International Conference on Web Reasoning and Rule Systems, June 2007, Innsbruck, Austria;
- IEEE/WIC/ACM International Conference on Web Intelligence (WI'06), Dec. 2006, Hong Kong SAR, China;
- International Semantic Web Conference (ISWC 2006). Nov. 2006, Athens, GA, USA
- Asian Semantic Web Conference (ASWC 2006), Sept., 2006, Beijing, China (Best Paper Award)

The work presented in the dissertation is in active community involvement. In particular, we organized or made significant contribution to several related workshops including:

- Second International Workshop on Modular Ontologies (WoMO 2007), Fourth International Conference on Knowledge Capture, Whistler, British Columbia, Canada, October 2007.
- IJCAI 2007 Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa 2007), International Joint Conference on Artificial Intelligence, Hyderabad, India, January 2007.
- First International Workshop on Modular Ontologies (WoMO 2006), International Semantic Web Conference, Athens, Georgia, USA, November 2006.
- AAAI 2006 Fall Symposium on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa 2006), Arlington, VA, USA, October 2006.

On the ground of preliminary results obtained from this study, research on this topic is currently supported by US National Science Foundation grant (IIS-0639230) SGER: Exploratory Investigation of Modular Ontologies (Sept. 2006-Feb. 2008).

The work presented in the dissertation has caused interests from industrial organizations and has been presented in the following invited talks:

- Collaborative Development, Selective Sharing and Reuse of Ontologies. SemGrail 2007 Workshop, organized by the Microsoft Corporation, June 21-22, 2007 Redmond, WA, USA;
- On Selective Sharing and Reuse of Ontologies, Semantic Technology Conference, May 18-22, 2007, San Jose, CA, USA.

8.2 Limitations and Future Work

The work presented in this dissertation may be extended in several promising directions.

8.2.1 Modular Ontology Study in General

Further investigation of semantics of modular ontologies. In chapter 3 we compared several semantics of modular ontologies. We will further compare expressivity and semantic correctness of those semantics. In particular, we believe that in general modular ontologies may be viewed as the hybrid of description logics (for each module) and rules (for semantic relations between modules). Lending recent progress in the study of combining DL and rules (e.g., logic programming), such an approach may provide us a more fundamental understanding of some properties of modular ontologies, e.g., decidability and semantic locality. Such a study may also deepen our understanding on the relation between combining many-sorted theories (Baader and Ghilardi, 2005) and modular ontologies.

Further comparative analysis of modular ontology languages. The comparison on modular ontology languages in Chapter 4 is limited to DDL with bridge rules between concepts, one-way binary $\mathcal{E}$-connections and P-DL $\mathcal{SHOIQP}$. We will further extend the comparison in more general setting, e.g., including DDL with heterogenous bridge rules; in particular, we will study the support of different formalisms on the reuse of nominals.

A modular syntax and semantics for RDF. Existing study on modular ontologies is focused on logical formalisms extending OWL and its corresponding DLs. In the semantic web language stack, OWL is on the top of RDF, which may also be used in applications that require strong modularity support. We will investigate the possibility of extending RDF as a modular ontology language.

Combining Conservative Extensions and Modular Ontology Languages. Conservative extensions offers design patterns in ontology language to ensure semantic modularity under the classic first-order semantics. On the other hand, specially designed modular ontology languages (e.g., P-DL) realize modularity using the local model semantics. Combining the two approaches may offer principled ways to guide the design of ontology modules, as well as provide the necessary, yet still unexplored, distribute reasoning support for modular ontologies based on conservative extensions.

8.2.2 Extending P-DL

Extending P-DL to support ABox modularity. Current work on P-DL is focused on TBox modularity only. However, many applications (e.g., distributed data retrieval on the semantic web) requires connecting multiple data-intensive sources which are best described using ABoxes. Extending P-DL to support modularity in both TBox and ABox would be useful for such scenarios.

Querying P-DL. RDF and OWL are supported by the query language SPARQL. Providing a similar query language is necessary for the success of modular ontology languages. We will investigate the possibility of extending SPARQL as a query language for web ontologies based on P-DLs.

Using Named Graph as the syntax for P-DL[1]. Instead of re-interpreting `owl:imports` and adopt OWL as a syntax for P-DL, an alternative approach is to use Names Graph as the syntax for P-DL, where each ontology file has clearly defined organization structure to embed multiple ontology modules. Such an approach may provide stronger support for partial ontology reuse and human readability of ontologies.

Formal semantic analysis on package hierarchies and scope limitations in P-DL. We extended P-DL with package hierarchies and scope limitations in Chapter 7. However, there

[1]We thank Jing Mei for discussions on this problems.

is still missing the formal semantic specification of those extension of P-DL and their impacts on semantic properties of P-DL.

Reusing P-DL Ontology Through Interfaces and Views. In (Bao et al., 2006f), P-DL ontologies may be partially reused through interfaces (for a single module) and views (for integrating contents from multiple modules), which are subset of the signatures of the reused modules. We will formally investigate such an extension to P-DL for better support of semantic partial reuse. In particular, we will investigate practical methods to identify axioms in reused modules that are relevant to the interface or view in question. Such a study may also be useful for the modularization of a large ontology.

8.2.3 Reasoning Algorithms for Modular Ontologies

Implementation of the $\mathcal{SHIQP}$ tableau algorithm. Current study on P-DL is theory-oriented. Providing a scalable and stable reasoner for P-DL is critical for its success. Since the $\mathcal{SHIQP}$ tableau algorithm as we described in Chapter 4 is a natural extension of the $\mathcal{SHIQ}$ reasoning algorithm, we may extend existing DL reasoners, e.g. Pellet (Sirin et al., 2007), to support reasoning with $\mathcal{SHIQP}$ (hence also for $\mathcal{ALCP}_{\mathcal{C}}^{-}$ and $\mathcal{ALCP}_{\mathcal{C}}$)[2].

Design federated reasoning algorithms for $\mathcal{SHOIQP}$ and $\mathcal{SROIQP}$. We believe the federated reasoning approach we applied for $\mathcal{SHIQP}$ can be extend to more expressive P-DL languages, such as $\mathcal{SHOIQP}$ and $\mathcal{SROIQP}$, on the basis of tableau algorithms for classic DLs $\mathcal{SHOIQ}$ (Horrocks and Sattler, 2005) and $\mathcal{SROIQ}$ (Horrocks et al., 2006), thus allowing each ontology module to be in OWL-DL and OWL 1.1.

Optimizing distributed reasoning with P-DL. The reasoning algorithms we presented in Chapter 5 are subject to optimization in several ways. Promising approaches include caching query results, pre-computing of message query and answering lookup table, and borrowing optimization techniques used in ExpTime tableau algorithms for DLs (Donini and Massacci, 2000; Ding and Haarslev, 2007).

Improving scalability of DL reasoners utilizing modularity in ontologies. The result obtained in reasoning with modular ontologies may also be applied in improving scalability of a DL reasoner. For example, if the ontology in question has good semantic modularity w.r.t. the query in question, the reasoning task may be preformed by using only knowledge in related

[2]Future progress will be released at https://sourceforge.net/projects/phoenix-project

parts of the ontology as needed instead of considering the whole ontology.

Reasoning with light-weight modular ontologies. Light-weight ontologies that are widely used in biological domains and social web (e.g., FOAF[3] and folksonomy) can be modeled by highly efficient subsets of OWL-DL, e.g., concept hierarchies or $\mathcal{EL}$ (Baader et al., 2005). Exploring modularity and efficient distributed reasoning algorithms for those ontologies will be extremely useful for tools on the semantic web, e.g., semantic search engine, to tackle very large sets of ontologies that might contain billions of terms. We will investigate the correspond subsets of P-DL to meet those needs.

Debugging modular ontologies. Debugging is a non-standard reasoning process of detecting and identifying possible reasons of problems (e.g., unsatisfiable concepts and ontology inconsistency) in an ontology (Kalyanpur et al., 2006). Reconciling inconsistencies in modular ontologies has been investigated only in several limited cases, such as generating consistent subset of an inconsistent modular ontology (Bao and Honavar, 2005b) and the debugging of DDL bridge rules (Meilicke et al., 2007). We will extend the work mentioned above to address the debugging of modular ontologies in the general setting. Results from this work will be useful to resolve inconsistencies between ontology modules in the course of collaborative ontology building.

8.2.4 Privacy-Preserving Reasoning in Modular Ontologies

Privacy-Preserving Reasoning with P-DL. In chapter 6 we discussed issues to be addressed in designing privacy-preserving reasoners for P-DL. We will further explore a practical distributed algorithm that is capable of doing reasoning with partially hidden knowledge in modular ontologies.

Design secure reasoners for RDF graphs. We will explore the possibility of extending the graph theory based analysis for privacy-preserving reasoning with hierarchical ontologies to ontologies that can be modeled using more complex graphs, e.g., RDF graphs. One promising approach is to extend the study of reachability in graphs with a single type of edges to reachability in graphs with labeled edges.

Privacy protection in medical ontologies. Medical information systems may require sharing of patient data among doctors, healthcare providers and health insurance companies,

[3] http://xmlns.com/foaf/spec/

in the form of ontologies, e.g., for describing metadata about medical videos (Bao et al., 2004). Our work on selective knowledge sharing allows us to address the necessity to protect sensitive information in medical knowledge bases from unauthorized access or inference. We will investigate algorithms and application tools for the secure knowledge discovery from medical and health knowledge bases.

8.2.5 Applications of Modular Ontologies

Collaborative Ontology Building. We will extend COB-Editor and WikiOnt for better usability. Currently, WikiOnt2, which will support a browser-based, interactive graphical user interface and a new wiki engine for editing modular ontologies, is under development[4]. Both COB-Editor and WikiOnt may be connected to a reasoner for modular ontologies for consistency checking between modules. A collaborative ontology building plugin for Protege driven by the modular ontology notion may also be developed in the future.

Semantic Data Integration. The massive size, the relative autonomy and the distributed nature of data sources on the semantic web requires tractable approaches to integrate information from multiple, semantically heterogeneous data sources. We have previously investigated Ontology-Extended Data Sources (OEDS) that are associated with explicit ontologies of schema and content of data sources (Caragea et al., 2005a,b) and query translation under such a setting (Bao et al., 2007a). Those work can be extended using results from modular ontologies in several ways: 1) modular ontologies may provide the right framework to capture the contextual and distributed nature of OEDS; 2) semantics-preserving query translation in OEDS may be studied as a special case of reasoning in modular ontologies; 3) translating retrieved data into the ontology that is understandable by the user may be studied as a special case of instance retrieval in modular ontologies.

[4]Future progress will be available at http://sourceforge.net/projects/wikiont/.

Appendix: Proof of Lemmas and Theorems

A.1 Proofs for Chapter 4

Some useful properties of image domain relations are listed in the below:

Lemma A.1 *Suppose C, D are concepts. A model $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{i \neq j} \rangle$ of a $\mathcal{SHOIQP}$ KB has the following properties:*

1. *If $P_i \in P_j^+$ and $P_j \in P_i^+$, then $r_{ij} = r_{ji}^-$.*
2. $r_{ij}(C^{\mathcal{I}_i} \cap D^{\mathcal{I}_i}) = r_{ij}(C^{\mathcal{I}_i}) \cap r_{ij}(D^{\mathcal{I}_i})$
3. $r_{ij}(C^{\mathcal{I}_i} \cup D^{\mathcal{I}_i}) = r_{ij}(C^{\mathcal{I}_i}) \cup r_{ij}(D^{\mathcal{I}_i})$
4. $r_{ij}(C^{\mathcal{I}_i}) \cap r_{ij}((\neg_i C)^{\mathcal{I}_i}) = \emptyset$
5. $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i} \rightarrow r_{ij}(C^{\mathcal{I}_i}) \subseteq r_{ij}(D^{\mathcal{I}_i})$
6. $r_{ij}(C^{\mathcal{I}_i} \setminus D^{\mathcal{I}_i}) = C^{\mathcal{I}_j} \setminus D^{\mathcal{I}_j}$

Proof: 1): We must have $r_{ij} \circ r_{ji} = r_{ii}$. If $(x, y) \in r_{ij}$ and $(y, z) \in r_{ji}$, we must have $x = z$, i.e. $(x, y) \in r_{ji}^-$. The other direction is similar.

2) and 3): They are true since domain relations are one-to-one.

4): A corollary of 2).

5): $x \in r_{ij}(C^{\mathcal{I}_i}) \rightarrow r_{ij}^-(x) \in C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i} \rightarrow x \in r_{ij}(D^{\mathcal{I}_i})$.

6): From $C^{\mathcal{I}_i} \setminus D^{\mathcal{I}_i} = C^{\mathcal{I}_i} \cap (\neg_i D)^{\mathcal{I}_i}$ and 2)

Proof of Lemma 4.1

- For Equation 1, we have

$$\begin{aligned}
(\top_i \sqcap \neg_j C)^{\mathcal{I}_j} &= \top_i^{\mathcal{I}_j} \cap (\neg_j C)^{\mathcal{I}_j} && \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
&= r_{ij}(\Delta^{\mathcal{I}_i}) \cap (\Delta^{\mathcal{I}_j} \backslash C^{\mathcal{I}_j}) && \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
&= r_{ij}(\Delta^{\mathcal{I}_i}) \backslash C^{\mathcal{I}_j} && \text{(since } r_{ij}(\Delta^{\mathcal{I}_i}) \subseteq \Delta^{\mathcal{I}_j}) \\
&= (\neg_i C)^{\mathcal{I}_j}. && \text{(by the definition of } (\neg_i C)^{\mathcal{I}_j})
\end{aligned}$$

- For Equation 2,

$$
\begin{aligned}
& (\neg_i(C \sqcap D))^{\mathcal{I}_j} \\
= \;& r_{ij}(\Delta^{\mathcal{I}_i}) \backslash (C \sqcap D)^{\mathcal{I}_j} && \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
= \;& r_{ij}(\Delta^{\mathcal{I}_i}) \backslash (C^{\mathcal{I}_j} \cap D^{\mathcal{I}_j}) && \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
= \;& (r_{ij}(\Delta^{\mathcal{I}_i}) \backslash C^{\mathcal{I}_j}) \cup (r_{ij}(\Delta^{\mathcal{I}_i}) \backslash D^{\mathcal{I}_j}) && \text{(set-theoretically)} \\
= \;& (\neg_i C)^{\mathcal{I}_j} \cup (\neg_i D)^{\mathcal{I}_j} && \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
= \;& (\neg_i C \sqcup \neg_i D)^{\mathcal{I}_j}. && \text{(by the definition of } \cdot^{\mathcal{I}_j})
\end{aligned}
$$

- The proof of Equation 3 is dual to that of Equation 2.

- For Equation 4, we first note that $P_k \xrightarrow{R} P_i$ and

$$
\begin{aligned}
\forall R.\neg_j C \;\; & \sqsubseteq_j \;\; \top_k && \text{(since } (\forall R.\neg_j C)^{\mathcal{I}_j} \subseteq (\top_k)^{\mathcal{I}_j} \text{ by the definitions)} \\
& \sqsubseteq_j \;\; \top_i, && \text{(since } (\top_k)^{\mathcal{I}_j} = r_{kj}(\Delta^{\mathcal{I}_k}) = r_{ij} \circ r_{ki}(\Delta^{\mathcal{I}_k}) \subseteq r_{ij}(\Delta^{\mathcal{I}_i}))
\end{aligned}
$$

$$
\begin{aligned}
(\neg_i(\exists R.C))^{\mathcal{I}_j} &= r_{ij}(\Delta^{\mathcal{I}_i}) \backslash (\exists R.C)^{\mathcal{I}_j} && \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
&= r_{ij}(\Delta^{\mathcal{I}_i}) \backslash \{x \in r_{kj}(\Delta^{\mathcal{I}_k}) | \exists y \in \Delta^{\mathcal{I}_j}, (x,y) \in R^{\mathcal{I}_j} \wedge y \in C^{\mathcal{I}_j}\} \\
& && \text{(by the definition of } (\exists R.C)^{\mathcal{I}_j}) \\
&= (r_{ij}(\Delta^{\mathcal{I}_i}) \backslash r_{kj}(\Delta^{\mathcal{I}_k})) \cup (r_{ij}(\Delta^{\mathcal{I}_i}) \cap \{x \in r_{kj}(\Delta^{\mathcal{I}_k}) | \\
& \qquad \forall y \in \Delta^{\mathcal{I}_j}, (x,y) \in R^{\mathcal{I}_j} \rightarrow y \notin C^{\mathcal{I}_j}\}) \\
& && \text{(set-theoretically)} \\
&= (\neg_i \top_k)^{\mathcal{I}_j} \cup (\top_i^{\mathcal{I}_j} \cap (\forall R.\neg_j C)^{\mathcal{I}_j}) \\
& && \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
&= (\neg_i \top_k \sqcup (\top_i \sqcap \forall R.\neg_j C))^{\mathcal{I}_j} && \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
&= (\neg_i \top_k \sqcup (\forall R.\neg_j C))^{\mathcal{I}_j}. && \text{(since } \forall R.\neg_j C \sqsubseteq_j \top_i)
\end{aligned}
$$

- The proof of Equation 5 is dual to that of Equation 4.

- For Equation 6, as for Equation 4, we have $\geq(n+1)R.C \sqsubseteq_j \top_i$.

$$
\begin{array}{rcll}
(\neg_i(\leq nR.C))^{\mathcal{I}_j} & = & r_{ij}(\Delta^{\mathcal{I}_i})\backslash(\leq nR.C)^{\mathcal{I}_j} & \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
& = & r_{ij}(\Delta^{\mathcal{I}_i})\backslash\{x \in r_{kj}(\Delta^{\mathcal{I}_k})| \ |\{y \in \Delta^{\mathcal{I}_j}|(x,y) \in R^{\mathcal{I}_j} & \\
& & \wedge\, y \in C^{\mathcal{I}_j}\}| \leq n\} & \\
& & & \text{(by the definition of } (\leq nR.C)^{\mathcal{I}_j}) \\
& = & r_{ij}(\Delta^{\mathcal{I}_i})\backslash r_{kj}(\Delta^{\mathcal{I}_k}) \cup (r_{ij}(\Delta^{\mathcal{I}_i}) \cap \{x \in r_{kj}(\Delta^{\mathcal{I}_k})| & \\
& & |\{y \in \Delta^{\mathcal{I}_j}|(x,y) \in R^{\mathcal{I}_j} \wedge y \in C^{\mathcal{I}_j}\}| \geq n+1\}) & \\
& & & \text{(set-theoretically)} \\
& = & (\neg_i \top_k)^{\mathcal{I}_j} \cup (\top_i^{\mathcal{I}_j} \cap (\geq(n+1)R.C)^{\mathcal{I}_j}) & \\
& & & \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
& = & (\neg_i \top_k \sqcup (\top_i \sqcap \geq(n+1)R.C))^{\mathcal{I}_j} & \\
& & & \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
& = & (\neg_i \top_k \sqcup \geq(n+1)R.C)^{\mathcal{I}_j}. & \\
& & & \text{(since } \geq(n+1)R.C \sqsubseteq_j \top_i)
\end{array}
$$

- The proof of Equation 7 follows the dual steps to those in the proof of Equation 6. Q.E.D.

Proof of Lemma 4.2

Proof: For a k-role name R, such that $R \in \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_j)$, we have $r_{ij}(R^{\mathcal{I}_i}) = r_{ij} \circ r_{ki}(R^{\mathcal{I}_k}) = r_{kj}(R^{\mathcal{I}_k}) = R^{\mathcal{I}_j}$.

To prove the claim for concepts, structural induction on the concept formula C will be used.

If C is a k-concept name or a k-nominal name, we have

$$
\begin{array}{rcll}
r_{ij}(C^{\mathcal{I}_i}) & = & r_{ij}(r_{ki}(C^{\mathcal{I}_k})) & \text{(by the definition of } C^{\mathcal{I}_i}) \\
& = & r_{kj}(C^{\mathcal{I}_k}) & \text{(by compositional consistency)} \\
& = & C^{\mathcal{I}_j}. & \text{(by the definition of } C^{\mathcal{I}_j})
\end{array}
$$

For $C = \neg_k D$ and $r_{ij}(D^{\mathcal{I}_i}) = D^{\mathcal{I}_j}$, we have

$$\begin{aligned}
r_{ij}(C^{\mathcal{I}_i}) &= r_{ij}((\neg_k D)^{\mathcal{I}_i}) && \text{(since } C = \neg_k D\text{)} \\
&= r_{ij}(r_{ki}(\Delta^{\mathcal{I}_k}) \backslash D^{\mathcal{I}_i}) && \text{(by the definition of } (\neg_k D)^{\mathcal{I}_i}\text{)} \\
&= r_{ij}(r_{ki}(\Delta^{\mathcal{I}_k})) \backslash r_{ij}(D^{\mathcal{I}_i}) && \text{(since } r_{ij} \text{ is one-to-one)} \\
&= r_{kj}(\Delta^{\mathcal{I}_k}) \backslash D^{\mathcal{I}_j} && \text{(by compositional consistency and} \\
& && \text{the induction hypothesis)} \\
&= (\neg_k D)^{\mathcal{I}_j} && \text{(by the definition of } (\neg_k D)^{\mathcal{I}_j}\text{)} \\
&= C^{\mathcal{I}_j}. && \text{(since } C = \neg_k D\text{)}
\end{aligned}$$

For $C = D \sqcap E$, assuming inductively that $r_{ij}(D^{\mathcal{I}_i}) = D^{\mathcal{I}_j}$ and $r_{ij}(E^{\mathcal{I}_i}) = E^{\mathcal{I}_j}$, we have

$$\begin{aligned}
r_{ij}(C^{\mathcal{I}_i}) &= r_{ij}((D \sqcap E)^{\mathcal{I}_i}) && \text{(since } C = D \sqcap E\text{)} \\
&= r_{ij}(D^{\mathcal{I}_i} \cap E^{\mathcal{I}_i}) && \text{(by the definition of } \cdot^{\mathcal{I}_i}\text{)} \\
&= r_{ij}(D^{\mathcal{I}_i}) \cap r_{ij}(E^{\mathcal{I}_i}) && \text{(since } r_{ij} \text{ is one-to-one)} \\
&= D^{\mathcal{I}_j} \cap E^{\mathcal{I}_j} && \text{(by the induction hypothesis)} \\
&= (D \sqcap E)^{\mathcal{I}_j} && \text{(by the definition of } \cdot^{\mathcal{I}_j}\text{)} \\
&= C^{\mathcal{I}_j}. && \text{(since } C = D \sqcap E\text{)}
\end{aligned}$$

Let $C = \leq n R.D$, with R a k-role, and assume inductively that $r_{ij}(D^{\mathcal{I}_i}) = D^{\mathcal{I}_j}$. We first prove two auxiliary claims.

Claim 1: Let $x' = r_{ij}(x)$. Then $r_{ij} : R^{\mathcal{I}_i}(x) \to R^{\mathcal{I}_j}(x')$ is a total bijection.

Proof: r_{ij} is a one-to-one function by definition. It is onto because

$$\begin{aligned}
R^{\mathcal{I}_j}(x') &= r_{ij}(R^{\mathcal{I}_i})(x') \\
&= (r_{ij} \circ R^{\mathcal{I}_i} \circ r_{ij}^{-})(r_{ij}(x)) \\
&= (r_{ij} \circ R^{\mathcal{I}_i})(x) \\
&= r_{ij}(R^{\mathcal{I}_i}(x))
\end{aligned}$$

By cardinality preservation (item 5 in Definition 5.1), r_{ij} is a total function from $R^{\mathcal{I}_i}(x)$ to $R^{\mathcal{I}_j}(x')$, whence r_{ij} is a total bijection from $R^{\mathcal{I}_i}(x)$ to $R^{\mathcal{I}_j}(x')$. Q.E.D.

Claim 2: Let $x' = r_{ij}(x)$. Then $r_{ij} : R^{\mathcal{I}_i}(x) \cap D^{\mathcal{I}_i} \to R^{\mathcal{I}_j}(x') \cap D^{\mathcal{I}_j}$ is also a total bijection.

Proof: r_{ij} is one-to-one and total on $R^{\mathcal{I}_i}(x)$ by Claim 1. Hence,

$$r_{ij}(R^{\mathcal{I}_i}(x) \cap D^{\mathcal{I}_i}) = r_{ij}(R^{\mathcal{I}_i}(x)) \cap r_{ij}(D^{\mathcal{I}_i}) = R^{\mathcal{I}_j}(x') \cap D^{\mathcal{I}_j}.$$

Thus, r_{ij} is onto $R^{\mathcal{I}_j}(x') \cap D^{\mathcal{I}_j}$, whence the claim holds. Q.E.D.

Using the two claims, we now obtain

$$\begin{aligned}
x' \in r_{ij}((\leq nR.D)^{\mathcal{I}_i}) \quad &\Leftrightarrow \quad \exists x \in (\leq nR.D)^{\mathcal{I}_i} \text{ such that } x' = r_{ij}(x) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{(by the definition of } r_{ij}) \\
&\Leftrightarrow \quad \exists x \in r_{ki}(\Delta^{\mathcal{I}_k}), |R^{\mathcal{I}_i}(x) \cap D^{\mathcal{I}_i}| \leq n \wedge x' = r_{ij}(x) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{(by the definition of } (\leq nR.D)^{\mathcal{I}_i}) \\
&\Leftrightarrow \quad x' \in r_{kj}(\Delta^{\mathcal{I}_k}), |R^{\mathcal{I}_j}(x') \cap D^{\mathcal{I}_j}| \leq n \\
&\qquad\qquad\qquad\qquad\qquad\qquad (\Rightarrow\text{: by compositional consistency and Claim 2}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad (\Leftarrow\text{: by compositional consistency and Claim 2}) \\
&\Leftrightarrow \quad x' \in (\leq nR.D)^{\mathcal{I}_j} \quad \text{(by the definition of } (\leq nR.D)^{\mathcal{I}_j})
\end{aligned}$$

Q.E.D.

Proof of Theorem 4.1: Sufficiency is proven in Lemma A.2 and necessity in Lemma A.3.

Lemma A.2 *Let Σ be a $\mathcal{SHOIQP}$ KB and P_w a package of Σ. If $\top_w$ is satisfiable with respect to $\Re(P_w^*)$, then Σ is consistent as witnessed by P_w.*

Proof: If $\top_w$ is satisfiable with respect to $\Re(P_w^*)$, then $\Re(P_w^*)$ has at least one model $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, such that $\top_w^{\mathcal{I}} \neq \emptyset$. Our goal is to construct a model of P_w^* from $\mathcal{I}$, such that $\Delta^{\mathcal{I}_w} \neq \emptyset$. For each package P_i, a local interpretation $\mathcal{I}_i$ is constructed as a projection of $\mathcal{I}$ in the following way:

- $\Delta^{\mathcal{I}_i} = \top_i^{\mathcal{I}}$.
- For every concept name C that appears in P_i, $C^{\mathcal{I}_i} = C^{\mathcal{I}} \cap \top_i^{\mathcal{I}}$.
- For every role name R that appears in P_i, $R^{\mathcal{I}_i} = R^{\mathcal{I}} \cap (\top_i^{\mathcal{I}} \times \top_i^{\mathcal{I}})$.
- For every nominal name o that appears in P_i, $o^{\mathcal{I}_i} = o^{\mathcal{I}}$.

For every pair i, j, such that $P_i \in P_j^*$, we define

$$r_{ij} = \{(x, x) | x \in \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j}\}.$$

Clearly, we have $\Delta^{\mathcal{I}_w} = \top_w^{\mathcal{I}} \neq \emptyset$, by the hypothesis. So it suffices, now, to show that $\langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^*} \rangle$ is a model of the modular ontology P_w^*, i.e., that it satisfies the seven conditions postulated in Definition 5.1.

First, it is clear from the definition that each r_{ij} is in fact a one-to-one relation.

Second, we must show that Compositional Consistency holds.

- Suppose that $P_i \in P_j^*$, $x \in \Delta^{\mathcal{I}_i}$, $y \in \Delta^{\mathcal{I}_j}$, and $(x, y) \in \rho_{ij}$. Therefore, x and y must be connected by some chain of domain relations and/or inverse domain relations according to the definition of ρ_{ij}. Because all domain relations are identities, this implies that $x = y \in \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j}$, whence, once more by the definition of r_{ij}, we obtain that $(x, y) \in r_{ij}$. This proves that $\rho_{ij} \subseteq r_{ij}$.

- Assume that i, j, k, with $i \neq j$, are such that $P_i \in P_k^*$, $P_k \in P_j^*$ and $(x, y) \in r_{ij}$. Then $x = y \in \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j}$. Since, in that case, $\top_i \sqcap \top_j \sqsubseteq \top_k$, this implies that $x \in \Delta^{\mathcal{I}_k}$, whence $x \in \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} \cap \Delta^{\mathcal{I}_k}$, showing that $(x, x) \in r_{ik}$ and $(x, x) \in r_{kj}$. Therefore $r_{ij} \subseteq r_{kj} \circ r_{ik}$.

- From the definition of ρ_{ij}, we have $r_{kj} \circ r_{ik} \subseteq \rho_{ij}$.

Hence, $\rho_{ij} = r_{ij} = r_{kj} \circ r_{ik}$, for $P_i \in P_k^*$ and $P_k \in P_j^*$.

Next, it is shown that Conditions 3,4 and 6 of Definition 5.1 hold for the distributed interpretation. Let X be an i-concept name or an i-nominal name. Then, we have that

$$\begin{aligned} r_{ij}(X^{\mathcal{I}_i}) &= X^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} && \text{(by the definition of } r_{ij}) \\ &= X^{\mathcal{I}} \cap \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} && \text{(by the definition of } X^{\mathcal{I}_i}) \\ &= X^{\mathcal{I}} \cap \Delta^{\mathcal{I}_j} && \text{(since } i : X \sqsubseteq \top_i) \\ &= X^{\mathcal{I}_j}. && \text{(by the definition of } X^{\mathcal{I}_j}) \end{aligned}$$

For X an i-role name, the same equalities hold with all local interpretation domains replaced by their cartesian squares.

To show Cardinality Preservation for Roles, suppose that R is an i-role in P_j and that $(x, x') \in r_{ij}$, i.e., $x = x' \in \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j}$. Then, we have

$$
\begin{array}{rlll}
y \in R^{\mathcal{I}_i}(x) & \text{iff} & (x,y) \in R^{\mathcal{I}_i} & \text{(by the definition of } R^{\mathcal{I}_i}(x)\text{)} \\
& \text{iff} & (x,y) \in R^{\mathcal{I}} & \text{(since } R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}\text{)} \\
& \text{iff} & (x',y) \in R^{\mathcal{I}} \cap (\Delta^{\mathcal{I}_j} \times \Delta^{\mathcal{I}_j}) & \text{(by the local domain and local} \\
& & & \text{range axioms, and } x = x' \in \Delta^{\mathcal{I}_j}\text{)} \\
& \text{iff} & (x',y) \in R^{\mathcal{I}_j} & \text{(by the definition of } R^{\mathcal{I}_j}\text{)} \\
& \text{iff} & y = r_{ij}(y) \in R^{\mathcal{I}_j}(x'). & \text{(by the definition of } R^{\mathcal{I}_j}(x')\text{)}
\end{array}
$$

Thus, cardinality preservation for roles holds.

Finally, it remains to show that Condition 7 of Definition 5.1 holds, i.e., that $\mathcal{I}_j$ is a model of P_j, for every j.

For every role inclusion of the form $R \sqsubseteq S$ in P_j, R and S must be j-roles (by our restriction on the use of imported roles), whence we have that

$$
\begin{array}{rcll}
R^{\mathcal{I}_j} & = & R^{\mathcal{I}} \cap (\Delta^{\mathcal{I}_j} \times \Delta^{\mathcal{I}_j}) & \text{(by the definition of } R^{\mathcal{I}_j}\text{)} \\
& \subseteq & S^{\mathcal{I}} \cap (\Delta^{\mathcal{I}_j} \times \Delta^{\mathcal{I}_j}) & \text{(since } R \sqsubseteq S \text{ holds in the integrated ontology)} \\
& = & S^{\mathcal{I}_j}. & \text{(by the definition of } S^{\mathcal{I}_j}\text{)}
\end{array}
$$

For a role R that appears in P_j, we have that $\mathsf{Trans}_j(R)$ if and only if $\mathsf{Trans}(R)$, whence

$$
\begin{array}{rcll}
(R^{\mathcal{I}_j})^+ & = & (R^{\mathcal{I}} \cap (\Delta^{\mathcal{I}_j} \times \Delta^{\mathcal{I}_j}))^+ & \text{(by the definition of } R^{\mathcal{I}_j}\text{)} \\
& = & (R^{\mathcal{I}})^+ \cap (\Delta^{\mathcal{I}_j} \times \Delta^{\mathcal{I}_j}) & \text{(set-theoretically)} \\
& = & R^{\mathcal{I}} \cap (\Delta^{\mathcal{I}_j} \times \Delta^{\mathcal{I}_j}) & \text{(since } \mathsf{Trans}(R) \text{ holds)} \\
& = & R^{\mathcal{I}_j}. & \text{(by the definition of } R^{\mathcal{I}_j}\text{)}
\end{array}
$$

Finally, suppose that $C \sqsubseteq D$ is a concept inclusion in P_j. Then we must have $\#_j(C)^{\mathcal{I}} \subseteq \#_j(D)^{\mathcal{I}}$, whence, to prove that $C^{\mathcal{I}_j} \subseteq D^{\mathcal{I}_j}$, it suffices to show that, for every concept formula X that appears in P_j, we have $\#_j(X)^{\mathcal{I}} = X^{\mathcal{I}_j}$. We do this by structural induction on X. We will consider in detail only concepts constructed using negation, conjunction and number restriction. All other constructors may be handled similarly.

For the basis of the induction, if X is a j-concept name or a j-nominal name, then we have $\#_j(X) = X$, whence $\#_j(X)^{\mathcal{I}} = X^{\mathcal{I}} = X^{\mathcal{I}} \cap \Delta^{\mathcal{I}_j} = X^{\mathcal{I}_j}$, whereas, if X is an i-concept name or an i-nominal name, with $i \neq j$, we have $\#_j(X)^{\mathcal{I}} = (X \sqcap \top_j)^{\mathcal{I}} = X^{\mathcal{I}} \cap \Delta^{\mathcal{I}_j} = X^{\mathcal{I}_j}$.

Suppose, next, as the induction hypothesis, that for concepts C and D appearing in P_j, $\#_j(C)^{\mathcal{I}} = C^{\mathcal{I}_j}$ and $\#_j(D)^{\mathcal{I}} = D^{\mathcal{I}_j}$, and also note that $\#_j(R) = R$ for every i-role R appearing in P_j. Thus, we have

$$
\begin{aligned}
\#_j(\neg_i C)^{\mathcal{I}} &= (\neg \#_j(C) \sqcap \top_i \sqcap \top_j)^{\mathcal{I}} && \text{(by the definition of } \#_j(\neg_i C)) \\
&= (\neg \#_j(C))^{\mathcal{I}} \cap \top_i^{\mathcal{I}} \cap \top_j^{\mathcal{I}} && \text{(by the definition of } \cdot^{\mathcal{I}}) \\
&= (\Delta^{\mathcal{I}} \backslash \#_j(C)^{\mathcal{I}}) \cap \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} && \text{(by the definition of } \cdot^{\mathcal{I}}) \\
&= (\Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j}) \backslash \#_j(C)^{\mathcal{I}} && \text{(since } \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} \subseteq \Delta^{\mathcal{I}}) \\
&= r_{ij}(\Delta^{\mathcal{I}_i}) \backslash C^{\mathcal{I}_j} && \text{(by the definition of } r_{ij} \text{ and} \\
& && \text{the induction hypothesis)} \\
&= (\neg_i C)^{\mathcal{I}_j}. && \text{(by the definition of } (\neg_i C)^{\mathcal{I}_j})
\end{aligned}
$$

$$
\begin{aligned}
\#_j(C \sqcap D)^{\mathcal{I}} &= (\#_j(C) \sqcap \#_j(D) \sqcap \top_j)^{\mathcal{I}} && \text{(by the definition of } \#_j(C \sqcap D)) \\
&= \#_j(C)^{\mathcal{I}} \cap \#_j(D)^{\mathcal{I}} \cap \Delta^{\mathcal{I}_j} && \text{(by the definition of } \cdot^{\mathcal{I}}) \\
&= C^{\mathcal{I}_j} \cap D^{\mathcal{I}_j} \cap \Delta^{\mathcal{I}_j} && \text{(by the induction hypothesis)} \\
&= C^{\mathcal{I}_j} \cap D^{\mathcal{I}_j} && \text{(since } C^{\mathcal{I}_j}, D^{\mathcal{I}_j} \subseteq \Delta^{\mathcal{I}_j}) \\
&= (C \sqcap D)^{\mathcal{I}_j}. && \text{(by the definition of } \cdot^{\mathcal{I}_j})
\end{aligned}
$$

$$
\begin{aligned}
\#_j(\leq nR.C)^{\mathcal{I}} &= ((\leq nR.\#_j(C)) \sqcap \top_i \sqcap \top_j)^{\mathcal{I}} \\
&= (\leq nR.\#_j(C))^{\mathcal{I}} \cap \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} \\
&= \{x \in \Delta^{\mathcal{I}} |\ |R^{\mathcal{I}}(x) \cap (\#_j(C))^{\mathcal{I}}| \leq n\} \cap \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} \\
&= \{x \in \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} |\ |R^{\mathcal{I}}(x) \cap C^{\mathcal{I}_j}| \leq n\} \\
&= \{x \in \Delta^{\mathcal{I}_i} \cap \Delta^{\mathcal{I}_j} |\ |R^{\mathcal{I}_j}(x) \cap C^{\mathcal{I}_j}| \leq n\} && (*) \\
&= \{x \in r_{ij}(\Delta^{\mathcal{I}_i}) |\ |R^{\mathcal{I}_j}(x) \cap C^{\mathcal{I}_j}| \leq n\} \\
&= (\leq nR.C)^{\mathcal{I}_j}
\end{aligned}
$$

(*) is because $R^{\mathcal{I}_j} \subseteq R^{\mathcal{I}}$ and for any $y \in R^{\mathcal{I}}(x) \cap C^{\mathcal{I}_j}$, $y \in \Delta^{\mathcal{I}_j}$, hence $y \in R^{\mathcal{I}_j}(x) \cap C^{\mathcal{I}_j}$. Q.E.D.

Next, we proceed to show the reverse implication.

Lemma A.3 *Let Σ be a $\mathcal{SHOIQP}$ KB. If Σ is consistent as witnessed by a package P_w, then $\top_w$ is satisfiable with respect to $\Re(P_w^*)$.*

Proof: Suppose that Σ is consistent as witnessed by P_w. Thus, it has a distributed model $\langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^*} \rangle$, such that $\Delta^{\mathcal{I}_w} \neq \emptyset$. We proceed to construct a model $\mathcal{I}$ of $\Re(P_w^*)$ by merging individuals that are related via chains of image domain relations or their inverses. More precisely, for every element x in the distributed model, we define its equivalence class

$\overline{x} = \{y|(x,y) \in \rho\}$ where ρ is the symmetric and transitive closure of the set $\bigcup_{P_i \in P_j^*} r_{ij}$. Moreover, for a set S, we define $\overline{S} = \{\bar{x}|x \in S\}$ and for a binary relation R, we define $\overline{R} = \{(\overline{x},\overline{y})|(x,y) \in R\}$.

Claim 3: (a) For all i and for all $\overline{x}$, $|\overline{x} \cap \Delta^{\mathcal{I}_i}| \leq 1$.

(b) For all i and any set $S \subseteq \Delta^{\mathcal{I}_i}$, $|S| = |\overline{S}|$.

(c) For all i and all sets $A_1, A_2 \subseteq \Delta^{\mathcal{I}_i}$, $\overline{A_1 \backslash A_2} = \overline{A_1} \backslash \overline{A_2}$.

(d) For all i and for all $S \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_i}$, $(\overline{S})^+ = (\overline{S^+})$.

Proof: (a) Suppose $u, v \in \overline{x} \cap \Delta^{\mathcal{I}_i}, u \neq v$. Then there must exist a $y \in (\overline{x} \backslash \Delta^{\mathcal{I}_i}) \cap \Delta^{\mathcal{I}_j}$ for some j, which implies that $\{(v,y),(u,y)\} \subseteq \rho_{ij} = r_{ij}$ or $\{(y,u),(y,v)\} \subseteq \rho_{ji} = r_{ji}$, contradicting the assumption that domain relations are one-to-one. Hence $|\overline{x} \cap \Delta^{\mathcal{I}_i}| \leq 1$.

(b) We prove this statement by showing that $f : x \rightarrow \overline{x}$ is a total bijection from S to $\overline{S}$. f is a total and onto function by the definition of $\overline{S}$. f is injective because for $\overline{x} \in \overline{S}$, if there are two distinct x_1, x_2 in S, such that $\overline{x_1} = \overline{x_2} = \overline{x}$, then $\{x_1, x_2\} \subseteq \overline{x} \cap \Delta^{\mathcal{I}_i}$, contradicting (a).

(c) This statement holds because

$$\begin{aligned}
\overline{x} \in \overline{A_1} \backslash \overline{A_2} &\leftrightarrow (\overline{x} \in \overline{A_1} \text{ and } \overline{x} \notin \overline{A_2}) \\
&\leftrightarrow \exists x', \{x'\} = \overline{x} \cap \Delta^{\mathcal{I}_i}, x' \in A_1 \backslash A_2 \qquad \text{(by Part (a))} \\
&\leftrightarrow \overline{x} \in \overline{A_1 \backslash A_2}
\end{aligned}$$

d) First we prove $(\overline{S})^+ \subseteq \overline{S^+}$. It is because $S \subseteq S^+$, hence $(\overline{S})^+ \subseteq (\overline{S^+})^+$; on the other hand, if $(x,y) \in (\overline{S^+})^+$, there exist $x_1, ..., x_n$ such that $(x,x_1), (x_n,y)$ and (x_{k-1},x_k) (for $k = 2, ..., n$) $\in \overline{S^+}$, hence $(x|_i, x_1|_i), (x_n|_i, y|_i)$ and $(x_{k-1}|_i, x_k|_i) \in S^+$ (for $k = 2, ..., n$), hence $(x|_i, y|_i) \in S^+$, thus $(x,y) \in \overline{S^+}$.

In the other direction, if $(x,y) \in \overline{S^+}$, then $(x|_i, y|_i) \in S^+$, hence there exist $x_1, ..., x_n$ such that $(x|_i, x_1), (x_n, y|_i)$ and $(x_{k-1}, x_k) \in S$ (for $k = 2, ..., n$), therefore $(x, \overline{x_1}), (\overline{x_n}, y)$ and $(\overline{x_{k-1}}, \overline{x_k}) \in \overline{S}$ (for $k = 2, ..., n$), thus $(x,y) \in (\overline{S})^+$.

Claim 3 Q.E.D.

We denote by $\overline{x}|_i$ the element (if it exists) in $\Delta^{\mathcal{I}_i}$ that belongs to $\overline{x}$, i.e., $\overline{x}|_i \in \Delta^{\mathcal{I}_i} \cap \overline{x}$.

We now proceed to define a model of Σ. Let $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ be defined as follows:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}} = \overline{\bigcup_i \Delta^{\mathcal{I}_i}}$, and $\bot^{\mathcal{I}} = \emptyset$.

- For every i-name X, $X^{\mathcal{I}} := \overline{X^{\mathcal{I}_i}}$.
- For every i, $\top_i^{\mathcal{I}} = \overline{\Delta^{\mathcal{I}_i}}$.

We must show that $\mathcal{I}$ is a model of $\Re(P_w^*)$, such that $\top_w^{\mathcal{I}} \neq \emptyset$.

We have $\top_w^{\mathcal{I}} = \overline{\Delta^{\mathcal{I}_w}} \neq \emptyset$, by the hypothesis.

a) Suppose, next that i, j, k are such that $i \neq j$, $P_i \in P_k^*$ and $P_k \in P_j^*$. To see that $\top_i \sqcap \top_j \sqsubseteq \top_k$ holds in $\mathcal{I}$, suppose that $\overline{x} \in (\top_i \sqcap \top_j)^{\mathcal{I}} = \top_i^{\mathcal{I}} \cap \top_j^{\mathcal{I}} = \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}}$. Then $\overline{x} \in \overline{\Delta^{\mathcal{I}_i}}$ and $\overline{x} \in \overline{\Delta^{\mathcal{I}_j}}$, therefore $(\overline{x}|_i, \overline{x}|_j) \in \rho_{ij} = r_{kj} \circ r_{ik}$. Hence, there exists $x' \in \Delta^{\mathcal{I}_k}$, such that $(\overline{x}|_i, x') \in r_{ik} \subseteq \rho$ and $(x', \overline{x}|_j) \in r_{kj} \subseteq \rho$, implying $\overline{x} = \overline{x'} \in \overline{\Delta^{\mathcal{I}_k}} = \top_k^{\mathcal{I}}$.

b) is discussed at the end of the proof.

c) For every role inclusion $R \sqsubseteq S$ in P_j, since both R and S must be j-roles, we obtain

$$\begin{aligned} R^{\mathcal{I}} &= \overline{R^{\mathcal{I}_j}} && \text{(by the definition of } R^{\mathcal{I}}) \\ &\subseteq \overline{S^{\mathcal{I}_j}} && \text{(since } R \sqsubseteq S \text{ is in } P_j) \\ &= S^{\mathcal{I}} && \text{(by the definition of } S^{\mathcal{I}}) \end{aligned}$$

d) If C is an i-concept name or an i-nominal name, then we do have $C \sqsubseteq \top_i$, since $C^{\mathcal{I}} = \overline{C^{\mathcal{I}_i}} \subseteq \overline{\Delta^{\mathcal{I}_i}} = \top_i^{\mathcal{I}}$.

e) If R is an i-role, then $R^{\mathcal{I}} = \overline{R^{\mathcal{I}_i}} \subseteq \overline{\Delta^{\mathcal{I}_i}} \times \overline{\Delta^{\mathcal{I}_i}} = \top_i^{\mathcal{I}} \times \top_i^{\mathcal{I}}$, whence the domain and range of $R^{\mathcal{I}}$ are both restricted to $\top_i^{\mathcal{I}}$.

f) Next, let R be an i-role name in P_j. It must be shown that $\exists R.\top_j \sqsubseteq \top_j$ and $\exists R^-.\top_j \sqsubseteq \top_j$ are both valid in $\mathcal{I}$. Only the first subsumption will be shown. The second follows using a similar argument. For any $\overline{x}$,

$$\begin{aligned} \overline{x} \in (\exists R.\top_j)^{\mathcal{I}} &\Rightarrow \exists \overline{y}, (\overline{x}, \overline{y}) \in R^{\mathcal{I}} \text{ and } \overline{y} \in \top_j^{\mathcal{I}} \\ &\Rightarrow \exists \overline{y}, (\overline{x}, \overline{y}) \in \overline{R^{\mathcal{I}_i}} \text{ and } \overline{y} \in \overline{\Delta^{\mathcal{I}_j}} \\ &\Rightarrow \exists x' = \overline{x}|_i, y' = \overline{y}|_i, (x', y') \in R^{\mathcal{I}_i} \text{ and } y'' = \overline{y}|_j, (y', y'') \in \rho_{ij} = r_{ij} \\ &\Rightarrow \exists x'' \in \Delta^{\mathcal{I}_j} \text{ and } (x', x'') \in r_{ij} = \rho_{ij} \\ &\qquad \text{(because } r_{ij} \text{ is a total bijection from } (R^-)^{\mathcal{I}_i}(y') \text{ to} \\ &\qquad (R^-)^{\mathcal{I}_j}(y'') \text{ by Claim 1 in the proof of Lemma 4.2.)} \\ &\Rightarrow \overline{x} = \overline{x'} = \overline{x''} \in \overline{\Delta^{\mathcal{I}_j}} = \top_j^{\mathcal{I}} \end{aligned}$$

g) For a transitive i-role R, we have $R^{\mathcal{I}^+} = (\overline{R^{\mathcal{I}_i}})^+ = \overline{R^{\mathcal{I}_i{}^+}} = \overline{R^{\mathcal{I}_i}} = R^{\mathcal{I}}$ (the second equality is by Claim 3 part (d)).

b): For concept inclusions, we first prove, by induction on the structure of concepts, that for any concept E appearing in P_j,

$$\#_j(E)^{\mathcal{I}} = \overline{E^{\mathcal{I}_j}}. \tag{A.1}$$

For the basis of the induction, let E be a concept such that $\mathsf{Sig}(E) \subseteq \mathsf{Sig}(P_i) \cap \mathsf{Sig}(P_j)$:

Claim 4: $\overline{E^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} = \overline{r_{ij}(E^{\mathcal{I}_i})} = \overline{E^{\mathcal{I}_j}}$

Proof:

$$\begin{aligned}
\overline{E^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} &= \{\overline{x} | x \in E^{\mathcal{I}_i}\} \cap \{\overline{x'} | x' \in \Delta^{\mathcal{I}_j}\} && \text{(by definition)} \\
&= \{\overline{x'} | \exists x \in E^{\mathcal{I}_i} \wedge x' \in \Delta^{\mathcal{I}_j} \wedge \overline{x} = \overline{x'}\} \\
&= \{\overline{x'} | \exists x \in E^{\mathcal{I}_i} \wedge x' \in \Delta^{\mathcal{I}_j} \wedge (x, x') \in \rho_{ij} = r_{ij}\} \\
& && \text{(by compositional consistency)} \\
&= \{\overline{x'} | x' \in r_{ij}(E^{\mathcal{I}_i})\} && \text{(by the definition of } r_{ij}(\cdot)) \\
&= \overline{r_{ij}(E^{\mathcal{I}_i})} && \text{(by definition)} \\
&= \overline{E^{\mathcal{I}_j}} && \text{(since } r_{ij}(E^{\mathcal{I}_i}) = E^{\mathcal{I}_j}) \text{ Q.E.D.}
\end{aligned}$$

The proof of the basis case of the induction is concluded as follows: if E is an i-concept name or an i-nominal name, then

$$\begin{aligned}
\#_j(E)^{\mathcal{I}} &= (E \sqcap \top_j)^{\mathcal{I}} \\
&= E^{\mathcal{I}} \cap \top_j^{\mathcal{I}} \\
&= \overline{E^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} \\
&= \overline{E^{\mathcal{I}_j}}. && \text{(by Claim 4)}
\end{aligned}$$

For the induction step, assume that for concepts C and D appearing in P_j, we have that $\#_j(C)^{\mathcal{I}} = \overline{C^{\mathcal{I}_j}}$ and $\#_j(D)^{\mathcal{I}} = \overline{D^{\mathcal{I}_j}}$.

If $E = \neg_i C$, then

$$
\begin{aligned}
\#_j(E)^{\mathcal{I}} &= \#_j(\neg_i C)^{\mathcal{I}} && \text{(since } E = \neg_i C\text{)} \\
&= (\neg \#_j(C) \sqcap \top_i \sqcap \top_j)^{\mathcal{I}} && \text{(by the definition of } \#_j(\neg_i C)\text{)} \\
&= (\Delta^{\mathcal{I}} \backslash \#_j(C)^{\mathcal{I}}) \cap \top_i^{\mathcal{I}} \cap \top_j^{\mathcal{I}} && \text{(by the definition of } \cdot^{\mathcal{I}}\text{)} \\
&= (\Delta^{\mathcal{I}} \backslash (\overline{C^{\mathcal{I}_j}})) \cap \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} && \text{(by the induction hypothesis)} \\
&= (\overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}}) \backslash (\overline{C^{\mathcal{I}_j}}) && \text{(since } \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} \subseteq \Delta^{\mathcal{I}}\text{)} \\
&= (\overline{r_{ij}(\Delta^{\mathcal{I}_i})}) \backslash (\overline{C^{\mathcal{I}_j}}) && \text{(by Claim 4)} \\
&= \overline{(r_{ij}(\Delta^{\mathcal{I}_i}) \backslash C^{\mathcal{I}_j})} && \text{(by Claim 3c)} \\
&= \overline{(\neg_i C)^{\mathcal{I}_j}} && \text{(by the definition of } (\neg_i C)^{\mathcal{I}_j}\text{)} \\
&= \overline{E^{\mathcal{I}_j}}.
\end{aligned}
$$

If $E = C \sqcap D$, then

$$
\begin{aligned}
\#_j(E)^{\mathcal{I}} &= \#_j(C \sqcap D)^{\mathcal{I}} && \text{(since } E = C \sqcap D\text{)} \\
&= (\#_j(C) \sqcap \#_j(D) \sqcap \top_j)^{\mathcal{I}} && \text{(by the definition of } \#_j(C \sqcap D)\text{)} \\
&= \#_j(C)^{\mathcal{I}} \cap \#_j(D)^{\mathcal{I}} \cap \top_j^{\mathcal{I}} && \text{(by the definition of } \cdot^{\mathcal{I}}\text{)} \\
&= \overline{C^{\mathcal{I}_j}} \cap \overline{D^{\mathcal{I}_j}} \cap \overline{\Delta^{\mathcal{I}_j}} && \text{(by the induction hypothesis)} \\
&= \overline{C^{\mathcal{I}_j}} \cap \overline{D^{\mathcal{I}_j}} && \text{(since } \overline{C^{\mathcal{I}_j}} \cap \overline{D^{\mathcal{I}_j}} \subseteq \overline{\Delta^{\mathcal{I}_j}}\text{)} \\
&= \{\overline{x} | x \in C^{\mathcal{I}_j}\} \cap \{\overline{x} | x \in D^{\mathcal{I}_j}\} && \text{(by the definition of } \overline{(\cdot)}\text{)} \\
&= \{\overline{x} | x \in C^{\mathcal{I}_j} \cap D^{\mathcal{I}_j}\} && \text{(follows from Claim 3a)} \\
&= \overline{(C \sqcap D)^{\mathcal{I}_j}} && \text{(by the definition of } \cdot^{\mathcal{I}_j}\text{)} \\
&= \overline{E^{\mathcal{I}_j}}.
\end{aligned}
$$

For $E = \leq n R.C$, where R is an i-role, we first need to show:

Claim 5: If $\overline{x} \in \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}}$, then $(\overline{x}, \overline{y}) \in \overline{R^{\mathcal{I}_i}}$ iff $(\overline{x}|_j, \overline{y}|_j) \in R^{\mathcal{I}_j}$, for any y.

Proof:

$\overline{x} \in \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}}$ and $(\overline{x}, \overline{y}) \in \overline{R^{\mathcal{I}_i}}$

$\Rightarrow$ $(\overline{x}|_i, \overline{y}|_i) \in R^{\mathcal{I}_i}$ and $(\overline{x}|_i, \overline{x}|_j) \in \rho_{ij} = r_{ij}$

$\Rightarrow$ $\exists y' \in \Delta^{\mathcal{I}_j}, (\overline{x}|_j, y') \in R^{\mathcal{I}_j},$ and $(\overline{y}|_i, y') \in r_{ij} = \rho_{ij}$

(because r_{ij} is a total bijection from $R^{\mathcal{I}_i}(\overline{x}|_i)$ to $R^{\mathcal{I}_j}(\overline{x}|_j)$ by Claim 1 in the proof of Lemma 4.2.)

$\Rightarrow$ $(\overline{x}|_j, \overline{y}|_j) = (\overline{x}|_j, y') \in R^{\mathcal{I}_j}$

and conversely

$$\begin{aligned}
& \overline{x} \in \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} \text{ and } (\overline{x}|_j, \overline{y}|_j) \in R^{\mathcal{I}_j} \\
\Rightarrow\ & (\overline{x}|_j, \overline{y}|_j) \in r_{ij}(R^{\mathcal{I}_i}) && \text{(since } r_{ij}(R^{\mathcal{I}_i}) = R^{\mathcal{I}_j}\text{)} \\
\Rightarrow\ & \exists x', y' \in \Delta^{\mathcal{I}_i}, (x', \overline{x}|_j) \in r_{ij}, (y', \overline{y}|_j) \in r_{ij}, (x', y') \in R^{\mathcal{I}_i} \\
\Rightarrow\ & (\overline{x}, \overline{y}) \in \overline{R^{\mathcal{I}_i}}. && \text{Q.E.D.}
\end{aligned}$$

Based on Claims 3,4 and 5, we have:

$$\begin{aligned}
\#_j(E)^{\mathcal{I}} &= \#_j(\leq nR.C)^{\mathcal{I}} && \text{(since } E = \leq nR.C\text{)} \\
&= (\leq nR.\#_j(C) \sqcap \top_i \sqcap \top_j)^{\mathcal{I}} && \text{(by the definition of } \#_j(\leq nR.C)\text{)} \\
&= \{\overline{x} \mid |\{\overline{y} | (\overline{x}, \overline{y}) \in R^{\mathcal{I}} \wedge \overline{y} \in \#_j(C)^{\mathcal{I}}\}| \leq n\} \cap \top_i^{\mathcal{I}} \cap \top_j^{\mathcal{I}} \\
& && \text{(by the definition of } \cdot^{\mathcal{I}}\text{)} \\
&= \{\overline{x} \mid |\{\overline{y} | (\overline{x}, \overline{y}) \in \overline{R^{\mathcal{I}_i}} \wedge \overline{y} \in \overline{C^{\mathcal{I}_j}}\}| \leq n\} \cap \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} \\
& && \text{(by the definitions of } R^{\mathcal{I}}, \top_i^{\mathcal{I}}, \top_j^{\mathcal{I}} \text{ and the induction hypothesis)} \\
&= \{\overline{x} \in \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} \mid |\{\overline{y}|_j \in \Delta^{\mathcal{I}_j} | (\overline{x}, \overline{y}) \in \overline{R^{\mathcal{I}_i}} \wedge \overline{y} \in \overline{C^{\mathcal{I}_j}}\}| \leq n\} \\
& && \text{(by Claim 3b)} \\
&= \{\overline{x} \in \overline{\Delta^{\mathcal{I}_i}} \cap \overline{\Delta^{\mathcal{I}_j}} \mid |\{\overline{y}|_j \in \Delta^{\mathcal{I}_j} | (\overline{x}|_j, \overline{y}|_j) \in R^{\mathcal{I}_j} \wedge \overline{y}|_j \in C^{\mathcal{I}_j}\}| \leq n\} \\
& && \text{(by Claim 5)} \\
&= \{\overline{x} \in \overline{r_{ij}(\Delta^{\mathcal{I}_i})} \mid |\{\overline{y}|_j \in \Delta^{\mathcal{I}_j} | (\overline{x}|_j, \overline{y}|_j) \in R^{\mathcal{I}_j} \wedge \overline{y}|_j \in C^{\mathcal{I}_j}\}| \leq n\} \\
& && \text{(by Claim 4)} \\
&= \{\overline{x} | x \in r_{ij}(\Delta^{\mathcal{I}_i}),\ |\{z \in \Delta^{\mathcal{I}_j} | (x, z) \in R^{\mathcal{I}_j} \wedge z \in C^{\mathcal{I}_j}\}| \leq n\} \\
& && \text{(by Claim 3a)} \\
&= \overline{(\leq nR.C)^{\mathcal{I}_j}} && \text{(by the definition of } (\leq nR.C)^{\mathcal{I}_j}\text{)} \\
&= \overline{E^{\mathcal{I}_j}}.
\end{aligned}$$

Finally, using Equation (A.1), we have that

$$\begin{aligned}
\#_j(C)^{\mathcal{I}} &= C^{\mathcal{I}_j} && \text{(by Equation (A.1))} \\
&\subseteq D^{\mathcal{I}_j} && \text{(since } C \sqsubseteq D \text{ is in } P_j\text{)} \\
&= \#_j(D)^{\mathcal{I}}. && \text{(by Equation (A.1))}
\end{aligned}$$

Lemma 4 Q.E.D.

That finishes the proof of Theorem 4.1

.

Proof of Theorem 4.6:

Given a DDL KB Σ and its integration $\#(\Sigma)$ as outlined in Section 4.4.1.2, we first prove that distributed models of Σ and the traditional DL models of $\#(\Sigma)$ can be mapped to each other.

For any distributed model $M_d = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$ of Σ, where $\mathcal{I}_i = \langle\Delta_i, (.)^{\mathcal{I}_i}\rangle$, we can construct a traditional DL model $\mathcal{I} = \langle\Delta^{\mathcal{I}}, (.)^{\mathcal{I}}\rangle$ by merging individual pairs connected by domain relations into single individuals in $\mathcal{I}$. For any concept or role $i : A$, $(i : A)^{\mathcal{I}}$ in $\mathcal{I}$ is the same as $A^{\mathcal{I}_i}$ in M_d. It is easy to see that

- $A^{\mathcal{I}_i} \subseteq B^{\mathcal{I}_i} \rightarrow (i : A)^{\mathcal{I}} \subseteq (i : B)^{\mathcal{I}}$
- $\Delta_i \subseteq \Delta^{\mathcal{I}}$
- $(i : A)^{\mathcal{I}} \subseteq \Delta_i$ for every atomic concept $i : A$
- for any $\langle x, y\rangle \in (i : s)^{\mathcal{I}}$, we have $x \in \Delta_i$ and $y \in \Delta_i$
- (INTO) $r_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_i} \rightarrow (i : C)^{\mathcal{I}} \subseteq (i : D)^{\mathcal{I}}$
- (ONTO) $r_{ij}(C^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_i} \rightarrow (i : C)^{\mathcal{I}} \supseteq (i : D)^{\mathcal{I}}$

The last two conditions are true since domain relations r_{ij} in M_d are one-to-one. Therefore, $\mathcal{I}$ is a model of $\#(\Sigma)$.

In the other direction, given a traditional DL model $\mathcal{I} = \langle\Delta^{\mathcal{I}}, (.)^{\mathcal{I}}\rangle$ of the integrated KB $\#(\Sigma)$, we may also construct a distributed model $M_d = \langle\{\mathcal{I}_i\}, \{r_{ij}\}\rangle$ (where $\mathcal{I}_i = \langle\Delta_i, (.)^{\mathcal{I}_i}\rangle$) of Σ in the following way: for any concept name $i : A$, Δ_i contains all individuals of $(i : A)^{\mathcal{I}}$. $(.)^{\mathcal{I}_i}$ maps each i-concept and i-role to the same set as $(.)^{\mathcal{I}}$ does. For any bridge rule $C \xrightarrow{\sqsubseteq} D$ or $C \xrightarrow{\sqsupseteq} D$ in B_{ij}, Δ_j also contains all individuals of $(i : C)^{\mathcal{I}}$ and the domain relation r_{ij} contains shared individual pairs in Δ_i and Δ_j.

It is easy to see that by making "copies" of some individuals in different local domains such a decomposition creates a distributed model for Σ. The domain relations in the distributed model are one-to-one.

Now we prove reasoning in Σ is exact w.r.t the integrated ontology $\#(\Sigma)$. For any i-concept $i : X$ and $i : Y$, if $\Sigma \models_d i : X \sqsubseteq i : Y$, we have that for any distributed model M_d of Σ, $X^{\mathcal{I}_i} \subseteq Y^{\mathcal{I}_i}$. Consequently, since any traditional DL model $\mathcal{I}$ of $\#(\Sigma)$ can be reduced to a

distributed model, $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$ must be hold, i.e. $\#(\Sigma) \models \#(i : X) \sqsubseteq \#(i : Y)$. Similarly, if $\#(\Sigma) \models \#(i : X) \sqsubseteq \#(i : Y)$, we also have $\Sigma \models_d i : X \sqsubseteq i : Y$.

For any i-concept $i : X$ and j-concept $j : Y$, if $\Sigma \models_d i : X \xrightarrow{\sqsubseteq} j : Y$, for any distributed model M_d of Σ, we have: $r_{ij}(X^{\mathcal{I}_i}) \subseteq Y^{\mathcal{I}_j}$. Consequently, for any traditional DL model $\mathcal{I}$ of $\#(\Sigma)$ reduced from M_d, since r_{ij} is one-to-one, $r_{ij}(X^{\mathcal{I}_i})$ will be merged with $X^{\mathcal{I}_i}$. Hence, $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$ in any $\mathcal{I}$, i.e., $\#(\Sigma) \models \#(i : X) \sqsubseteq \#(j : Y)$. Similarly, if $\Sigma \models_d i : X \xrightarrow{\sqsupseteq} j : Y$, we will have $\#(\Sigma) \models \#(i : X) \sqsupseteq \#(j : Y)$.

For any i-concept $i : X$ and j-concept $j : Y$, if $\#(\Sigma) \models \#(i : X) \sqsubseteq \#(j : Y)$, for any classical model $\mathcal{I}$ of $\#(\Sigma)$, we have: $X^{\mathcal{I}} \subseteq Y^{\mathcal{I}}$. For any distributed model M_d that is obtained from $\mathcal{I}$, either $r_{ij}(X^{\mathcal{I}_i}) \subseteq Y^{\mathcal{I}_j}$ or $r_{ji}(Y^{\mathcal{I}_j}) \supseteq X^{\mathcal{I}_i}$, i.e., $\Sigma \models_d (X \xrightarrow{\sqsubseteq} Y$ or $Y \xrightarrow{\sqsupseteq} X)$.

Therefore, if domain relations in DDL are one-to-one, the results of reasoning in DDL are identical to the results of reasoning with the integrated ontology. Q.E.D.

Proof of Theorem 4.7: A $C^{\mathcal{E}}_{\mathcal{HQ}}(\mathcal{SHOIQ})$ knowledge base $\mathcal{T} = \langle\{\mathcal{T}_i\}, \{\mathcal{E}_{ij}\}_{i\neq j}\rangle$ contains a set of local TBoxes $\mathcal{T}_i$, each of which is expressed over a subset of $\mathcal{SHOIQ}$, and a set of one-way $\mathcal{E}$-connections $\mathcal{E}_{ij}$. A reduction from $\mathcal{T}$ to a $\mathcal{SHOIQP}$ knowledge base $\Sigma = \{P_i\}$ can be defined in the following way. For a formula in $\mathcal{T}_i$, we define a translation function $.^{\#}$ (in what following, nominal names are treated as concept names):

$$
\begin{aligned}
A^{\#} &:= A \quad \text{for a concept name A} \\
\top_i^{\#} &:= \top_i' \quad \text{where } \top_i' \text{ is a new concept name} \\
\bot_i^{\#} &:= \bot \\
(C \sqcap D)^{\#} &:= C^{\#} \sqcap D^{\#} \\
(\neg C)^{\#} &:= \top_i' \sqcap \neg_i C^{\#} \\
(\exists R.C)^{\#} &:= \top_i' \sqcap \exists R.C^{\#} \quad \text{where } R \text{ is an } i\text{-role} \\
(\leq nR.C)^{\#} &:= \top_i' \sqcap \leq nR.C^{\#} \quad \text{where } R \text{ is an } i\text{-role} \\
(\exists E.C)^{\#} &:= \top_i' \sqcap \exists E.C^{\#} \quad \text{for } E \in \mathcal{E}_{ij}, \text{ where } C \text{ is a } j\text{-concept name} \\
(\leq nE.C)^{\#} &:= \top_i' \sqcap \leq nE.C^{\#} \quad \text{for } E \in \mathcal{E}_{ij}, \text{ where } C \text{ is a } j\text{-concept name}
\end{aligned}
$$

For simplicity, we do not discuss $\forall$ and $\geq n$ constructors since they can be reduced to other concept constructors.

For each $\mathcal{T}_i$, we define a package P_i that contains

- $\mathsf{Loc}(P_i) = \{i\text{-concept names and } i\text{-role names}\} \cup \{E | E \in \mathcal{E}_{ij}, \forall j\} \cup \{\top_i\} \cup \{\top'_i\} \cup \{\bot\}$;
- $\mathsf{Ext}(P_i) = \{\text{non-}i \text{ concept names appears in } \mathcal{T}_i\} \cup \{\top'_j | \mathcal{E}_{ij} \neq \emptyset, \forall j\}$;
- $C^{\#} \sqcap \top'_i \sqsubseteq D^{\#} \sqcap \top'_i$, for each GCI $C \sqsubseteq D$ in $\mathcal{T}_i$;
- $R \sqsubseteq S$, for each role inclusion $R \sqsubseteq S$ in $\mathcal{T}_i$;
- $E_1 \sqsubseteq E_2$, for each link inclusion $E_1 \sqsubseteq E_2$, where $E_1, E_2 \in \mathcal{E}_{ij}$, for some j;
- For every i-concept name C, add $C \sqsubseteq \top'_i$;
- For every j-concept name C ($i \neq j$) appearing in $\mathcal{T}_i$, add $C \sqsubseteq \top'_j$;
- For every i-role name R, add $domain(R) = range(R) = \top'_i$;
- For every $E \in \mathcal{E}_{ij}$, add $domain(R) = \top'_i$ and $range(R) = \top'_j$;
- For every j such that $\mathcal{E}_{ij} \neq \emptyset$, add $\top'_j \sqcap \top'_i \sqsubseteq \bot$.

Now we prove that a concept $i : C$ is satisfiable in $\mathcal{T}$ iff $C^{\#} \sqcap \top'_i$ is satisfiable in Σ as witnessed by P_i.

If $i : C$ is satisfiable in $\mathcal{T}$, there is a model $\mathcal{J} = \langle \{\mathcal{J}_i\}, \{r_E\}_{E \in \mathcal{E}_{ij}} \rangle$ of $\mathcal{T}$, such that $C^{\mathcal{J}_i} \neq \emptyset$. We construct an interpretation $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^*} \rangle$ of Σ as follows: for every i,

1 $\top'^{\mathcal{I}_i}_i = \Delta^{\mathcal{J}_i}$, $\cdot^{\mathcal{I}_i} \leftarrow \cdot^{\mathcal{J}_i}$;

2 For every $\langle x, y \rangle \in r_E$, where $E \in \mathcal{E}_{ij}$, if $r_{ij}(y) = \emptyset$, add y' in $\Delta^{\mathcal{I}_i}$ and let $\langle y, y' \rangle \in r_{ji}$, otherwise let $y' = r_{ji}(y)$; let $\langle x, y' \rangle \in E^{\mathcal{I}_i}$;

3 For every i, j, add minimal new domain relation instances that satisfies $\rho_{ij} = r_{ij} = r_{kj} \circ r_{ik}$, for every k such that $P_i \in P_k^*$ and $P_k \in P_j^*$;

4 For every j and every j-concept name C in P_i, let $C^{\mathcal{I}_i} = r_{ji}(C^{\mathcal{I}_j})$;

5 Let $\top'^{\mathcal{I}_j}_i = r_{ij}(\Delta^{\mathcal{I}_i})$.

We now prove $\mathcal{I}$ is a model of Σ and $C^{\#} \sqcap \top'_i$. First, we note the following properties of $\mathcal{I}$:

- Compositional consistency is satisfied from the construction of $\mathcal{I}$;
- For every i, j such that $P_i \in P_j^+$, r_{ij} is one-to-one; that is because that 1) for $\{(x, y) \in r_{ij}\}$ added in step 2 above, it must be an one-to-one function, 2) for any $(x, y) \in r_{ij}$ added in step 3 above, it must be the case that $\overline{x} = \overline{y}$ before it was added, hence the one-to-one property still holds.
- For a link $E \in \mathcal{E}_{ij}$, we note that $E^{\mathcal{I}_i} = r_{ji} \circ r_E$, which follows from the construction of $E^{\mathcal{I}_i}$.
- For every i, j such that $P_i \in P_j^+$ and $x \in \Delta^{\mathcal{I}_j}$, $\Delta^{\mathcal{J}_i} \cap r_{ji}(x) = \emptyset$, that is because $r_{ji}(x)$ must be a "new" element that does not exist in $\Delta^{\mathcal{J}_i}$. Hence, for any $S \subseteq \Delta^{\mathcal{I}_j}$, we have $\Delta^{\mathcal{J}_i} \cap r_{ji}(S) = \emptyset$.

Next, we prove $(X^{\#} \sqcap \top_i')^{\mathcal{I}_i} = X^{\mathcal{J}_i}$ for every concept X and every i.
If X is $\top_i$ or $\bot$, $(X^{\#} \sqcap \top_i')^{\mathcal{I}_i} = X^{\mathcal{J}_i}$ trivally.
If X is an i-concept name,

$$\begin{aligned}
(X^{\#} \sqcap \top_i')^{\mathcal{I}_i} &= (X \sqcap \top_i')^{\mathcal{I}_i} \quad \text{(because } X^{\#} = X\text{)} \\
&= X^{\mathcal{I}_i} \quad \text{(because } X^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i} = \top_i'^{\mathcal{I}_i}\text{)} \\
&= X^{\mathcal{J}_i} \quad \text{(from the construction of } \mathcal{I}_i\text{)}
\end{aligned}$$

If X is a j-concept name ($i \neq j$)

$$\begin{aligned}
(X^{\#} \sqcap \top_i')^{\mathcal{I}_i} &= (X \sqcap \top_i')^{\mathcal{I}_i} \quad \text{(because } X^{\#} = X\text{)} \\
&= X^{\mathcal{I}_i} \cap \Delta^{\mathcal{J}_i} \quad \text{(from the construction of } \mathcal{I}_i\text{)} \\
&= \emptyset \quad \text{(because } X^{\mathcal{I}_i} = r_{ji}(X^{\mathcal{I}_j}) \text{ is disjoint from } \Delta^{\mathcal{J}_i}\text{)} \\
&= X^{\mathcal{J}_i} \quad \text{(from the construction of } \mathcal{J}_i\text{)}
\end{aligned}$$

In what follows, we suppose, as induction hypothesis, $(C^{\#} \sqcap \top_i')^{\mathcal{I}_i} = C^{\mathcal{J}_i}$ and $(D^{\#} \sqcap \top_i')^{\mathcal{I}_i} = D^{\mathcal{J}_i}$.

If $X = C \sqcap D$, we have:

$$\begin{aligned}
(X^{\#} \sqcap \top_i')^{\mathcal{I}_i} &= (C^{\#} \sqcap D^{\#} \sqcap \top_i')^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}\text{)} \\
&= (C^{\#} \sqcap \top_i')^{\mathcal{I}_i} \cap (D^{\#} \sqcap \top_i')^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}\text{)} \\
&= C^{\mathcal{J}_i} \cap D^{\mathcal{J}_i} \quad \text{(from induction hypothesis)}
\end{aligned}$$

If $X = \neg C$, we have:

$$\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \neg_i C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= (\Delta^{\mathcal{I}_i} \backslash (C^{\#})^{\mathcal{I}_i}) \cap (\top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= (\top'_i)^{\mathcal{I}_i} \backslash (C^{\#})^{\mathcal{I}_i} \quad \text{(because } (\top'_i)^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= (\top'_i)^{\mathcal{I}_i} \backslash (C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from set theory)} \\
&= \Delta^{\mathcal{J}_i} \backslash C^{\mathcal{J}_i} \quad \text{(from induction hypothesis and the construction of } \mathcal{I}_i) \\
&= (\neg C)^{\mathcal{J}_i} \quad \text{(from the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad \text{(since } X = \neg C)
\end{aligned}$$

If $X = \exists R.C$, we have:

$$\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \exists R.C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= \top'^{\mathcal{I}_i}_i \cap (\exists R.C^{\#})^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \Delta^{\mathcal{J}_i} \cap \{x \in \Delta^{\mathcal{I}_i} | \exists y, (x,y) \in R^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in R^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \quad \text{(since } \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in R^{\mathcal{J}_i} \wedge y \in C^{\mathcal{J}_i}\} \\
&\qquad \text{(from induction hypothesis and } R^{\mathcal{I}_i} = R^{\mathcal{J}_i}) \\
&= (\exists R.C)^{\mathcal{J}_i} \quad \text{(the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad \text{(since } X = \exists R.C)
\end{aligned}$$

If $X =\leq nR.C$, we have:

$$\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \leq nR.C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= \top'^{\mathcal{I}_i}_i \cap (\leq nR.C^{\#})^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \Delta^{\mathcal{J}_i} \cap \{x \in \Delta^{\mathcal{I}_i} | \mathsf{card}\{y | (x,y) \in R^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \leq n\} \\
&\qquad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y | (x,y) \in R^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \leq n\} \quad \text{(since } \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y | (x,y) \in R^{\mathcal{J}_i} \wedge y \in C^{\mathcal{J}_i}\} \leq n\} \\
&\qquad \text{(from induction hypothesis and } R^{\mathcal{I}_i} = R^{\mathcal{J}_i}) \\
&= (\leq nR.C)^{\mathcal{J}_i} \quad \text{(the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad \text{(since } X =\leq nR.C)
\end{aligned}$$

If $X = \exists E.C$ where $E \in \mathcal{E}_{ij}$ and C is a j-concept name, we have:

$$
\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \exists E.C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= \top'^{\mathcal{I}_i}_i \cap (\exists E.C^{\#})^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \Delta^{\mathcal{J}_i} \cap \{x \in \Delta^{\mathcal{I}_i} | \exists y, (x,y) \in E^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in E^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \quad \text{(since } \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in r_{ji} \circ r_E \wedge y \in C^{\mathcal{J}_i}\} \\
&\quad \text{(from induction hypothesis and } E^{\mathcal{I}_i} = r_{ji} \circ r_E) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in r_{ji} \circ r_E \wedge y \in r_{ji}(C^{\mathcal{J}_j})\} \\
&\quad \text{(since } C^{\mathcal{J}_i} = r_{ji}(C^{\mathcal{J}_j})) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, y', y' \in \Delta^{\mathcal{J}_i}, y = r_{ji}(y'), (x,y') \in r_E \wedge y' \in C^{\mathcal{J}_j}\} \\
&\quad \text{(from the definition of } r_{ij}(.)) \\
&= (\exists E.C)^{\mathcal{J}_i} \quad \text{(the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad \text{(since } X = \exists R.C)
\end{aligned}
$$

If $X = \leq nE.C$ where $E \in \mathcal{E}_{ij}$ and C is a j-concept name, we have:

$$
\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \leq nE.C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= \top'^{\mathcal{I}_i}_i \cap (\leq nE.C^{\#})^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \Delta^{\mathcal{J}_i} \cap \{x \in \Delta^{\mathcal{I}_i} | \mathsf{card}\{(x,y) \in E^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \leq n\} \\
&\quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y|(x,y) \in E^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \leq n\} \quad (\text{since } \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y|(x,y) \in r_{ji} \circ r_E \wedge y \in C^{\mathcal{J}_i}\} \leq n\} \\
&\quad \text{(from induction hypothesis and } E^{\mathcal{I}_i} = r_{ji} \circ r_E) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y|(x,y) \in r_{ji} \circ r_E \wedge y \in r_{ji}(C^{\mathcal{J}_j})\} \leq n\} \\
&\quad \text{(since } C^{\mathcal{J}_i} = r_{ji}(C^{\mathcal{J}_j})) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y|\exists y' \in \Delta^{\mathcal{J}_i}, y = r_{ji}(y'), (x,y') \in r_E \wedge y' \in C^{\mathcal{J}_j}\} \leq n\} \\
&\quad \text{(from the definition of } r_{ij}(.)) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y' \in \Delta^{\mathcal{J}_i} | (x,y') \in r_E \wedge y' \in C^{\mathcal{J}_j}\} \leq n\} \\
&\quad \text{(because } r_{ij} \text{ is one-to-one from its construction)} \\
&= (\leq nE.C)^{\mathcal{J}_i} \quad \text{(the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad (\text{since } X = \leq nR.C)
\end{aligned}
$$

Other constructors can be handled similarly.

Next, we return to the proof of that $\mathcal{I}$ is a model of Σ and $C^{\#} \sqcap \top'_i$. That is because for every i, j:

- $(C^{\#} \sqcap \top'_j)^{\mathcal{I}_j} = C^{\mathcal{J}_j} \neq \emptyset$ must be true;
- Condition 1 and 2 in Definition 5.1, i.e., one-to-one and compositional consistency of domain relations, are satisfied by construction as we already shown;
- $r_{ij}(X^{\mathcal{I}_i}) = X^{\mathcal{I}_j}$ (Condition 3 and 6 in Definition 5.1) is true, for every concept or nominal name X;
- Since there is no role importing, Condition 4 and 5 in Definition 5.1 are trivially satisfied.

Next we prove that the Condition 7 in Definition 5.1 holds, i.e., $\mathcal{I}_j$ is a model of P_j, for every j:

- For every concept inclusion $C \sqsubseteq D$ in $\mathcal{T}_j$, since $C^{\mathcal{J}_j} \subseteq D^{\mathcal{J}_j}$, we have $(C^{\#} \sqcap \top'_j)^{\mathcal{I}_j} = C^{\mathcal{J}_j} \subseteq D^{\mathcal{J}_j} = (D^{\#} \sqcap \top'_j)^{\mathcal{I}_j}$;
- For every role inclusion $R \sqsubseteq S$ in $\mathcal{T}_j$, $R^{\mathcal{I}_j} = R^{\mathcal{J}_j} \subseteq S^{\mathcal{J}_j} = S^{\mathcal{I}_j}$;
- For every link inclusion $E_1 \sqsubseteq E_2$, $E_1, E_2 \in \mathcal{E}_{ji}$, we have $r_{E_1} \subseteq r_{E_2}$, and $E_k^{\mathcal{I}_j} = r_{ij} \circ r_{E_k}, k = 1, 2$, hence $E_1^{\mathcal{I}_j} \subseteq E_2^{\mathcal{I}_j}$;
- $X^{\mathcal{I}_j} = X^{\mathcal{J}_j} \subseteq \Delta^{\mathcal{J}_j} = (\top'_j)^{\mathcal{I}_j}$, for every j-concept name X;
- $X^{\mathcal{I}_j} = r_{ij}(X^{\mathcal{I}_i}) \subseteq r_{ij}(\Delta^{\mathcal{I}_i}) = (\top'_i)^{\mathcal{I}_j}$, for every i-concept or nominal name X imported in P_j;
- For every j-role R, $R^{\mathcal{I}_j} = R^{\mathcal{J}_j} \subseteq \Delta^{\mathcal{J}_j} \times \Delta^{\mathcal{J}_j} = (\top'_j)^{\mathcal{I}_j} \times (\top'_j)^{\mathcal{I}_j}$, hence its domain and range restrictions are satisfied;
- For every link $E \in \mathcal{E}_{ij}$, $E^{\mathcal{I}_i} = r_{ji} \circ r_E \subseteq \Delta^{\mathcal{J}_i} \times r_{ji}(\Delta^{\mathcal{J}_j}) = (\top'_i)^{\mathcal{I}_i} \times (\top'_j)^{\mathcal{I}_i}$, hence its domain and range restrictions are satisfied.
- For every j such that $\mathcal{E}_{ij} \neq \emptyset$, $(\top'_j \sqcap \top'_i)^{\mathcal{I}_i} = r_{ji}(\Delta^{\mathcal{I}_i}) \cap \Delta^{\mathcal{J}_i} \subseteq \emptyset$.

On the other direction, if $C^{\#} \sqcap \top'_i$ is satisfiable in Σ as witnessed by P_i, there is a model $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^*} \rangle$ of P_i^* such that $(C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \neq \emptyset$. We construct a model $\mathcal{J} = \langle \{\mathcal{J}_i\}, \{r_E\}_{E \in \mathcal{E}_{ij}} \rangle$ of $\mathcal{T}$, with $C^{\mathcal{J}_i} \neq \emptyset$, in the following way:

- For every i, let $\Delta^{\mathcal{J}_i} = \top_i'^{\mathcal{I}_i}$;
- For every i and for every i-name t occurring in $\mathcal{T}_i$, let $t^{\mathcal{J}_i} = t^{\mathcal{I}_i}$;
- For each $E \in \mathcal{E}_{ij}$, let $r_E = r_{ji}^- \circ E^{\mathcal{I}_i}$; hence $r_{ji} \circ r_E = r_{ji} \circ r_{ji}^- \circ E^{\mathcal{I}_i} = E^{\mathcal{I}_i}$ (since r_{ij} is one-to-one).

Now we prove $\mathcal{J}$ is a model of $\mathcal{T}$ with $C^{\mathcal{J}_i} \neq \emptyset$. We first prove that $(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} = X^{\mathcal{J}_i}$ for every concept X and every i.

If X is $\top_i$ or $\bot$, $(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} = X^{\mathcal{J}_i}$ trivally.

If X is an i-concept name,

$$\begin{aligned}(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (X \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(because } X^{\#} = X\text{)} \\ &= X^{\mathcal{I}_i} \quad \text{(because } X \sqsubseteq \top'_i\text{)} \\ &= X^{\mathcal{J}_i} \quad \text{(from the construction of } \mathcal{J}_i\text{)}\end{aligned}$$

If X is a j-concept name ($i \neq j$)

$$\begin{aligned}(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (X \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(because } X^{\#} = X\text{)} \\ &= \emptyset \quad \text{(because } X \sqsubseteq \top'_j \text{ and } \top'_j \sqcap \top'_i \sqsubseteq \bot\text{)} \\ &= X^{\mathcal{J}_i} \quad \text{(from the construction of } \mathcal{J}_i\text{)}\end{aligned}$$

In what follows, we suppose, as induction hypothesis $(C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} = C^{\mathcal{J}_i}$ and $(D^{\#} \sqcap \top'_i)^{\mathcal{I}_i} = D^{\mathcal{J}_i}$.

If $X = C \sqcap D$, we have:

$$\begin{aligned}(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (C^{\#} \sqcap D^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}\text{)} \\ &= (C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \cap (D^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}\text{)} \\ &= C^{\mathcal{J}_i} \cap D^{\mathcal{J}_i} \quad \text{(from induction hypothesis)}\end{aligned}$$

If $X = \neg C$, we have:

$$\begin{aligned}(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \neg_i C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}\text{)} \\ &= (\Delta^{\mathcal{I}_i} \backslash (C^{\#})^{\mathcal{I}_i}) \cap (\top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}\text{)} \\ &= (\top'_i)^{\mathcal{I}_i} \backslash (C^{\#})^{\mathcal{I}_i} \quad \text{(because } (\top'_i)^{\mathcal{I}_i} \subseteq \Delta^{\mathcal{I}_i}\text{)} \\ &= (\top'_i)^{\mathcal{I}_i} \backslash (C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from set theory)} \\ &= \Delta^{\mathcal{J}_i} \backslash C^{\mathcal{J}_i} \quad \text{(from induction hypothesis and the construction of } \mathcal{J}_i\text{)} \\ &= (\neg C)^{\mathcal{J}_i} \quad \text{(from the definition of } .^{\mathcal{J}_i}\text{)} \\ &= X^{\mathcal{J}_i} \quad \text{(since } X = \neg C\text{)}\end{aligned}$$

If $X = \exists R.C$, we have:

$$\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \exists R.C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= \top'^{\mathcal{I}_i}_i \cap (\exists R.C^{\#})^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \Delta^{\mathcal{J}_i} \cap \{x \in \Delta^{\mathcal{I}_i} | \exists y, (x,y) \in R^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in R^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \quad (\text{since } \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in R^{\mathcal{J}_i} \wedge y \in C^{\mathcal{J}_i}\} \\
&\quad \text{(from induction hypothesis and } R^{\mathcal{I}_i} = R^{\mathcal{J}_i}) \\
&= (\exists R.C)^{\mathcal{J}_i} \quad \text{(the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad (\text{since } X = \exists R.C)
\end{aligned}$$

If $X =\leq nR.C$, we have:

$$\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \leq nR.C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= \top'^{\mathcal{I}_i}_i \cap (\leq nR.C^{\#})^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \Delta^{\mathcal{J}_i} \cap \{x \in \Delta^{\mathcal{I}_i} | \mathsf{card}\{y | (x,y) \in R^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \leq n\} \\
&\quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y | (x,y) \in R^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \leq n\} \quad (\text{since } \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y | (x,y) \in R^{\mathcal{J}_i} \wedge y \in C^{\mathcal{J}_i}\} \leq n\} \\
&\quad \text{(from induction hypothesis and } R^{\mathcal{I}_i} = R^{\mathcal{J}_i}) \\
&= (\leq nR.C)^{\mathcal{J}_i} \quad \text{(the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad (\text{since } X =\leq nR.C)
\end{aligned}$$

If $X = \exists E.C$ where $E \in \mathcal{E}_{ij}$ and C is a j-concept name, we have:

$$
\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \exists E.C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= \top'^{\mathcal{I}_i}_i \cap (\exists E.C^{\#})^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \Delta^{\mathcal{J}_i} \cap \{x \in \Delta^{\mathcal{I}_i} | \exists y, (x,y) \in E^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in E^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \quad (\text{since } \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in r_{ji} \circ r_E \wedge y \in C^{\mathcal{J}_i}\} \\
&\quad \text{(from induction hypothesis and } E^{\mathcal{I}_i} = r_{ji} \circ r_E) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, (x,y) \in r_{ji} \circ r_E \wedge y \in r_{ji}(C^{\mathcal{J}_j})\} \\
&\quad (\text{since } C^{\mathcal{J}_i} = r_{ji}(C^{\mathcal{J}_j})) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \exists y, y', y' \in \Delta^{\mathcal{J}_i}, y = r_{ji}(y'), (x,y') \in r_E \wedge y' \in C^{\mathcal{J}_j}\} \\
&\quad \text{(from the definition of } r_{ij}(.)) \\
&= (\exists E.C)^{\mathcal{J}_i} \quad \text{(the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad (\text{since } X = \exists R.C)
\end{aligned}
$$

If $X =\leq nE.C$ where $E \in \mathcal{E}_{ij}$ and C is a j-concept name, we have:

$$
\begin{aligned}
(X^{\#} \sqcap \top'_i)^{\mathcal{I}_i} &= (\top'_i \sqcap \leq nE.C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\#}) \\
&= \top'^{\mathcal{I}_i}_i \cap (\leq nE.C^{\#})^{\mathcal{I}_i} \quad \text{(from the definition of } .^{\mathcal{I}_i}) \\
&= \Delta^{\mathcal{J}_i} \cap \{x \in \Delta^{\mathcal{I}_i} | \mathsf{card}\{(x,y) \in E^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \leq n\} \\
&\quad \text{(from the definition of } .^{\mathcal{I}_i}\text{)} \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y | (x,y) \in E^{\mathcal{I}_i} \wedge y \in (C^{\#})^{\mathcal{I}_i}\} \leq n\} \quad \text{(since } \Delta^{\mathcal{J}_i} \subseteq \Delta^{\mathcal{I}_i}) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y | (x,y) \in r_{ji} \circ r_E \wedge y \in C^{\mathcal{J}_i}\} \leq n\} \\
&\quad \text{(from induction hypothesis and } E^{\mathcal{I}_i} = r_{ji} \circ r_E) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y | (x,y) \in r_{ji} \circ r_E \wedge y \in r_{ji}(C^{\mathcal{J}_j})\} \leq n\} \\
&\quad \text{(since } C^{\mathcal{J}_i} = r_{ji}(C^{\mathcal{J}_j})) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y | \exists y' \in \Delta^{\mathcal{J}_i}, y = r_{ji}(y'), (x,y') \in r_E \wedge y' \in C^{\mathcal{J}_j}\} \leq n\} \\
&\quad \text{(from the definition of } r_{ij}(.)) \\
&= \{x \in \Delta^{\mathcal{J}_i} | \mathsf{card}\{y' \in \Delta^{\mathcal{J}_i} | (x,y') \in r_E \wedge y' \in C^{\mathcal{J}_j}\} \leq n\} \\
&\quad \text{(because } r_{ij} \text{ is one-to-one)} \\
&= (\leq nE.C)^{\mathcal{J}_i} \quad \text{(the definition of } .^{\mathcal{J}_i}) \\
&= X^{\mathcal{J}_i} \quad \text{(since } X =\leq nR.C)
\end{aligned}
$$

Other constructors can be handled similarly.

Now we prove $\mathcal{J}$ is a model of $\mathcal{T}$ and C.

- $C^{\mathcal{J}_i} = (C^{\#} \sqcap \top'_i)^{\mathcal{I}_i} \neq \emptyset$;
- For every concept inclusion $C \sqsubseteq D$ in $\mathcal{T}_j$, we have $C^{\mathcal{J}_j} = (C^{\#} \sqcap \top'_j)^{\mathcal{I}_j} \subseteq (D^{\#} \sqcap \top'_j)^{\mathcal{I}_j} = D^{\mathcal{J}_j}$;
- For every role inclusion $R \sqsubseteq S$ in $\mathcal{T}_j$, $R^{\mathcal{J}_j} = R^{\mathcal{I}_j} \subseteq S^{\mathcal{I}_j} = S^{\mathcal{J}_j}$;
- For every link inclusion $E_1 \sqsubseteq E_2$, $E_1, E_2 \in \mathcal{E}_{ij}$, we have $E_1^{\mathcal{I}_i} \subseteq E_2^{\mathcal{I}_i}$, $r_{E_k} = r_{ij}^{-} \circ E_k^{\mathcal{I}_i}, k = 1, 2$ and r_{ij} is one-to-one, hence $r_{E_1} \subseteq r_{E_2}$;
- For every i-role R, $R^{\mathcal{J}_j} = R^{\mathcal{I}_j} \subseteq (\top'_j)^{\mathcal{I}_j} \times (\top'_j)^{\mathcal{I}_j} = \Delta^{\mathcal{J}_j} \times \Delta^{\mathcal{J}_j}$, as being required for an i-role;

- For every link $E \in \mathcal{E}_{ij}$,

$$\begin{aligned} r_E &= r_{ji}^{-} \circ E^{\mathcal{I}_i} \\ &\subseteq r_{ji}^{-} \circ \left((\top_i')^{\mathcal{I}_i} \times (\top_j')^{\mathcal{I}_i} \right) \quad \text{(from the domain and range restriction of } E \text{ in } P_i) \\ &\subseteq r_{ji}^{-} \circ \left(\Delta^{\mathcal{J}_i} \times r_{ji}((\top_j')^{\mathcal{I}_j}) \right) \quad \text{(since } \top_j' \text{ is an imported concept)} \\ &\subseteq \Delta^{\mathcal{J}_i} \times \left(r_{ji}^{-} \circ r_{ji}((\top_j')^{\mathcal{I}_j}) \right) \quad \text{(from the definition of function composition)} \\ &\subseteq \Delta^{\mathcal{J}_i} \times (\top_j')^{\mathcal{I}_j} \quad \text{(since } r_{ji} \text{ is one-to-one)} \\ &\subseteq \Delta^{\mathcal{J}_i} \times \Delta^{\mathcal{J}_j} \quad \text{(from the construction of } \mathcal{J}) \end{aligned}$$

  as being required for a link in $\mathcal{E}_{ij}$;

Hence, every axiom in $\mathcal{T}$ is satisfied by $\mathcal{J}$. Hence, Σ can answer any query that might be answered by $\mathcal{T}$. Q.E.D.

A.2 Proofs for Chapter 5

Proof of Lemma 5.1: The statement is obvious if $C = \top_i$ or $C = \bot$. For any model $\mathcal{I}$ of P_j^*, we have:

- if C is a concept name or a local top concept, $(\dot{\neg}_i C)^{\mathcal{I}_j} = (\top_i \sqcap (\neg_j C))^{\mathcal{I}_j} = r_{ij}(\Delta^{\mathcal{I}_i}) \cap \Delta^{\mathcal{I}_j} \backslash C^{\mathcal{I}_j} = r_{ij}(\Delta^{\mathcal{I}_i}) \backslash C^{\mathcal{I}_j} = (\neg_i C)^{\mathcal{I}_j}$;

- if $C = \neg_k D$, then

$$\begin{array}{rcll}
(\dot{\neg}_i C)^{\mathcal{I}_j} & = & (\top_i \sqcap (D \sqcup \neg_i \top_k))^{\mathcal{I}_j} & \text{(by NNF transformation rules)} \\
& = & \top_i^{\mathcal{I}_j} \cap (D^{\mathcal{I}_j} \cup (\neg_i \top_k)^{\mathcal{I}_j}) & \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
& = & r_{ij}(\Delta^{\mathcal{I}_i}) \cap (D^{\mathcal{I}_j} \cup (r_{ij}(\Delta^{\mathcal{I}_i}) \backslash r_{kj}(\Delta^{\mathcal{I}_k}))) & \text{(by the definition of } \mathcal{I}) \\
& = & (r_{ij}(\Delta^{\mathcal{I}_i}) \backslash (r_{kj}(\Delta^{\mathcal{I}_k}) \backslash D^{\mathcal{I}_j})) & \text{(set-theorectically)} \\
& = & (\neg_i (\neg_k D))^{\mathcal{I}_j} & \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
& = & (\neg_i C)^{\mathcal{I}_j}; & \text{(since } C = \neg_k D)
\end{array}$$

- if $C = C_1 \sqcap C_2$, then

$$\begin{array}{rcll}
(\dot{\neg}_i C)^{\mathcal{I}_j} & = & (\neg_i C_1 \sqcup \neg_i C_2)^{\mathcal{I}_j} & \text{(by NNF transformation rules)} \\
& = & (r_{ij}(\Delta^{\mathcal{I}_i}) \backslash C_1^{\mathcal{I}_j}) \cup (r_{ij}(\Delta^{\mathcal{I}_i}) \backslash C_2^{\mathcal{I}_j}) & \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
& = & r_{ij}(\Delta^{\mathcal{I}_i}) \backslash (C_1^{\mathcal{I}_j} \cap C_2^{\mathcal{I}_j}) & \text{(set-theorectically)} \\
& = & (\neg_i (C_1 \sqcap C_2))^{\mathcal{I}_j} & \text{(by the definition of } \cdot^{\mathcal{I}_j}) \\
& = & (\neg_i C)^{\mathcal{I}_j}; & \text{(since } C = C_1 \sqcap C_2)
\end{array}$$

- if $C = C_1 \sqcup C_2$, the proof is similar;

- if $C = \exists R.D$, then

$$\begin{array}{rcl}
(\dot{\neg}_i C)^{\mathcal{I}_j} & = & (\top_i \sqcap \forall R.\neg_j D)^{\mathcal{I}_j} \\
& = & \{x \in r_{ij}(\Delta^{\mathcal{I}_i}) \cap \Delta^{\mathcal{I}_j} | (\forall y \in \Delta^{\mathcal{I}_j})((x,y) \in R^{\mathcal{I}_j} \rightarrow y \in (\neg_j D)^{\mathcal{I}_j})\} \\
& = & \{x \in r_{ij}(\Delta^{\mathcal{I}_i}) | (\forall y \in \Delta^{\mathcal{I}_j})((x,y) \in R^{\mathcal{I}_j} \rightarrow y \notin D^{\mathcal{I}_j})\} \\
& = & r_{ij}(\Delta^{\mathcal{I}_i}) \backslash \{x \in \Delta^{\mathcal{I}_j} | (\exists y \in \Delta^{\mathcal{I}_j})((x,y) \in R^{\mathcal{I}_j} \wedge y \in D^{\mathcal{I}_j})\} \\
& = & (\neg_i (\exists R.D))^{\mathcal{I}_j} \\
& = & (\neg_i C)^{\mathcal{I}_j};
\end{array}$$

- if $C = \forall R.D$, the proof is similar to the previous case.

Hence, $(\dot{\neg}_i C)^{\mathcal{I}_j} = (\neg_i C)^{\mathcal{I}_j}$ holds, for every model $\mathcal{I}$ of P_j^*, whence $\dot{\neg}_i C \equiv_j \neg_i C$. Q.E.D.

Proof of Lemma 5.2: Proof: For the "if" direction, suppose that $\langle \{T_i\}, \{t_{ij}\}_{P_i \in P_j^+} \rangle$, with $T_i = (\mathbf{S}_i, \mathcal{L}_i, \mathcal{E}_i)$, is a tableau for D w.r.t. P_w^*. Then, a model $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^+} \rangle$ of P_w^* may be defined as follows:

$$\begin{aligned} \Delta^{\mathcal{I}_i} &= \mathbf{S}_i; \\ A^{\mathcal{I}_i} &= \{x | A \in \mathcal{L}_i(x)\}, \text{ for every concept name } A; \\ R^{\mathcal{I}_i} &= \mathcal{E}_i(R), \text{ for every } i\text{-role name } R; \\ r_{ij} &= t_{ij}. \end{aligned}$$

By using induction on the structure of concepts, we show that

$$C \in \mathcal{L}_i(x) \quad \text{implies} \quad x \in C^{\mathcal{I}_i}. \tag{A.2}$$

- If C is a concept name, then the statement follows by the definition of $C^{\mathcal{I}_i}$.
- If $C = \neg_i E$, where E is a concept name, then, by Property (A1) of the tableau, $E \notin \mathcal{L}_i(x)$, whence, by the definition of $E^{\mathcal{I}_i}$, $x \notin E^{\mathcal{I}_i}$ and, hence, $x \in \Delta^{\mathcal{I}_i} \backslash E^{\mathcal{I}_i} = C^{\mathcal{I}_i}$.
- If $C = C_1 \sqcap C_2$, then, by Property (A2), $C_1 \in \mathcal{L}_i(x)$ and $C_2 \in \mathcal{L}_i(x)$, whence, by the induction hypothesis, $x \in C_1^{\mathcal{I}_i}$ and $x \in C_2^{\mathcal{I}_i}$ and, therefore, $x \in (C_1 \sqcap C_2)^{\mathcal{I}_i}$.
- The case $C = C_1 \sqcup C_2$ may be handled similarly.
- If $C = \forall R.E$ and $\langle x, y \rangle \in R^{\mathcal{I}_i}$, then $\langle x, y \rangle \in \mathcal{E}_i(R)$ and, by Property (A4), $E \in \mathcal{L}_i(y)$, whence, by the induction hypothesis, $y \in E^{\mathcal{I}_i}$ and, hence, $x \in (\forall R.E)^{\mathcal{I}_i}$.
- If $C = \exists R.E$, then, by Property (A5), there exists $y \in \mathbf{S}_i$, such that $\langle x, y \rangle \in \mathcal{E}_i(R)$ and $E \in \mathcal{L}_i(y)$, whence, by definition, $\langle x, y \rangle \in R^{\mathcal{I}_i}$ and, by the induction hypothesis, $y \in E^{\mathcal{I}_i}$, and, therefore, $x \in (\exists R.E)^{\mathcal{I}_i}$.

Next, using Implication (A.2), it is shown that all $\mathcal{ALCP}_{\mathcal{C}}^{-}$ restrictions on domain relations are satisfied.

- First, $D^{\mathcal{I}_w}$ is not empty, since there exists, by hypothesis, $x \in \mathbf{S}_w$, such that $D \in \mathcal{L}_w(x)$.
- The image domain relations r_{ij} are one-to-one and compositionally consistent by tableau Properties (B1) and (B2).
- For every concept importing $P_i \xrightarrow{C} P_j$, where C is an i-concept name, we have $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$, by Property (B3).
- For every $P_i \in P_w^*$, every axiom $C \sqsubseteq D \in P_i$ and every individual $x \in \mathbf{S}_i$, we have, using tableau Properties (A0) and (A2), that $\neg_i C \sqcup D \in \mathcal{L}_i(x)$. Thus, by Property (A3), either $\neg_i C \in \mathcal{L}_i(x)$ or $D \in \mathcal{L}_i(x)$. Hence, by Implication (A.2), $x \notin C^{\mathcal{I}_i}$ or $x \in D^{\mathcal{I}_i}$, whence $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$, and, therefore, $\mathcal{I}_i \models P_i$.

For the "only if" direction, if $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{P_i \in P_j^+} \rangle$ is a model of P_w^*, with $C^{\mathcal{I}_w} \neq \emptyset$, then a tableau $T = \langle \{T_i\}, \{t_{ij}\}_{P_i \in P_j^+} \rangle$ for P_w^* may be defined as follows:

$$\begin{aligned}
\mathbf{S}_i &= \Delta^{\mathcal{I}_i}; \\
\mathcal{L}_i(x) &= \{C \in sub(C_{\mathcal{T}_i}) | x \in C^{\mathcal{I}_i}\}, x \in \Delta^{\mathcal{I}_i}, i \neq w; \\
\mathcal{L}_w(x) &= \{C \in sub(D) \cup sub(C_{\mathcal{T}_w}) | x \in C^{\mathcal{I}_w}\}, x \in \Delta^{\mathcal{I}_w}; \\
\mathcal{E}_i(R) &= R^{\mathcal{I}_i}; \\
t_{ij} &= r_{ij}.
\end{aligned}$$

We now verify that T is indeed a tableau for D w.r.t. P_w, i.e., that it satisfies all conditions in Definition 5.2:

- **(E)**: Since $C^{\mathcal{I}_w} \neq \emptyset$, there exists $x \in \mathbf{S}_w$, such that $C \in \mathcal{L}_w(x)$.
- **(A0)**: Since $\mathcal{I}_i$ is a model of P_i, we have, for every $x \in \mathbf{S}_i$, $x \in C_{\mathcal{T}_i}^{\mathcal{I}_i}$, whence $C_{\mathcal{T}_i} \in \mathcal{L}_i(x)$.
- **(A1)**: If $C \in \mathcal{L}_i(x)$, then $x \in C^{\mathcal{I}_i}$, whence $x \notin (\neg_i C)^{\mathcal{I}_i} = \Delta^{\mathcal{I}_i} \backslash C^{\mathcal{I}_i}$, and, hence, $\neg_i C \notin \mathcal{L}_i(x)$.
- **(A2)**: If $C_1 \sqcap C_2 \in \mathcal{L}_i(x)$, then $x \in (C_1 \sqcap C_2)^{\mathcal{I}_i} = C_1^{\mathcal{I}_i} \cap C_2^{\mathcal{I}_i}$, hence $C_1 \in \mathcal{L}_i(x)$ and $C_2 \in \mathcal{L}_i(x)$.
- **(A3)**: The proof is similar to the previous one.

- **(A4)**: If $\forall R.C \in \mathcal{L}_i(x)$ and $\langle x, y\rangle \in \mathcal{E}_i(R)$, we have $x \in (\forall R.C)^{\mathcal{I}_i}$ and $\langle x, y\rangle \in R^{\mathcal{I}_i}$, whence, according to the semantics of $\forall R.C$, $y \in C^{\mathcal{I}_i}$ and, hence, $C \in \mathcal{L}_i(y)$.

- **(A5)**: If $\exists R.C \in \mathcal{L}_i(x)$, then there exists $y \in \Delta^{\mathcal{I}_i} = \mathbf{S}_i$, such that $\langle x, y\rangle \in R^{\mathcal{I}_i} = \mathcal{E}_i(R)$ and $y \in C^{\mathcal{I}_i}$, whence $C \in \mathcal{L}_i(y)$.

- **(B1)**: $t_{ij} = r_{ij}$ must be an one-to-one partial function, for all i, j.

- **(B2)**: By the compositional consistency of the r_{ij}, we have, for all i, j, k, $i \neq j$, such that $P_i \in P_k^*$ and $P_k \in P_j^*$, that $\rho_{ij} = r_{ij} = r_{kj} \circ r_{ik}$, whence, by the definition of $\{t_{ij}\}$, we have $\rho_{ij}^t = t_{ij} = t_{kj} \circ t_{ik}$.

- **(B3)**: If C is an i-concept name, $P_i \xrightarrow{C} P_j$, $j \neq i$, then $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$, whence, since $t_{ij} = r_{ij}$, $(\forall x' \in \mathbf{S}_j)((\exists x \in \mathbf{S}_i)(\langle x, x'\rangle \in t_{ij}$ and $C \in \mathcal{L}_i(x))$ iff $C \in \mathcal{L}_j(x'))$.

Q.E.D.

Proof of Lemma 5.3: We start with a set of observations:

- *For every node that has no local predecessor (called* local top node *henceforth), its local descendants have a tree shape.* This observation follows from the form of the expansion rules.

- *For every local top node $j : x$, $j \neq w$, x must be a preimage of a node in another local completion graph G_i, such that $P_j \mapsto P_i$.* This holds because such an x must be created by a backward concept reporting message triggered by an application of the CReport-rule or of the r-rule. Suppose, for the sake of concreteness, that the message is $r^{j \leftarrow i}(x', C)$, $i \neq j$, where $x' \in V_i$, $origin(x) = origin(x')$ and C is $\top_j$ or a j-concept name. Note that, since x does not exist before the message, C is not added to $\mathcal{L}_i(x')$ by a concept reporting message, whence it must be case that C appears in P_i. Thus, P_i imports P_j and, hence, x is a preimage of x'.

- *For any j,x, all local descendants of $j : x$ in G_j are not preimages of nodes in any other local completion graph.* This holds because a local descendant of $j : x$ is generated only by an application of the $\sqcup$-rule, while a preimage node is created only by an application of the CReport-rule or of the r-rule.

Hence, 1) each local completion graph is a forest; 2) the root of every tree, i.e., a local top node, in a local completion graph, except for the root of G_w, is "copied" from, i.e., it is the preimage of, a node in another local completion graph.

Next we prove that the size of each local completion graph, hence also the total size of the "global completion graph", is limited. For convenience, we define a function $f(x) = 2^{2^{x \times \log x}}$.

First, due to subset blocking, for any local top node in G_j, the depth of its local descendant tree is bounded by $O(2^{n_j})$ and its breadth is bounded by the number of "$\exists$" in $C_{\mathcal{T}_j}$, for $j \neq w$, or in $C_{\mathcal{T}_w} \sqcap D$, for $j = w$, which is smaller than n_j. Thus, the size of the tree is bounded by $O({n_j}^{2^{n_j}}) = O(f(n_j))$.

Since there is only acyclic importing, we can put all packages in P_w^* in an ordered list $\mathfrak{L}$, such that $\mathfrak{L}_1 = P_w$ and each package comes in $\mathfrak{L}$ before all packages in its importing transitive closure, in a way similar to topological sorting in DAG. Let $\#(\mathfrak{L}_j)$ be the subscript of the package at $\mathfrak{L}_j$. Then, we have that the size of $G_{\#(\mathfrak{L}_j)}$ is bounded by:

$$\begin{aligned} |G_{\#(\mathfrak{L}_1)}| &: O(f(n_w)) \\ |G_{\#(\mathfrak{L}_j)}| &: O\Big(\sum_{k<j} |G_{\#(\mathfrak{L}_k)}| \times f(n_{\#(\mathfrak{L}_j)})\Big), \text{ for } j > 1 \end{aligned}$$

This holds because there is only one local top node in $G_{\#(\mathfrak{L}_1)} = G_w$ (the original node), and, for every $j > 1$ and $p = \#(\mathfrak{L}_j)$, the number of local top nodes in G_p is limited by $\sum_{P_p \mapsto P_q} |G_q|$, i.e., by the total size of the local completion graphs of packages that directly import P_p, since all nodes in P_p must be preimage nodes of nodes in those local completion graphs. In the worst case, $\{P_q | P_p \mapsto P_q\}$ contains all packages that are before j in $\mathfrak{L}$. On the other hand, the size of a tree under a local top node in G_k is limited by $f(n_k)$.

Setting $|G_{\#(\mathfrak{L}_j)}| = t_j$ and $e_j = f(n_{\#(\mathfrak{L}_j)})$, we obtain that t_j is bounded by

$$O\big((t_1 + t_2 + ... + t_{j-1}) \times e_j\big). \tag{A.3}$$

Using induction, it will now be shown that t_j is bounded by

$$O\big(2^{j-2} \times e_1 \times ... \times e_j\big), \text{ for } j > 1. \tag{A.4}$$

By Equation (A.3), when $j = 2$, t_2 is bounded by $O(t_1 \times e_2) = O(e_1 \times e_2)$, whence Equation (A.4) holds. Let $j > 2$. Assuming, as the induction hypothesis, that, for every $1 < k < j$,

Equation (A.4) holds, we have, by Equation (A.3), that t_j is bounded by

$$\begin{aligned} O\big((t_1 + t_2 + \cdots + t_{j-1}) \times e_j\big) &< O\big((e_1 + 2^0 e_1 e_2 + \cdots + 2^{j-3} e_1 e_2 \cdots e_{j-1}) e_j\big) \\ &< O\big((1 + 2^0 + \cdots + 2^{j-3}) \times e_1 e_2 \cdots e_j\big) \\ &= O\big(2^{j-2} e_1 e_2 \cdots e_j\big) \end{aligned}$$

This finishes the induction step and concludes the proof of Equation (A.4). Hence, the size of all local completion graphs is bounded by:

$$\begin{aligned} O\Big(e_1 + \sum_{2 \le j \le m} \Big(2^{j-2} \prod_{k \le j} e_j\Big)\Big) &\le O\Big(2^{m-1} \times \prod_{P_j \in P_w^*} f(n_j)\Big) \\ &< O\Big(2^m \times \prod_{P_j \in P_w^*} 2^{2^{n_j \times \log n_j}}\Big) \end{aligned}$$

Q.E.D.

Proof of Lemma 5.5: Let $G = \{G_i\}$, with $G_i = (V_i, E_i, \mathcal{L}_i^g)$, be a complete and clash-free distributed completion graph generated by the $\mathcal{ALCP}_{\mathcal{C}}^-$ algorithm. We will obtain a tableau by "unraveling" blocked nodes and tableau relations. For a directly blocked node x, we denote by $bk(x)$ the node that directly blocks x. Thus, we have $\mathcal{L}_i^g(x) \subseteq \mathcal{L}_i^g(bk(x))$. We can define a tableau $T = \langle \{T_i\}, \{t_{ij}\}_{P_i \in P_j^+} \rangle$, with $T_i = (\mathbf{S}_i, \mathcal{L}_i^t, \mathcal{E}_i)$, for D w.r.t. P_w in the following way:

$$\begin{aligned} \mathbf{S}_i &= \{x \in V_i | \text{ neither } x \text{ nor any image or preimage node of } x \text{ is blocked}\}; \\ \mathcal{L}_i^t(x) &= \mathcal{L}_i^g(x); \\ \mathcal{E}_i(R) &= \{\langle x, y \rangle \in V_i \times V_i | \ y \text{ is an R-successor of } x, y \text{ is not blocked}\}; \\ &\quad \cup \{\langle x, bk(y) \rangle \in V_i \times V_i | \ y \text{ is an R-successor of } x, y \text{ is directly blocked}\}; \\ t_{ij} &= \{\langle x, y \rangle \in \mathbf{S}_i \times \mathbf{S}_j | \ origin(x) = origin(y)\}, \text{ for } P_i \in P_j^+. \end{aligned}$$

We show that T satisfies all tableau properties.

- Property (A0) holds due to the CE-rule.
- Property (A1) holds since G is clash-free.
- Properties (A2) and (A3) hold because of the $\sqcap$- and $\sqcup$-rules and the fact that G is complete.

- To show Property (A4), suppose that $\forall R.C \in \mathcal{L}_i^t(x) = \mathcal{L}_i^g(x)$ and $\langle x, y\rangle \in \mathcal{E}_i(R)$. Then it must be the case that either 1) $\langle x, y\rangle \in E_i$, $R \in \mathcal{L}_i^g(\langle x, y\rangle)$, whence, according to the $\forall$-rule, $C \in \mathcal{L}_i^g(y) = \mathcal{L}_i^t(y)$; or 2) there exists a y', such that $y = bk(y')$, whence $\mathcal{L}_i^g(y') \subseteq \mathcal{L}_i^g(y)$, $\langle x, y'\rangle \in E_i$, $R \in \mathcal{L}_i^g(\langle x, y'\rangle)$. Thus, according to the $\forall$-rule and the fact that G is complete, $C \in \mathcal{L}_i^g(y') \subseteq \mathcal{L}_i^g(y) = \mathcal{L}_i^t(y)$. Therefore, in both cases, Property (A4) holds.

- Property (A5) may be shown to hold by a proof dual to that of Property (A4).

- Property (B1) holds because, according to the concept reporting message, for any i, j, a node $i : x$ has at most one node of the same origin in G_j.

- For Property (B2) we have: 1) For any $P_i \in P_j^+$, $(x, y) \in t_{ij}$ iff $origin(x) = origin(y)$, whence $(x, y) \in \rho_{ij}^t$ iff $origin(x) = origin(y)$ and, therefore, $\rho_{ij}^t = t_{ij}$. 2) For all $P_i \in P_k^+$ and $P_k \in P_j^+$, $i \neq j$, if there exist $x \in \mathbf{S}_i$ and $x' \in \mathbf{S}_j$, such that $origin(x) = origin(x')$, then, according to the r-rule and the fact that G is complete, there must also exist an $x'' \in V_k$, such that $origin(x'') = origin(x) = origin(x')$. This x'' cannot be blocked, since it must be a local top node, whence $x'' \in \mathbf{S}_k$. Therefore, it follows that $t_{ij} \subseteq t_{kj} \circ t_{ik}$. On the other hand, $t_{kj} \circ t_{ik} \subseteq t_{ij}$ follows by construction.

- Finally, the "only if" direction of Property (B3) holds because of the CPush-rule and the "if" direction because of the CReport-rule.

Q.E.D.

Proof of Lemma 5.6: Let $T = \langle \{T_i\}, \{t_{ij}\}_{P_i \in P_j^+} \rangle$, with $T_i = (\mathbf{S}_i, \mathcal{L}_i^t, \mathcal{E}_i)$, be a tableau for D w.r.t. P_w. Following Horrocks et al. (1999), we will use T to guide the application of the non-deterministic $\sqcup$-rule in a way that yields a complete and clash-free distributed completion graph $G = \{G_i\}$, with $G_i = (V_i, E_i, \mathcal{L}_i^g)$.

To construct G, we start with a single node x_0 in the local tableau T_w, with $D \in \mathcal{L}_w^t(x_0)$. Such an x_0 exists, since T is a tableau for D w.r.t. P_w. Let $\pi \subseteq \bigcup_i (V_i \times \mathbf{S}_i)$ be a function that maps all individuals in local completion graphs to individuals in corresponding local tableaux. Initially, we have $V_w = \{x_0\}$, $\mathcal{L}_w^g(x_0) = \{D\}$, $\pi(x_0) = x_0$ and all G_i, $i \neq w$, being empty. Next, we apply $\mathcal{ALCP}_{\mathcal{C}}^-$ expansion rules to extend G and π, in such a way that the following

conditions always (inductively) hold:

$$\begin{cases} \mathcal{L}_i^g(x) \subseteq \mathcal{L}_i^t(\pi(x)) \\ \text{if } R \in \mathcal{L}_i^g(\langle x, y\rangle), \text{ then } \langle \pi(x), \pi(y)\rangle \in \mathcal{E}_i(R) \\ \text{if } \mathit{origin}(i:x) = \mathit{origin}(j:y) \text{ in } G, \text{ then } \langle \pi(x), \pi(y)\rangle \in t_{ij} \text{ in } T, \text{ for } P_i \in P_j^+ \end{cases} \tag{A.5}$$

- CE-rule: **if** $C_{\mathcal{T}_i} \notin \mathcal{L}_i^g(x)$, **then** $\mathcal{L}_i^g(x)$+= $\{C_{\mathcal{T}_i}\}$. Since, by Property (A0), $C_{\mathcal{T}_i} \in \mathcal{L}_i^t(\pi(x))$, this rule can be applied without violating Conditions (A.5).

- $\sqcap$-rule: **if** $C_1 \sqcap C_2 \in \mathcal{L}_i^g(x)$, x is not blocked, and $\{C_1, C_2\} \not\subseteq \mathcal{L}_i^g(x)$, **then** $\mathcal{L}_i^g(x)$+= $\{C_1, C_2\}$. Since, by Property (A2) of $\mathcal{ALCP}_{\mathcal{C}}^-$ tableaux, $C_1 \sqcap C_2 \in \mathcal{L}_i^t(\pi(x))$ implies $C_1 \in \mathcal{L}_i^t(\pi(x))$ and $C_2 \in \mathcal{L}_i^t(\pi(x))$, Conditions (A.5) are not violated.

- $\sqcup$-rule: **if** $C_1 \sqcup C_2 \in \mathcal{L}_i^g(x)$, x is not blocked, and $\{C_1, C_2\} \cap \mathcal{L}_i^g(x) = \emptyset$, **then** $\mathcal{L}_i^g(x)$+= $\{C\}$, for some $C \in \{C_1, C_2\} \cap \mathcal{L}_i^t(\pi(x))$. Such a C must exist because T is a tableau and, hence, satisfies Property (A3), and $C_1 \sqcup C_2 \in \mathcal{L}_i^t(\pi(x))$, by the induction hypothesis. Hence, in this case, Conditions (A.5) are not violated either.

- $\forall$-rule: **if** $\forall R.C \in \mathcal{L}_i^g(x)$, x is not blocked, and there is a local R-successor y of x in G_i with $C \notin \mathcal{L}_i^g(y)$, **then** $\mathcal{L}_i^g(y)$+= $\{C\}$. By the induction hypothesis, $\langle \pi(x), \pi(y)\rangle \in \mathcal{E}_i(R)$ and $\forall R.C \in \mathcal{L}_i^t(\pi(x))$, whence, by Property (A4), $C \in \mathcal{L}_i^t(\pi(y))$. Thus, Conditions (A.5) are not violated.

- $\exists$-rule: **if** $\exists R.C \in \mathcal{L}_i^g(x)$, x is not blocked, and x has no local R-successor y of x in G_i, with $C \in \mathcal{L}_i^g(y)$, **then** 1) create a new node y, with $\mathit{orgin}(y) = y$, $\mathcal{L}_i^g(\langle x, y\rangle) = \{R\}$ and $\mathcal{L}_i^g(y) = \{C\}$; 2) let $\pi(y) = y'$, where $y' \in \mathbf{S}_i$, $\langle \pi(x), y'\rangle \in \mathcal{E}_i(R)$ and $C \in \mathcal{L}_i^t(y')$. Such a y' must exist because T is a tableau and, hence, it satisfies Property (A5) and, by the induction hypothesis, $\exists R.C \in \mathcal{L}_i^t(\pi(x))$. Therefore, Conditions (A.5) are not violated.

- r-rule: **if** $\mathit{origin}(i:x) = \mathit{origin}(j:x')$, there exists k such that $P_i \in P_k^+$, $P_k \in P_j^+$ and there is no $k : x''$ with $\mathit{origin}(j:x') = \mathit{origin}(k:x'')$, **then** 1) transmit $r^{k \leftarrow j}(x', \top_k)$. This will create $k : x''$, such that $\mathit{origin}(k:x'') = \mathit{origin}(j:x') = \mathit{origin}(i:x)$; 2) let $\pi(x'') = z$, where $z \in \mathbf{S}_k$, $\langle \pi(x'), z\rangle \in t_{ik}$ and $\langle z, \pi(x')\rangle \in t_{kj}$; such a z must exist because, by the induction hypothesis, $\langle \pi(x), \pi(x')\rangle \in t_{ij}$ and, by the tableau Property (B2), $t_{ij} = t_{kj} \circ t_{ik}$. After this operation $\mathcal{L}_k^g(x'') = \emptyset$. Therefore, Conditions (A.5) are not violated.

- CPush-rule: **if** $C \in \mathcal{L}_i^g(x)$, where C is an i-concept name, $P_i \xrightarrow{C} P_j$, x is not blocked and there exists an $x' = x^{i \to j} \in V_j$, such that $C \notin \mathcal{L}_j^g(x')$, **then** transmit $r^{i \to j}(x, C)$. This will set $\mathcal{L}_j^g(x') \mathrel{+}= \{C\}$. By the induction hypothesis, $\langle \pi(x), \pi(x') \rangle \in t_{ij}$, $C \in \mathcal{L}_i^t(\pi(x))$, whence, by Property (B3), $C \in \mathcal{L}_j^t(\pi(x'))$. Hence, Conditions (A.5) are not violated.

- CReport-rule: **if** $C \in \mathcal{L}_i^g(x)$, where C is $\top_j$ or a j-concept name, x is not blocked and there is no $x' = x^{j \leftarrow i} \in V_j$ such that $C \in \mathcal{L}_j^g(x')$, **then** 1) transmit $r^{j \leftarrow i}(x, C)$. This will create $x' = x^{j \leftarrow i}$, with $origin(x') = origin(x)$, if such an x' had not already been created, and set $\mathcal{L}_j^g(x') \mathrel{+}= \{C\}$; 2) let $\pi(x') = x''$, if $\pi(x')$ has not yet been given, where $x'' \in \mathbf{S}_j$, $\langle x'', \pi(x) \rangle \in t_{ji}$ and $C \in \mathcal{L}_i^t(x'')$. Such a, x'' must exist because, by the induction hypothesis, $C \in \mathcal{L}_i^t(\pi(x))$ and T satisfies tableau Property (B3). Therefore, Conditions (A.5) are not violated in this case either.

G must be clash-free, since, if there existed i, x, C, such that $\{C, \neg_i C\} \subseteq \mathcal{L}_i^g(x)$, then, by Conditions (A.5), $\{C, \neg_i C\} \subseteq \mathcal{L}_i^t(\pi(x))$, which would contradict tableau Property (A1) for T. Hence, whenever an expansion rule is applicable to G, it can be applied in such a way that maintains Conditions (A.5). By the Termination Lemma, any sequence of rule applications must terminate. Hence, we will obtain a complete and clash-free completion graph G for D from T. Q.E.D.

Proof of Lemma 5.9: For the if direction, if $\langle \{T_i\}, \{t_{ij}\}_{i \neq j} \rangle$ (where $T_i = \langle V_i, E_i, \mathcal{L}_i \rangle$) is a tableau for D w.r.t. P_w, a model $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{i \neq j} \rangle$ of D and P_w^* can be defined as :

$$
\begin{aligned}
\Delta^{\mathcal{I}_i} &= \mathbf{S}_i \\
A^{\mathcal{I}_i} &= \{x | A \in \mathcal{L}_i(x)\}, \text{for all concept name } A \\
R^{\mathcal{I}_i} &= \mathcal{E}_i(R)^+, \text{for all } i\text{-role name } R \text{ that is transitive} \\
R^{\mathcal{I}_i} &= \mathcal{E}_i(R) \cup \bigcup_{S \sqsubseteq R, S \neq R} S^{\mathcal{I}_i}, \text{for all } i\text{-role name } R \text{ that is not transitive} \\
r_{ij} &= t_{ij}
\end{aligned}
$$

By the induction on the structure of concepts, we show that, if $C \in \mathcal{L}_i(x)$, then $x \in C^{\mathcal{I}_i}$ and all $\mathcal{SHIQP}$ restrictions on domain relations are satisfied.

- If C is an i-concept name, then $x \in C^{\mathcal{I}_i}$ by definition.

- The $C = \neg_j E$ case is similar to the proof of Lemma 5.2.
- Similar to the proof for $\mathcal{SHIQ}$ (Tobies, 2001), we can prove if $C \in \mathcal{L}_i(x)$, then $x \in C^{\mathcal{I}_i}$ for other cases that using $\sqcap, \sqcup, \exists, \forall, \leq, \geq$ constructs .
- r_{ij} is one-to-one and compositional consistent by tableau property B1 and B2.
- For every concept importing $P_i \xrightarrow{C} P_j$ where C is an i-concept name, due to tableau property B3, we have $r_{ij}(C^{\mathcal{I}_i}) = C^{\mathcal{I}_j}$.
- Similar to the concept importing case, for every role importing $P_i \xrightarrow{R} P_j$ where R is an i-role name, we have $r_{ij}(R^{\mathcal{I}_i}) = R^{\mathcal{I}_j}$ due to tableau property B5.
- Cardinality Preservation: For every i-role name R that appears in P_j and every $(x, x') \in r_{ij}$, if $(x', r_{ij}(y)) \in R^{\mathcal{I}_j} = \mathcal{E}_j(R)$, then by Property (B4) we have $(x, y) \in R^{\mathcal{I}_i} = \mathcal{E}_i(R)$; on the other hand, if $(x, y) \in R^{\mathcal{I}_i} = \mathcal{E}_i(R)$, by Property (B5) we have $(x', r_{ij}(y)) \in R^{\mathcal{I}_j} = \mathcal{E}_j(R)$.
- For every $P_i \in P_w^*$, every axiom $C \sqsubseteq D \in \mathcal{T}_i$, for every individual $x \in \mathbf{S}_i$, we have $\neg_i C \sqcup D \in \mathcal{L}_i(x)$; by induction, either $x \notin C^{\mathcal{I}_i}$ or $x \in D^{\mathcal{I}_i}$, hence $C^{\mathcal{I}_i} \subseteq D^{\mathcal{I}_i}$, therefore $\mathcal{I}_i \models \mathcal{T}_i$.
- $\mathcal{I}_i \models \mathcal{R}_i$ due to tableau property A8.
- $D^{\mathcal{I}_w}$ is not empty, since the root node x_0 in T_w has the label D.

For the only if direction, if $\mathcal{I} = \langle \{\mathcal{I}_i\}, \{r_{ij}\}_{i \neq j} \rangle$ is a model of P_w^*, then a distributed tableau $\langle \{T_i\}, \{t_{ij}\}_{i \neq j} \rangle$ for P_w^* can be defined as:

$$\begin{aligned}
\mathbf{S}_i &= \Delta^{\mathcal{I}_i} \\
\mathcal{L}_i(x) &= \{C \in clos_i(C_{\mathcal{T}_i}) | x \in C^{\mathcal{I}_i}\}, i \neq w \\
\mathcal{L}_w(x) &= \{C \in clos_w(D) \cup clos_w(C_{\mathcal{T}_w}) | x \in C^{\mathcal{I}_w}\} \\
\mathcal{E}_i(R) &= R^{\mathcal{I}_i} \\
t_{ij} &= r_{ij}
\end{aligned}$$

Similar to the $\mathcal{SHIQ}$ proof (Tobies, 2001), the tableau holds property A0-A11. Property B1-B5 follow from the semantics of $\mathcal{SHIQP}$. Q.E.D.

Proof of Theorem 5.3 Because each thread can be handled separately, in what follows we only discuss for one thread.

Termination: we prove termination by showing that the "combined" completion graph G', resulting from the various local completion graphs by merging all nodes of the same origin into one node (and the corresponding edges), is finite.

For a local completion graph G_i, let $m_i = \#(X)$, where X is $(clos_w(D) \cup clos_w(C_{\mathcal{T}_w}))$ for G_w or $clos_w(C_{\mathcal{T}_w})$ for $i \neq w$, $k_i = \#\overline{\mathsf{NR}}_i$, and b_i the maximum n that occurs in a concept of the form $\leq nS.C$ or $\geq nS.C$ in X.

Let and $n_i = |C_{\mathcal{T}_i}| + |\mathcal{R}_i|$ (for $i \neq w$) and $n_w = |C_{\mathcal{T}_w}| + |\mathcal{R}_w| + |D|$. Let $n_\Sigma = \sum\limits_j n_j$. Let $l = |P_w^*|$ be the number of packages involved. We have $m_i = O(n_i^2)$, $k_i = O(n_i)$, and $b_i = O(2^{n_i})$.

For any node in G_j, the number of its children that generated by the application of the $\exists$-rule is bounded by the number of "$\exists$" in $C_{\mathcal{T}_j}$, for $j \neq w$, or in $C_{\mathcal{T}_w} \sqcap D$, for $j = w$, which is smaller than $m_j b_j$ (Detailed analysis is similar to the $\mathcal{SHIQ}$ tableau algorithm (Tobies, 2001)). In the combined graph G', the total number of children of a node can be, in worst case, as many as $\sum\limits_j m_j b_j$.

In the combined graph G', the size of labels of a node is limited by $\sum\limits_j m_j k_j$; thus it is blocked at least at the depth $2^{2\sum\limits_j m_j k_j}$ due to double blocking. Hence, the number of nodes in G' is bounded by

$$\begin{aligned}
& O\left((\sum_j m_j b_j)^{2^{2\sum_j m_j k_j}}\right) \\
= \; & O\left((\sum_j n_j^2 \cdot 2^{n_j})^{2^{\sum_i (2n_i^3)}}\right) \\
\leq \; & O\left((\sum_j 2^{2n_j})^{2^{2n_\Sigma^3}}\right) \\
\leq \; & O\left((2^{2n_\Sigma})^{2^{2n_\Sigma^3}}\right) \\
\leq \; & O\left(2^{((2^{2n_\Sigma^3}) \cdot 2n_\Sigma)}\right)
\end{aligned}$$

Hence, the $\mathcal{SHIQP}^-$ algorithm runs in worst case 2NExpTime w.r.t. the size of the concept D plus the total size of all packages.

Soundness: If the $\mathcal{SHIQP}$ algorithm generates a complete and clash-free distributed completion graph for a concept D w.r.t. a witness package P_w, then D has a tableau w.r.t. P_w.

Let $G = \langle \{G_i\}, \{\gamma_{ij}\}_{i \neq j} \rangle$ be a complete and clash-free distributed completion graph, we can obtain a tableau by "unraveling" blocked nodes. The construction extends a similar process for $\mathcal{SHIQ}$ (Lemma 6.38 of (Tobies, 2001)). The basic idea is that, since $\mathcal{SHIQP}$ does not have the finite model property ($\mathcal{SHIQ}$ already has no finite model property), the constructed tableau may also be infinite. Hence, instead of directly translating nodes in the completion graph into individuals in the tableau as we did for $\mathcal{ALCP}_{\mathcal{C}}$, we will construct (possibly infinitely long) paths from the graph as individuals in the tableau, and each blocked nodes will be translated into infinitely many such paths. The details go as the follows.

A path is a sequence of pairs of nodes of the form $p = [\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}]$. For such a path p, we define $\mathsf{Tail}(p) = x_n$ and $\mathsf{Tail}'(p) = x'_n$. We denote by $[p|\frac{x_{n+1}}{x'_{n+1}}]$ the path $[\frac{x_0}{x'_0}, \ldots, \frac{x_n}{x'_n}, \frac{x_{n+1}}{x'_{n+1}}]$. We denote by $\mathsf{rank}(p)$ as the *rank* of p, which may be countably infinite, that will be assigned as the following.

For a local tableau T_i, the set of all paths of it is denoted as $\mathsf{Path}(T_i)$. $\{\mathsf{Path}(T_i)\}$ are inductively defined as:

- When $i = w$, let x be the root node of T_i, $p = [\frac{x}{x}] \in \mathsf{Path}(T_i)$ and $rank(p) = 1$;
- If $p \in \mathsf{Path}(T_i)$, then
  - if y is a successor of $\mathsf{Tail}(p)$ and y is not blocked, then $q = [p|\frac{y}{y}] \in \mathsf{Path}(T_i)$ and $\mathsf{rank}(q) = \mathsf{rank}(p)$;
  - if y is a successor of $\mathsf{Tail}(p)$ and y is blocked by z, then $q = [p|\frac{z}{y}] \in \mathsf{Path}(T_i)$ and $\mathsf{rank}(q) = \mathsf{rank}(p) + 1$;
  - if y is a local top node in G_j $(j \neq i)$ such that $origin(y) = origin(\mathsf{Tail}(p))$, and there is no path $r \in \mathsf{Path}(T_j)$ such that $\mathsf{rank}(r) = \mathsf{rank}(p)$ and $origin(\mathsf{Tail}(r)) = origin(\mathsf{Tail}(p))$, then $q = [p|\frac{y}{y}] \in \mathsf{Path}(T_j)$ and $\mathsf{rank}(q) = \mathsf{rank}(p)$.

Note that, for any i, $\mathsf{Path}(T_i)$ may be an infinite set. We now define a tableau $T = \langle\{T_i\}, \{t_{ij}\}\rangle$ (where $T_i = (\mathbf{S}_i, \mathcal{L}_i, \mathcal{E}_i)$) for D w.r.t. P_w^* in the following way:

$$
\begin{aligned}
\mathbf{S}_i &= \mathsf{Path}(T_i) \\
\mathcal{L}_i(p) &= \mathcal{L}_i(\mathsf{Tail}(p)) \\
\mathcal{E}_i(R) &= \{(p,q) \in \mathbf{S}_i \times \mathbf{S}_i | \text{ either } q = [p|\frac{x}{x'}] \text{ and } x' \text{ is an } R\text{-neighbor of } \mathsf{Tail(p)} \\
&\quad \text{or } p = [q|\frac{x}{x'}] \text{ and } x' \text{ is an } R^-\text{-neighbor of } \mathsf{Tail(q)}\} \\
t_{ij} &= \{(p,q) \in \mathbf{S}_i \times \mathbf{S}_j | \mathsf{rank}(p) = \mathsf{rank}(q), \mathit{origin}(\mathsf{Tail}(p)) = \mathit{origin}(\mathsf{Tail}(q))\}, \text{for } P_i \in P_j^+
\end{aligned}
$$

We show that T satisfies all tableau properties. A0-A11 hold similarly as for the analysis of $\mathcal{SHIQ}$ (Tobies, 2001). We focus on properties B1-B5.

- Property (B1) holds because for each path $p \in \mathsf{Path}(T_i)$ there is at most one path $q \in \mathsf{Path}(T_j)$ such that $\mathsf{rank(p)}$=$\mathsf{rank(q)}$, and, according to the concept reporting message, the node $\mathsf{Tail}(p)$ has at most one node of the same origin in G_j.

- For Property (B2) we have: 1) $\rho_{ij}^t = t_{ij}$ because "=" is an equivalency relation over path ranks and node origins. 2) For all $P_i \in P_k^+$ and $P_k \in P_j^+$, $i \neq j$, if there exist $p \in \mathbf{S}_i$ and $q \in \mathbf{S}_j$, such that $\mathsf{rank}(p) = \mathsf{rank}(q)$ and $\mathit{origin}(\mathsf{Tail}(p)) = \mathit{origin}(\mathsf{Tail}(q))$, then, according to the r-rule and the fact that G is complete, there must also exist an $x \in V_k$, such that $\mathit{origin}(\mathsf{Tail}(p)) = \mathit{origin}(x) = \mathit{origin}(\mathsf{Tail}(q))$. By the construction of the tableau, there must be a path $r \in \mathbf{S}_k$ such that $\mathsf{Tail}(r) = x$ and $\mathsf{rank}(r) = \mathsf{rank}(q)$. Therefore, it follows that $t_{ij} \subseteq t_{kj} \circ t_{ik}$. On the other hand, $t_{kj} \circ t_{ik} \subseteq t_{ij}$ follows by construction.

- The "only if" direction of Property (B3) holds because of the CPush-rule and the "if" direction because of the CReport-rule. Let C be an i-concept name, $P_i \xrightarrow{C} P_j, i \neq j$,

  - $\forall q \in \mathbf{S}_j$, if $\exists p \in \mathbf{S}_i$ such that $\langle p, q\rangle \in t_{ij}$ and $C \in \mathcal{L}_i(p))$, then let $x = \mathsf{Tail}(p)$, $x' = \mathsf{Tail}(q)$, we have that $C \in \mathcal{L}_i(x))$ and $\mathit{origin}(x) = \mathit{origin}(x')$, hence by the CPush-rule $C \in \mathcal{L}_j(x'))$, therefore $C \in \mathcal{L}_j(q))$.
  - $\forall q \in \mathbf{S}_j$, if $C \in \mathcal{L}_j(q))$, then let $x' = \mathsf{Tail}(q)$, we have that $C \in \mathcal{L}_j(x'))$; by the CReport-rule, there must be a $x \in V_i$ such that $C \in \mathcal{L}_i(x))$ and $\mathit{origin}(x) = \mathit{origin}(x')$. Hence, there must exist a $p \in \mathbf{S}_i$ with $x = \mathsf{Tail}(p)$, such that $\langle p, q\rangle \in t_{ij}$ and $C \in \mathcal{L}_i(p))$.

- B4 holds because of the RPush- rule (for the only if direction) and RReport-rule (for the if direction). Detailed proof is similar to the Property B3.

- B5 holds because of the RPush-rule. If $\langle p, q\rangle \in \mathcal{E}_i(R)$, R is an i-role that appears in P_j ($i \neq j$), and there are some $p' \in \mathbf{S}_j$ such that $\langle p, p'\rangle \in t_{ij}$, then $\mathsf{rank}(p) = \mathsf{rank}(p')$; let $x = \mathsf{Tail}(p)$, $x' = \mathsf{Tail}(p')$, and there are two possibilities:

  - $q = [p|\frac{z}{y}]$ and y is an R-neighbor of x; since $origin(x) = origin(x')$, then by the RPush-rule, there is a $y' \in V_j$ such that $origin(y) = origin(y')$ and y' is an R-neighbor of x'. Thus, there is some $q' = [p'|\frac{z'}{y'}] \in \mathbf{S}_j$ such that $\langle x', y'\rangle \in \mathcal{E}_j(R)$; also note that since z blocks y iff z' blocks y' due to double blocking, we also have that $\mathsf{rank}(q') = \mathsf{rank}(p')$.
  - $p = [q|\frac{x}{y}]$ and y is an R^--neighbor of $z = \mathsf{Tail}(q)$; then by the RPush-rule, there is a $z' \in V_j$ such that $origin(z) = origin(z')$ and z' is an R^--neighbor of x'; according to the construction of the tableau, there must be a $q' \in \mathsf{Path}(T_j)$ such that $\mathsf{Tail}(q') = z'$ and $\mathsf{rank}(q') = \mathsf{rank}(q)$ (due to double blocking); hence $\langle p', q'\rangle \in t_{ij}$ and $\langle p', q'\rangle \in \mathcal{E}_j(R)$.

  In both cases, there is some $q' \in \mathbf{S}_j$ such that $\langle q, q'\rangle \in t_{ij}$ and $\langle p', q'\rangle \in \mathcal{E}_j(R)$. Hence Property (B5) holds.

Completeness (proof sketch): If the concept D has a tableau w.r.t. the witness package P_w, then the $\mathcal{SHIQP}$ algorithm can generate a complete and clash-free distributed completion graph for D w.r.t.P_w.

Let $T = \langle\{T_i\}, \{\gamma_{ij}\}_{i\neq j}\rangle$ be a tableau for D w.r.t.P_w. We can construct T by triggering $\mathcal{SHIQP}$ rules. We start with a single node x_0 in the local tableau T_w with $\mathcal{L}_k(x_0) = \{C_{\mathcal{T}_w}\}$. Similar to (Horrocks et al., 1999), we can use T to guide the application of the non-deterministic $\sqcup, choose, \leq$-rule in a way that yields a complete and clash-free distributed graph.

Bibliography

Adjiman, P., Chatalic, P., Goasdou, F., Rousset, M.-C., and Simon, L. (2006). Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web . *Journal of Artificial Intelligence Research*, 25:269,314. 5.6

Alani, H., Harris, S., and O'Neil, B. (2006). Winnowing ontologies based on application use. In *ESWC*, pages 185–199. 3.1.2.2

Amir, E. and McIlraith, S. A. (2000). Partition-based logical reasoning. In *KR*, pages 389–400. 3.4, 4.4.1.1, 5.6

Amir, E. and McIlraith, S. A. (2005). Partition-based logical reasoning for first-order and propositional theories. *Artif. Intell.*, 162(1-2):49–88. 3.1.1.1

Antoniou, G. (1992). Modularity for logical knowledge bases. In *Fourth International Conference on Software Engineering and Knowledge Engineering*, pages 87–93. 3.4

Antoniou, G. and Sperschneider, V. (1992). Modularity for logic programs. In *ALPUK*, pages 97–107. 3.4

Auer, S., Dietzold, S., and Riechert, T. (2006). Ontowiki - a tool for social, semantic collaboration. In Cruz et al. (2006), pages 736–749. 7.5

Avery, J. and Yearwood, J. (2003). DOWL: A dynamic ontology language. In *ICWI*, pages 985–988. 4.3.1, 4.4.5.1

Baader, F., Brandt, S., and Lutz, C. (2005). Pushing the EL envelope. In *IJCAI*, pages 364–369. 8.2.3

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. 2

Baader, F. and Ghilardi, S. (2005). Connecting many-sorted theories. In *CADE*, pages 278–294. 8.2.1

Baader, F., Lutz, C., Sturm, H., and Wolter, F. (2000). Fusions of description logics. In *Description Logics*, pages 21–30. 3.2.1.5, 3.3.3.5, 4.4.1.4, 4.4.2

Baader, F. and Nutt, W. (2003). Basic description logics. In Baader, F., Calvanese, D., and et.al., D. M., editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press. 2.1.1

Baader, F. and Sattler, U. (2001). An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40. 2.2.2, 2.2.2

Backhaus, M., Kelso, J., Bacher, J., Herre, H., Hoehndorf, R., Loebe, F., and Visagie, J. (2007). Bowiki - a collaborative annotation and ontology curation framework. In *Workshop on Social and Collaborative Construction of Structured Knowledge at 16th International World Wide Web Conference (WWW2007)*. 7.5

Bao, J., Cao, Y., Tavanapong, W., and Honavar, V. (2004). Integration of domain-specific and domain-independent ontologies for colonoscopy video database annotation. In Arabnia, H. R., editor, *Proceeding of International Conference on Information and Knowledge Engineering (IKE 04)*, pages 82–88. CSREA Press. 8.2.4

Bao, J., Caragea, D., and Honavar, V. (2006a). A distributed tableau algorithm for package-based description logics. In *the 2nd International Workshop On Context Representation And Reasoning (CRR 2006), co-located with ECAI 2006*. 1.2.1, 4.2.1.2, 5, 8.1

Bao, J., Caragea, D., and Honavar, V. (2006b). Modular ontologies - a formal investigation of semantics and expressivity. In *Asian Semantic Web Conference (ASWC)*, pages 616–631. 3.2.1, 3.3.2, 4.1, 4.2.6.3, 4.4.5.3, 8.1

Bao, J., Caragea, D., and Honavar, V. (2006c). On the semantics of linking and importing in modular ontologies. In *International Semantic Web Conference (ISWC)*, pages 72–86. 1.2, 3.3.2, 4.1, 4.2.6.3, 4.4.2.3, 4.4.4.2, 8.1

Bao, J., Caragea, D., and Honavar, V. (2006d). Package-based description logics - preliminary results. In Cruz et al. (2006), pages 967–969. 8.1

Bao, J., Caragea, D., and Honavar, V. (2006e). A tableau-based federated reasoning algorithm for modular ontologies. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 404–410. IEEE Press. 1.2.1, 4.2.1.2, 4.2.2.1, 4.2.6.4, 5, 5.4, 8.1

Bao, J., Caragea, D., and Honavar, V. (2006f). Towards collaborative environments for ontology construction and sharing. In *International Symposium on Collaborative Technologies and Systems (CTS 2006)*, pages 99–108. IEEE Press. 1.2, 1.2.1, 1.2.2, 4.1, 6.3.1, 7.1.1, 7.3.1, 8.1, 8.2.2

Bao, J., Caragea, D., and Honavar, V. (2007a). Query translation for ontology-extended data sources. In *AAAI'07 Workshop on Semantic e-Science (SeS'07)*. 1.2.2, 8.2.5

Bao, J. and Honavar, V. (2004a). Collaborative ontology building with wiki@nt - a multi-agent based ontology building environment. In *ISWC 2004 Workshop on Evaluation of Ontology-based Tools (EON)*, pages 37–46. 1.2.2, 7, 8.1

Bao, J. and Honavar, V. (2004b). Ontology language extensions to support localized semantics, modular reasoning, and collaborative ontology design and ontology reuse. Technical report, TR-341, Computer Science, Iowa State University. 8.1

Bao, J. and Honavar, V. (2004c). Ontology language extensions to support localized semantics, modular reasoning, collaborative ontology design and reuse. In *3rd International Semantic Web Conference (ISWC2004), Poster Track*. 4.3.4

Bao, J. and Honavar, V. (2005a). Collaborative package-based ontology building and usage. In *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources, ICDM 2005*, pages 35–44. 3.1.2.3, 7

Bao, J. and Honavar, V. (2005b). Reconciling inconsistencies between package-extended ontology modules. Technical report, TR-403, Computer Science, Iowa State University. 8.2.3

Bao, J. and Honavar, V. (2006a). Adapt OWL as a modular ontology language. In *OWL: Experiences and Directions (OWLED 2006), CEUR Workshops Vol. 216*. 4.1, 8.1

Bao, J. and Honavar, V. (2006b). Divide and conquer semantic web with modular ontologies - a brief review of modular ontology language proposals. In *ISWC 2006 Workshop on Modular Ontologies (WoMo 2006)*. 4.1, 8.1

Bao, J. and Honavar, V. (2006c). Representing and reasoning with modular ontologies. In *AAAI Fall Symposium on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa 2006), Arlington, VA, USA, October 2006*. 6, 8.1

Bao, J., Hu, Z., Caragea, D., Reecy, J., and Honavar, V. (2006g). Developing frameworks and tools for collaborative building of large biological ontologies. In *the 4th International Workshop on Biological Data Management (BIDM), DEXA Workshops*, pages 191–195. 1.2.2, 3.1.3.2, 7, 8.1

Bao, J., Slutzki, G., and Honavar, V. (2007b). Distributed reasoning with expressive modular ontologies on the semantic web. Technical report, Computer Science, Iowa State University, http://www.cs.iastate.edu/~baojie/iswc2007tr.pdf. 5, 8.1

Bao, J., Slutzki, G., and Honavar, V. (2007c). *Ontology Modularization*, chapter P-DL: A Semantic Importing Approach to Selective Knowledge Reuse In Modular Ontologies. Berlin: Springer (In Press). 4.1

Bao, J., Slutzki, G., and Honavar, V. (2007d). Privacy-preserving reasoning on the semantic web, tr 544. Technical report, Computer Science, Iowa State University, http://archives.cs.iastate.edu/documents/disk0/00/00/05/44/. 6, 8.1

Bao, J., Slutzki, G., and Honavar, V. (2007e). A semantic importing approach to knowledge reuse from multiple ontologies. In *AAAI*, pages 1304–1309. 1.2, 4.1, 4.2.1.2, 4.3.1, 4, 8.1

Bassiliades, N., Antoniou, G., and Vlahavas, I. P. (2004). Dr-device: A defeasible logic system for the semantic web. In *PPSWR*, pages 134–148. 3.2.1.6

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. (2004). Owl web ontology language reference. http://www.w3.org/TR/owl-ref/. 2.3

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43. 1.1

Bonatti, P. A., Duma, C., Fuchs, N., Nejdl, W., Olmedilla, D., Peer, J., and Shahmehri, N. (2006). Semantic web policies - a discussion of requirements and research issues. In *ESWC*, pages 712–724. 6.7.1

Bonifacio, M., Bouquet, P., Busetta, P., Danieli, A., Donà, A., Mameli, G., and Nori, M. (2004). Keex: A peer-to-peer solution for distributed knowledge management. In *P2PKM*. 5.1

Bontas, E. P. (2004). Context representation and usage for the semantic web: A state of the art. Technical Report B-04-30, Freie Universität Berlin, http://www.inf.fu-berlin.de/inst/pubs/tr-b-04-30.abstract.html. 3.1.2.1

Borgida, A. and Serafini, L. (2002). Distributed description logics: Directed domain correspondences in federated information sources. In *CoopIS*, pages 36–53. 1.2.1, 4.4.1.2, 4.4.2.1, 4.3, 4.5, 4.4.2.3, 4.4.3.1, 4.4.3.1, 4.4.3.1, 4.4.3.1, 4.4.5.2

Borgida, A. and Serafini, L. (2003). Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184. 3.2.1.3, 3.2.1.6, 3.2.2, 3.3.2, 4.2.4

Bouquet, P., Dona, A., Serafini, L., and Zanobini, S. (2002). Conceptualized local ontologies specification via CTXML. In *AAAI-02 Workshop on Meaning Negotiation (MeaN-02) July 28, 2002, Edmonton, Canada*. 4.4.1.2, 4.4.5.2

Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., and Stuckenschmidt, H. (2003). C-OWL: Contextualizing ontologies. In Fensel et al. (2003), pages 164–179. 3.1.2.1, 4.1, 4.3.1, 4.4.1.2, 4.4.3.1, 4.4.3.1, 4.4.5.2

Bouquet, P. and Serafini, L. (2003). On the difference between bridge rules and lifting axioms. In *CONTEXT*, pages 80–93. 3.4.2

Bryson, J. and Stein, L. A. (2001). Modularity and design in reactive intelligence. pages 1115–1120. 3.4

Buffa, M. and Gandon, F. (2006). Sweetwiki: semantic web enabled technologies in wiki. In *Int. Sym. Wikis*, pages 69–78. 7.5

Bugliesi, M., Lamma, E., and Mello, P. (1994). Modularity in logic programming. *J. Log. Program.*, 19/20:443–502. 3.4

Buvac, S., Buvac, V., and Mason, I. A. (1995). Metamathematics of contexts. *Fundam. Inform.*, 23(2/3/4):263–301. 3.4.2

Buvac, S. and Kameyama, M. (1998). Introduction: Toward a unified theory of context? *Journal of Logic, Language and Information*, 7(1):1. 3.1.2.1

Buvac, S. and Mason, I. A. (1993). Propositional logic of context. In *AAAI*, pages 412–419. 3.1.2.1, 3.4.2

Calvanese, D., Damaggio, E., Giacomo, G. D., Lenzerini, M., and Rosati, R. (2003). Semantic data integration in p2p systems. In *DBISP2P*, pages 77–90. 3.4.3, 3.4.3

Calvanese, D., Giacomo, G. D., Lenzerini, M., and Rosati, R. (2004). Logical foundations of peer-to-peer data integration. In *PODS*, pages 241–251. 3.4.3, 3.4.3, 3.4.3

Caragea, D., Bao, J., Pathak, J., Silvescu, A., Andorf, C. M., Dobbs, D., and Honavar, V. (2005a). Information integration from semantically heterogeneous biological data sources. In *Proceedings of the 3rd International Workshop on Biological Data Management (BIDM'05) at DEXA 2005*, pages 580–584. 8.2.5

Caragea, D., Pathak, J., Bao, J., Silvescu, A., Andorf, C. M., Dobbs, D., and Honavar, V. (2005b). Information integration and knowledge acquisition from semantically heterogeneous biological data sources. In *Proceedings of the 2nd International Workshop on Data Integration in Life Sciences (DILS'05), San Diego, CA*, pages 175–190. 8.2.5

Catarci, T. and Lenzerini, M. (1993). Representing and using interschema knowledge in cooperative information systems. In *CoopIS*, pages 55–62. 3.4.3, 4.2.2.2

Chen, Y., Mihaila, G. A., Bordawekar, R., and Padmanabhan, S. (2004). L-tree: A dynamic labeling structure for ordered xml data. In *EDBT Workshops*, pages 209–218. 5.3.1

Christophides, V., Karvounarakis, G., Plexousakis, D., Scholl, M., and Tourtounis, S. (2004). Optimizing taxonomic semantic web queries using labeling schemes. *J. Web Sem.*, 1(2):207–228. 5.3.1

Cohen, R., Fraigniaud, P., Ilcinkas, D., Korman, A., and Peleg, D. (2005). Labeling schemes for tree representation. In *IWDC*, pages 13–24. 5.3.1

Cruz, I. F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., and Aroyo, L., editors (2006). *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*, volume 4273 of *Lecture Notes in Computer Science*. Springer. A.2

Cuppens, F. (1990). An epistemic and deontic logic for reasoning about computer security. In *First European Symposium On Research In Computer Security (ESORICS 90)*, pages 135–145. 6.7.3

Damianou, N., Dulay, N., Lupu, E., and Sloman, M. (2001). The ponder policy specification language. In *POLICY*, pages 18–38. 6.7.1

D'Aquin, M., Schlicht, A., Stuckenschmidt, H., and Sabou, M. (2007). Ontology modularization for knowledge selection: Experiments and evaluations. In *DEXA*. 4.4.3.2

Day-Richter, J. (2004). DAG-Edit: A controlled vocabulary editor. In *http://www.godatabase.org/dev/java/dagedit/docs/*. 7.1.1, 7.5

Denny, M. (2002). Ontology building: A survey of editing tools. Technical report, XML.com, http://www.xml.com/pub/a/2002/11/06/ontologies.html. 7.5

Denny, M. (2004). Ontology tools survey, revisited. Technical report, XML.com, http://www.xml.com/pub/a/2004/07/14/onto.html. 7.5

Ding, Y. and Haarslev, V. (2007). An exptime tableau decision procedure for ALCQI. In *Description Logics Workshop, CEUR-WS Vol 250*. 8.2.3

Donini, F. M., Lenzerini, M., Nardi, D., Nutt, W., and Schaerf, A. (1998). An epistemic operator for description logics. *Artificial Intelligence*, 100(1-2):225–274. 3.4.3, 6.7.3

Donini, F. M., Lenzerini, M., Nardi, D., Schaerf, A., and Nutt, W. (1992). Adding epistemic operators to concept languages. In *KR*, pages 342–353. 3.4.3

Donini, F. M. and Massacci, F. (2000). Exptime tableaux for alc. *Artif. Intell.*, 124(1):87–138. 8.2.3

Farkas, C. (2006). *Web and Information Security*, chapter Data Confidentiality on The Semantic Web: Is There an Inference Problem? Chapter IV, pages 73–91. Idea Group Inc. 6.7.2

Farkas, C., Brodsky, A., and Jajodia, S. (2006). Unauthorized inferences in semi-structured databases. *Information Sciences*, 176(22):3269–3299. 6.3, 6.7.2

Farquhar, A., Fikes, R., Pratt, W., and Rice, J. (1995). Collaborative ontology construction for information integration. In *Technique Reports of Knowledge Systems Laboratory, Department of Computer Science, KSL-95-63*. 7.5

Fensel, D., Sycara, K. P., and Mylopoulos, J., editors (2003). *The Semantic Web - ISWC 2003, Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003, Proceedings*, volume 2870 of *Lecture Notes in Computer Science*. Springer. A.2

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., and Patel-Schneider, P. F. (2001). OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent System*, 16(2):38–45. 2.3

Fikes, R., Farquhar, A., and Rice, J. (1997). Tools for assembling modular ontologies in ontolingua. In *AAAI/IAAI*, pages 436–441. 4.3.1, 4.3.4, 6.7.1

Fischer, J., Gantner, Z., Rendle, S., Stritt, M., and Schmidt-Thieme, L. (2006). Ideas and improvements for semantic wikis. In *ESWC*, pages 650–663. 7.5

Franconi, E., Kuper, G. M., Lopatenko, A., and Serafini, L. (2003). A robust logical and computational characterisation of peer-to-peer database systems. In *DBISP2P*, pages 64–76. 4, 3.4.3

Gardiner, T., Horrocks, I., and Tsarkov, D. (2006). Automated benchmarking of description logic reasoners. In *Proc. of the 2006 Description Logic Workshop (DL 2006)*, volume 189. 3.1.3.2, 5.1

Garson, J. (1989). Modularity and relevant logic. *Notre Dame Journal of Formal Logic*, 30(2):207–223. 3.4

Gene Ontology Consortium (2005). Go editor guides. In *http://www.geneontology.org/GO.contents.curator.guides.shtml*. 7.2

Ghidini, C. and Giunchiglia, F. (2001). Local models semantics, or contextual reasoning=locality+compatibility. *Artificial Intelligence*, 127(2):221–259. 3.1.2.1, 3.3, 3.4.1

Ghidini, C. and Serafini, L. (1998). *Frontiers Of Combining Systems 2, Studies in Logic and Computation*, chapter Distributed First Order Logics, pages 121–140. Research Studies Press. 3.3.1, 3.3.2, 3.4.1, 3.4.3, 4.4.1.2, 8.1

Ghidini, C. and Serafini, L. (2006a). Mapping properties of heterogeneous ontologies. In *1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC*. 3.2.2, 4.4.2.1, 5.6

Ghidini, C. and Serafini, L. (2006b). Reconciling concepts and relations in heterogeneous ontologies. In *ESWC*, pages 50–64. 4.4.1.2, 4.4.2.1

Ghidini, C., Serafini, L., and Tessaris, S. (2007). On relating heterogeneous elements from different ontologies. In *Description Logics Workshop, CEUR-WS Vol 250*. 4.4.1.2

Ghilardi, S., Lutz, C., and Wolter, F. (2006). Did i damage my ontology? a case for conservative extensions in description logics. In *KR*, pages 187–197. 3.4.4

Giereth, M. (2005). On partial encryption of rdf-graphs. In Gil et al. (2005), pages 308–322. 6.7.1

Gil, Y., Motta, E., Benjamins, V. R., and Musen, M. A., editors (2005). *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, volume 3729 of *Lecture Notes in Computer Science*. Springer. A.2

Giunchiglia, F. and Ghidini, C. (1998). Local models semantics, or contextual reasoning = locality + compatibility. In *KR*, pages 282–291. 3.1.2.1, 4.4.1.2

Glasgow, J. I., MacEwen, G. H., and Panangaden, P. (1992). A logic for reasoning about security. *ACM Trans. Comput. Syst.*, 10(3):226–264. 6.7.3

Godik, S. and Moses, T. (2002). Oasis extensible access control markup language (xacml). OASIS Committee Secification cs-xacml-specification-1.0, November 2002, http://www.oasis-open.org/committees/xacml/. 6.1, 6.7.1

Gomez-Perez, A., Angele, J., Fernandez-Lopez, M., Christophides, V., Stutt, A., and Sure, Y. (2002). Ontoweb deliverable 1.3: A survey on ontology tools, http://ontoweb.org/about/deliverables/d13_v1-0.zip/. 1.1, 7.5

Grädel, E. (2001). Why are modal logics so robustly decidable? In *Current Trends in Theoretical Computer Science*, pages 393–408. 5.2

Grau, B. C. (2005). *Combination and Integration of Ontologies on the Semantic Web*. PhD dissertation, Dpto. de Informatica, Universitat de Valencia, Spain. 1.2.1, 2, 3.3.2, 4.4.1.4, 4.4.2.2, 4.4.2.3, 4.3, 4.4.2.3, 4.5, 4.4.3.1, 4.8, 4.4.3.2, 4.4.3.2, 4.4.3.2, 4.4.5.3, 5.6

Grau, B. C., Horrocks, I., Kazakov, Y., and Sattler, U. (2007). Just the right amount: Extracting modules from ontologies. In *Proc. of the Sixteenth International World Wide Web Conference (WWW 2007)*. 3.1.2.2, 3.4.4, 4.3.4

Grau, B. C., Horrocks, I., Kazakov, Y., and Sattler, U. (2007). A logical framework for modularity of ontologies. In *IJCAI*, pages 298–303. 3.4.4, 4.2.5, 4.4.2, 6.4, 6.6, 6.4, 6.2, 6.6.3, 8.1

Grau, B. C., Horrocks, I., Kutz, O., and Sattler, U. (2006a). Will my ontologies fit together? In *Proc. of the 2006 Description Logic Workshop (DL 2006)*, volume 189. CEUR (http://ceur-ws.org/). 3.4.4, 6.1, 6.3.3, 6.5

Grau, B. C., Horrocks, I., Parsia, B., Patel-Schneider, P., and Sattler, U. (2006b). Next steps for OWL. In *OWL: Experiences and Directions (OWLED 2006), CEUR Workshops*. 2.3

Grau, B. C. and Kutz, O. (2007). Modular ontology languages revisited. In *Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa), co-located with IJCAI*. 3.4.4, 4.2.6.3, 4.2.6.4, 5, 4.2.6.4, 4.2.6.4, 4.4.2.2, 4.4.2.3

Grau, B. C., Parsia, B., and Sirin, E. (2004a). Tableau algorithms for e-connections of description logics. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), TR 2004-72. 4.2.6.4, 4.4.3.2

Grau, B. C., Parsia, B., and Sirin, E. (2004b). Working with multiple ontologies on the semantic web. In McIlraith et al. (2004), pages 620–634. 3.2.2, 4.2.6.4, 4.3.1, 4.4.1.4, 4.4.2.2, 4.4.3.1, 4.8, 4.4.3.2, 4.4.3.2, 4.4.4.1, 4.4.5.3, 5.4, 5.6

Grau, B. C., Parsia, B., and Sirin, E. (2006c). Combining owl ontologies using epsilon-connections. *J. Web Sem.*, 4(1):40–59. 4.4.1.4, 4.4.2.2, 5.6

Grau, B. C., Parsia, B., Sirin, E., and Kalyanpur, A. (2005). Automatic partitioning of owl ontologies using -connections. In *Description Logics*. 4.4.3.2

Grau, B. C., Parsia, B., Sirin, E., and Kalyanpur, A. (2006d). Modularity and web ontologies. In *KR*, pages 198–209. 3.2.1.6, 3.4.4

Gray, J. W. and Syverson, P. F. (1998). A logical approach to multilevel security of probabilistic systems. *Distributed Computing*, 11(2):73–90. 6.6.1, 6.7.3

Guha, R. V. (1991). *Contexts: a formalization and some applications*. PhD dissertation, Stanford University. 3.1.2.1

Guha, R. V. and McCarthy, J. (2003). Varieties of contexts. In *CONTEXT*, pages 164–177. 3.1.2.1

Guha, R. V., McCool, R., and Fikes, R. (2004). Contexts for the semantic web. In McIlraith et al. (2004), pages 32–46. 3.1.2.1

Guo, Y., Pan, Z., and Heflin, J. (2005). Lubm: A benchmark for owl knowledge base systems. *J. Web Sem.*, 3(2-3):158–182. 3.1.3.2

Haarslev, V. and Möller, R. (2001). Racer system description. In *IJCAR*, pages 701–706. 2.2.1

Halevy, A. Y., Ives, Z. G., Suciu, D., and Tatarinov, I. (2003). Schema mediation in peer data management systems. In *ICDE*, pages 505–. 3.4.3

Hallam-Baker, P. and Maler, E. (2002). Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML). OASIS Committee Specification sstc-core, May 2002. 6.7.1

Hayes, P., Saavedra, R., and Reichherzer, T. (2003). A collaboration development environment for ontologies. In *Proceedings of the Semantic Integration Workshop, Sanibel Island, Florida,*. 7.5

Herzig, A. and Varzinczak, I. J. (2004). On the modularity of theories. In *Advances in Modal Logic*, pages 93–109. 3.4

Heymans, S. and Vermeir, D. (2002). A defeasible ontology language. In *CoopIS/DOA/ODBASE*, pages 1033–1046. 3.2.1.6

Hladik, J. (2004). A tableau system for the description logic shio. In *IJCAR Doctoral Programme*. 2.1.3

Hollunder, B., Nutt, W., and Schmidt-Schauß, M. (1990). Subsumption algorithms for concept description languages. In *ECAI*, pages 348–353. 2.2.2

Homola, M. (2007). Distributed description logics revisited. In *Description Logics Workshop, CEUR-WS Vol 250*. 4.4.3.1

Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., and Wang, H. (2006). The Manchester OWL syntax. In *OWL: Experiences and Directions (OWLED 2006), CEUR Workshops*. 2.3

Horrocks, I. (2002). DAML+OIL: a description logic for the semantic web. *IEEE Data Engineering Bulletin*, 25(1):4–9. 2.1.3, 2.3

Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible $\mathcal{SROIQ}$. In *KR*, pages 57–67. 2.3, 8.2.3

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From SHIQ and RDF to OWL: the making of a web ontology language. *J. Web Sem.*, 1(1):7–26. 2.1.3, 2.3

Horrocks, I. and Sattler, U. (1999). A description logic with transitive and inverse roles and role hierarchies. 9(3):385–410. 2.1.3

Horrocks, I. and Sattler, U. (2001). Ontology reasoning in the SHOQ(D) description logic. In Nebel, B., editor, *Proceeding of the 17th Int. Joint Conf. on Artificial Intelligence*, pages 199–204. AAAI, Morgan Kaufmann. 2.1.3, 5.1

Horrocks, I. and Sattler, U. (2005). A Tableaux Decision Procedure for $\mathcal{SHOIQ}$. In *IJCAI*, pages 448–453. 1.2.1, 2.1.3, 2.2.3, 5.1, 5.5.1, 8.2.3

Horrocks, I. and Sattler, U. (2007). A tableaux decision procedure for SHOIQ. *Journal of Automated Reasoning*, To Appear. 2.2.3

Horrocks, I., Sattler, U., and Tobies, S. (1999). Practical reasoning for expressive description logics. In *LPAR*, pages 161–180. 2, 2.1.3, 4.2.1.3, 5.2.2, 5.5.1, 5.5.3, 5.5.3, 5.5.4.2, A.2, A.2

Horrocks, I., Sattler, U., and Tobies, S. (2000). Practical reasoning for very expressive description logics. *Logic Journal of the Interest Group in Pure and Applied Logic (IGPL)*, 8(3):239–264. 2.1.3

Hu, Z., Bao, J., Rothschild, M. F., Honavar, V., and Reecy, J. M. (2006). Developing frameworks and tools for animal trait ontology (ato). In *Plant and Animal Genome XIV Conference. Poster Track. January 14-18, 2006,San Diego, CA*. 7.1.1

Huang, Z., van Harmelen, F., and ten Teije, A. (2005). Reasoning with inconsistent ontologies. In *IJCAI*, pages 454–459. 3.2.1.6

Jain, A. and Farkas, C. (2006). Secure resource description framework: an access control model. In *SACMAT*, pages 121–129. 6.7.2

Jajodia, S. and Wijesekera, D. (2001). Recent advances in access control models. In *DBSec*, pages 3–15. 6.1, 6.7.1

Kagal, L., Finin, T. W., and Joshi, A. (2003). A policy based approach to security for the semantic web. In Fensel et al. (2003), pages 402–418. 6.3.1, 6.7.1

Kagal, L., Paolucci, M., Srinivasan, N., Denker, G., Finin, T. W., and Sycara, K. P. (2004). Authorization and privacy for semantic web services. *IEEE Intelligent Systems*, 19(4):50–56. 6.7.1

Kalyanpur, A., Parsia, B., and Hendler, J. A. (2005). A tool for working with web ontologies. *Int. J. Semantic Web Inf. Syst.*, 1(1):36–49. 4.4.5.3, 7.5

Kalyanpur, A., Parsia, B., Sirin, E., and Grau, B. C. (2006). Repairing unsatisfiable concepts in owl ontologies. In *ESWC*, pages 170–184. 3.1.3.1, 3.2.1.6, 8.2.3

Kawamoto, K., Kitamura, Y., and Tijerino, Y. A. (2006). Kawawiki: A semantic wiki based on rdf templates. In *IAT Workshops*, pages 425–432. 7.5

Kazakov, Y., Sattler, U., and Zolin, E. (2007). How many legs do i have? non-simple roles in number restrictions revisited. In *Proc. of the 14th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'2007)*, volume To appear. 2

Koch, C. (2002). Query rewriting with symmetric constraints. In *FoIKS*, pages 130–147. 3.4.3

Kolovski, V., Hendler, J., and Parsia, B. (2007). Analyzing web access control policies. In *WWW*, pages 677–686. 6.7.1

Korman, A., Peleg, D., and Rodeh, Y. (2004). Labeling schemes for dynamic tree networks. *Theory Comput. Syst.*, 37(1):49–75. 5.3.1

Kriple, S. A. (1962). Semantical considerations on modeal logic. In *A Colloquium on Modal and Many-Valued Logics*, Helsinki. 3.4.3

Krorzsch, M., Vrandecic, D., and Volkel, M. (2005). Wikipedia and the semantic web - the missing links. In *Proceedings of the WikiMania2005*. 7.5

Kutz, O., Lutz, C., Wolter, F., and Zakharyaschev, M. (2003). E-connections of description logics. In *Description Logics Workshop, CEUR-WS Vol 81*. 4.4.1.4, 4.4.2.2

Kutz, O., Lutz, C., Wolter, F., and Zakharyaschev, M. (2004). E-connections of abstract description systems. *Artif. Intell.*, 156(1):1–73. 4.4.2.3, 4.3, 4.5, 4.4.4.1

Kutz, O., Wolter, F., and Zakharyaschev, M. (2002). Connecting abstract description systems. In *KR*, pages 215–226. 4.4.1.4

Lenat, D. B. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):32–38. 3.1, 4.4.1.1

Levy, A. Y. and Rousset, M.-C. (1998). Combining horn rules and description logics in carin. *Artif. Intell.*, 104(1-2):165–209. 3.4.3

Lifschitz, V. (1991). Nonmonotonic databases and epistemic queries. In *IJCAI*, pages 381–386. 3.4.3

Loebe, F. (2006). Requirements for logical modules. In *1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC*. 3.1, 3.1.2.3, 3.1.2.3, 3.2.1.2

Lutz, C., Walther, D., and Wolter, F. (2007). Conservative extensions in expressive description logics. In *IJCAI*, pages 453–458. 3.4.4, 6, 6.6.3

McCarthy, J. (1993). Notes on formalizing context. In *IJCAI*, pages 555–562. 3.1.2.1

McIlraith, S. A., Plexousakis, D., and van Harmelen, F., editors (2004). *The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings*, volume 3298 of *Lecture Notes in Computer Science*. Springer. A.2

Meghini, C. and Straccia, U. (1996). A relevance terminological logic for information retrieval. In *SIGIR*, pages 197–205. 3.4

Meilicke, C., Stuckenschmidt, H., and Tamilin, A. (2007). Repairing ontology mappings. In *AAAI*. 3.2.1.6, 8.2.3

Motik, B. (2005). On the properties of metamodeling in owl. In Gil et al. (2005), pages 548–562. 2.3

Motik, B. and Horrocks, I. (2006). Problems with OWL syntax. In *OWL: Experiences and Directions (OWLED 2006), CEUR Workshops*. 11, 12

Motik, B., Horrocks, I., Rosati, R., and Sattler, U. (2006). Can owl and logic programming live together happily ever after? In Cruz et al. (2006), pages 501–514. 3.4.3

Motik, B. and Rosati, R. (2007). A faithful integration of description logics with logic programming. In *IJCAI*, pages 477–482. 3.4.3

Mukerjee, A. and Mali, A. D. (2002). Modular models of intelligence - review, limitations and prospects. *Artif. Intell. Rev.*, 17(1):39–64. 3.4

Muljadi, H., Takeda, H., Shakya, A., Kawamoto, S., Kobayashi, S., Fujiyama, A., and Ando, K. (2006). Semantic wiki as a lightweight knowledge management system. In *ASWC*, pages 65–71. 7.5

Mungall, C. (2005). Integrated ontologies for biological annotation: The national center for biomedical ontologies. In *The First International Biocurator Meeting. Pacific Grove, CA, December 8-11, 2005*. 7.1.1

Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In *FOIS*, pages 2–9. 3.1

Oren, E., Völkel, M., Breslin, J. G., and Decker, S. (2006). Semantic wikis for personal knowledge management. In *DEXA*, pages 509–518. 7.5

Ouellet, R. and Ogbuji, U. (2002). Introduction to DAML. http://www.daml.org/about.html. accessed Jan. 2007. 2.3

Pan, J., Serafini, L., and Zhao, Y. (2006). Semantic import: An approach for partial ontology reuse. In *1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC*. 3.1.2.2, 4.2.2.1, 4.4.4.3

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12):1053–1058. 3.1.3.1

Parsia, B. and Grau, B. C. (2005). Generalized link properties for expressive epsilon-connections of description logics. In *AAAI*, pages 657–662. 4.4.1.4, 4.4.2.2, 14, 4.4.3.2, 4.2, 4.4.4.4

Parsia, B., Sirin, E., and Kalyanpur, A. (2005). Debugging owl ontologies. In *WWW*, pages 633–640. 3.2.1.6

Patel-Schneider, P., P.Hayes, and Horrocks, I. (2003). Web ontlogy language (owl) abstract syntax and semantics. http://www.w3.org/TR/owl-semantics/. 1.2.2, 4.3.1

Plaisted, D. A. and Yahya, A. H. (2003). A relevance restriction strategy for automated deduction. *Artif. Intell.*, 144(1-2):59–93. 3.4

Polleres, A. (2006). Logic programs with contextually scoped negation. In *WLP*, pages 129–136. 4.2.6.2

Polleres, A., Feier, C., and Harth, A. (2006). Rules with contextually scoped negation. In *ESWC*, pages 332–347. 4.2.6.2

Schaffert, S. (2006). Ikewiki: A semantic wiki for collaborative knowledge management. In *WETICE*, pages 388–396. 7.5

Schlicht, A. and Stuckenschmidt, H. (2006). Towards structural criteria for ontology modularizationc. In *1st International Workshop on Modular Ontologies (WoMo 2006), co-located with ISWC*. 3.1.1.1

Schmidt, R. and Tishkovsky, D. (2007). Deciding ALBO with tableau. In *Proceedings of the 20th International Workshop on Description Logics (DL-2007)*, pages 135–146. Bozen-Bolzano University Press. 2

Schmidt-Schauß, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26. 2.1.2

Schreiber, G. and Dean, M. (2004). Owl web ontology language reference. http://www.w3.org/TR/2004/REC-owl-ref-20040210/. 2.1.3, 2.3, 4.3.1, 5.1

Seidenberg, J. and Rector, A. L. (2006). Web ontology segmentation: analysis, classification and use. In *WWW*, pages 13–22. 4.4.3.2

Serafini, L., Borgida, A., and Tamilin, A. (2005a). Aspects of distributed and modular ontology reasoning. In *IJCAI*, pages 570–575. 4.2.6.4, 4.4.2.3, 4.4.3.1, 4.4.3.1, 4.4.3.1, 4.9, 4.10, 5.6

Serafini, L. and Bouquet, P. (2004). Comparing formal theories of context in ai. *Artif. Intell.*, 155(1-2):41–67. 3.1.2.1, 3.4.2

Serafini, L., Stuckenschmidt, H., and Wache, H. (2005b). A formal investigation of mapping language for terminological knowledge. In *IJCAI*, pages 576–581. 3.3.2, 4, 5, 4.4.1.3, 4.4.2, 4.4.2.2, 4.4.2.3

Serafini, L. and Tamilin, A. (2004). Local tableaux for reasoning in distributed description logics. In *Description Logics Workshop 2004, CEUR-WS Vol 104*. 4.2.6.4, 5.6

Serafini, L. and Tamilin, A. (2005a). Distributed instance retrieval in heterogeneous ontologies. In *Proceedings of SWAP 2005, CEUR Workshop Vol 166*. 5.6

Serafini, L. and Tamilin, A. (2005b). Drago: Distributed reasoning architecture for the semantic web. In *ESWC*, pages 361–376. 4.4.3.1, 4.4.5.3

Sirin, E. and Parsia, B. (2004). Pellet: An OWL DL Reasoner. In *Description Logics Workshop*. 4.4.5.3, 5.1

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *J. Web Sem.*, 5(2). 2.2.1, 5.6, 8.2.3

Spaccapietra (Coordnator), S. (2005). Report on modularization of ontologies. Deliverable D2.1.3.1, Knowledge Web, http://wasp.cs.vu.nl/knowledgeweb/Deliverables/D2.1.3.1/D2.1.3.1-Modularization.pdf. 3.1.3.2

Stoermer, H. (2006). Introducing Context into Semantic Web Knowledge Bases. In *Proceedings of the CAISE*06 Doctoral Consortium*. http://dme.uma.pt/caise06dc/papers/Stoermer.pdf. 3.1.2.1

Stoermer, H., Palmisano, I., Redavid, D., Iannone, L., Bouquet, P., and Semeraro, G. (2006). Contextualization of a rdf knowledge base in the vikef project. In *ICADL*, pages 101–110. 3.1.2.1

Stuckenschmidt, H. and Klein, M. (2003a). Modularization of ontologies - wonderweb: Ontology infrastructure for the semantic web. http://wonderweb.semanticweb.org/ deliverables/documents/D21.pdf. 4.4.1.3

Stuckenschmidt, H. and Klein, M. C. A. (2003b). Integrity and change in modular ontologies. In *IJCAI*, pages 900–908. 4.4.1.3

Stuckenschmidt, H. and Klein, M. C. A. (2004). Structure-based partitioning of large concept hierarchies. In McIlraith et al. (2004), pages 289–303. 3.1.1.1

Stuckenschmidt, H., Serafini, L., and Wache, H. (2006). Reasoning about ontology mappings. In *ECAI 2006 Workshop on Context Representation and Reasoning (CRR)*. 4.4.2.3, 4.10, 6

Tamilin, A. (2007). *Distributed Ontological Reasoning: Theory, Algorithms, And Applications*. Phd dissertation, University of Trento. 5.6

Tobies, S. (2000). The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. Artif. Intell. Res. (JAIR)*, 12:199–217. 4.2.5

Tobies, S. (2001). *Complexity results and practical algorithms for logics in Knowledge Representation*. Phd thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany. 4.2.5, 5.5.3, 5.5.4.3, A.2

Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., and Uszok, A. (2003). Semantic web languages for policy representation and reasoning: A comparison of kaos, rei, and ponder. In Fensel et al. (2003), pages 419–437. 6.1, 6.7.1

Tsarkov, D. and Horrocks, I. (2004). Efficient reasoning with range and domain constraints. In *Description Logics*. FaCT++. 2.2.1, 5.1

Tudorache, T. and Noy, N. (2007). Collaborative protege. In *Workshop on Social and Collaborative Construction of Structured Knowledge (CKC) at 16th International World Wide Web Conference (WWW2007)*. 7.5

Uszok, A., Bradshaw, J. M., Jeffers, R., Suri, N., Hayes, P. J., Breedy, M. R., Bunch, L., Johnson, M., Kulkarni, S., and Lott, J. (2003). Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *POLICY*, pages 93–96. 6.3.1, 6.7.1

Vardi, M. Y. (1996). Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models*, pages 149–184. 2.2.2, 5.2

Vega, J. C. A., Corcho, Ó., Fernández-López, M., and Gómez-Pérez, A. (2001). WebODE: a scalable workbench for ontological engineering. In *K-CAP*, pages 6–13. 7.5

Völkel, M., Schaffert, S., and Decker, S., editors (2006a). *First Workshop on Semantic Wikis: From Wiki to Semantic (SemWiki2006), at the 3rd Annual European Semantic Web Conference (ESWC) in Budva, Montenrego*. 7.5

Völkel, M., Schaffert, S., Pasaru-Bontas, E., and Auer, S., editors (2006b). *Wiki-based knowledge engineering: second workshop on semantic Wikis, workshop at Int. Sym. Wikis*. 7.5

Wahlster, W. (2006). Smartweb: Getting answers on the go. In *ECAI (Invited Talk)*, page 6. 3.1.3.2

Wang, H., He, H., Yang, J., Yu, P. S., and Yu, J. X. (2006). Dual labeling: Answering graph reachability queries in constant time. In *ICDE*, page 75. 6.5

Wang, Y., Bao, J., Haase, P., and Qi, G. (2007). Evaluating formalisms for modular ontologies in distributed information systems. In *RR*, pages 178–193. 8.1

Yeh, I., Karp, P. D., Noy, N. F., and Altman, R. B. (2003). Knowledge acquisition, consistency checking and concurrency control for gene ontology (go). *Bioinformatics*, 19(2):241–248. 7.2

Zimmermann, A. (2007). Integrated distributed description logics. In *Description Logics Workshop, CEUR-WS Vol 250*. 4.4.3.1

Zimmermann, A. and Euzenat, J. (2006). Three semantics for distributed systems and their relations with alignment composition. In *International Semantic Web Conference*, pages 16–29. 4.2.6.1, 4.4.3.1, 5.3

Index

$\mathcal{E}$-Connections, 96, 100, 111, 114, 162

ABox, 9
Abstract Modular Ontology, AMO, 43
    Image Domain Relation, 46

COB Editor, 194
Collaborative Ontology Building, 4, 33, 188
Conservative Extension, 62
Contextual Semantics, 28
Contextualized Constructors, 49
Contextualized Negation, 80

Description Logics, 8
    $\mathcal{SHOIQ}$, 11, 17
    $\mathcal{ALC}$, 10
    Reasoning Tasks, 13
    Semantics, 10
Distributed Description Logics, DDL, 94, 98, 106, 115, 161
Distributed First Order Logic, 57
Distributed Tableau, 128

Entailment, 9
Epistemic Semantics, 59

General Concept Inclusion, GCI, 9

Interpretation, 9

Local Model Semantics, 57, 67
Local Semantics, 28, 35, 54

Model, 9
Modular Ontology, 24
    Decidability, 38, 54, 77
    Directionality, 36, 55, 80
    Exact Reasoning, 35, 56, 78
    Monotonicity, 37, 55, 78
    Transitive Reusability, 37, 55, 78

NNF, Negation Normal Form, 15, 128

Open World Assumption, OWA, 168
Organizational Structure, 195
OWL, 20, 84
    OWL 1.1, 23

Package, 65
    Nesting, 195
Package-based Description Logics, P-DL, 2, 64
    $\mathcal{SHOIQP}$, 65
    $\mathcal{SHIQP}$, 150
    $\mathcal{ALCP}_{\mathcal{C}}$, 144
    $\mathcal{ALCP}_{\mathcal{C}}^{-}$, 125
Partial Ontology Reuse, 4, 27, 29
Privacy-Preserving Inference, 168
    $\mathcal{SHIQ}$, 173

General Strategy, 171
Hierarchical Ontologies, 175
Propositional Logic of Context, PLC, 58

RBox, 9

Satisfiability, 13, 70
Scope Limitation Modifier, 167
Semantic Encapsulation, 30
Semantic Modularity, 28
Semantic Web, 1
Subsumption, 13
Syntactical Modularity, 25

Tableau Algorithm, 14
for $\mathcal{ALC}$, 15
for $\mathcal{SHOIQ}$, 17
for $\mathcal{SHIQP}$, 150
for $\mathcal{ALCP}_{\mathcal{C}}$, 144, 147
for $\mathcal{ALCP}_{\mathcal{C}}^{-}$, 125
TBox, 9

WikiOnt, 199