NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D3.8.2 Evaluation of Methods for Contextualized Learning of Networked Ontologies

Deliverable Co-ordinator: **Johanna Völker**

Deliverable Co-ordinating Institution: **University of Karlsruhe (UKARL)**

Other Authors: **Johanna Völker, Eva Blomqvist**

With Contributions from: Claudio Baldassarre

In this deliverable, we report on several evaluation experiments which aim to demonstrate the synergies arising from a combination of ontology design patterns and ontology learning techniques. Furthermore, we present a novel method for ontology refinement, i.e. the semi-automatic acquisition of complex domain-range restrictions, and its evaluation on the well-known SWRC ontology. A variant of this method is currently being implemented as a plugin for the NeOn Toolkit to enable future user studies within the NeOn project.

Document Identifier:	NEON/2009/D3.8.2/v1.1	Date due:	February 28, 2009
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 28, 2009
Project start date	March 1, 2006	Version:	v1.1
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier, France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield, United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Kolenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Change Log

Version	Date	Amended by	Changes
0.1	04-01-2009	Johanna Völker	Initial setup
0.2	26-01-2009	Eva Blomqvist	First draft of OntoCase chapter
0.3	02-02-2009	Johanna Völker	First draft of RoLExO chapter
0.4	08-02-2009	Johanna Völker	Description of RELExO plugin
0.5	10-02-2009	Johanna Völker	Abstract and introduction
0.6	22-02-2009	Eva Blomqvist	Updates in OntoCase chapter and parts in introduction and conclusion
0.7	23-02-2009	Johanna Völker	Overall revision and executive summary
0.8	05-03-2009	Johanna Völker	Revision of Chapters 1, 2 and 4
0.9	07-03-2009	Eva Blomqvist	Revision of Chapter 3
1.0	08-03-2009	Johanna Völker	Minor changes and corrections
1.1	10-03-2009	Eva Blomqvist	Final changes to Chapter 3

Executive Summary

In D3.8.1, we reported on the evaluation of our approach to learning disjointness axioms for the purpose of mapping debugging, i.e. the improvement of automatically generated alignments by means of ontology reasoning. Moreover, first experiments on FAO data demonstrated the basic feasibility of our method for learning complex class descriptions from natural-language definitions. This deliverable continues our series of evaluation experiments and describes new methods for learning networked ontologies. We start by a short overview of NeOn, focusing on WP3 and the definition of context, and then report on the implementation and evaluation of two FCA-based approaches to ontology refinement. More specifically, we describe a novel method for the semi-automatic acquisition of complex property restrictions and its evaluation on the well-known SWRC ontology as well as a new plugin that serves as a graphical frontend for RELExO. In the second part of the deliverable, we present the results of several experiments that aim to demonstrate possible synergies between ontology learning and pattern-based ontology engineering with the help of OntoCase.

Contents

List of Figures

Chapter 1

Introduction

While the realization of the semantic web as envisioned by Tim Berners-Lee is still hampered by the lack of ontological resources, more and more people tend to see the future of semantic technologies in application scenarios such as e-business or life sciences, which require large scale reasoning over very complex domains. But these knowledge-intensive applications even more than the semantic web depend on the availability of expressive, high-quality ontologies. Approaches towards the automatic or semi-automatic construction of ontologies could help to overcome this knowledge acquisition bottleneck – and indeed, a huge number of ontology learning tools and frameworks have been developed in recent years. They all aim to enable the automatic or semi-automatic generation of ontologies from various kinds of resources. Nevertheless, both quality and expressivity of the ontologies which can be acquired by the current state-of-the-art in ontology learning have failed to meet the expectations of people who argue in favor of powerful, knowledge-intensive applications based on ontological reasoning.

Our work aims at providing methods and tools to facilitate the generation of expressive ontologies suitable for reasoning-based applications. The methods we propose are suitable for enriching (or refining) any OWL-based ontology such that previously implicit assumptions are explicated in a formal way. Both formalness and explicitness of knowledge are indispensable requirements for automated logical inference and indeed, reasoning-based applications immediately benefit from the enrichment of an ontology, as the enhanced axiomatization increases the number of possible conclusions. However, in particular, such knowledge-intensive applications are very sensitive to modeling errors and incorrect formalizations as well as logical inconsistencies can have a serious impact on the overall usefulness of the underlying ontologies. Our methods and experiments therefore not only concentrate on the acquisition or enrichment of ontologies, but also aim to ensure the quality of the learned ontologies by automated means. Ontology design patterns are an effective and efficient means to incorporate best-practices of ontology construction into manually or automatically acquired axiomatizations. At the same time, automated approaches to applying ontology design patterns can benefit from ontology learning techniques. In this deliverable and the experiments described in the following, we therefore aim to outline possible synergies between ontology learning and pattern-based ontology engineering.

In D3.8.1 [VB08], we reported on the evaluation of our approach to learning disjointness axioms for the purpose of mapping debugging, i.e. the improvement of automatically generated alignments by means of ontology reasoning. Moreover, first experiments on FAO data demonstrated the basic feasibility of our method for learning complex class descriptions from natural-language definitions. This deliverable continues our series of evaluation experiments and describes new methods for learning networked ontologies. We start with a short overview of NeOn, focusing on WP3 and the definition of context (cf. Section 1.1, which we will occasionally refer to in the remainder of this deliverable). Chapter 2 describes the implementation and evaluation of two FCA-based approaches to ontology refinement. More specifically, in Section 2.2.4, we present a novel method for the semi-automatic acquisition of complex property restrictions and its evaluation on the well-known SWRC ontology. Section 2.3 introduces a new plugin that serves as a graphical frontend for RELExO (see NeOn D3.8.1, Chapter 4). In Chapter 3 two sets of experiments involving the OntoCase method are

presented and analysed. OntoCase was discussed in D3.8.1 [VB08] and now this discussion is followed by evaluations of the OntoCase approach. We mainly show that OntoCase improves on learnt ontologies generated by the ontology learning tool Text2Onto and that on the other hand such ontology learning tools can facilitate pattern matching through enriching the input ontologies used by OntoCase. Finally, Chapter 4 concludes with a summary and an outlook to future work.

1.1 The Big Picture: NeOn and WP3

The research reported in this deliverable is part of NeOn WP3 – "Context sensitivity for Networked Ontologies". As illustrated by Figure 1.1, WP3 constitutes a central component of the overall NeOn architecture. Among the most important goals of this workpackage is the development of formalisms and methods for dealing with context-sensitivity of ontologies, including specific solutions for applications such as ontology alignment or reasoning.

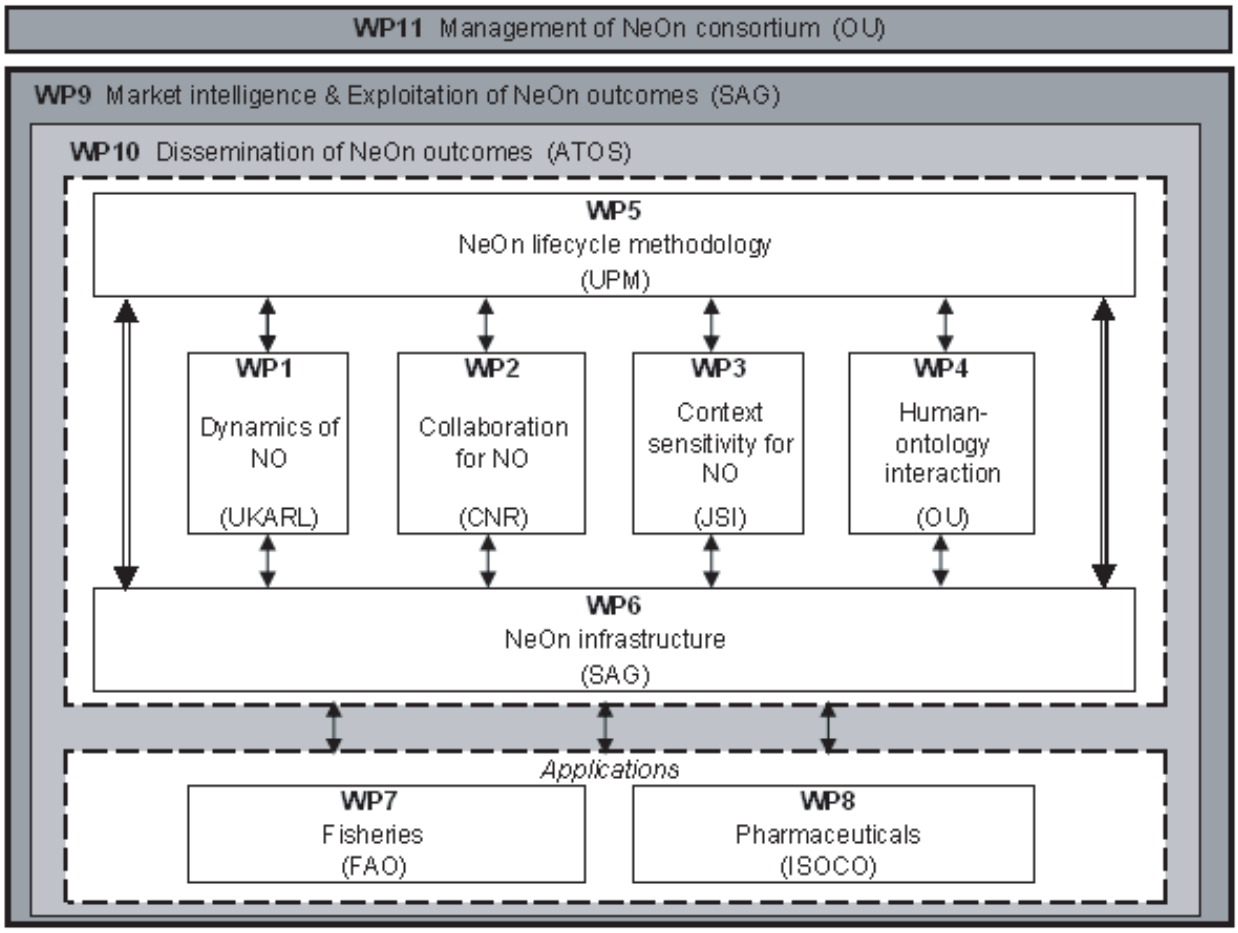

Figure 1.1: Relationships between different workpackages in NeOn

Context sensitivity for networked ontologies. For this deliverable, we adopt the notion of context as a *semantic modifier*, which is considered something that changes our interpretation of a knowledge base. This definition was first introduced in NeOn D3.1.1 and refined later by further definitions, e.g., in NeOn D3.1.3. The latter deliverable instantiates the original abstract notion of context in three different ways: *provenance*, *argumentation* and *mapping*.

- *Provenance* refers to the origin of an ontology element and constitutes the most important type of context in this deliverable. By associating an entity or axiom with provenance information we can represent, for example, knowledge about its creator (typically, a human or automatic agent), the time it was added to the ontology or lexical resources providing a justification for its existence.
- *Mappings* are directed semantic relationships (also called "correspondences") between elements in different ontologies. As suggested in D3.1.3, *un*directed correspondences can be represented by bi-directional mappings. In the following, we will use the term *ontology alignment* to refer to a set of correspondences including, e.g., subsumption or equivalence relationships.
- *Argumentation* is the exchange of arguments in favor of or against particular ontology design decisions.

1.2 Related Deliverables

Deliverable D3.8.1 [VB08] presents the OntoCase method that is used in the experiments of Chapter 3. Ontology content design patterns were presented in D5.1.1 [SFBG+07] and more recently in D2.5.1 [PGD+08]. Evaluations of content design patterns as such are presented in D5.6.2 [DSFGP+09].

1.3 Acknowledgements

The work on exploration-based ontology engineering reported in Chapter 2 of this deliverable is partly based on a collaboration with Sebastian Rudolph from Karlsruhe University. Data for the OntoCase experiments was collected and pre-processed by Claudio Baldassarre at the FAO.

Chapter 2

Exploration-based Ontology Refinement

In this chapter, we describe two complementary approaches that aim to support the user in the difficult and time-consuming process of ontology refinement. One of these approaches, which is instantiated by a tool named RELExO (*Relational Exploration for Learning Expressive Ontologies*), has already been introduced in D3.8.1, but so far lacked a proper integration with the NeOn Toolkit. In Section 2.3, we therefore present a new plugin that serves as a graphical frontend for RELExO and extends the Toolkit's core functionality by automatic support for ontology evaluation and refinement. The second approach is similar to the aforementioned one in that it is based on relational exploration and formal concept analysis. However, the axioms acquired by this approach and its implementation in the RoLExO tool[1] (*Role Exploration for Learning Expressive Ontologies*, cf. Section 2.2.4) belong to a different logical fragment which is not covered by RELExO. In fact, RoLExO has been specifically developed to assist the ontology engineer in specifying complex domain-range restrictions of object properties, i.e. non-taxonomic relationships.

2.1 Context Sensitivity for Networked Ontologies

The semi-automatic refinement of ontologies by means of RoLExO (cf. Section 2.2.4) and RELExO (cf. Section 2.3) relates to our understanding of context as a semantic modifier as follows.

Ontology alignment. Both relational and role exploration can be used to support the process of ontology alignment and integration, because the construction of a formal context from classes in two different ontologies enables us to interactively acquire subsumption axioms corresponding to ontology mappings. This kind of semi-automatic ontology alignment is particularly useful when it comes to the integration of domain and top-level ontologies as the latter feature a very high level of abstraction that it is very difficult to handle for automatic alignment approaches, e.g., based on label similarity. However, note that the application of methods for exploration-based ontology refinement is left for future work (cf. Chapter 4).

2.2 Semi-automatic Acquisition of Property Restrictions

The more complex an ontology or the bigger a knowledge base is, the more difficult is its extension, evaluation and refinement. In practical scenarios with medium to large size ontologies, it is almost impossible even for an experienced ontology engineer to anticipate all the logical consequences of a modeling activity such as the addition of a class or axiom. At the same time, domain experts who are presented with possible formalizations of their knowledge, e.g., automatically generated by an ontology learning method (cf. D3.8.1, Chapter 4), often cannot tell if a given set of axioms sufficiently approximates their conceptualization.

[1] So far, RoLExO is only available as a stand-alone version. The development of a NeOn Toolkit plugin is left for future work.

In D3.8.1 [VB08], we therefore proposed a systematic, reasoner-aided approach to ontology acquisition and refinement. The instantiation of this approach, the RELExO framework (see D3.8.1, Section 4.4), combines a method for learning complex class descriptions from textual definitions with the FCA-based technique of *relational exploration*. By asking decisive questions to the user, hence forcing important modeling decisions, the exploration of class extension relationships guarantees logical completeness and at the same time reveals unintended logical consequences. This way, RELExO not only supports the user in a stepwise refinement of the ontology, but also helps to ensure the compatibility of a logical axiomatization with the user's conceptualization.

Finally, the exploration leads to an ontology that is expressive enough to provide unique answers to all the possible queries (such as subsumption or class membership) with regard to the set of atomic classes in the exploration scope. This type of completeness increases the overall quality of the ontology and its usefulness for any sort of reasoning-based application, since it enables new conclusions and the derivation of previously unknown facts. When it comes to modeling object properties and complex domain-range restrictions, however, concentrating on classes and class subsumption relationships is not a suitable way any more to acquire the knowledge we need. For this reason, we implemented a complementary approach, which uses *role exploration* to support the acquisition of complex domain-range restrictions [VR08a]. In this section, we describe in depth the conceptual design and implementation of this approach and motivate its feasibility by means of a practical ontology engineering scenario.

2.2.1 Introduction

Ontologies constitute the backbone of Semantic Web technologies and hence provide an essential ingredient for Web Intelligence. The increasing uptake of these technologies by industry intensifies the need for medium- to large-size, yet expressive and high-quality ontologies. However, the modeling and maintenance tasks required to satisfy those needs easily surpass the capabilities of human knowledge engineers if not thoroughly assisted by automatic or semi-automatic methods. The incompleteness of information or even modeling errors often remain undetected until their accidental discovery due to unexpected query results.

Let us illustrate this claim by a small example: Querying our institute's Web portal ontology, SWRC (Semantic Web for Research Communities), for all class members of PhDStudent, we find among the results several post-doctoral researchers like Philipp Cimiano. Since Philipp's PhD thesis is recorded properly in SWRC, the ontology intuitively provides enough evidence for him *not* being a PhD student any more. However, the necessary background knowledge needed to logically infer this piece of information is obviously missing, and a class Postdoc does not exist in SWRC.

In general, it is not always obvious which kind of knowledge is required, nor to formalize it in a way that is consistent with the existing knowledge base. In our case, a person's position within the academic staff – e.g., postdoctoral researcher, PhD student or undergraduate – is strongly correlated with his or her authorship properties. Hence, it seems sensible to add more background knowledge specifying which publications and authors can take part in an authorship relation. We will refer to these kinds of binary relations as *roles*.

In the following, we propose a systematic methodology based on a reasoner-aided approach to ontology acquisition and refinement, which combines techniques from natural language processing (NLP) with formal concept analysis (FCA). We start by elaborating on our conceptual framework that integrates the core components of our implementation (Section 2.2.2). Preliminaries – our ontology model and basics of FCA – are introduced in Section 2.2.3, where we also describe the refinement of complex class descriptions and the acquisition of missing subsumption relationships. Section 2.2.4 introduces our approach to a semi-automatic specification of complex domain-range restrictions as well as its implementation in the RoLExO application. A detailed example in Section 2.2.5 illustrates the integrated use of our method within the overall process of ontology refinement. Section 2.2.6 concludes with some related and future work.

As we will see, our methodic approach to ontology enrichment can help to increase the expressive power of any ontology on the Web. By enabling ontology-based applications to draw additional conclusions about previously hidden class memberships while at the same time revealing undesired logical implications, it

fosters Web and Intranet Intelligence.

2.2.2 Ontology Refinement Process

The overall process of semi-automatic ontology refinement, that we envision for future ontology engineering environments, consists of four steps.

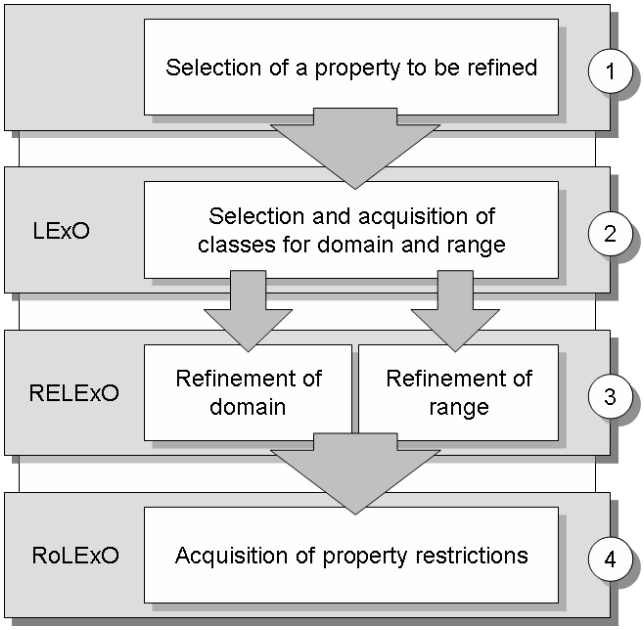

Figure 2.1: Ontology Refinement Process

Step 1: *Selection of a role to be refined.* In the first step the user, possibly assisted by automatically computed heuristics, selects a role whose domain and range characteristics are to be refined.

Step 2: *Specification of relevant class descriptions.* Users are asked to name a set of classes that they consider relevant for the domain (or range, respectively) of the role. These classes (like PhDStudent or Person) constitute the focus for the subsequent exploration process (see *Step 3* and *Step 4*). If any of these classes is not yet specified in the ontology, the users are prompted to enter a natural language definition for it (e.g. "A *postdoc* is a graduate who has written a doctoral thesis."). An ontology learning component automatically transforms this informal definition into a complex class description, whose parts are mapped to already existing, atomic classes or roles.

Step 3: *Refinement of domain and range.* Once the focus of the exploration (i.e. the classes most relevant for domain or range, respectively) has been defined, the user is recommended to complete the ontology with respect to these classes.[2] For this purpose, he or she is guided through an interrogation process in the course of which the system asks smart[3] questions with respect to the domain (e.g. "Can an article be a PhD thesis?"). By answering them and occasionally giving counterexamples the user efficiently acquires missing subsumption and disjointness relationships.

Step 4: *Acquisition of role restrictions.* In the final step of the ontology refinement process, users are supported in specifying complex domain-range restrictions for a given role (e.g. AUTHOR). Again, they are guided by a methodical interrogation, whose implementation (cf. Section 2.2.4) relies on a reasoner for minimizing the user involvement. The result of this phase is a set of axioms that constrain the allowed usage of the initially selected role.

[2]This step is optional, but it significantly facilitates and shortens the acquisition of domain-range restrictions that follows (*Step 4*).
[3]These questions are smart in a sense that they are non-redundant and will be posed to the user only if they cannot be answered automatically by a reasoner or ontology learning component.

The advantages of our approach are manifold:

Integration of lexical and logical approaches to knowledge acquisition. The NLP-based ontology learning component assists the user in formalizing her knowledge and increasing the expressivity of the ontology by adding complex class descriptions (*Step 2*). The subsequent exploration of domain and range (*Step 3*), helps to clarify previously underspecified logical dependencies, thereby integrating the newly acquired axioms into the ontology. Implementations of additional, automatic experts allow for a seamless integration of supplementary lexical or logical ontology acquisition methods into the engineering process.

Efficient acquisition of logically complete knowledge. Relational exploration of classes and role restrictions guarantees completeness,[4] while at the same time minimizing the human modeling effort. Guiding the user through the ontology refinement process by asking smart questions, the system enforces modeling decisions that might have otherwise been ignored though being important for an appropriate representation of the user's domain knowledge. Since it is most often easier for humans to give concrete examples than to come up with abstract axioms, our approach facilitates the acquisition of even complex subsumption relationships (*Step 3*) and generalized domain-range restrictions (*Step 4*). The reasoner support underlying our implementation reduces the amount of user interaction required in the exploration process even further and ensures logical consistency of the ontology.

Interactive evaluation of learned or manually engineered ontologies. The systematic exploration of class and role extension relationships helps to detect underspecified logical dependencies. At the same time, modeling errors and wrong facts are relentlessly revealed by unforeseen questions and automatically retrieved, erroneous counterexamples. By interactively refining the ontology, the user can effectively determine whether a given formalization of domain knowledge matches her conceptualization.

Open-source Implementation. Our implementation is open-source and publicly available under the LGPL license. Sources, binaries and examples can be downloaded from a dedicated Web page.[5]

2.2.3 Preliminaries

In what follows, we will introduce the preliminaries of knowledge specification in the description logic $\mathcal{SHOIN}$ as well as the basic notions of formal concept analysis (FCA), a mathematical discipline also dealing with conceptual knowledge specification.

Knowledge Specification in $\mathcal{SHOIN}$. The description logic $\mathcal{SHOIN}$ serves as the theoretical basis for the Web Ontology Language OWL DL as defined in [MvH04]. For a thorough treatise on the rich field of description logics, see [BCM$^+$03].

A $\mathcal{SHOIN}$ knowledge base (KB, also: ontology) is based on sets N_R (*role names*) $\mathbf{C}$ (*atomic classes*) and $\mathbf{I}$ (*individuals*).[6] In the following, we leave this vocabulary implicit and assume that A, B are atomic classes, a, b, i are individuals, and R, S are roles. Those can be used to define class descriptions employing the constructors from the upper part of Table 2.1. We use C, D to denote class descriptions. Moreover, a $\mathcal{SHOIN}$ KB consists of two finite sets of axioms that are referred to as *TBox* and *ABox*. The possible axiom types for each are displayed in the lower part of Table 2.1.[7] We use the common model-theoretic semantics for $\mathcal{SHOIN}$: an interpretation $\mathcal{I}$ consists of a set Δ called *domain* together with a function $\cdot^{\mathcal{I}}$ mapping individual names to elements of Δ, class names to subsets of Δ, and role names to subsets of $\Delta \times \Delta$. This function is then extended to complex expressions and axioms (cf. Table 2.1).

The following tiny examples illustrate how knowledge specification in $\mathcal{SHOIN}$ works: The fact that Sydney is an Australian city could be expressed by $\mathsf{City} \sqcap \mathsf{Australian}(\mathsf{Sydney})$, whereas Nicole being an inhabitant

[4] Full completeness is not mandatory as users may quit the exploration at any point of time, e.g., if they are tired of answering questions or already satisfied with the results. In this case, they can either stay with the partially refined ontology or resume the exploration later on.

[5] `http://relexo.ontoware.org`

[6] In DL settings one usually speaks of *concepts* and *roles*, the synonym terms used in OWL are *classes* and *properties*. For clarity, we will consequently use *classes* but *roles*.

[7] As usual, we require to restrict number restrictions to simple roles, being (roughly speaking and omitting further technical details) roles without transitive subroles.

Name	Syntax	Semantics	
inverse role	R^-	$\{(x,y) \mid (y,x) \in R^{\mathcal{I}}\}$	
top	$\top$	Δ	
bottom	$\bot$	$\emptyset$	
nominal	$\{i\}$	$\{i^{\mathcal{I}}\}$	
negation	$\neg C$	$\Delta \setminus C^{\mathcal{I}}$	
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	
universal restriction	$\forall R.C$	$\{x \mid (x,y) \in R^{\mathcal{I}}$ implies $y \in C^{\mathcal{I}}\}$	
existential restriction	$\exists R.C$	$\{x \mid$ for some $y \in \Delta,\ (x,y) \in R^{\mathcal{I}}, y \in C^{\mathcal{I}}\}$	
(unqualified) number restriction	$\leq n\, R$	$\{x \mid \#\{y \in \Delta \mid (x,y) \in R^{\mathcal{I}}\} \leq n\}$	
	$\geq n\, R$	$\{x \mid \#\{y \in \Delta \mid (x,y) \in R^{\mathcal{I}}\} \geq n\}$	
role inclusion	$S \sqsubseteq R$	$S^{\mathcal{I}} \subseteq R^{\mathcal{I}}$	TBox
transitivity	$\mathsf{Trans}(S)$	$S^{\mathcal{I}}$ is transitive	TBox
general class inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	TBox
class membership	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	ABox
role membership	$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$	ABox

Table 2.1: Concept constructors and axioms in $\mathcal{SHOIN}$

of Sydney can be represented as INHABITANTOF(Nicole, Sydney). We can further model the proposition that all Australian state capitals are located at the coast (Australian $\sqcap$ State_capital $\sqsubseteq \exists$LOCATEDAT.Coast) and make the claim that all Australian cities have only polite and beautiful citizens: Australian $\sqcap$ City $\sqsubseteq$ $\forall$INHABITANTOF$^-$.(Polite $\sqcap$ Beautiful). Note the knowledge increase obtainable by this role restriction! Given the above facts about Sydney and Nicole, one can derive the class memberships Polite(Nicole) and Beautiful(Nicole).

	east_coast	south_cost	population_3.000.000+	founded_1.800+
Sydney	×		×	
Melbourne		×	×	×
Brisbane	×			×
Perth				×
Adelaide		×		×

Table 2.2: Example context.

Formal Concept Analysis (FCA). The primal notion of formal concept analysis (see [GW97]) is that of a FORMAL CONTEXT $\mathbb{K} := (G, M, I)$, which formally consists of two sets G (objects) and M (attributes) as well as an incidence relation $I \subseteq G \times M$. Intuitively, it can be conceived as a cross table indicating whether an object has an attribute as depicted by Figure 2.2.

An important means of expressing information in FCA is via ATTRIBUTE IMPLICATIONS, written as $A \rightarrow B$ where A and B are sets of attributes: $A \rightarrow B$ is valid in a given formal context, if every object that has all attributes from A also has all attributes from B. In our example, $\{\text{founded\_1800+}, \text{pop3000000+}\} \rightarrow \{\text{south\_cost}\}$ would be a valid implication, expressing that all Australian cities established after 1800 with more than 3 million inhabitants are situated at the south cost.

The method of *attribute exploration* [Gan84] tackles the problem of determining all implications valid in a formal context which might be still partially unknown to the computer (but is completely known by a human expert). Essentially, the algorithm enumerates non-redundant, hypothetic implications and asks the human expert for their validity in a domain of interest. The expert then has to decide: if the implication in question is

valid, it will be added to an implication set called the STEM BASE. If not, the expert has to provide an object (the so-called COUNTEREXAMPLE) that refutes the hypothesis. In the end, after this algorithm has terminated, the acquired knowledge (consisting of implications and counterexamples) is complete in the following sense: every implication is either a logical consequence of the stem base or there is a recorded counterexample for it.

Explorative Refinement of Class Interdependencies. Comparing the knowledge representation approaches of DLs and FCA, one finds considerable similarities: FCA objects can be conceived as individuals of a KB, while FCA attributes resemble DL classes with the formal context displaying class memberships. In particular, for a given set of DL class descriptions $\mathcal{C}$ and a DL interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, one can construct the formal context $\mathbb{K}_{\mathcal{I},\mathcal{C}} := (\Delta, \mathcal{C}, \models)$ (where $\delta \models \mathsf{C}$ denotes $\delta \in \mathsf{C}^{\mathcal{I}}$). In this context, an arbitrary implication $\mathsf{C}_1, \ldots \mathsf{C}_n \rightarrow \mathsf{D}_1, \ldots \mathsf{D}_m$ is valid exactly if the class subsumption axiom $\mathsf{C}_1 \sqcap \ldots \sqcap \mathsf{C}_n \sqsubseteq \mathsf{D}_1 \sqcap \ldots \sqcap \mathsf{D}_m$ is valid in $\mathcal{I}$ (cf. [Rud06], Theorem 4.2). Hence, attribute exploration techniques can be used for the interactive refinement of an ontology's class interdependencies. Existing approaches [Baa95, Rud04, Rud06, BGSS07, VR08b] mainly differ in the supported logical fragment to be explored and the opportunity of providing only partially specified counterexamples.

This refinement of class hierarchies, more specifically, the acquisition of missing subsumption relationships (*Step 2* and *3* in Section 2.2.2), is supported by our previous implementation of reasoner-aided relational exploration with partial contexts [VR08b]. In the following, we will therefore concentrate on the more ambitious goal of acquiring fine-grained role restrictions (*Step 4*) as motivated in the introduction.

2.2.4 Acquisition of Role Restrictions

As mentioned earlier, most exploration methods focus on the interdependencies of class descriptions, whereas roles merely occur as building blocks for complex class descriptions. However, as argued in our introductory example, it is worth focusing on specific roles and to model logical interrelationships between its domain and range individuals, since those can entail valuable conclusions about class memberships.

Indeed, the use of the attribute exploration technique from formal concept analysis is not limited to the exploration of the conceptual hierarchy of class descriptions (and their conjunctions). Rather, by using other mappings from interpretations to formal contexts, it is possible to explore different logical fragments. Recently, the fragment of generalized domain-range restrictions (GDRRs) has been proposed as a both interesting and intuitive fragment eligible for exploration [Rud08].

Given a set $\mathcal{C}$ of named classes and a role R, a GENERALIZED DOMAIN-RANGE RESTRICTION (short: GDRR) is a rule

$$\mathsf{R}(X,Y) \wedge \bigwedge_{\mathsf{A}\in\mathbf{A}} \mathsf{A}(X) \wedge \bigwedge_{\mathsf{B}\in\mathbf{B}} \mathsf{B}(Y) \rightarrow \bigwedge_{\mathsf{C}\in\mathbf{C}} \mathsf{C}(X) \wedge \bigwedge_{\mathsf{D}\in\mathbf{D}} \mathsf{D}(Y) \quad (*)$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D} \subseteq \mathcal{C}$ and R is a role name. Note that for $\mathbf{C} \cup \mathbf{D} = \emptyset$, the rule will have an empty head (also denoted by $\square$) and hence, will be interpreted as integrity constraint. So, the GDRR presented in the above definition would mean the following: "*For any two elements X and Y of the domain of interest that are connected by a role* R *and where X fulfills (all of) $\mathbf{A}$ as well as Y fulfills (all of) $\mathbf{B}$, we know that X additionally fulfills $\mathbf{C}$ and Y additionally fulfills $\mathbf{D}$.*"

It has been proven that every GDRR can be equivalently expressed by a DL axiom. The rule scheme (*) corresponds to the axiom:

$$\prod_{\mathsf{A}\in\mathbf{A}} \mathsf{A} \sqcap \exists \mathsf{R}.\Big(\prod_{\mathsf{B}\in\mathbf{B}} \mathsf{B}\Big) \sqsubseteq \prod_{\mathsf{C}\in\mathbf{C}} \mathsf{C} \sqcap \forall \mathsf{R}.\Big(\Big(\bigsqcup_{\mathsf{B}\in\mathbf{B}} \neg\mathsf{B}\Big) \sqcup \Big(\prod_{\mathsf{D}\in\mathbf{D}} \mathsf{D}\Big)\Big).$$

GDRRs allow for the specification of semantic ramifications caused by the presence of a specific role between two individuals (some of whose class memberships are also known). Exploring the fragment of GDRRs therefore enables a novel role-focused way of ontology refinement.

Technically this is done – as opposed to all other current exploration methods for ontologies – by exploring a formal context whose objects are *not* single individuals from the knowledge base (like Sydney, see Figure 2.2), but *pairs* of individuals which instantiate the role R. Consequently, also the counterexamples are individual pairs.

For a given interpretation $\mathcal{I}$ together with two sets $\mathcal{C}_{\text{domain}}$ and $\mathcal{C}_{\text{range}}$ of named classes and a role R, the ROLE CONTEXT $\mathbb{K}_{\mathsf{R}}$ is defined as formal context (G, M, I) with

- $G := \mathsf{R}^{\mathcal{I}} = \{(\delta_1, \delta_2) \mid \delta_1, \delta_2 \in \Delta, (\delta_1, \delta_2) \in \mathsf{R}^{\mathcal{I}}\}$
- $M := \{\mathsf{C}_1 \mid \mathsf{C} \in \mathcal{C}_{\text{domain}}\} \cup \{\mathsf{C}_2 \mid \mathsf{C} \in \mathcal{C}_{\text{range}}\} \cup \{\bot\}$
- $I \subseteq G \times M$ with $(\delta_1, \delta_2) \not I \bot$ and $(\delta_1, \delta_2) I \mathsf{C}_1 \iff \delta_1 \in \mathsf{C}^{\mathcal{I}}$ as well as $(\delta_1, \delta_2) I \mathsf{C}_2 \iff \delta_2 \in \mathsf{C}^{\mathcal{I}}$

The following theorem shows how the validity of a GDRR in an interpretation can be read from a corresponding role context.

Theorem 1 An interpretation $\mathcal{I}$ satisfies a GDRR of the shape (*) if and only if the corresponding role context $\mathbb{K}_{\mathsf{R}}$ satisfies the implication

$$\{\mathsf{A}_{\mathrm{d}} \mid \mathsf{A} \in \mathbf{A}\} \cup \{\mathsf{B}_{\mathrm{r}} \mid \mathsf{B} \in \mathbf{B}\} \to \begin{cases} \bot & \text{if } \mathbf{C} \cup \mathbf{D} = \emptyset, \text{otherwise:} \\ \{\mathsf{C}_{\mathrm{d}} \mid \mathsf{C} \in \mathbf{C}\} \cup \{\mathsf{D}_{\mathrm{r}} \mid \mathsf{D} \in \mathbf{D}\}. \end{cases}$$

This theorem enables us to "translate" any implication in a role context into an equivalent GDRR and via the abovementioned correspondence further into a DL axiom.

Now, the basic idea for the knowledge acquisition method we are going to propose is to carry out attribute exploration on the context $\mathbb{K}_{\mathsf{R}}$. Thereby, our basic assumption is that there exists a distinguished interpretation $\mathcal{I}'$ entirely (but implicitly) known by the human expert that we want to specify in terms of GDRRs.

After role exploration, the updated knowledge base is *complete* in the following sense: any GDRR (referring to R, $\mathcal{C}_{\text{domain}}$ and $\mathcal{C}_{\text{range}}$) can either be deduced from the updated knowledge base or there is a pair of individuals in the ontology witnessing that this GDRR does not hold.

We instantiated our approach described in Section 2.2.4 by implementing RoLExO[8] (*Role Exploration for Learning Expressive Ontologies*), an interactive application supporting the acquisition of complex role restrictions (cf. *Step 4* in Section 2.2.2). The architecture of RoLExO relies upon KAON2[9] as an ontology management back-end and features a simple graphical user interface. The ontology refinement process, depicted by Figure 2.2, is handled by a role exploration component which manages a partial context and an implication set. Both are updated based on answers obtained from the "expert team" constituted by a KAON2 reasoner and the human knowledge engineer.

As we will see in Section 2.2.5, RELExO and RoLExO can be synergetically combined in a process of interactive ontology refinement.

2.2.5 Example Scenario

We now illustrate the practical relevance of our approach to ontology refinement by means of a real-world example.

Ontology. The SWRC (Semantic Web for Research Communities)[10] [SBH+05] ontology is a well-known ontology modeling the domain of Semantic Web research. Version 0.3 containing 55 classes as well as 41 roles serves as a basis for the AIFB Web portal[11] which manages information about 2,982 persons, projects, and publications. For our experiment, we exported all the instance data stored in the AIFB portal into one single OWL file (more than 5 MB in RDF syntax), and merged it with the corresponding TBox, i.e. the imported version of SWRC. This way, we obtained an ontology with 33,426 axioms.

[8] http://relexo.ontoware.org
[9] http://kaon2.semanticweb.org
[10] http://ontoware.org/projects/swrc/
[11] http://www.aifb.uni-karlsruhe.de

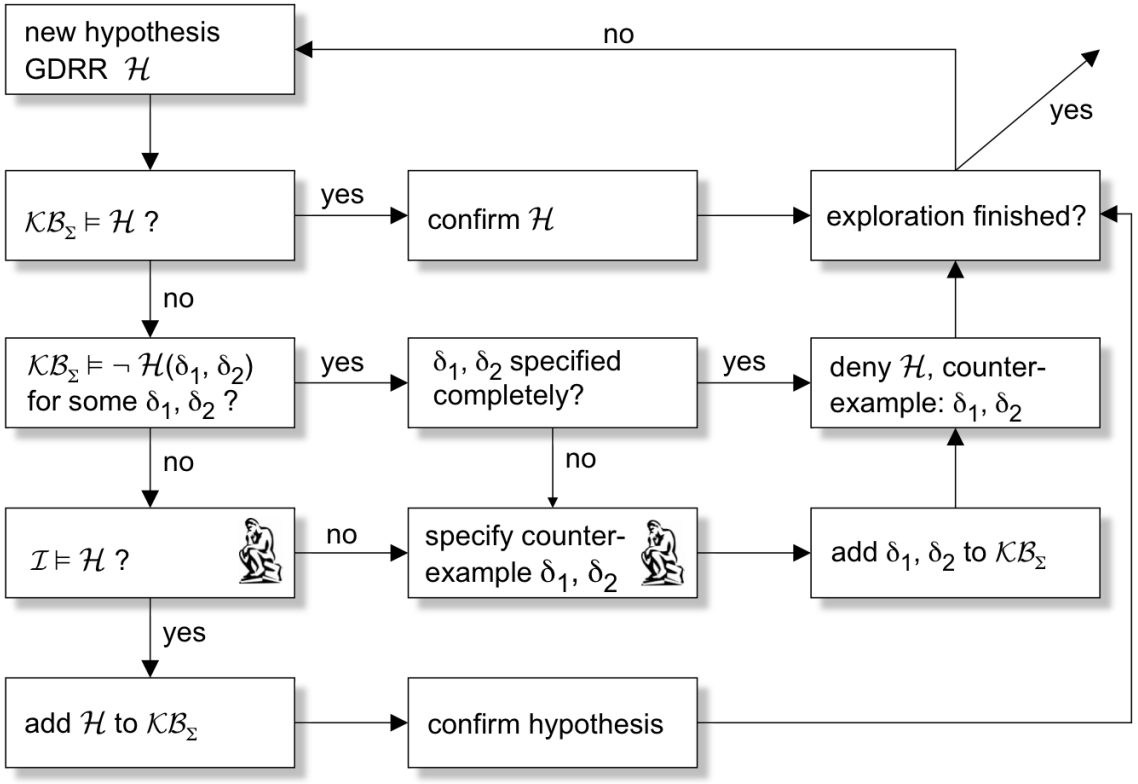

Figure 2.2: RoLExO Implementation. "The Thinker" indicates user involvement.

Refinement Process. As motivated in Section 2.2.1, we first decide to refine the AUTHOR role of the SWRC ontology (*Step 1*, Section 2.2.2). Subsequently, we select several classes that we assumed to be relevant for the domain and range of this role (*Step 2*), thereby noticing that an equivalent to "postdoctoral researcher" is missing in the ontology. After some discussion about our understanding of this class, we enter the following definition into LExO [VHC07] (see also D3.8.1 [VB08], Chapter 4), a tool for transforming natural language definitions into DL class descriptions.

"*A postdoc is a graduate who has written a doctoral thesis.*"

The result is a set of axioms representing a formal description of the class a_postdoc.[12]

$$\begin{aligned}
\text{a_postdoc} &\equiv \text{a_graduate_who_has_written_a_doctoral_thesis} \\
\text{a_graduate_who_has_written_a_doctoral_thesis} &\equiv \text{a_graduate} \sqcap \text{has_written_a_doctoral_thesis} \\
\text{has_written_a_doctoral_thesis} &\equiv \exists \text{HAS_WRITTEN.a_doctoral_thesis} \\
\text{a_doctoral_thesis} &\equiv \text{a_thesis} \sqcap \text{doctoral}
\end{aligned}$$

Since some of the atomic classes obviously map to existing classes in SWRC, we further add the following mapping axioms: swrc:Thesis $\equiv$ lexo:thesis, swrc:Graduate $\equiv$ lexo:a_graduate, swrc:PhDThesis $\equiv$ lexo:a_thesis $\sqcap$ lexo:doctoral and lexo:has_written $\sqsubseteq$ swrc:author$^-$. The resulting, extended version of SWRC is input to the exploration process described in the sequel.

Refinement of Domain and Range. We only briefly display the results of the two relational exploration processes dedicated to the domain and range of the considered AUTHOR role (cf. Section 2.2.2, *Step 3*). For a more detailed description, we refer the reader to [VR08b]. Altogether, the following 7 new axioms were acquired during the process:

[12] a_postdoc $\equiv$ a_graduate $\sqcap \exists$HAS_WRITTEN.(a_thesis $\sqcap$ doctoral)

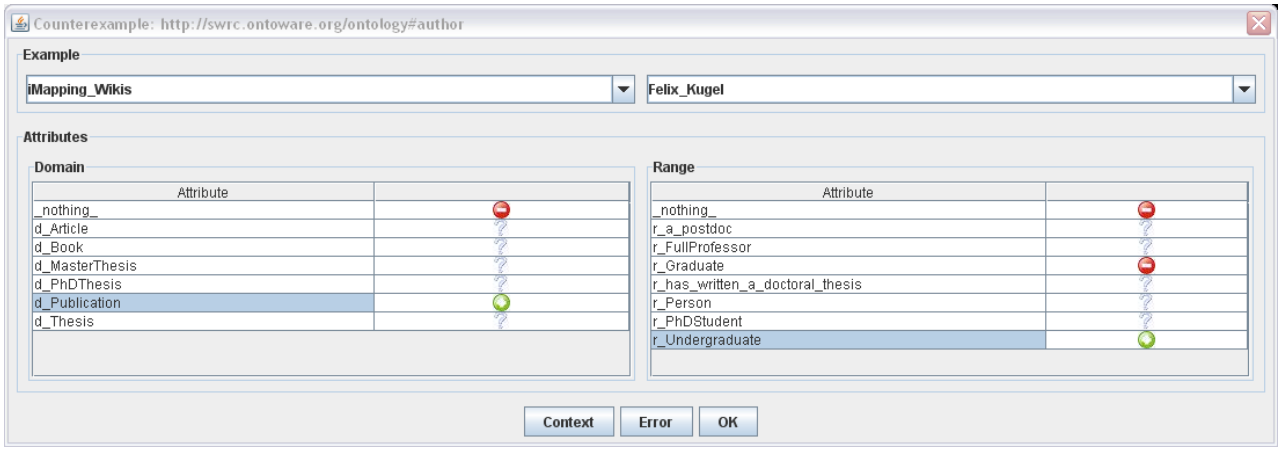

Figure 2.3: Entering a counterexample, i.e. a pair of individuals related by the *author* role.

$$\begin{aligned} \text{PhDThesis} &\sqsubseteq \text{Book} \\ \text{Article} \sqcap \text{Thesis} &\sqsubseteq \bot \\ \text{Book} \sqcap \text{Thesis} &\sqsubseteq \text{PhDThesis} \\ \text{Book} \sqcap \text{MasterThesis} &\sqsubseteq \bot \\ \text{FullProfessor} &\sqsubseteq \text{a\_postdoc} \\ \text{PhDStudent} &\sqsubseteq \text{Graduate} \\ \text{has\_written\_a\_doctoral\_thesis} &\sqsubseteq \text{Graduate} \end{aligned}$$

Acquisition of Role Restrictions. Subsequent to these preprocessing steps, we are now ready to carry out the role exploration focusing on the AUTHOR role as well as the classes Article, Book, MasterThesis, PhDThesis, Publication, Thesis on the domain side and the classes a_postdoc, FullProfessor, Graduate, has_written_a_doctoral_thesis, Person, PhDStudent, Undergraduate on the range side (see *Step 4* in Section 2.2.2).

After a few trivial hypotheses, which can be refuted by automatically retrieved counterexamples, the first question requiring human interaction is posed. It asks for the validity of the rule $\text{author}(X, Y) \rightarrow \text{Publication}(X) \wedge \text{Graduate}(Y) \wedge \text{Person}(Y)$, investigating whether, if some X is authored by some Y, X must be a publication and Y must be both a graduate and a person. As there are no records in SWRC of any authors not being graduates, no counterexample can be brought up automatically. Instead, after denying this hypothesis, the human expert is asked to add a publication-author pair disproving it. We do so by adding information about a workshop paper [HVK06] and its student author, Felix Kugel (cf. Figure 2.3).

The next hypothesis brought up by the system, $\text{author}(X, Y) \rightarrow \text{Publication}(X) \wedge \text{Person}(Y)$, can be clearly confirmed by the human expert as it gives straightforward domain and range restrictions for the author role: every author is a person and everything authored is a publication.

In the subsequent execution, the algorithm poses the GDRR $\text{author}(X, Y) \wedge \text{Book}(X) \rightarrow \text{Graduate}(Y)$ expressing that everybody publishing a book must be a graduate. Though quite arguable in general, in the scientific area considered by us, we judge that this axiom constitutes a sensible restriction and therefore confirm it.

As opposed to the previous, the next presented potential axiom, $\text{author}(X, Y) \wedge \text{Thesis}(X) \rightarrow \text{Graduate}(Y)$ can be accepted without much argument. Clearly everybody having written a thesis qualifies as a graduate – be it a master or a PhD thesis.

The last axiom in question, displayed in Figure 2.4, encodes that any thesis that has been written by somebody being now a PhD student must be a master thesis. We confirm this (as we assume any thesis to be either a PhD or a master thesis and furthermore, somebody having written a PhD thesis cannot be a PhD student any more).

In the end, we have acquired the following four new axioms that characterize how class memberships of

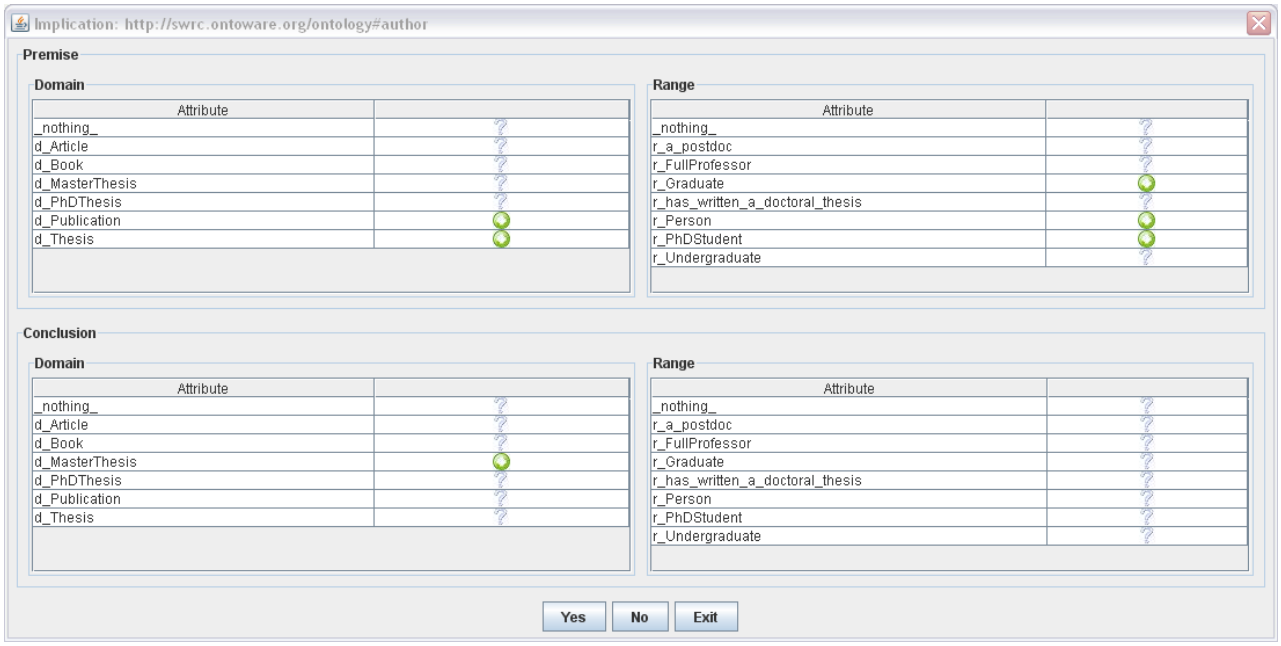

Figure 2.4: How a hypothetic GDRR is displayed.

domain and range individuals "influence" each other:

$$\begin{aligned} \exists \text{AUTHOR}.\top &\sqsubseteq \text{Publication} \sqcap \forall \text{AUTHOR}.\text{Person} \\ \text{Book} \sqcap \exists \text{AUTHOR}.\top &\sqsubseteq \forall \text{AUTHOR}.\text{Graduate} \\ \text{Thesis} \sqcap \exists \text{AUTHOR}.\top &\sqsubseteq \forall \text{AUTHOR}.\text{Graduate} \\ \text{Thesis} \sqcap \exists \text{AUTHOR}.\text{PhDStudent} &\sqsubseteq \text{MasterThesis}. \end{aligned}$$

The fact that this constitutes a rather small set of axioms is thanks to the preprocessing via domain and range exploration. Otherwise, numerous additional questions actually relating to domain- or range-inherent dependencies would have been brought up. By proceeding as we propose, the knowledge acquired in this exploration step only incorporates logical ramifications where both domain and range are involved. A statistical summary of the entire exploration process is given in Table 2.3.

	domain	range	RoLExO	$\sum$
Reasoner-answered questions	9	8	19	36
Questions decided by human	6	5	13	24
New TBox axioms	5	3	4	12
New individuals	1	2	14	17
New class memberships $C(a)$	18	27	36	81
New role memberships $R(a, b)$	0	0	7	7

Table 2.3: Summary of the exploration process. The columns correspond to the different phases of the refinement as described above, i.e. the exploration of *domain* and *range* or the acquisition of role restrictions (*RoLExO*), respectively.

Revisiting our initially described problems, we find them solved. It is now possible to correctly classify our colleague, Philipp Cimiano, (and others) based on ontological knowledge: The fact that Philipp is author of an individual classified as a PhD thesis was originally stated in SWRC. Now, from the knowledge acquired via LExO from natural language resources defining the notion of a postdoc (together with

the introduced mappings), we can infer that Philipp is indeed a postdoc. Likewise, thanks to the GDRR Thesis $\sqcap \exists$AUTHOR.$\top \sqsubseteq \forall$AUTHOR.Graduate, every person specified as being the author of a master thesis will from now on be inferred to be a graduate. Hence, by employing a combination of ontology learning and logical exploration techniques we have considerably increased the knowledge inferable from the SWRC ontology (cf. Table 2.4).

2.2.6 Related Work and Conclusion

The incremental refinement of ontologies has become an important issue that is addressed by several ontology learning tools and methodologies such as Text2Onto [CV05] or the work by Navigli and Velardi [NV06] (for an overview, see also [BC08]). However, only a few of them support the acquisition of logically complex axioms, and most lexically inspired techniques lack a proper integration with reasoning-based methods for ontology evaluation or debugging. This also holds for approaches to learning domain-range restrictions of binary relations [CHR06, CGR$^+$05]. Parallel to these more or less NLP-based ontology learning methods, approaches relying on Inductive Logic Programming (ILP) [LE05, FIPS04, CH94] and Formal Concept Analysis (FCA) [Rud06] have been developed in the logics community. But although there are a few approaches, similar to ours, aiming to reconcile the two worlds of lexical and logical ontology acquisition by either FCA [SM01, CHS05] or ILP [Ned99], none of them has been designed specifically for refining OWL DL ontologies.

	SWRC	RELExO range	RELExO domain	RoLExO
Article	189	189	189	190
Book	36	36	94	95
MasterThesis	0	0	1	4
PhDThesis	58	58	58	59
Publication	1499	1499	1500	1507
Thesis	58	58	59	63
a_postdoc	0	63	63	67
FullProfessor	6	6	6	9
Graduate	52	111	111	139
has_written_a_doctoral_thesis	0	63	63	67
Person	1213	1215	1215	1222
PhDStudent[12]	50	46	46	47
Undergraduate	6	7	7	9
$\sum$	3167	3351	3418	3478

Table 2.4: Inferred class memberships.

In this section, we have therefore sketched a way to combine two complementary approaches to the acquisition and refinement of OWL DL ontologies: the more intentional approach of distilling conceptual information from lexical resources and the extensional method of extracting hypothetical axioms from manually specified or automatically retrieved domain entities. We have instantiated our approach by designing and implementing a framework that integrates Relational Exploration and Role Exploration – two techniques for the systematic refinement of OWL ontologies based on FCA. To the best of our knowledge, RoLExO is the first publicly available implementation of an exploration-based approach to ontology refinement. In an example using the

[12] Exploring the range of AUTHOR for the first time, we noticed a modeling error in SWRC. It was the second question, automatically answered by the reasoner, which brought up the counterexample Peter Haase. A simple query to SWRC revealed that he was explicitly stated to be a PhDStudent and inferred to be a_postdoc. Since we assumed both classes to be disjoint, we changed the explicit classification of Peter and two other postdocs to Person and restarted the exploration with the corrected ontology.

well-known SWRC ontology we have demonstrated the feasibility of semi-automatic ontology refinement, and its applicability to real-world ontology engineering tasks.

After all, we identify several issues for future research. Firstly, we will further extend RoLExO by an additional automatic expert which uses ontology learning techniques or Web resources, including other ontologies, for confirming hypotheses and suggesting counterexamples. Moreover, FCA-based exploration can be applied to a set of classes from multiple distributed knowledge bases, in order to clarify logical dependencies across ontologies. Hence our approach can be extended to support Web ontology alignment and integration. Finally, we will integrate RoLExO into an ontology engineering environment such as the NeOn Toolkit[13] and improve its usability by adding a natural language generation component for translating hypotheses, i.e. logical implications, into natural language questions. In the end, we are confident that our planned user studies will demonstrate the advantages of interactive ontology refinement and we hope that our approach will initiate the development of new tools for enhancing Web Intelligence.

2.3 A NeOn Toolkit Plugin for Ontology Refinement

In this section, we present a tightly-coupled GUI plugin that serves as a graphical frontend for RELExO (see NeOn D3.8.1 [VB08], Chapter 4).

2.3.1 User Guide

Preferences. The *preference page* is accessible from the main menu of Eclipse (*Window* → *Preferences...* → *RELExO Preferences*). As shown by Figure 2.5, it allows for setting a variety of parameters:

- **Find complete ontology examples:** Completely specified counterexamples (i.e. individuals whose attributes are all known) can significantly speed up the exploration process in terms of questions asked. Hence, the ontology expert can be configured to search for complete examples in case it is asked to provide a counterexample for a given hypothesis. However, depending on the size of the underlying knowledge base and its computational complexity, this search can be very time consuming. *Default: false*
- **Print debug information:** If this parameter is set, additional debug information will be printed to the console window or log file. *Default: true*
- **Minimize reasoning:** Depending on the complexity of the underlying ontology, this parameter can significantly speed up the overall exploration process. Especially in the case of very lightweight ontologies, it is possible to avoid many of the costly reasoner calls by checking for explicit subsumption or class instantiation relationships first. *Default: false*
- **Show incomplete ontology examples:** If this flag is set, all of the automatically generated counterexamples, i.e. retrieved by the reasoning-based expert, will be shown to the user. This way, users get the opportunity to specify additional attributes and thus shorten the exploration process in terms of questions asked by the system. *Default: true*
- **Complete user examples:** This parameter triggers the automatic completion of manually specified counterexamples. For each attribute which is left determined by the user, the reasoner tries to infer or disprove the corresponding class membership from the ontology. *Default: true*
- **Open browser:** With this option switched on, the plugin will try to open an individual's URI in a Web browser when this individual is brought up as an automatically generated counterexample. This feature can be very useful in case the plugin is unable to access the label information associated with the entities in a particular ontology. *Default: true*

[13] `http://www.neon-toolkit.org`

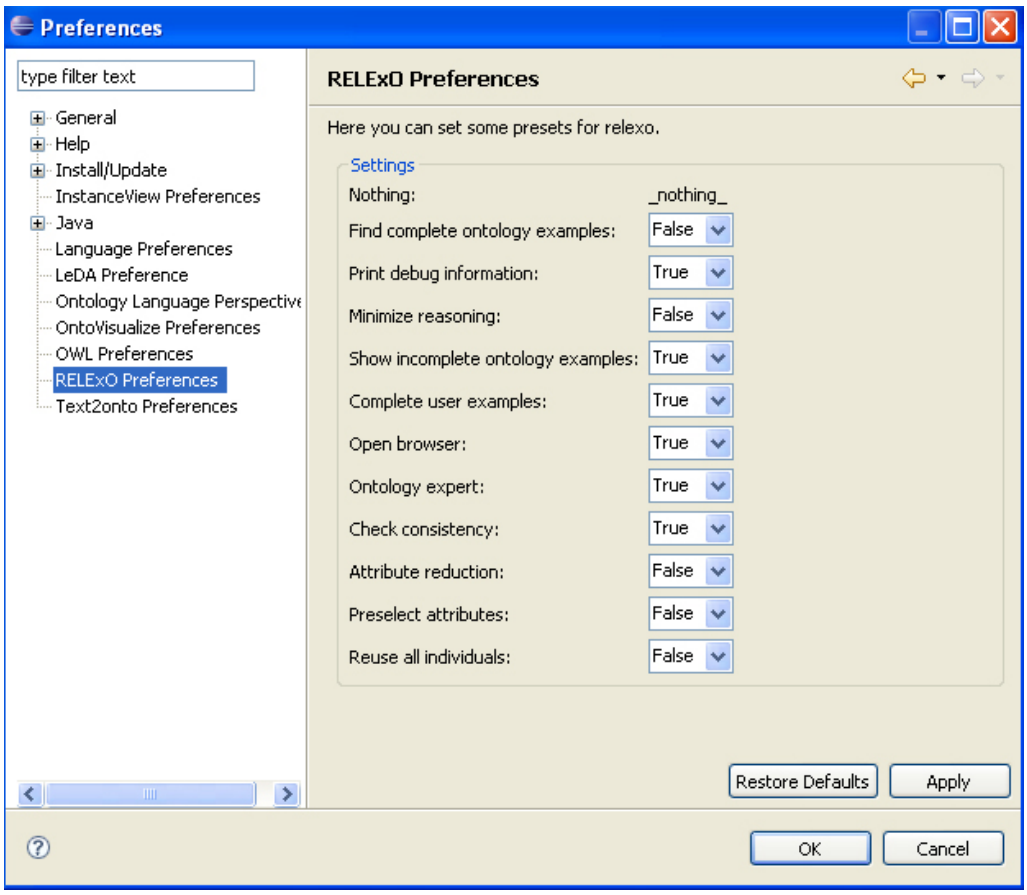

Figure 2.5: RELExO: preference page

- **Ontology expert:** By means of this parameter, users can configure the expert team that takes part in the exploration process. If it is set to true, the human expert will be supported by an ontology reasoner, i.e. KAON2. *Default: true*

- **Check consistency:** If this flag is set, the plugin will check for logical contradictions each time the user enters a new counterexample. *Default: true*

- **Attribute reduction:** In case one of the attributes, i.e. atomic classes in the ontology, is known to be equivalent to the intersection of any other two or more classes, it can be removed from the list of attributes without causing any loss of information. That sort of pruning can be turned on and off by means of this parameter. *Default: false*

- **Preselect attributes:** Whenever the plugin is used to integrate learned class descriptions into an existing ontology (see NeOn D3.8.1 [VB08], Chapter 4), it is advisable to focus on those attributes (i.e. atomic classes) which have been introduced by the respective ontology learning method. By setting this parameter, the plugin can be configured to preselect these attributes from the list of all the atomic classes in the extended ontology (cf. Figure 2.6). *Default: false*

- **Reuse all individuals:** When the human expert is asked to provide a counterexample, he or she can either enter a new individual, choose one from the list of previously specified counterexamples, or – if this parameter is set to *true* – select any individual from the ontology's ABox. Note that a huge ABox will significantly slow down the exploration in case this option is switched on, because the plugin will have to determine the attributes of each individual shown in the list of reusable examples. *Default: false*

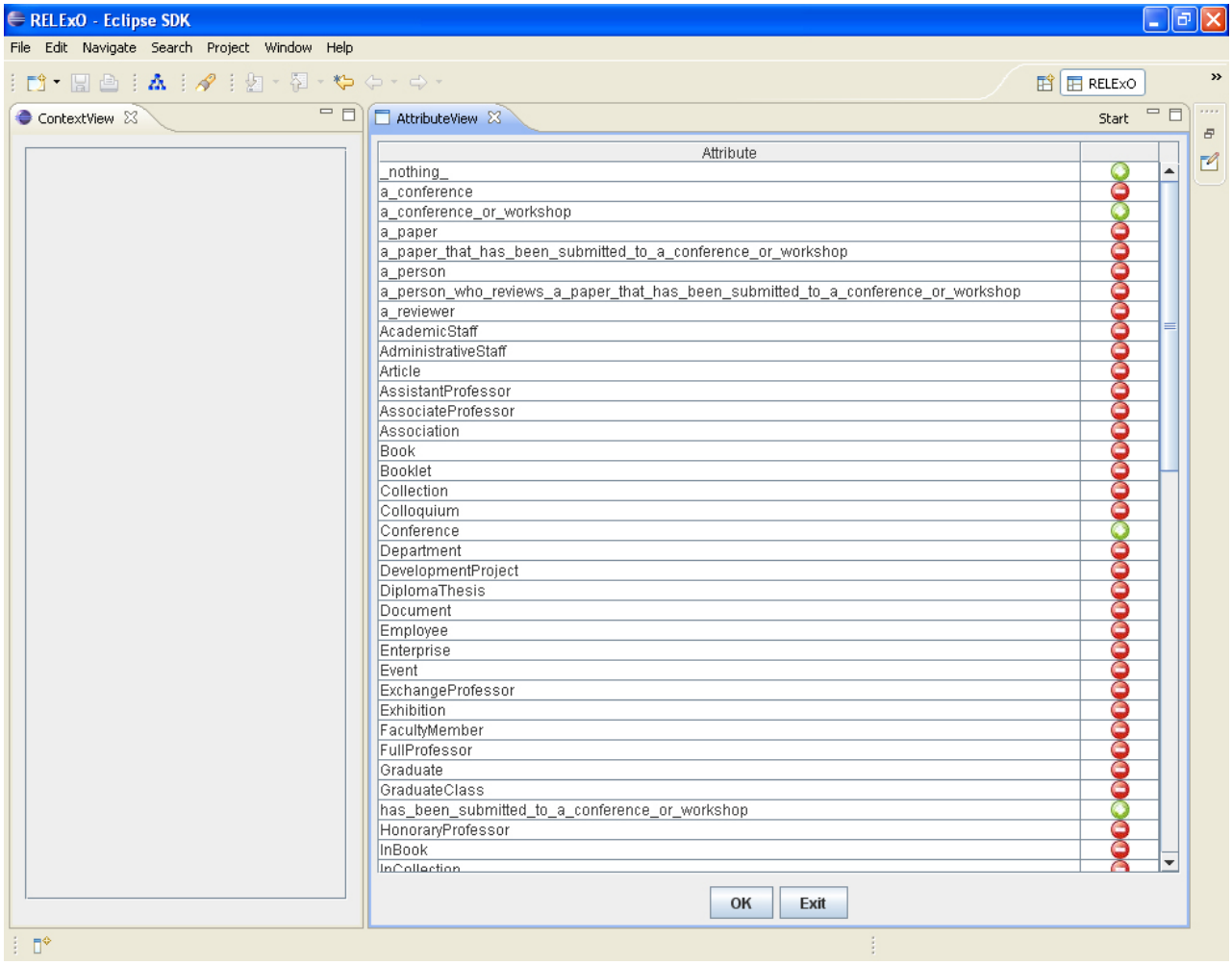

Figure 2.6: RELExO: selection of attributes

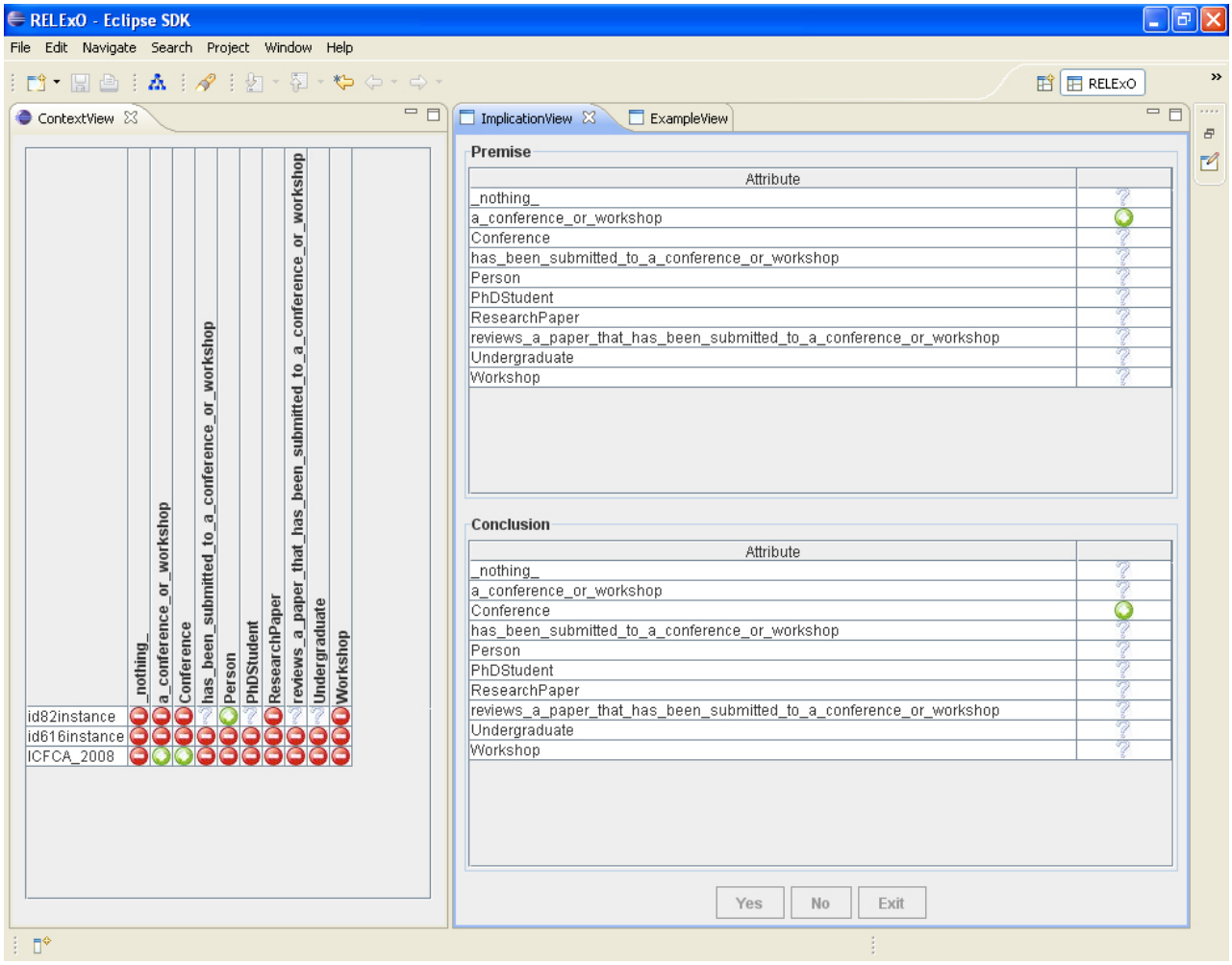

Figure 2.7: RELExO: hypothesis

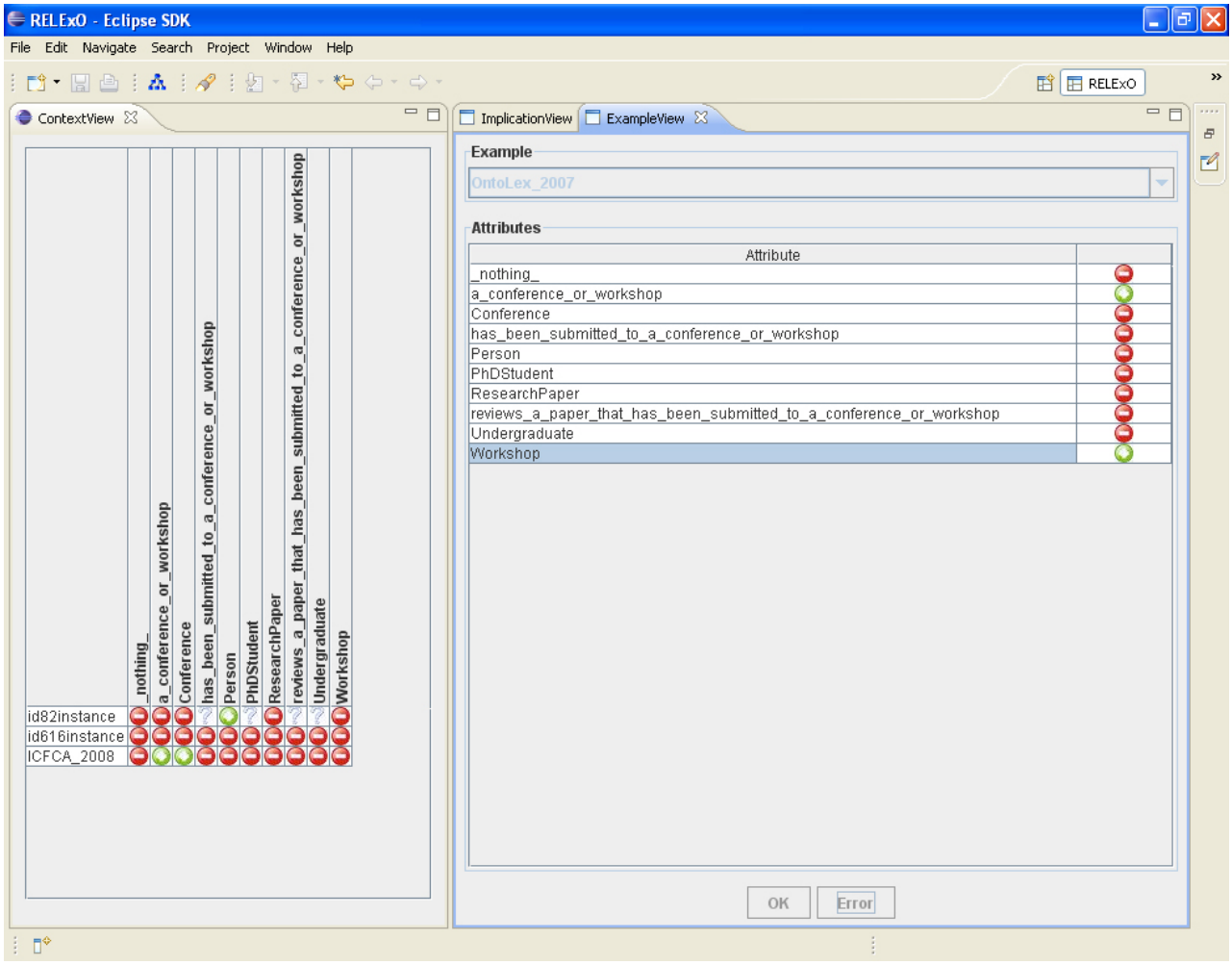

Figure 2.8: RELExO: counterexample

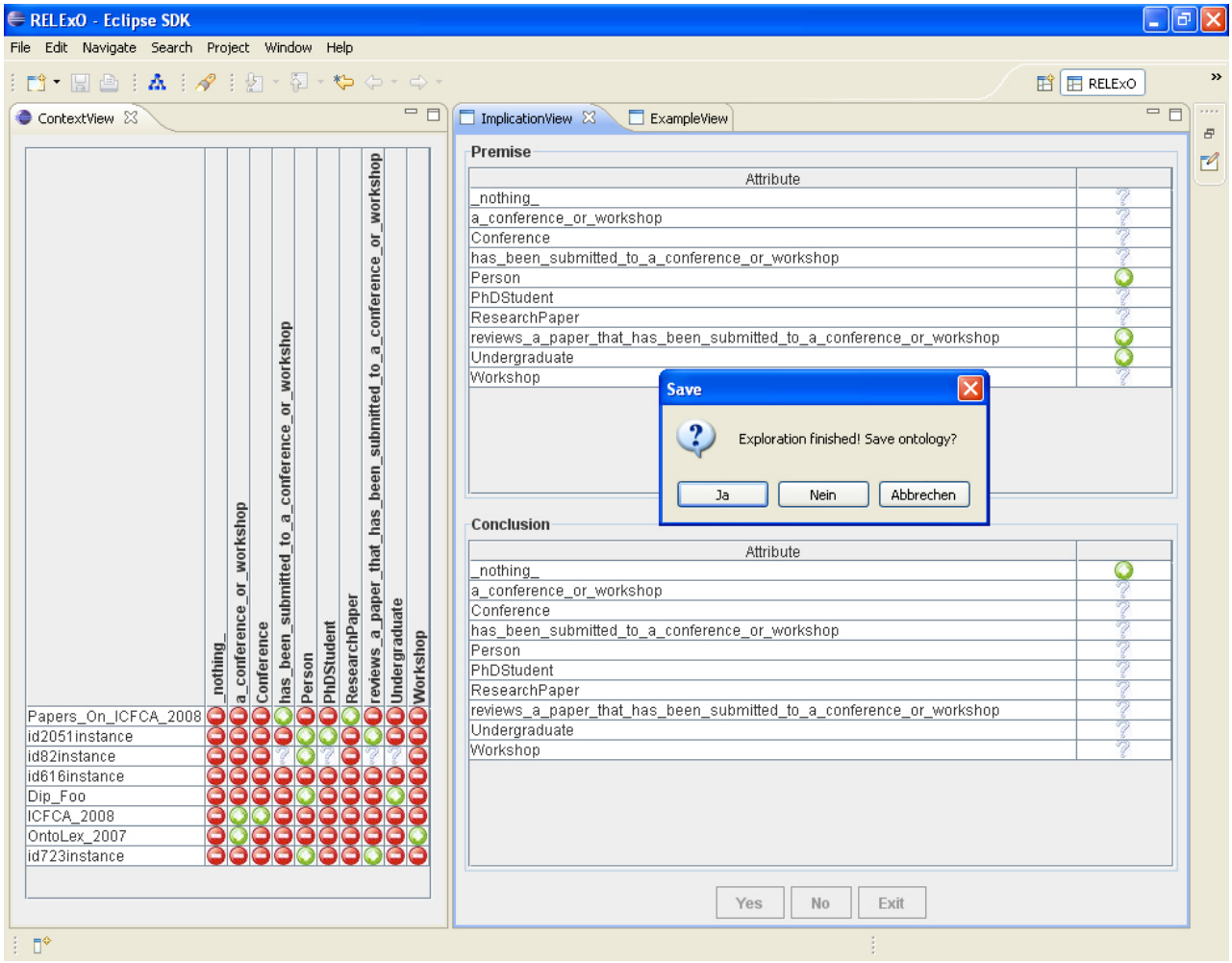

Figure 2.9: RELExO: save ontology

Attribute View. The first step of every ontology refinement process based on formal concept analysis is the manual (or automatic) selection of relevant attributes (cf. Figure 2.6). In the case of standard relational exploration as implemented by the RELExO plugin, any atomic class of the underlying ontology can serve as an attribute. However, since the computational complexity of relational exploration is exponential in the number of selected attributes, it is advisable to choose around ten classes.

Implication View. In the course of the ontology refinement process, the human expert is presented with hypotheses about underspecified subsumption relationships. For example, Figure 2.7 shows the hypothetical axiom Conference_or_workshop $\sqsubseteq$ Conference ("Every conference or workshop is a conference."), which of course should be rejected by the user.

Example View. A counterexample to the hypothesis depicted by Figure 2.7 could be the OntoLex workshop that was held in conjunction with ISWC 2007. Figure 2.8 shows the example view, which opens automatically in case the human expert rejects a hypothesis.

Saving the ontology. When the exploration is finished, i.e. each of the possible hypotheses has either been accepted or disproved by means of a counterexample, the user is prompted to save the refined ontology (see Figure 2.9).

2.3.2 Installation

The installation of the RELExO plugin does not require any third-party software or specific configuration steps that might demand for in depth explanations. Hence, as soon as the official version has been released, the users will be able to install this plugin by using the standard update mechanism of the NeOn Toolkit.

Chapter 3

Modeling Patterns for Ontology Engineering

In D3.8.1 [VB08], a method for automatically matching, retrieving and reusing ontology content design patterns, called OntoCase, was presented. In this chapter a first evaluation of this approach on FAO data is presented. The evaluation aims to show the feasibility of the approach, the characteristics of the output ontologies and that they improve the learnt ontologies that were the input to the method. Through a small evaluation at the end of this chapter we also show that ontology learning systems can improve the performance of OntoCase by first enriching the input ontology. First, we briefly comment on the connection between OntoCase and context sensitivity of ontologies, next we give an overview of OntoCase, then the experiments are presented and their results analysed.

3.1 Context Sensitivity for Networked Ontologies

The notion of ontology patterns, specifically content design patterns, and the usage of such patterns for ontology design is related to context-sensitivity in several ways. Most patterns are extracted from general top-level ontologies, and the reuse of content patterns may thereby be viewed as an alignment of ontologies to top-level ontologies. Below some brief introduction to context sensitivity and content patterns is given.

Ontology engineering patterns. When ontology engineering patterns, specifically ontology content design patterns, are used on top of other OL techniques, as is the aim of the OntoCase approach described already in D3.8.1 [VB08], the patterns represent a set of best practices of the community. Thereby, relating a learnt ontology to a pattern means to relate the learnt ontology to the modelling best practices. Content patterns usually encode domain independent common-sense knowledge that can be included in the constructed ontology by applying the pattern. Such common-sense knowledge adds missing background knowledge and puts the existing ontology elements into their appropriate context.

Alignment to top-level ontologies. Ontology content design patterns are in most cases "pieces" extracted from a larger construct, such as a top-level ontology. In this case, relating the learnt ontology to the pattern additionally means relating to the top-level ontology from which the pattern was extracted. A top-level ontology provides the basic definitions and the axiomatisation that can put the learnt ontology into a broader context. Different top-level ontologies might result in different interpretations of the learnt ontology.

3.2 Overview of OntoCase

In D3.8.1 [VB08], OntoCase was presented as a method that complements other ontology learning (OL) methods and tools, both by improving the results of the other OL methods and by putting the produced ontologies in a context, with respect to patterns and the top-level ontologies that the patterns are extracted from.

OntoCase is a general framework for iterative and experience-based semi-automatic ontology construction, based on the notion of ontology patterns. OntoCase proposes a cycle of learning from experience and using the learnt information to solve new problems. Intuitively patterns are in fact encoded experiences, that are used to solve new problems. In Figure 3.1 an overview of the OntoCase approach is presented, including the tasks that are performed in each of the phases. In D3.8.1 [VB08] and in the implementation used for these experiments only the first two phases of OntoCase were included, i.e. retrieval and reuse of patterns.

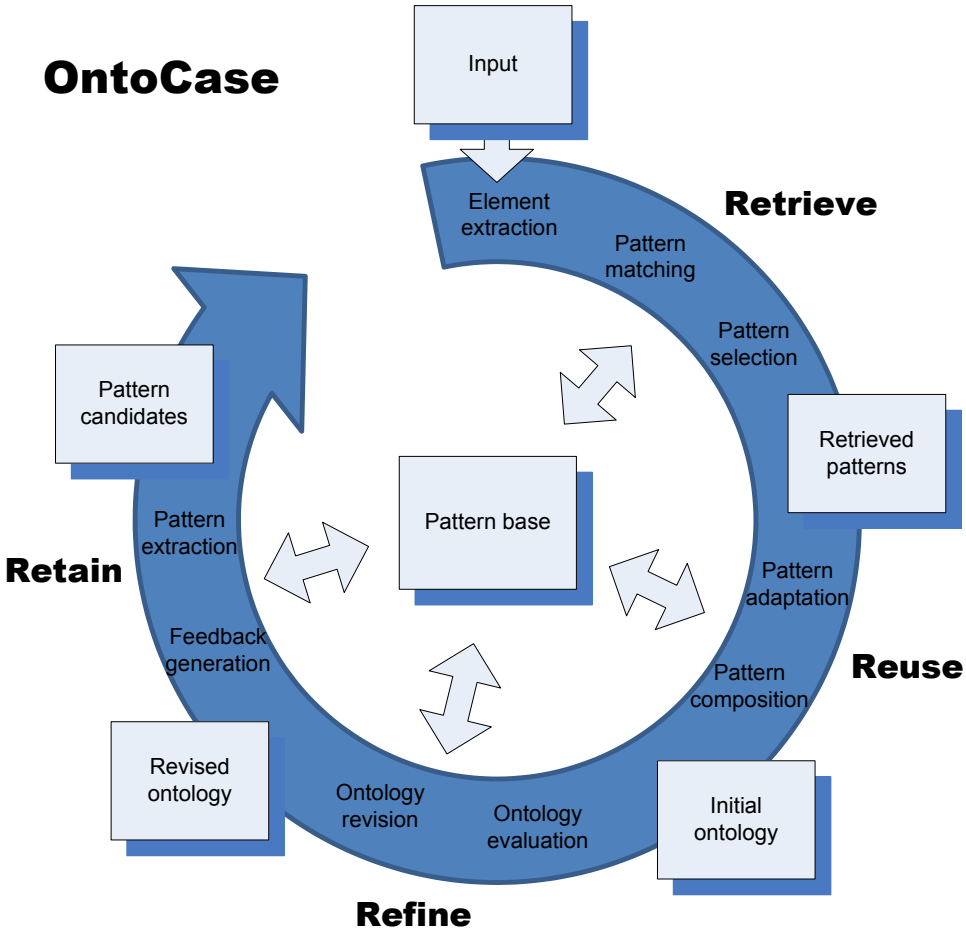

Figure 3.1: The OntoCase framework.

The first phase of OntoCase, called 'retrieve', involves extracting ontological elements from a text corpus, or alternatively inputting a seed ontology, then matching these elements to the available patterns, and subsequently selecting a set of suitable patterns. The element extraction is actually an OL process, where almost any existing OL tool producing OWL ontologies could be used. In the experiments presented in this deliverable the Tex2Onto[1] tool was used. Pattern matching and selection in OntoCase is done based on a pattern ranking scheme presented in [Blo08], using techniques from ontology search, ranking, and matching, to find suitable patterns for inclusion. Some background knowledge is used in this process, currently WordNet is exploited, but in future experiments also domain specific knowledge can be used. Throughout the matching process confidence values are used to represent the uncertainty of the matching, taking into account the polysemy of the terms when WordNet is used for matching for instance. The confidence values are stored together with the matches and used also in the next phase.

The second phase of OntoCase, called 'reuse', involves using the matching information to reuse the selected patterns together with the extracted elements. The result is an initial ontology, suitable for further refinement by an ontology engineer. There are two basic options for how to conduct the ontology construction, either the

[1] http://ontoware.org/projects/text2onto/

patterns are used as a starting-point and the input, i.e. extracted elements, is pruned of all parts not matching any pattern element, or the input ontology is used as a starting point and the patterns are merely added to the input, including the correspondences found in the matching step. In both cases heuristics are used to improve pattern composition and ontology building. For the experiments presented in this deliverable the pruning alternative was used. The confidence values resulting from the matching step above are transferred to the resulting ontology, whereby some parts can be seen as more 'certain' than others, also in the output ontology. In the experiments presented here these confidence values have not been taken into account, all added elements were included in the output OWL-file. More detailed future evaluations could also include the confidence of the evaluated elements.

OntoCase was initially implemented as a command-line tool in Java, based on the Jena API for OWL ontology management. The method has not yet been incorporated into the NeOn toolkit, but a plug-in version is planned for the long term. The first improvement step is however to develop a suitable user interface for the method. The input to the method is currently an OWL ontology, either generated directly by Text2Onto, or any other ontology the user chooses to provide. Links to the patterns are collected in the so called 'pattern base', which is a database containing links to all available patterns and some metadata about the patterns. The patterns themselves are small self-contained OWL ontologies. OntoCase uses only content design patterns at the moment. One example of such a pattern is the Agent-role pattern, present in the ODP portal,[2] that can be seen in Figure 3.2. The pattern is small, it includes only three concepts. Agents, which are types of objects that in turn can be classified by certain roles. An example instantiation would be to specialise the 'agent' concept and add 'person' as a subconcept, then add 'parenting role' as a subconcept of 'role'. Such an instantiation can then be used to store information about the parenting roles, i.e. instances of the 'parenting role' concept such as 'father' and 'mother', of people, i.e. concrete instances of the 'person' concept. The arrow with no label in the figure signifies a disjointness axiom between agents and roles.

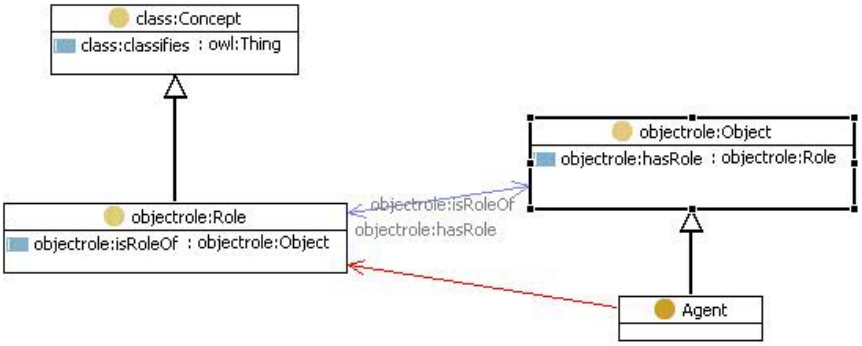

Figure 3.2: The Agent-role pattern.

Another example of such a pattern is the Information realisation pattern, as seen in Figure 3.3. The pattern represents the semiotic notion of information, i.e. the abstract information as such and its concrete realisation in the world.

[2] http://ontologydesignpatterns.org

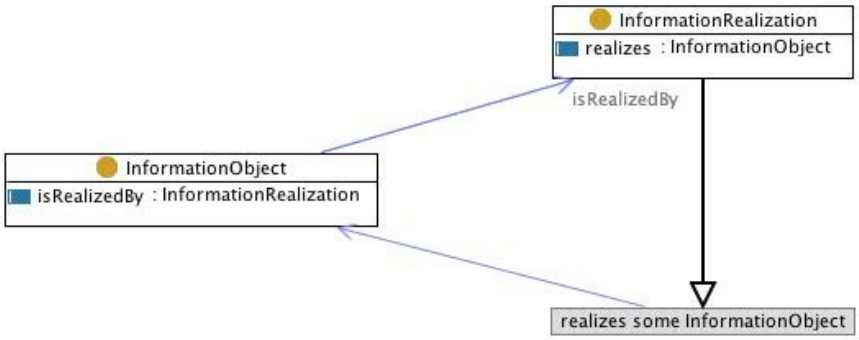

Figure 3.3: The Information realization pattern.

3.3 Experiment motivation and goals

In this deliverable the focus is on showing experimental results supporting the claim that there is actually a benefit from the interaction between OntoCase and the other OL methods proposed. Two hypotheses were stated in order to be tested during these experiments. The hypotheses were the following:

1. Patterns can improve the structure of learnt ontologies by connecting unconnected parts, and adding relations, without increasing the error rate.

2. Ontology enrichment using other OL methods can facilitate and improve pattern matching in OntoCase.

The first hypothesis states that applying OntoCase on top of results from other OL methods, such as Text2Onto, will assist in connecting unconnected parts, such as unconnected concepts, and give the ontology a more general top structure. Of course this needs to be done without increasing the error rate of the ontology, i.e. the fraction of erroneous concepts and relations in the ontology. Hence, it is important to assess the correctness of the ontologies both before and after applying OntoCase in addition to showing that relations are added and a general top structure is achieved. The second hypothesis on the other hand states that applying other OL methods, such as Text2Onto, as a pre-processing step in order to enrich the input ontology before applying OntoCase might actually improve the pattern matching, so that more relevant patterns can be identified. This shows the beneficial interplay between OntoCase and other OL methods.

3.3.1 Example illustrating goals

To clarify these hypotheses we introduce an example using the two patterns Agent-role, as seen in Figure 3.2 above, and Object-role. Object-role is a more general pattern that is imported and specialised in Agent-role. In Figure 3.2 we can see that the only concept native to Agent-role is the Agent concept, the rest is imported from Object-role, using the namespace prefix 'objectrole', and by that pattern concepts are in turn imported from the classification pattern. To illustrate how OntoCase works and to explain the above hypotheses we start by assuming that the two concepts 'grader' and 'puddler' have been extracted from a text corpus, e.g. using Text2Onto. These concepts are present in the input ontologies used in the actual experiments, but the illustrations of the matching and reuse of patterns below is not present exactly in this form, it should be seen as a simplified example. The input ontology containing only the two unconnected concepts of this example can be seen in Figure 3.4, using a UML notation. These two concepts can also have a confidence value associated, representing the confidence with which we believe that they were correctly included in the ontology.

Now, let us assume that the background knowledge used by OntoCase produces the two hypotheses that 'grader' is either a vehicle for grading or the role of setting grades for something, i.e. rating the quality of

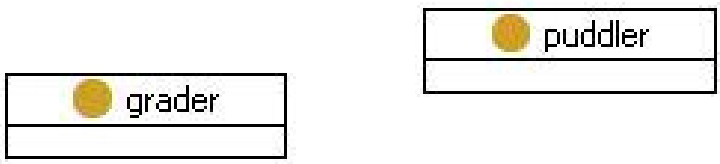

Figure 3.4: Two unconnected concepts as input ontology.

something. Such a hypothesis would be provided together with a confidence value, i.e. if WordNet was used the confidence would among other things be based on the number of senses of both the terms involved, since the more senses that exist the more uncertain is our hypothesis. The hypotheses mentioned above match the Object-role pattern, to a certain extent, since a vehicle is a kind of object and in the other sense the concept is a role. Similarly background knowledge might tell us that 'puddler' is either the role performing iron or clay puddling, or the tool used for this purpose, i.e. again either an object or a role. These hypotheses provide a match to the Object-role pattern, and it can be automatically instantiated as in Figure 3.5. When the pattern is instantiated the confidence values are transferred to the output ontology.

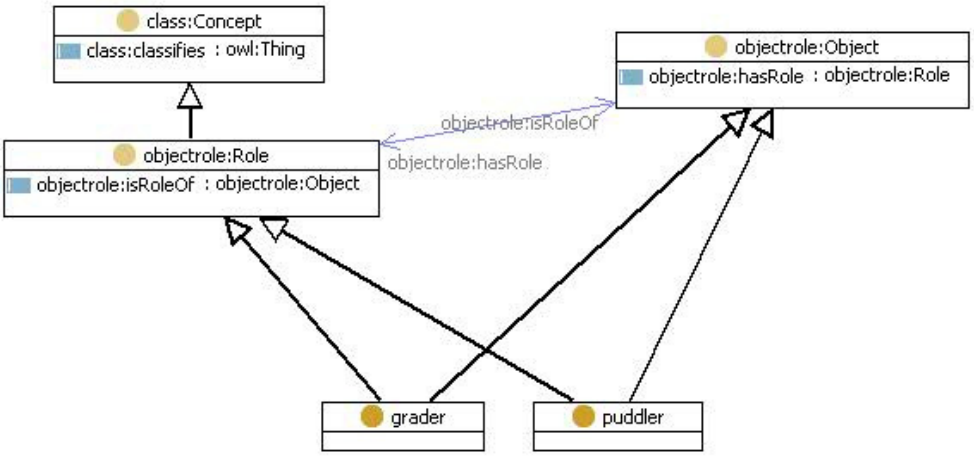

Figure 3.5: The ontology after applying the Object-role pattern.

The figure illustrates the way OntoCase connects unconnected concepts, through adding hypotheses about their connections to general patterns. The ontology is now 'connected', in a graph sense, and also contains a more general top-structure than before. The two hypotheses for each concept are both correct in the domain, even though probably in a final version of the ontology only one of these will be selected for each concept, depending on the focus and task of the ontology. We have however no way of knowing which one of the alternatives is the correct one, either the role or the object or perhaps both, only with this limited evidence we have at the moment. One part of this evidence is the confidence values, but rather than using only these as basis for a decision we would like to let the ontology engineer decide, with the confidence values as one guiding factor. A result such as the above example would support the first experiment hypothesis, since we have added a general top structure and connected unconnected concepts, without increasing the error rate.

To illustrate the aim of the second hypothesis, we can use some tool to enrich the input ontology. This could be an OL tool such as Text2Onto. With 'enrich' we mean to add additional knowledge and context to the extracted elements, e.g. to add more concepts and relations connected to the original elements. Let us assume that we apply such a tool and manage to extract one more concept, 'person', and two relations

stating that 'puddler' and 'grader' are both subclasses of 'person'. This new input ontology can be seen in Figure 3.6. The added relations are not certain to be completely correct however, the enrichment might contain errors as well. Although a puddler is in fact a person, to model the concepts as subclass of person might not be the best solution, i.e. since it is not an inherent property of a human being to be a puddler or not to be a puddler. However, this is a typical uncertain result that may be provided when using OL methods.

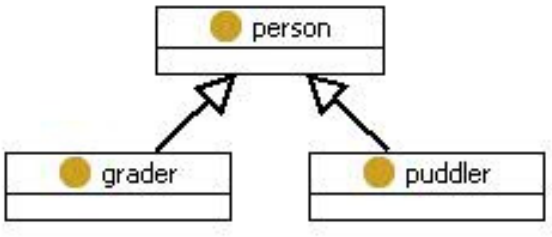

Figure 3.6: The enriched input ontology.

Using this new input ontology we can derive the same conclusions as previously, matching the Object-role pattern. In addition we could now discover a connection to the Agent-role pattern, if our background knowledge contains the common sense information that a 'person' is a type of 'agent'. Assuming this then we can see that by enriching the ontology we were able to match an additional pattern and add a more detailed structure to the ontology than before. The resulting ontology can be seen in Figure 3.7. This ontology is actually inconsistent, due to the 'grader' and 'puddler' being both persons, i.e. agents, and roles, which are disjoint in the Agent-role pattern. Although this is inherently an error this is also a good starting point for refinement, since the ontology engineer then has to choose how to model these concepts. Most likely the conclusion would be that it is the subclass relations of person that introduce the errors, and these can be replaced by specialisations of the 'hasRole'/'roleOf' relations between objects and roles. However, this correction has to be done manually at the moment. The two senses of 'grader' and 'puddler' as vehicles/tools or roles are harder to resolve, and depend more on the task and intention of the ontology, but by including both the possibilities we at least suggest both options to the ontology engineer.

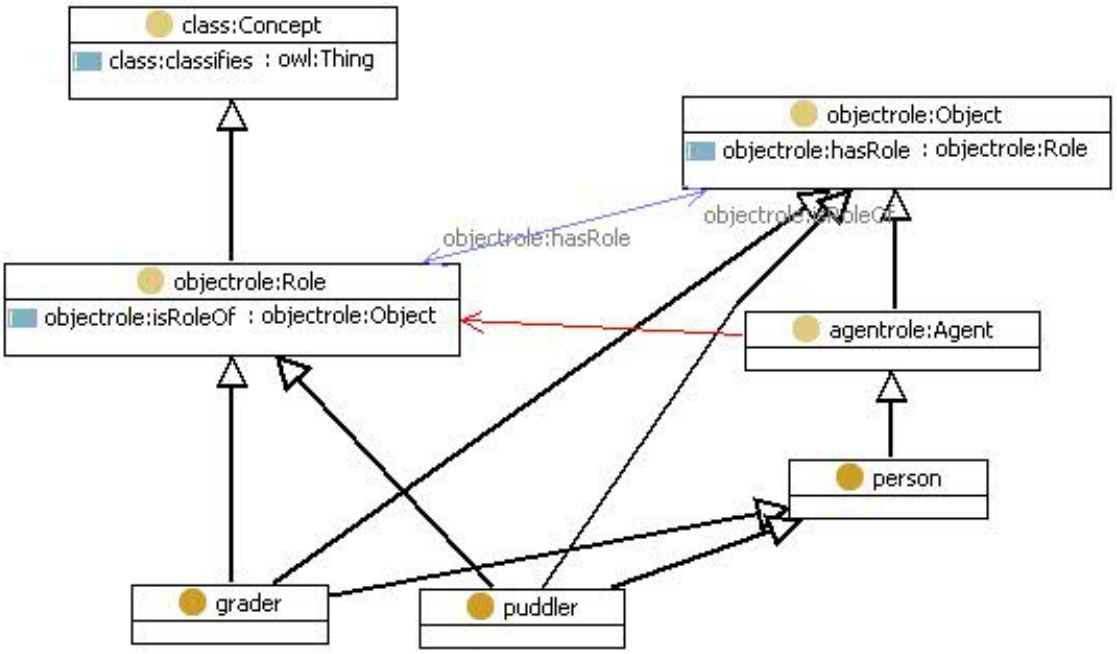

Figure 3.7: The output ontology after applying the Agent-role pattern.

3.4 Experiment setup

The data used for the experiments was provided by the FAO and originates within the agriculture domain. First, a simple lightweight ontology was provided, generated and translated into OWL from a manually engineered concept network representation. The ontology focuses on concepts related to the concept of "rice". This ontology will here be denoted *Ontology_rice*, and some information about the ontology can be seen in Figure 3.1. Additionally three ontologies were produced using the Text2Onto tool based on texts provided by FAO. Out of the three text corpora two were generated by extracting DBPedia[3] abstracts and comments respectively, collected by using the concepts in *Ontology_rice* as search terms. The third text corpora was produced by FAO based on abstracts related to the term “rice” in the AGROVOC thesaurus. Additional information about the ontologies generated from these text corpora using Text2Onto can be seen in Figure 3.1, they are named *T2O_DBPabstracts*, *T2O_DBPcomments*, and *T2O_AgrovocAbs* respectively.

	No. of concepts	No. of top concepts	No. of subclass relations	No. of properties	Avg. depth
Ontology_rice	266	155	110	37	1.68
T2O_DBPabstracts	1086	1018	89	17	1.06
T2O_DBPcomments	365	290	189	3	1.34
T2O_AgrovocAbs	3575	1822	1954	49	1.68

Table 3.1: The experiment input ontologies.

For the second part of the experiments the *Ontology_rice* ontology was then enriched using Text2Onto and the same data as for producing *T2O_DBPabstracts* and *T2O_DBPcomments*. Two new ontologies were thereby produced, named *OntologyRice_EnrichedAbs* and *OntologyRice_EnrichedComm* and details about these can be seen in Table 3.2.

Next, OntoCase was run with all these ontologies as input, in the pruning mode, meaning that only the parts matching any patterns were included in the output, mainly for minimising the evaluation effort. The ranking

[3] http://dbpedia.org/

	No. of concepts	No. of top concepts	No. of subclass relations	No. of properties	Avg. depth
OntologyRice_EnrichedAbs	1256	1080	199	53	1.20
OntologyRice_EnrichedComm	535	352	299	39	1.57

Table 3.2: The enriched ontologies for the second part of the experiment.

threshold for pattern selection was based on previous experiments using the tool, and it was fixed throughout the whole experiment. A catalogue of 41 ontology patterns was used, containing all content patterns that were at the time available (and correctly represented) on the ODP portal and in addition a set of patterns previously used with OntoCase (see [Blo05]).

After producing the ontologies, next is the evaluation of the ontologies. First general characteristics were gathered for all the ontologies, see tables above and results in the following section. This is in order to give an idea of the structure of the ontologies, for example indicating the number of top concepts and average depth of inheritance etc. Also the size of the ontologies is an interesting feature, in order to show that these are not toy examples. After this collection of basic measures, the correctness needed to be evaluated, in order to show that the added structure does not affect the correctness negatively.

Since some of the ontologies were quite large and the evaluation had to be done manually we had to use a random sample of concepts and relations for the evaluation and not the complete ontology. This of course introduces some uncertainty into the results, but in this case we do not aim to show the exact amount of improvement but only the general trend that the correctness is not worse than before, hence this uncertainty can be accepted. Additionally due to time and resource restrictions the ontologies were only evaluated by one person (ontology expert), which also introduces an uncertainty. The effect of this is reduced by the fact that the same method and the same judgements were applied to all ontologies, whereas the relations between the error rates of the input and the output will ideally stay the same. In summary, the numbers presented below should not be taken as absolute numbers, but can be seen as reliable for comparing the sets of values for the input and the output ontologies, and for showing the general trend.

Moreover, since the evaluator was an ontology expert and not a domain expert, i.e. not from the agriculture domain, the evaluation had to be based on reliable sources of agriculture information. These sources were primarily the AGROVOC thesaurus of agricultural terms and their relations, and secondarily agriculture information available on the web. Nevertheless, there were cases when only a domain expert could have evaluated the correctness properly, in these cases the evaluator had the choice of marking the concept or relation with an "I'm not sure" annotation. The other two alternatives were to state that the concept or relation was “correct” or “not correct”, for inclusion in a domain ontology within the agriculture domain.

The sample size for each ontology was between 104 and 180 randomly selected concepts, and between 51 and 102 randomly selected relations. The relations were a mix of both subclass statements and properties. The exact sample sizes for each ontology can be seen in Table 3.3. The difference in sample size is due to the random selection method, which was dependent on the ontology size and concrete representation in the OWL-file. The aim was initially to evaluate at least around 10% of the concepts and relations of each ontology, but practical constraints were set by the available resources, i.e. the feasibility of one evaluator evaluating a high number of concepts and relations during a limited amount of time. However, this aim was reached in all but one case, the largest ontology T2O_AgrovocAbs and its output counterpart, where only around 4% of the concepts were evaluated. Similarly the aim was to evaluate at least around 5% of the relations, but in T2O_AgrovocAbs and its output counterpart we only managed to evaluate between 2% and 4.5% of the relations due to its total size. Despite these shortcomings it should be noted that many of the samples were much larger with respect to the actual size, the most reliable results probably being reached for Ontology_rice and OC_Ontology_rice since there the sample sizes for concepts were 42% and 50% respectively, and for relations 48% and 24% respectively.

The initial plan was to also include one manually engineered translation of a part of the AGROVOC thesaurus, connected to rice-concepts, in the experiment. However, this ontology was examined and it was concluded

Ontology	Concepts	Relations	Concept sample	Relation sample %
Ontology_rice	266	147	111	70
OC_Ontology_rice	280	291	139	69
T2O_DBPabstracts	1086	106	117	86
OC_T2O_DBPabstracts	812	1499	132	80
T2O_DBPcomments	365	192	104	78
OC_T2O_DBPcomments	321	644	115	79
T2O_AgrovocAbs	3575	2003	159	91
OC_T2O_AgrovocAbs	2823	4225	104	87
OntologyRice_EnrichedAbs	1256	252	180	84
OC_OntologyRice_EnrichedAbs	1293	1752	126	86
OntologyRice_EnrichedComm	535	338	162	102
OC_OntologyRice_EnrichedComm	570	983	110	51

Table 3.3: The total sizes and sample sizes for each ontology.

that OntoCase is currently not applicable for such ontologies. We provide a short discussion here about why this is the case and in what way this ontology was not possible to use with the current version of OntoCase, in order to illustrate the limitations of the current approach. Inherent in OntoCase are some common principles of ontological modelling, such as the common practice that concepts are given humanly understandable names or at least labels in natural language. The ontology intended to be used was in fact an ontology, in the sense that it was represented in OWL and used some of the features in OWL, but it was in all other respects more of a thesaurus. The transformation was done more or less as a “direct translation”, whereby the structure was not a common ontological structure. Almost none of the concepts had natural language names or labels, but were instead encoded as numbers, i.e. *c_3259*. The ontology provided a very general top structure which was basically a metamodel of a thesaurus, containing concepts such as ’lexicalization’, ’term’, ’noun’, ’category’, and ’domain concept’. There was a taxonomy and properties specialising this structure, but all actual domain concepts and terms were instances of the concepts in this structure and the concept instances were only connected to their lexicalization through properties.

Some of those general metamodel concepts can be recognised by OntoCase, and could match general pattern concepts, thereby some kind of result can actually be produced by OntoCase through pattern reuse. This would not be a comprehensive result however, since at the moment OntoCase does not treat instances, whereas the part where the information of the input ontology actually resides would be completely ignored by the method. The pruning mode used for all the experiments even prunes all the instances of the ontology, whereas the output would be complete nonsense since all concepts would have been stripped of their lexicalizations, which in this case was the only form of concept definitions existing. The output ontology would be just a set of numbered concepts put in a logical structure without meaning, hence it would also be impossible to evaluate the correctness of this structure. This is a type of ontology that OntoCase in its current version cannot handle.

3.5 Experiment results and analysis

Below the results of the two experiments are presented and analysed. Some conclusions are drawn with respect to the hypotheses.

3.5.1 Providing structure to learnt ontologies

The ontologies resulting from running OntoCase on all the above presented ontologies can be seen in Table 3.4, together with their characteristics. Without going into details of the ontologies it may be noted that for most ontologies the number of top concepts is reduced and the average depth is increased, this is due to the

added top structure provided by the quite general ontology design patterns that were matched and reused by OntoCase. Additionally the number of subclass relations has increased for most ontologies, even though they were pruned of some of their original concepts and thereby reduced in size. This is due to the fact that OntoCase generally adds all “hypotheses” found in the pattern matching process, i.e. all correspondences between pattern concepts and input concepts that may be interpreted as subclass relations. This results in a certain amount of redundancy, and can even result in cycles in the taxonomy. At a first glance these may be considered as errors, but it is actually an intended feature of OntoCase, since one of its goals is to provide the ontology engineer with suggestions for alternative modelling choices. The ontologies are also enriched with more general properties from the patterns, which can be noted in the properties column of Table 3.4.

	No. of concepts	No. of top concepts	No. of subclass relations	No. of properties	Avg. depth
OC_Ontology_rice	280	112	245	46	2.19
OC_T2O_DBPabstracts	812	22	1471	28	2.37
OC_T2O_DBPcomments	321	22	628	16	2.39
OC_T2O_AgrovocAbs	2823	27	4162	63	2.84
OC_OntologyRice_EnrichedAbs	1293	758	1679	73	3.36
OC_OntologyRice_EnrichedComm	570	248	921	62	3.06

Table 3.4: The experiment output ontologies.

The following step was to evaluate the ontologies, using the method stated in the previous section. The results of this evaluation can be seen in Table 3.5. From the results in the table we can note that the correctness is generally slightly better for the output ontologies than for the original ontologies. However, the increase in accuracy might be mostly a side-effect of applying the method rather than a direct feature, since OntoCase does not attempt to explicitly filter out “irrelevant” parts. Rather, the results originate from the pattern matching, since clearly incorrect concepts will not match anything in a pattern and will therefore be pruned. Examples of ’clearly incorrect concepts’ are misspelled terms and strangely combined multi-word concept labels, such as word combinations denoting a somewhat unclear concept, i.e. ’people worldwide’, or common word combinations not really denoting a concept at all, i.e. ’sativa subsp’ which is a subset of the words in a sentence discussing subspecies of the species ’sativa’ but where the actual subspecies was not included, only the abbreviation for subspecies, i.e. ’subsp’. In some domains on the other hand, if the terms are very specific, some additional domain specific background knowledge might be needed for the pattern matching in order not to prune too many domain specific concepts and thereby reduce the ontology quality in this way. At the moment only domain independent background knowledge is used for the matching, whereby these results are really encouraging, showing that even in a quite specific domain, such as agriculture, this will be sufficient for most case.

To summarise these results we may note that OntoCase in fact gives an added structure to the ontologies, and does connect unconnected parts of the ontologies produced by other OL methods. This can be seen by studying the number of top-concepts of each ontology, and the number of taxonomic and non-taxonomic relations present, but is of course best viewed in the actual ontology. Unfortunately the ontologies are large structures and it is not possible to present them here. However, the results presented above provide clear support for the first hypothesis, stated at the beginning of the chapter. OntoCase is able to connect large parts of the ontologies to the patterns, even with this very small pattern catalogue used. For this experiment we used the pruning version of OntoCase, for convenience of evaluating the added parts, but also the version including all input elements could be used, in order not to loose any of the input elements. Adding structure to the ontologies is additionally done without reducing ontology quality, in terms of overall correctness of concepts and relations. It has to be noted however that the ontologies produced are intended as the basis for further development, manually or semi-automatically. They contain, for example multiple modelling choices in parallel from which the ontology engineer can choose the needed ones.

Ontology	ConceptsProperties	Correct %	Not sure %	Incorrect %
Ontology_rice	C	92.8	5.4	1.8
	P	90.0	5.7	4.3
OC_Ontology_rice	C	97.8	2.2	0.0
	P	92.8	2.9	4.4
T2O_DBPabstracts	C	85.5	3.4	11.1
	P	61.6	12.8	25.6
OC_T2O_DBPabstracts	C	87.9	4.6	7.6
	P	77.5	2.5	20.0
T2O_DBPcomments	C	94.2	2.9	2.9
	P	64.1	11.5	24.4
OC_T2O_DBPcomments	C	93.0	4.4	2.6
	P	86.1	6.3	7.6
T2O_AgrovocAbs	C	84.9	5.7	9.4
	P	65.9	13.2	20.9
OC_T2O_AgrovocAbs	C	88.5	5.8	5.8
	P	79.3	6.9	13.8
OntologyRice_EnrichedAbs	C	87.8	5.6	6.7
	P	76.2	10.7	13.1
OC_OntologyRice_EnrichedAbs	C	88.1	5.6	6.4
	P	79.1	8.1	12.8
OntologyRice_EnrichedComm	C	92.0	4.3	3.7
	P	78.4	4.9	16.7
OC_OntologyRice_EnrichedComm	C	93.6	3.6	2.7
	P	88.2	5.9	5.9

Table 3.5: The evaluation results for the ontologies.

3.5.2 Enrichment as support for pattern matching

To find support for the second hypothesis the pattern matching results from running OntoCase on the original ontology called *Ontology_rice* above and the two ontologies where this ontology was enriched by the results from Text2Onto were recorded. The general characteristics of the input ontology *Ontology_rice* were presented in Table 3.1 and the characteristics of the enriched ontologies in Table 3.2. Results of running OntoCase on these ontologies are shown in Table 3.4. The enrichment primarily consisted in adding additional concepts and relations to the original ontology, it can be seen as a merging of the original ontology and the results from Text2Onto using the respective text corpora. OntoCase was then run with the same parameters as before.

Ontology	No. of selected patterns	No longer used	Not previously used
Ontology_rice	19	NA	NA
OntologyRice_EnrichedAbs	24	-2	+7
OntologyRice_EnrichedComm	23	-3	+7

Table 3.6: The number of patterns selected.

The intention in this case is to show that enrichment using OL methods ensures that more patterns are identified to match the input ontology, without introducing any errors. In this case errors could be both incorrect concepts and relations in the ontology, as discussed above, or if an inappropriate pattern was matched and included. For the first type of errors we have already seen in the last section (see Table 3.5) that the enrichment in itself will in fact introduce some errors, since the OL methods used are not exact and

the original ontology was manually engineered and thereby quite accurate in its definitions. It is of course a trade-off to be considered, if we should extend the ontology or not, but this is not the main focus of this evaluation. We focus only on the effects of the enrichment on the ontology design pattern inclusion.

Ontology_rice	OntologyRice_EnrichedAbs	OntologyRice_EnrichedComm
Actions.owl	Actions.owl	Actions.owl
AgentRole.owl	AgentRole.owl	AgentRole.owl
Classification.owl	Classification.owl	Classification.owl
CollectionEntity.owl	CollectionEntity.owl	CollectionEntity.owl
Constituency.owl	Constituency.owl	Constituency.owl
CoParticipation.owl	CoParticipation.owl	CoParticipation.owl
Description.owl	Description.owl	Description.owl
GOtop.owl	GOtop.owl	GOtop.owl
ObjectRole.owl	ObjectRole.owl	ObjectRole.owl
Participation.owl	Participation.owl	Participation.owl
Person.owl	Person.owl	Person.owl
Precedence.owl	Precedence.owl	Precedence.owl
Product.owl	Product.owl	Product.owl
SpeciesModel.owl	SpeciesModel.owl	SpeciesModel.owl
System.owl	System.owl	System.owl
TypesOfEntities.owl	TypesOfEntities.owl	TypesOfEntities.owl
Organisation.owl	Organisation.owl	EmployeeDepartment.owl
ProductAssociations.owl	EmployeeDepartment.owl	Metonymy.owl
ProductCategory.owl	Metonymy.owl	NaryParticipation.owl
	NaryParticipation.owl	Party.ow
	Party.owl	Situation.owl
	Situation.owl	SystemSynthesis.owl
	SystemSynthesis.owl	TaskRole.owl
	TaskRole.owl	

Table 3.7: The patterns selected.

The results in terms of the number of selected patterns can be viewed in Table 3.6. 19 patterns out of the catalogue of 41 were selected for inclusion by OntoCase based on the *Ontology_rice* ontology. This number was increased to 24 and 23 for the two enriched ontologies respectively. It may also be noted that patterns are not only added (see 'Not previously used' column), some are also no longer selected (see the 'No longer used' column). This is due to that the enrichment also slightly changes the focus of the ontologies, whereas some patterns might now be considered more relevant and some less relevant.

To show that the added patterns are really a positive result we will also study the patterns that were additionally selected in more detail. It is generally not possible to make a statement if a pattern is valid or not for a specific domain, rather we have to look at what parts of the pattern could actually be reused in the ontology and are compatible with the focus of the case at hand. There may be concepts and properties from a pattern that are valid in a certain domain, even though the complete pattern is not valid in this domain. In this case most of the patterns in the pattern catalogue are quite general and may be appropriate to include in any domain, whereas the task of deciding if a pattern was correctly selected is more connected to if the matches found are indeed correct. Below, in Table 3.7, the patterns actually selected are listed, for each of the ontologies.

The patterns were judged in a similar way as the correctness of concepts and properties previously, assigned either "complete", "incorrect" or "partial" by an ontology engineer. "Complete" denoting that the complete pattern fits the domain and case at hand, "partial" meaning that there are some parts of the pattern that may be used for this domain and the case at hand, and finally "incorrect" meaning that this pattern does not fit the

domain nor the case at hand. The task of the evaluator was to judge if the pattern was correctly or incorrectly selected, based on if any part of the pattern could be applicable in the domain, rather than to evaluate all the actual matching results. Missing patterns were not considered. In Table 3.8 the number of patterns in each category can be seen for each ontology. In parentheses the number of added and removed patterns in comparison to the matching of Ontology_rice are listed.

Ontology	Complete (added/removed)	Partial (added/removed)	Incorrect (added/removed)
Ontology_rice	15	4	0
OntologyRice_EnrichedAbs	20 (5/0)	4 (2/2)	0 (0/0)
OntologyRice_EnrichedComm	19 (5/1)	4 (2/2)	0 (0/0)

Table 3.8: The number of completely and partially applicable patterns.

The four patterns considered to only partially match the domain of *Ontology_rice* are the *Person*, *Product*, *ProductAssociations*, and *ProductCategory* patterns. The *Person* pattern is not completely suitable since it is intended for ontologies about people and the information associated to individuals, such as social security numbers, age, height, weight, which in case of an agricultural ontology might be considered irrelevant. The other three patterns are domain specific patterns from the product developments domain, still some general parts of these patterns treating products and their features are also applicable in the agriculture domain, agriculture does in fact produce some types of products.

Based on the results presented above it can be noted that enrichment of ontologies, whether learnt or hand-crafted, can lead to the selection of more relevant patterns for inclusion in the ontology. The enrichment in itself might introduce some errors compared to a manually constructed ontology, but this is natural since enrichment is in this case done in an automatic manner using OL methods. Nevertheless, the error rate is not increased by applying OntoCase on top of the results, and the additional patterns selected are to a high extent correctly selected. The overall conclusion is clear, the second hypothesis is supported, enrichment support the OntoCase matching and selection of patterns.

Chapter 4

Conclusion and Outlook

In this deliverable, we presented a method for exploration-based ontology refinement which supports the user in specifying complex property restrictions (see Section 2.2.4) as well as a novel plugin for the NeOn Toolkit (cf. Section 2.3). Initial experiments with the well-known SWRC ontology demonstrated the usefulness of our methods for semi-automatic ontology refinement (Section 2.2.5).

In Chapter 3, we reported on a set of evaluation experiments that aim to demonstrate the synergies arising from a combination of ontology design patterns and ontology learning techniques. The first experiment aimed to show that applying OntoCase on top of the results from the OL methods in Text2Onto improve the learnt ontologies, in terms of adding more relations, thus connecting unconnected parts, and adding missing background knowledge, thus giving the ontologies a general top-structure. The experiments support this hypothesis and show that the resulting ontologies include a richer structure, in terms of relations, and at the same time have a lower error rate than the input ontologies. The second experiment aimed to show that enriching existing ontologies using an OL tool, such as Text2Onto, will facilitate and improve the pattern matching of OntoCase, and result in more relevant patterns being selected and reused. The experimental results show that more patterns were selected and reused, and that those patterns were in fact relevant for inclusion in the ontology, thus also the second hypothesis is supported.

In the near future, we will extend our framework for exploration-based ontology refinement by additional, non-logical experts. Support for reasoning with uncertain and contradictory knowledge might be required, in order to enable an even higher degree of automation. Notwithstanding, we are also planning to develop a second plugin for the NeOn Toolkit to assist ontology engineers in refining the domain and range restrictions of ontological properties. Further evaluation experiments will be conducted to demonstrate the feasibility of learning ontology alignments by exploring and bridging the *semantic gap* between different ontologies. With respect to OntoCase, the method will be further developed and a main focus is to add a graphical user interface, thus enabling the user to select among the modelling choices generated by the pattern matching process. Further evaluation experiments are also planned, in order to more precisely determine the accuracy of the approach. The pattern catalogue will also be extended with more domain-specific patterns, and experiments to study the effects of providing such patterns are planned.

Bibliography

[Baa95] F. Baader. Computing a minimal representation of the subsumption lattice of all conjunctions of concepts defined in a terminology. In *Proc. Int. Symposium on Knowledge Retrieval, Use, and Storage for Efficiency (KRUSE)*, pages 168–178, Santa Cruz, USA, 1995.

[BC08] Paul Buitelaar and Philipp Cimiano, editors. *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, volume 167 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, JAN 2008.

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

[BGSS07] Franz Baader, Bernhard Ganter, Baris Sertkaya, and Ulrike Sattler. Completing description logic knowledge bases using formal concept analysis. In Manuela M. Veloso, editor, *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 230–235, 2007.

[Blo05] Eva Blomqvist. Fully Automatic Construction of Enterprise Ontologies Using Design Patterns: Initial Method and First Experiences. In *Proceedings of OTM 2005 Conferences, Ontologies, DataBases, and Applications of Semantics (ODBASE), Agia Napa, Cyprus, Oct 31- Nov 4*, 2005.

[Blo08] E. Blomqvist. Pattern ranking for semi-automatic ontology construction. In *Proccedings of SAC'08: Track on Semantic Web and Applications (SWA)*, Fortaleza, Ceará, Brazil, March 16-20 2008.

[CGR+05] Massimiliano Ciaramita, Aldo Gangemi, Esther Ratsch, Jasmin Saric, and Isabel Rojas. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 659–664. Professional Book Center, 2005.

[CH94] William W. Cohen and Haym Hirsh. Learning the classic description logic: Theoretical and experimental results. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 121–133. Morgan Kaufmann, 1994.

[CHR06] Philipp Cimiano, Matthias Hartung, and E. Ratsch. Finding the appropriate generalization level for binary relations extracted from the genia corpus. In *Proc. Int. Conf. on Language Resources and Evaluation (LREC)*, pages 161–169. ELRA, MAY 2006.

[CHS05] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning concept hierarchies from text corpora using formal concept analysis. *JAIR*, 24:305–339, AUG 2005.

[CV05] Philipp Cimiano and Johanna Völker. Text2Onto - a framework for ontology learning and data-driven change discovery. In *Proc. 10th Int. Conf. on Applications of Natural Language to Information Systems*, pages 227–238. Springer, JUN 2005.

[DSFGP+09] Martin Dzbor, Mari Carmen Suarez-Figueroa, Asuncion Gomez-Perez, Eva Blomqvist, Holger Lewen, and Mauricio Espinoza. D5.6.2 experimentation with parts of neon methodology. Technical report, NeOn Project., 2009.

[FIPS04] N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro. Concept formation in expressive description logics. In *Proc. 15th European Conf. on Machine Learning (ECML)*. Springer Verlag, 2004.

[Gan84] Bernhard Ganter. Two basic algorithms in concept analysis. Technical Report 831, FB4, TH Darmstadt, 1984.

[GW97] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[HVK06] Heiko Haller, Max Völkel, and Felix Kugel. imapping wiks - towards a graphical environment for semantic knowledge management. In Sebastian Schaffert, Max Völkel, and Stefan Decker, editors, *Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics*, volume 1, JUN 2006.

[LE05] Francesca A. Lisi and Floriana Esposito. Ilp meets knowledge engineering: A case study. In Stefan Kramer and Bernhard Pfahringer, editors, *ILP*, volume 3625 of *LNCS*, pages 209–226. Springer, 2005.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. `CTAN:http://www.w3.org/TR/2004/REC-owl-features-20040210/`, FEB 2004.

[Ned99] Claire Nedellec. Corpus-based learning of semantic relations by the ILP system, Asium. In James Cussens and Saso Dzeroski, editors, *Learning Language in Logic*, volume 1925 of *LNCS*, pages 259–278. Springer, 1999.

[NV06] Roberto Navigli and Paola Velardi. Ontology enrichment through automatic semantic annotation of on-line glossaries. In Steffen Staab and Vojtech Svátek, editors, *Proc. 15th Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW)*, volume 4248 of *LNCS*, pages 126–140. Springer, 2006.

[PGD+08] V. Presutti, A. Gangemi, S. David, G. Aguado de Cea, M. C. Suárez-Figueroa, E. Montiel-Ponsoda, and M. Poveda. D2.5.1: A library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. NeOn Deliverable 2.5.1, NeOn Consortium, 2008.

[Rud04] Sebastian Rudolph. Exploring relational structures via FLE. In Karl Erich Wolff, Heather D. Pfeiffer, and Harry S. Delugach, editors, *Proc. 12th Int. Conf. on Conceptual Structures (ICCS)*, volume 3127 of *LNCS*, pages 196 – 212. Springer, JUL 2004.

[Rud06] Sebastian Rudolph. *Relational Exploration - Combining Description Logics and Formal Concept Analysis for Knowledge Specification*. Universitätsverlag Karlsruhe, 2006. Dissertation.

[Rud08] Sebastian Rudolph. Acquiring generalized domain-range restrictions. In Raoul Medina and Sergei Obiedkov, editors, *Proc. 6th Int. Conf. on Formal Concept Analysis (ICFCA)*, volume 4933 of *LNAI*, pages 32–45. Springer, FEB 2008.

[SBH+05] York Sure, Stephan Bloehdorn, Peter Haase, Jens Hartmann, and Daniel Oberle. The SWRC ontology - semantic web for research communities. In Carlos Bento, Amilcar Cardoso, and Gael Dias, editors, *Proc. 12th Portuguese Conf. on Artificial Intelligence*, pages 218 – 231. Springer, DEC 2005.

[SFBG+07] Mari Carmen Suárez-Figueroa, Saartje Brockmans, Aldo Gangemi, Asunción Gómez-Pérez, Jos Lehmann, Holger Lewen, Valentina Presutti, and Marta Sabou. D 5.1.1 neon modelling components. Technical report, NeOn Project, March 2007. Available at: http://www.neon-project.org.

[SM01] Gerd Stumme and Alexander Maedche. FCA-merge: Bottom-up merging of ontologies. In *Proc. 17th Int. Conf. on Artificial Intelligence (IJCAI)*, pages 225–230, 2001.

[VB08] Johanna Völker and Eva Blomqvist. D3.8.1 prototype for learning networked ontologies. Technical report, Institute AIFB, University of Karlsruhe, FEB 2008. NeOn Deliverable.

[VHC07] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of OWL DL axioms from lexical resources. In *Proc. 4th European Semantic Web Conf. (ESWC)*. Springer, JUN 2007.

[VR08a] Johanna Völker and Sebastian Rudolph. Fostering web intelligence by semi-automatic owl ontology refinement. In *Proceedings of the 7th International Conference on Web Intelligence (WI)*, DEC 2008. regular paper.

[VR08b] Johanna Völker and Sebastian Rudolph. Lexico-logical acquisition of OWL DL axioms - an integrated approach to ontology refinement. In Raoul Medina and Sergei Obiedkov, editors, *Proceedings of the 6th Int. Conf. on Formal Concept Analysis (ICFCA)*, volume 4933 of *LNAI*, pages 62–77. Springer, FEB 2008.