NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D3.1.2 Context Representation Formalism

Deliverable Co-ordinator: **Guilin Qi**

Deliverable Co-ordinating Institution: **Universität Karlsruhe (TH) (UKARL)**

Other Authors: **Peter Haase (UKARL), with contributions from Sofia Pinto (TU Lisbon)**

In this deliverable we discuss formalisms for context representation in NeOn project. We first recall the general definition of context given in NeOn deliverable D3.1.1. Then we instantiate the general definition by describing *provenance* and *argumentation*. After that, we consider how to represent context in OWL ontologies. Finally, some approaches are given to illustrate how to obtain context information automatically from an *incoherent* ontology.

Document Identifier:	NEON/2007/D3.1.2/v1.0	Date due:	March 30, 2007
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 28, 2007
Project start date	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities, 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 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: Richard Benjamins E-mail adress: rbenjamins@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	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield United Kingdom Contact person: Hamish Cunningham
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	**Asociación Española de Comercio Electrónico (AECE)** C/Icalde Barnils, Avenida Diagonal 437 08036 Barcelona Spain Contact person: Jose Luis Zimmerman E-mail address: jlzimmerman@fecemd.org
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

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed writing parts of this document:

- University of Karlsruhe

Executive Summary

In D3.1.1 we surveyed the state-of-the-art on dealing with context and identified approaches for representing and reasoning with context which may be relevant for NeOn. In terms of the usages of context, we found that *supporting viewpoints and perspectives* and *dealing with inconsistent, uncertain and vague information*, will play a paramount role in NeOn. Considering these findings, in this deliverable we define the NeOn formalisms for context representation. Based on a generic and abstract definition of context, we specifically describe two specific forms of context: *Provenance* and *Arguments*.

Provenance includes context information about when and how ontology elements where introduced, from which information sources they have been obtained as well as information about the relevance of and confidence in ontology elements. This context information can then be exploited in dealing with various forms of imperfection, e.g. by interpreting the confidence values in a setting of probabilistic logics. Provenance information can easily be generated in approaches of automated ontology construction, e.g. ontology learning. We therefore specifically describe a model for provenance in ontology learning.

Arguments are another important form of context that captures reasons why particular elements in the ontology have been introduced in a particular way, but also decision procedures for the case of disagreements about the ontology. Such context information can again be exploited in resolving conflicts within an ontology, or selecting particular subsets of an ontology for a given context.

Another important aspect we need to address is how the context can be syntactically represented to be able to relate an ontology with its context. We therefore propose so called groundings of the context representation within OWL that allow us to specify the context itself in the form of an OWL ontology.

Finally, we describe some approaches for measuring incoherence in OWL ontologies that allows us to automatically obtain context information. This context – in the form of ranking information – can be exploited in dealing with imperfect information, where there are usually several alternative solutions available and we need to explore ranking information to select the best solution.

Contents

List of Figures

Chapter 1

Introduction

1.1 NeOn Big Picture

Real life ontologies and corresponding data are produced by individuals or groups in certain settings for specific purposes. Because of this, they can almost never be considered as something absolute in their semantics and are often inconsistent with ontologies created by other parties under other circumstances. In order to fully utilize networked ontologies, those disagreements must be identified prior to using them for reasoning. Each ontology can be viewed as valid (or appropriate) in a certain context. The context can be seen as a set of all circumstances, properties and facts within which the ontology has the desired semantics. From the theoretical side, we could say that whenever the contextual information is necessary, the target ontology cannot have fully defined static semantics because it depends on some external information which we call *context*. We could call such ontologies *parametric ontologies* because their semantics depends on the value of contextual *parameters*. It is the goal of the work performed in WP3 to develop appropriate techniques for dealing with context. As shown in Figure 1.1, this work belongs to the central part of the research and development WPs in NeOn. One of the key points of this workpackage is to model and provide a formalization of the context. This model will support both a proper representation of the information particular to the context and its formalization that allows reasoning with the modeled context.

1.2 Context Representation in NeOn

The notion of context has a very long history within several research communities, leading to vast studies about how to define a context, how to take into account information coming from the context, how to contextualize or de-contextualize knowledge and information, etc.

In D3.1.1 we surveyed the state-of-the-art on dealing with context. We identified several possible usages of context for ontologies and gave an overview of some present approaches for representing and reasoning with context which may be relevant for NeOn. We provided an abstract and generic mathematical definition of context, based on which we compared the different approaches. In terms of the usages of context, we found that *supporting viewpoints and perspectives* and *dealing with inconsistent, uncertain and vague information*, will play a paramount role in NeOn. To be able to address these usage scenarios for context, we identified that the following approaches for contexts are relevant for NeOn: The *networked ontology model* developed in WP1 provides the most obvious form of context: Ontologies will be embedded in a network of ontologies, which forms the context for its interpretation. In the networked scenarios of NeOn, ontologies are not treated as isolated entities, but are related to other ontologies in various networked ways, including versioning and mapping information etc. These other ontologies together with these links can be understood as a context for the ontology, as they will (in some cases) alter the knowledge which can be inferred from the ontology. The discussion of the networked ontology model will be provided by WP1. So we will not include it in the deliverable. *Reasoning with inconsistent ontologies* exploiting context information is important when different information sources with contradicting information will be integrated. Contextual information can be used

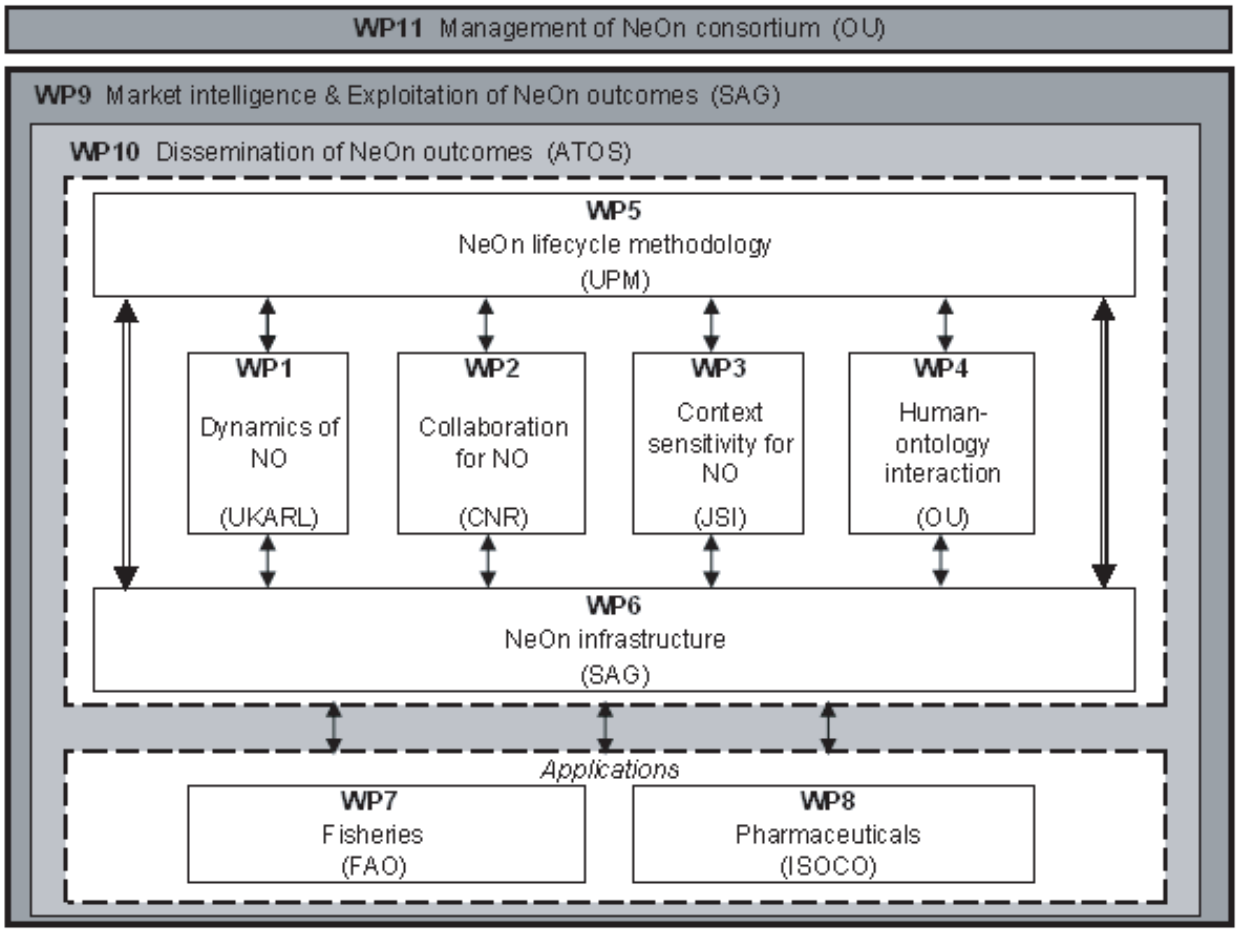

Figure 1.1: Relationships between different workpackages in NeOn

to resolve such conflicts. It can be used to select relevant consistent parts of the knowledge base which suffice for the task at hand. Contextual information provides guidance for this selection process, as usually different possibilities exist for resolving an inconsistency. *Context-based selection functions* appear promising for addressing a number of different problems. Finally, a combination of *possibilistic* and *probabilistic logics* seems to be required to deal with the various forms of vagueness and uncertainty in a contextualized way. The numerical values or priority information attached to elements of an ontology will provide important context information to deal with imprecision in an ontology.

Considering these findings, in this deliverable we define the NeOn formalism for context representation. In particular, we describe instantiate our generic definition of context for two specific forms of context: *Provenance* and *Arguments*.

Another important aspect we need to address is how the context can be syntactically represented to be able to relate an ontology with its context. We therefore propose so called groundings of the context representation within OWL that allow to specify the context itself in the form of an OWL ontology.

When dealing with inconsistency and uncertainty, there are usually several alternative solutions available and we need to explore ranking information to select the best solution. This ranking information can be generated by ontology learning. We can also obtain the ranking information by measuring incoherence in an incoherent OWL ontology. For example, by measuring the extent of incoherence of different ontologies, we can give a rank on them. That is, an ontology is more reliable than another one if it contains less incoherent information. Similarly, by measuring the extent of incoherence of different axioms, we get some ranking information on axioms which can be used to resolve incoherence [KPSG06].

1.3 Overview of the Deliverable

In this deliverable we start with a generic and abstract definition of context in Chapter 2. This definition was already used in the state-of-the-art deliverable on context representation languages D3.1.1 [HHR+06]. In Chapter 3 we then present instantiations of this generic definition for a number of types of contexts that we identified as particularly interesting for the NeOn project. We provide an OWL-based syntax for context in Chapter 4. Finally, in Chapter 5 we give an example on how context information can be used in reasoning with ontologies, specifically for measuring incoherence in ontologies. We conclude with a roadmap for future work in Chapter 6.

Chapter 2

A Generic Definition for Context

In this chapter we recapitulate the formal definition of *context of an ontology*[1], which we first provided in D3.1.1 [HHR+06]. This definition does not attempt to give a notion of "context" in a more general sense. It is also not intended to be operationalized in a reasoner. The purpose of this definition rather is to provide a basis for a being able to characterize different context representation formalisms within a common framework by instantiating the generic definition.

Contexts as modifiers of semantics. We are interested in knowledge expressed as a set of assertions and rules. Examples are knowledge bases (or ontologies), and relations between such knowledge bases.

If such a set of assertions is put into a context, then this means that the context alters some of the meaning of the set of assertions. In other words, the context acts as a *modifier* for the semantics of a knowledge base.

2.1 Formal abstract definition of context for an ontology

Let K be a knowledge base, which comes with an associated semantics $S(K)$. Thus, S is a function which associates a semantics to any knowledge base K.

Now, given a context C and a knowledge base K, we denote by $S'(K, C)$ the semantics of K *in the context* C. Thus, S' is a function which associates to any knowledge base K and context C a semantics, e.g. expressed by the set of all logical consequences of K in the context C.

If we have empty context (denoted by $\emptyset$), then often we require $S'(K, \emptyset) = S(K)$.

Note that there is a convenient way to describe the function S' in many cases. Given a knowledge base K and context C, it will often be possible to create a knowledge base K' such that $S'(K, C) = S(K')$. In these cases, reasoning within a context can be reduced to changes of the knowledge base K (converting it into K'), and by reusing existing reasoners.

Formal Definition of Context We will now go into further detail. Taking some language L, a *knowledge base on* L is a (possibly infinite) set of statements over L. Let $\mathcal{KB}(L)$ denote the set of all knowledge bases expressible in L.

Now, we consider a language L_{k} called *knowledge language*. A semantics for L_{k} can be formalized as a function $S : \mathcal{KB}(L_{\mathrm{k}}) \rightarrow \mathcal{KB}(L_{\mathrm{k}})$ assigning to a knowledge base K a knowledge base $S(K)$ containing all logical consequences of K expressible in the knowledge language.

Let furthermore be L_{c} a language called *context language* for expressing contextual knowledge. An L_{c}-context semantics for L_{k} is then a function $S : \mathcal{KB}(L_{\mathrm{k}}) \times \mathcal{KB}(L_{\mathrm{c}}) \rightarrow \mathcal{KB}(L_{\mathrm{k}})$. (The overloading of the symbol S is by purpose.)

[1] Please note that within this deliverable we do not distinguish between the use of the notion of "ontology" and that of "knowledge base". However, for the logical characterization, we tend to prefer the term "knowledge base".

In practice, one will mostly impose further restrictions on the knowledge base one works with. E.g., a knowledge base could be required to contain only certain kinds of expressions from L_{k} – as an easy example, take a database containing only tuples of entities (or, similarly, a logic program containing only ground facts) while the entailed knowledge (respectively the expressible queries) could have a much more complex structure. Another common constraint to knowledge bases is that they have to be finite (or at least finitely representable in some sense). The set of finite knowledge bases over some language L_{k} will be denoted by $\mathcal{KB}_{\mathrm{fin}}(L_{\mathrm{k}})$.
In many cases, additional constraints will be reasonable. In particular, we will call a context semantics

- *conservative*, if $S(K, \emptyset) = S(K)$ for all $K \in \mathcal{KB}(L_{\mathrm{k}})$. This means that, if an empty context (i.e. no contextual information) is provided, the semantics coincides with the "pure" semantics of the knowledge language.
- *extensive*, if $K \subseteq S(K, C)$ for all $K \in \mathcal{KB}(L_{\mathrm{k}})$ and for all $C \in \mathcal{KB}(L_{\mathrm{c}})$, i.e., all statements of the knowledge base are as well logical consequences of it. In other words, any information stated in the knowledge base can be deduced to be valid (and cannot be spoilt by whatever context provided).
- *knowledge-monotone*, if $K_1 \subseteq K_2$ implies $S(K_1, C) \subseteq S(K_2, C)$ for all $K_1, K_2 \in \mathcal{KB}(L_{\mathrm{k}})$ and $C \in \mathcal{KB}(L_{\mathrm{c}})$, i.e., all logical consequences remain valid if the knowledge base is augmented and the context does not change. Note, that this is not always the case (cf. non-monotonic semantics by closed world assumption).
- *context-monotone*, if $C_1 \subseteq C_2$ implies $S(K, C_1) \subseteq S(K, C_2)$ for all $C_1, C_2 \in \mathcal{KB}(L_{\mathrm{c}})$ and $K \in \mathcal{KB}(L_{\mathrm{k}})$, i.e., if the information given by the context increases, the derivable information does so as well. In particular, no previously valid consequence can be invalidated by adding more contextual knowledge.
- *idempotent*, if $S(S(K, C), C) = S(K, C)$ for all $C \in \mathcal{KB}(L_{\mathrm{c}})$ and $K \in \mathcal{KB}(L_{\mathrm{k}})$, i.e., taking all consequences of a knowledge base under a certain context and then taking again all consequences under the same context will yield nothing new.
- *dependently reducible*, if there is a function $\sigma : \mathcal{KB}(L_{\mathrm{k}}) \times \mathcal{KB}(L_{\mathrm{c}}) \to \mathcal{KB}(L_{\mathrm{k}})$, such that $S(K, C) = S(\sigma(K, C), \emptyset)$, i.e., knowing a knowledge base K and a context C, one can determine a new finite knowledge base with the same set of consequences as K with context C. I.e. for every contextualized knowledge base we can determine a logically equivalent knowledge base without context.
- *independently reducible*, if there is a function $\tau : \mathcal{KB}(L_{\mathrm{c}}) \to \mathcal{KB}(L_{\mathrm{k}})$ such that $S(K, C) = S(K \cup \tau(C), \emptyset)$ for all $K \in \mathcal{KB}(L_{\mathrm{k}})$ and $C \in \mathcal{KB}(L_{\mathrm{c}})$, i.e., any context can be "translated" into L_{k} (independently from K) and simply added to the knowledge base. In this case, contextual reasoning could be reduced to pure reasoning over L_{k}, such that existing methods could easily be employed for this.

The above definition is very abstract. This is done on purpose to accommodate the many practically important ways of context usage. In Chapter 3, we give some examples of concrete instances of the abstract definition. Many more notions of context fit our general definition. In the project, we will have to determine which concrete instances will be used and supported by the NeOn system. These instances will have to be dealt with on an individual basis when realizing the NeOn system.

Chapter 3

Instantiation

In this chapter we present instantiations of the generic definition from the previous chapter for a number of types of contexts. NeOn has in its core the ambitious scenario that ontologies are developed in the open environment in a distributed fashion. Moreover, it is not just the ontologies and meta-data that are distributed, but we also assume that they are built by distributed teams. In terms of the usages of context, this means that *supporting viewpoints and perspectives* will play a paramount role in NeOn. In the scenarios addressed by NeOn, information sources typically cannot be easily integrated without violating the overall consistency of the system. Thus *dealing with inconsistent information* will be another important usage of context, where information about the provenance of ontological structures, about various contexts and user profiles leads to the generation of local consistent views out of a globally inconsistent network of ontologies. Closely related is the problem of *dealing with uncertain and vague information*, which will play an important role in the NeOn scenarios, where ontologies are generated from a variety of sources that may be imprecise, vague and contextualized in the first place (e.g. natural language text) and where automated ontology learning algorithms introduce an additional dimension of uncertainty. In D3.1.1 we identified the following forms of reasoning with context as particularly interesting for the NeOn project:

- The *networked ontology model* developed in WP1 will provide the most obvious form of context: Ontologies will be embedded in a network of ontologies, which forms the context for its interpretation. Depending on the types of relationships in the network of ontologies, different forms of context can be realized: The context may be a temporal one if the network represents a version space; it may be used to connect different viewpoints via alignments, etc.

- *Reasoning with inconsistent ontologies* exploiting context information will be important when different information sources with contradicting information will be integrated.

- *Context-based selection functions* appear promising for addressing a number of different problems. However, the development of such functions that go beyond the state-of-the-art syntactic-based functions will require significant research efforts.

- Finally, a combination of *possibilistic* and *probabilistic logics* seems to be required to deal with the various forms of vagueness and uncertainty in a contextualized way.

As the semantics of the networked ontology model is specified as part of WP1, we do not provide its formalization as part of this deliverable. Instead, we focus on the types of context to support the latter three aspects of context reasoning. In particular, we describe *Provenance* as a form of context that is typically available for automatically generated ontologies – e.g. in ontology learning – and *Argumentation Structures* as a form of context that is obtained in collaborative ontology engineering processes.

3.1 Provenance as Context in Dealing with Imperfect Information

In many cases, the information derived from diverse sources leads to inconsistencies. This is especially the case if information is derived using automatic knowledge acquisition tools such as wrappers [FK00] or information extraction systems (e.g. [Cir01], [BCRS06]) or tools for automatic or semi-automatic ontology learning such as OntoLT [BOS03], OntoLearn [NVCN04], ASIUM [FN98] or Text2Onto [CV05] that aim at the semi- or even fully automatic extraction of ontologies from sources of textual data. Common to all of them is the need for handling the uncertainty which is inherent in any kind of knowledge acquisition process. Moreover, ontology-based applications which rely on learned ontologies have to face the challenge of reasoning with large amounts of imperfect information resulting from automatic ontology generation systems.

Different causes for the imperfection of information can be found identified. According to [AM97] imperfection can be due to *imprecision*, *inconsistency* or *uncertainty*. Imprecision and inconsistency are properties of the information itself - either more than one world (in the case of ambiguous, vague or approximate information) or no world (if contradictory conclusions can be derived from the information) is compatible with the given information. Uncertainty means that an agent, i.e. a computer or a human, has only partial knowledge about the truth value of a given piece of information. One can distinguish between objective and subjective uncertainty. Whereas objective uncertainty relates to randomness referring to the propensity or disposition of something to be true, subjective uncertainty depends on an agent's opinion about the truth value of the information. In particular, the agent can consider information as unreliable or irrelevant.

Depending on the type of imperfection, different approaches for interpreting the information may be adequate. For each of them, the provenance of information can be exploited, as we have outlined in the deliverable D3.1.1 [HHR$^+$06].

There are mainly two classes of languages for representing dealing with uncertainty: probabilistic logic and possibilistic logic. Dealing with probabilistic uncertainty in the Semantic Web has been recognized as an important problem in the recent decades. Many approaches have been proposed to extend description logics with probabilistic reasoning [Jae94, Hei94, GL02, DS05, NF04, DP04, KLP97]. These approaches can be classified according to ontology languages, the supported forms of probabilistic knowledge and the underlying probabilistic reasoning formalism. In probabilistic extensions of description logics, a probability value (or an interval) is often attached to a *conditional constraints* of the form $(D|C)$, where C and D are concepts. By contrast, there is relatively few work on combining possibilistic logic and description logic. Possibilistic logic [DLP94] or possibility theory offers a convenient tool for handling uncertain or prioritized formulas and coping with inconsistency. It is very powerful to represent partial or incomplete knowledge [BLP04]. There are two different kinds of possibility theory: one is qualitative and the other is quantitative. Qualitative possibility theory is closely related to default theories and belief revision [DP91, BDP92] while quantitative possibility can be related to probability theory and can be viewed as a special case of belief function [DP98]. One of the major problems with the quantitative possibility theory is that the weights attached to formulas are usually hard to obtain. When numerical information is not available, we often use qualitative possibility theory. In this case, a possibilistic knowledge base can be viewed as a stratified knowledge base, i.e. knowledge bases in which all pieces of information are assigned a rank. The confidence values attached to axioms in possibilistic description logics are often used to represent priority levels of the axioms. A challenging problem here is to extract ranking information from an inconsistent or incoherent ontology. In Chapter 5, we will provide some approaches to automatically computing the ranking on axioms in an ontology.

Fusing mutually inconsistent knowledge extracted from heterogeneous sources in essence requires (i) an algorithm to pinpoint down where the inconsistencies arise, (ii) a procedure to resolve inconsistencies by removing axioms leading to these inconsistencies, as well as (iii) a representation of the provenance of axioms as context information on the basis of which to decide which axioms should be removed.

For the knowledge fusion scenario, we intend to build on the approach of [HV05] to find *minimally inconsistent axioms sets* within a given ontology. The idea behind minimally inconsistent axiom sets is that the removal of one axiom in each set will lead to consistency. The decision which axiom to remove can then indeed be

guided by the provenance context as described above.

In the following, we introduce a model for representing provenance in ontology learning. The main part of the model is shown in Figure 3.1. We associate with each of the learned ontology elements a confidence and a relevance value which allow for a more target-oriented inspection of the learned ontology. Both confidence and relevance values can be considered as context annotations defined as follows:

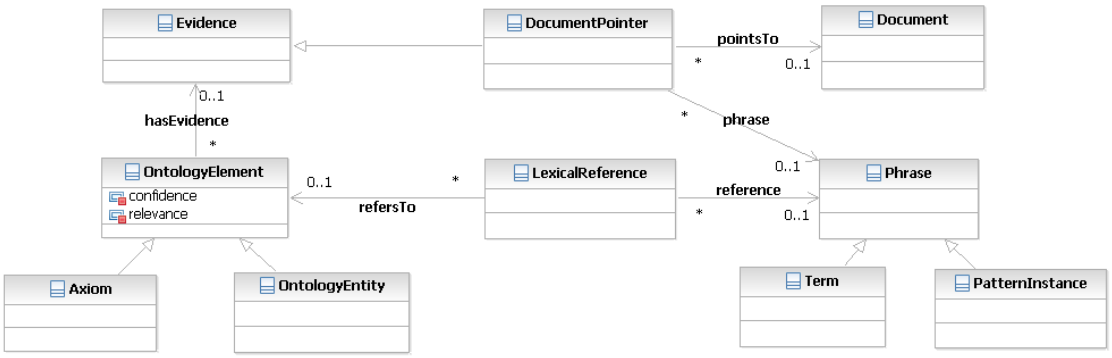

Figure 3.1: Provenance Context for Ontology Learning

- We use a **confidence** rating to indicate how confident the system is about the correctness of an ontology element. The confidence rating is determined by the number and quality of evidences found in the corpus.

- We use a **relevance** rating to denote the relevance of an ontology element with respect to a particular domain given by the corpus.

In addition to confidence and relevance values each ontology element is associated with evidences and references that can be used to generate formal or informal explanations for particular results.

Evidences are the basis for any computation of a confidence value. Typical examples for evidences which may lead to the creation of a `subclass-of` relation, for instance, include Hearst patterns [Hea92] and hyponymy relationships in WordNet [Fel98].

References are pointers to occurrences of the regarding element in the corpus or other underlying knowledge sources such as ontologies or lexical resources. Adding a list of references to each ontology element not only increases the traceability of the ontology learning process, but also facilitates the detection of ontology changes in case of changes to the corpus.

Further extensions can be made to the model to represent information about the algorithms which suggested the addition or removal of a particular ontology element, learning parameters and timestamps.

Axiom t	Confidence
$Architecture \sqsubseteq \neg Tool$	0.10
$Methodology \sqsubseteq \neg Tool$	0.10
$Tool \sqsubseteq Implementation$	0.40
$Application(kavido)$	0.46
$Tool(kavido)$	0.46
$Tool(amilcare)$	1.0
$Tool \sqsubseteq \neg Application$	0.3

Table 3.1: Sample ontology with confidence values

3.1.1 Ontology learning example

In this subsection, we use an example from [HV05] to illustrate how confidence values attached to axioms of an ontology can be used to deal with imperfect information. In this example, the ontology is obtained by ontology learning.

Let us consider the ontology in Table 3.1 below. The example exhibits two forms of imperfection: First, the elements of the learned ontology are uncertain, as indicated by the confidence values that are attached to the axioms in the ontology. Second, the obtained knowledge is inconsistent: Here $KaViDo$ was identified to be both an instance of $Application$ and a $Tool$, however, $Application$ and $Tool$ were learned to be disjoint concepts, so the ontology is inconsistent. We may consider two ways to deal with inconsistency in the ontology: we can either delete some axioms to restore inconsistency or tolerate the inconsistency and apply possibilistic logic approach.

To resolve inconsistency, we first need to find the minimally inconsistent sub-ontologies of the inconsistent ontology based on the algorithm. There is only one minimally inconsistent sub-ontology which contains the following axioms: $Tool \sqsubseteq \neg Application$, $Application(kavido)$, $Tool(kavido)$. Taking the confidence values of the axioms into account, we can resolve the inconsistency by removing the disjointness axiom whose confidence value is the lowest. In this case, removing the axiom $Tool \sqsubseteq \neg Application$ would yield a consistent ontology.

We next illustrate how to reason with the inconsistent ontology using possibilistic logic. In this case, the confidence values attached to axioms are explained as certainty degrees of the axioms. We first need to find the *inconsistency degree* of this ontology, which is the maximal weight of axioms such that all the axioms whose weights are greater than or equal to it are inconsistent. An axiom can be inferred from the ontology using possibilistic inference if and only if it can be inferred from axioms whose weights are greater than the inconsistency degree. For the ontology in this example, its inconsistency degree is 0.3. So, we can infer $Tool(kavido)$ whose weight is 0.46, which is greater than 0.3. Furthermore, we can also infer $Implementation(amilcare)$ with confidence degree 0.40 using possibilistic inference.

3.1.2 Instantiation of the Generic Definition

Let K be a knowledge base in an ontology language L. The context information is the confidence and relevance values attached to formulae in the knowledge base. In the case of probabilistic logic, the context language is the conditional probabilistic terminology for probabilistic logic or conditional probability table for Bayesian networks. The semantics of the context language is a knowledge base which consists of K and probabilistic terminologies. Whilst in the case of possibilistic logic, the context information can be used to obtain the weights or priority levels attached to formulae in the knowledge base. That is, $S(K, C)$ is a weighted knowledge base or a prioritized knowledge base.

The context information can be also used to guide how to resolve inconsistency in the knowledge base. It is clear that this context language is conservative. However, it is neither extensive nor knowledge-monotonic,

because the underlying logic (i.e. possibilistic logic) is inherently nonmonotonic.

3.2 Argumentation Structures as Context

Argumentation frameworks usually provide information that can be useful to provide information about the context of ontology elements.

The two classical approaches on argumentation theory [Tou58, POT70] provide general schemas for representing argumentative processes. However, argumentation is a large field in continuous evolution and proposed frameworks vary from informal to rigorous formal ones. One acknowledged problem of these theories is the fact that it is difficult to find the appropriate level of detail with which to represent arguments. For some more details about argumentation theories see D2.1.1. In the case of ontology based argumentation frameworks, DILIGENT [TPSS05] proposes an argumentation ontology that can be used to formally represent the arguments exchanged by ontology engineers in ontology building processes. Some initial experiments have been conducted. Another framework currently under development under the Neon Project is C-ODO [CGL$^+$06].

These frameworks usually provide important information that can be used in context representation and moreover that can be used in inference processes. In the following paragraphs we are taking as starting point the DILIGENT argumentation framework [TPSS05, Tem06]. The DILIGENT argumentation ontology adapts, for ontology engineering purposes, the IBIS methodology, which proposes one model to formally structure arguments. The DILIGENT argumentation framework consists of two building blocks; a process and an argumentation ontology. In the argumentation process five main activities take place: choose moderator, choose decision procedure, specify issues, provide arguments and ideas, decide on issues and ideas. The argumentation ontology formalizes the arguments exchanged during ontology engineering discussions.

The DILIGENT Argumentation Ontology is visualized in Figure 3.2. It formalizes the arguments exchanged during ontology engineering discussions. The main concepts in this ontology are issues, solution proposals and arguments. An *issue* introduces a new requirement or topic in a discussion from a conceptual point of view. Issues may refer to particular ontology elements. They are used to discuss problems in the definition of ontology elements without yet taking into account how the problems should be resolved and implemented in the ontology. *Solution proposals* are put forward as ideas to *address issues* and refer to their formalization in the ontology, for instance as a class, instance, relation or axiom. Typically, a solution proposal encompasses one or more ontology changes that may affect the definition of ontology elements. New requirements or topics are introduced as issues, which can be extended or refined by *generalizing* or *specializing* an issue. When the experts start discussing how a given issue, a domain concept, can be represented in the ontology, they discuss in terms of solution proposals, how domain knowledge can or should be formalized. Accepted solution proposals trigger concrete ontology change operations. *Arguments* can be exchanged on particular solution proposals, either supporting an idea or objecting (counter-argument).

The concepts an ontology represents should be consensual, this requires some consensus building discussions. In DILIGENT processes, concepts are only added to the ontology if they can be agreed upon, that is after some arguments have been exchanged, positions by different actors have been issued on them and some decisions have been made. DILIGENT proposes examples, evaluations and justifications as particularly useful argument types. Those involved in discussions can state *positions*. They clarify the position on one particular solution proposal under discussion. Possible positions are *agreement* and *disagreement*. Once enough arguments have been provided and positions have been stated on them, decisions can be made. In general, positions lead to decisions. Decisions are taken on issues. A decision has a status that can vary from under-discussion, postponed, discarded and agreed. A decision records not only the issue on which it was taken, but also both the positions issued when final with-votes (several positions) were cast and the line of reasoning (a sequence of arguments) underlying the decision on that issue.

The argumentation structures can be exploited as context information in various ways in the ontology reasoning. In the following we will illustrate how this can be done for the task of diagnosis and repair. Diagnosis

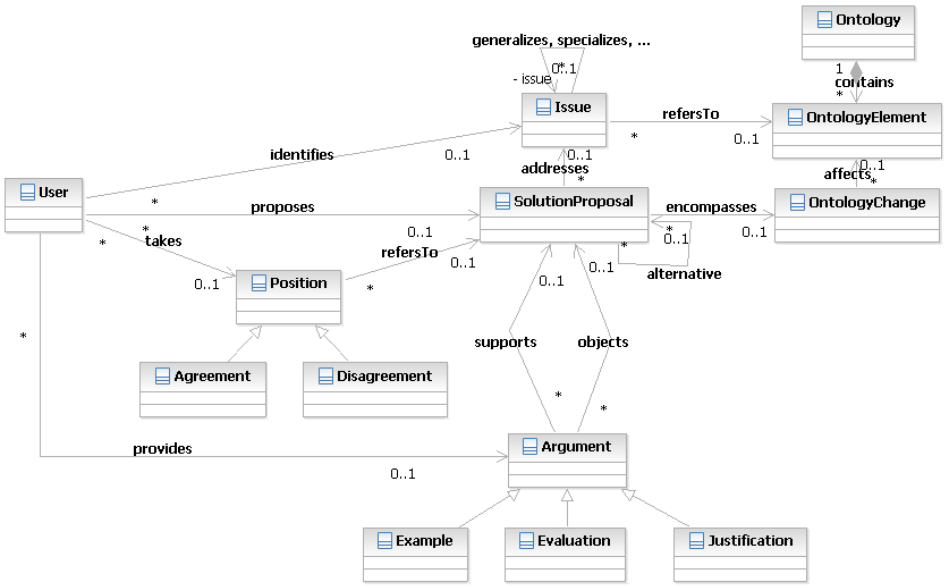

Figure 3.2: The major concepts of the argumentation ontology and their relations

and repair of ontologies is an important aspect of ontology engineering. This is especially the case in collaborative engineering environments, where there may be conflicts and disagreements about the meaning and definition of concepts among the ontology engineers.

Reasoning agents can be useful in different steps of the argumentation process: Indeed, in every phase they can act in place of a user, i.e. they can *identify issues*, they can *propose solutions*, they can *take positions* and they can *provide arguments*. Identifying *issues* corresponds to the identification of a logical contradiction in the ontology. To resolve the logical contradiction, it is important to pinpoint the erroneous class definitions in the ontology which are responsible for the contradiction. (Please refer to [QHJ07] for an overview of different approaches for this task.) When repairing an ontology, we usually have different alternatives to resolve the incoherence, i.e. the diagnosis algorithms may generate a number of different *solution proposals*. Some context information such as ranking information is often needed to select the best solution [KPSG06]. The idea here is to use the argumentation structures as context information to define useful rankings for the different solution proposals. Based on the ranking, the reasoning agent would define a *position* on a solution proposal, i.e. recommending one (or multiple) solution proposals to implement. The final decision can then either be done automatically (by accepting the recommendation of the reasoning agent) or it can be left to the user. If the decision is left to the user, the individual arguments can be presented to the user (based on the argumentation ontology) in order to support the decision process.

3.2.1 The University example

Let us suppose we want to model the University domain. In a DILIGENT ontology engineering process participants would start by proposing issues and after enough arguments had been provided and that participants had reached an agreement as to what should be represented in the ontology, the arguments would be attached to ontology elements. We will be using the real data from the discussions that took place in one of the DILIGENT case studies, partially reported in [PST04]. We here assume that the ontology targeted by Figure 3.3 contains concepts such as organization, and in particular university, persons, employees, Professors, students, PhD students, study programs. All these concepts are all related since university employees work at universities, both Professors and PhD students are university employees, students are enrolled in study programs and these are offered by universities. In the figure, we use the symbol circle to denote the disjointness of two concepts Students and (Uni)Employees.

This example contains a contradiction: PhDstudents are both subclass of Students (PhDstudents$\sqsubseteq$Students) and University Employees (PhDstudents$\sqsubseteq$Employees), and these classes are disjoint (Students$\sqcap$Employees$\sqsubseteq\bot$). To resolve the contradiction, we can get an ordering relation on the axioms which are involved in the conflict based on argumentation information. Then we can simply delete the axiom which has least priority.

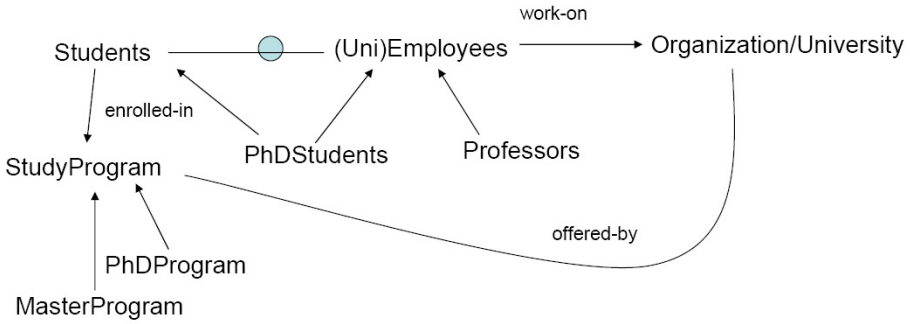

Figure 3.3: University ontology

In the argumentation ontology, we may have the following structures:

- SolutionProposal to introduce PhD students being students:
  Arguments:
  support: Students are enrolled in student program.
  objects: Student need to pay taxes.
  Voting on this SolutionProposal: 4 agreements; 2 disagreements

- SolutionProposal to model PhDStudent as subclass of Employee
  Arguments:
  support: PhD students work on projects; They have a salary.
  objects: PhD students have a scholarship; They do not have a contract;
  Voting on this SolutionProposal: 4 agreements; 3 disagreements

- SolutionProposal to define students disjoint from employees
  Arguments:
  support: Students cannot be employed by the university, as they are enrolled in study program.
  objects: Students may work as as research assistants and tutors, thus as employees
  Voting: on SolutionProposal: 3 agreements; 3 disagreements

To resolve the incoherence in the University ontology, we have at least the following three solutions: (1) delete the class definition PhDStudents is a subclass of Students, or (2) delete the class definition PhDStudents is a subclass of Employees, or (3) delete the class definition “Students disjoint from Employees". However, without extra information, we do not know which class definition should be deleted. In this case, we can turn to argumentation ontology to get some ranking information on the class definitions. In the argumentation ontology, the difference between numbers of argument support and against “PhD students being students" is 0, the difference between numbers of argument support and against “PhD students being employees" is 1, and the difference between numbers of argument support and against “students disjoint from employees" is 2. Therefore, we can conclude that the class definition that “PhD students being students" has the least priority. So we can simply delete this class definition to restore coherence.

3.2.2 Instantiation of the Generic Definition

Let K be a knowledge base in an ontology language L. The context information C is then the set of arguments which are provided by actors and is stored in an argumentation ontology. The semantics of K in the context C is then an ontology which contains both K and C. It is clear that the context semantics is conservative and idempotent. According to the University example, it is clear that the context semantics is not extensive because the axiom “PhD students being students" is not included in the final result. It is neither knowledge-monotone nor context-monotone. For knowledge-monotone, if we add an axiom $PhDStudents(John)$ to state that John is a PhD student and an argument for it and an argument again it. Then we may delete the axiom $PhDStudents(John)$ and keep the class definition that “PhD students being students". So knowledge-monotone is violated. For context-monotone, suppose we have more arguments support the class definition “PhD students being students", for example, PhD students need to attend lectures and PhD students have a student card, then we may delete the class definition “PhD students being employees" to restore consistency.

3.3 Summary

In this chapter, we considered two types of context which are of particular interest to NeOn project: provenance and argumentation structures. We then related these two types of context to the general definition of context given in Chapter 2. In the next chapter we will discuss how the context can be syntactically represented to be able to relate an ontology with its context. After that, we will give some approaches for measuring inconsistency to illustrate how to automatically get ranking information on axioms from an incoherent ontology.

Chapter 4

Representing Context in OWL Ontologies

In this chapter we address the question how the context can be syntactically represented to be able to relate an ontology with its context. We therefore propose so called groundings of the context representation within OWL that allow to specify the context itself in the form of an OWL ontology. This requires to talk about the elements of the ontology, its facts and axioms as first class objects. However, when talking about ontologies and their elements themselves – i.e. annotating them with context, for example about their provenance, trustworthiness, arguments – we are severely restricted. For example, treating classes and individuals on the same metalevel, introduces difficulties and is not possible in the decidable variants of the OWL ontology language. Also, in OWL it is currently not possible to talk about axioms as first class objects in the language. Our approach – as originally presented in [VVH+06] – builds on the metamodel for networked ontologies as described in D1.1.1 [HRW+06]. As extension, we also include axioms as elements in the metamodel (c.f. Figure 4.1). We propose two different possible groundings of the metamodel in the OWL DL language. The first approach relies on a meta-ontology to ground the metamodel in a way compatible with the current OWL 1.0 standard. The other grounding requires an extension of OWL 1.0 by introducing URIs for axioms in the ontology. This extension will be available in the OWL 1.1 language.

4.1 Alternatives for Representing Context in OWL

4.1.1 Annotation Compatible with OWL 1.0 using a Meta-Ontology

We have defined an OWL DL meta-ontology[1] to capture the metamodel as presented in the previous section. In this ontology we explicitly also capture axioms. Ontologies can be transformed to become instance data with regards to the vocabulary of the meta-ontology. We here first present a small fragment of the meta-ontology that reflects the part of the metamodel shown in Figure 4.1. Then we will give a small example, how an ontology looks like when transformed. Note that axioms, in this case, translate to proper individuals as well, and thus become annotatable, as we will show in the example.

(1)Class $\sqsubseteq$ OntologyEntity
(2)Axiom $\sqsubseteq$ OntologyElement
(3)SubClassOfAxiom $\sqsubseteq$ Axiom
(4)EquivalentClassAxiom $\sqsubseteq$ Axiom
(5)DisjointWithAxiom $\sqsubseteq$ Axiom
$(6)\top \sqsubseteq \forall$subClassOfSubClass.Class
$(7)\top \sqsubseteq \forall$subClassOfSubClass^{-1}.SubClassOfAxiom
$(8)\top \sqsubseteq \leq 1$ subClassOfSubClass.$\top$
$(9)\top \sqsubseteq \forall$subClassOfSuperClass.Class
$(10)\top \sqsubseteq \forall$subClassOfSuperClass^{-1}.SubClassOfAxiom
$(11)\top \sqsubseteq \leq 1$ subClassOfSuperClass.$\top$

[1] http://owlodm.ontoware.org/OWL1.0

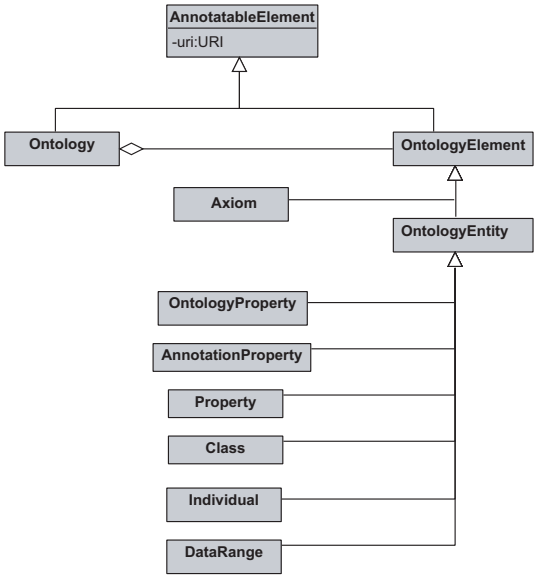

Figure 4.1: Main Elements of the Ontology Definition Metamodel

$(12) \top \sqsubseteq \forall$EQUIVALENTCLASS.CLASS
$(13) \top \sqsubseteq \forall$EQUIVALENTCLASS^{-1}.EQUIVALENTCLASSAXIOM
$(14) \top \sqsubseteq \forall$DISJOINTWITH.CLASS
$(15) \top \sqsubseteq \forall$DISJOINTWITH^{-1}.DISJOINTWITHAXIOM

Axioms 1-5 define the terms used. Every axiom type is defined by a class of its own (refer to the following example). The rest of the axioms defines the domain and ranges of the used properties.

In the following, we demonstrate the use of the meta-ontology based on the small ontology from the Ontology Learning example in Section 3.1.1, stating that tools and applications are disjoint concepts:

TOOL $\sqsubseteq \neg$APPLICATION

Using the meta-ontology, we can represent this ontology as follows:

CLASS(*Tool*)
CLASS(*Application*)
DISJOINTWITHAXIOM(*axiom1*)
DISJOINTWITH(*axiom1*, *Tool*)
DISJOINTWITH(*axiom1*, *Application*)

Note that the class TOOL is something else than its representing individual in the meta-ontology, which is the individual *Tool*. The axiom of the original ontology is reified explicitly as the individual *axiom1*, an instance of the class DISJOINTWITHAXIOM (as it is an axiom that represents a subclass relation between to classes). The axiom is connected to the entities taking part in that axiom with the given properties (i.e., the DISJOINTWITH property points to the classes in the axiom that are stated to be disjoint with one another).

Now it is possible to state further facts about this axiom, like its source or the confidence we put into the axiom, within the ontology:

CREATOR(*axiom1*, *text2onto*)
CONFIDENCE(*axiom1*, 0.3)

Naturally, we also can talk about the entities of the ontology in the same manner:

CREATOR(*Tool*, *text2onto*)

We have implemented the presented approach of creating a meta-ontology out of OWL DL ontologies within OWL DL. The implementation is available as part of the KAON2 OWL tools[2], and is based on the KAON2

[2] http://owltools.ontoware.org

reasoner and ontology management infrastructure [MS05]. Two tools are currently implemented:

- `owl meta` creates a meta-ontology from an input ontology. Both the ontological entities like individuals, properties and classes, as well as the axioms of the ontology are transformed. The ontology is transformed in such a way that it can be safely merged with the original ontology itself, thus allowing for rich metadata within the ontology without breaking it.
- `owl demeta` on the other hand takes a meta-ontology and creates the ontology that would have caused the given meta-ontology.

A problem with the meta-ontology is that it is considerably bigger than the original ontology. As every ontology entity is described with an extra axiom, and every already existing axiom is described with at least two further axioms, the meta-ontology often is three or four times as big as the original ontology. Typically, the meta-ontology is only processed automatically, and not presented to the user. Therefore, it remains a question of scalability of the tools and higher requirements regarding resources, but the user does not have to deal with the growing size of the ontology. The actual growth rate depends on structural properties of the ontology, but is basically a constant factor to the size.

4.1.2 Annotations of Axioms with URIs

In this section, we discuss an alternative grounding for the proposed metamodel. In contrast to the one above, these approaches require an extension of the current OWL 1.0 standard in order to assign URIs to axioms. These extensions will be available in the OWL1.1 language. That is, axioms become entities in the ontology that can be referred to, i.e. they are reified in the ontology. In the following, we denote the URI of an axiom within square brackets following the axiom[3]:

TOOL $\sqsubseteq \neg$APPLICATION $[$*axiom1*$]$

We are now able to refer to the axiom using its URI. Depending on how this is done, we distinguish two approaches discussed in the following.

Annotations using a Meta-Ontology

In this approach, a separate meta-ontology in OWL DL is employed in order to make statements about ontology elements in a similar way as in the approach described in Section 4.1.1. However, it is not required anymore to reproduce the entire ontology within the meta-ontology, as we are now able to refer to all elements of the original ontology – including its axioms – via their URI:

AXIOM$($*axiom1*$)$
CREATOR$($*axiom1*, *text2onto*$)$
CONFIDENCE$($*axiom1*$, 0.3)$

Annotations using Annotation Properties

The final grounding we propose abandons the use of a meta-ontology. Instead, annotation properties are used for the representation of metadata about ontology elements. In the current version of the specification of OWL DL, it is possible to annotate ontology entities (i.e., classes, properties and individuals) in this way. We here propose to allow such annotations also for axioms. The above example would then look as follows:

TOOL $\sqsubseteq$ APPLICATION $[$*axiom1*$]$
CREATOR$($*axiom1*, *text2onto*$)$
CONFIDENCE$($*axiom1*$, 0.3)$

[3] For the serialization of this extension in an XML based syntax, we follow the approach taken in the current proposal for the OWL 1.1. language.

In contrast to the grounding described in Section 4.1.2, CREATOR and CONFIDENCE here denote *annotation properties*. These annotations constitute non-logical information, meaning that their treatment is outside the (regular) semantics of OWL. They can thus be stored within the original ontology without causing semantic conflicts.

4.2 Summary

In this chapter, we have presented a proposal on how to represent context information as annotations to ontology elements in OWL DL ontologies. The proposal is based on an extension to the networked ontology model, in which axioms are also treated as ontology elements and identifiable via a URI. We are thus able to capture context of axioms, which is critical for many use cases of context. We have presented a number of different groundings that are compatible with the current OWL 1.0 standard.

The extensions to refer to axioms in the ontology via a URI will be directly available in the OWL 1.1 language. The grounding of our proposal will then be much more straight-forward.

Chapter 5

Measuring Incoherence in OWL DL

Incoherence can occur for several reasons, such as modeling errors when constructing an ontology and migration or merging of ontologies [Sch05]. For example, two upper ontologies SUMO and CYC are used in a single document, there exist over 1000 unsatisfiable concepts. Currently, there are many discussions on how to debug and diagnose terminologies in ontologies [KPSG06, SC03, Sch05]. Therefore, incoherence is often viewed as negative information in an ontology. However, by measuring incoherence, we can get some useful context information for maintaining and evaluating an ontology. For example, by measuring the extent of incoherence of different ontologies, we can rank them. That is, an ontology is of better quality than another one if it contains less incoherent information. Similarly, by measuring the extent of incoherence of different axioms, we get some ranking information on axioms which can be used to resolve incoherence [KPSG06]. Furthermore, there is a trade-off between the amount of useful information in an ontology and the amount of coherent information. For example, in the extreme, we could guarantee a coherent ontology by only having the empty ontology. Obviously, this would not be acceptable, and so we need to tolerate the possibility of some incoherence in an ontology. Once, we do tolerate this possibility, we need to consider keeping track of it, perhaps as part of a process of improvement, or as a way of isolating the problematical parts of the ontology until we decide to fix those parts. So for these tasks, it is helpful to know where and by how much there is incoherence. This may include consideration of whether it is widespread, or localized. So the size and overlaps of incoherent subsets can provide important context information for dealing with incoherence and inconsistency. A simple way to measure incoherence is to consider the arguments for and against incoherent concepts.

In this chapter, we propose some approaches for measuring incoherence in DL $\mathcal{SHOIN}$ which provides the foundation for the W3C Web Ontology Language OWL. In our framework, based on scoring functions [Hun04], we present two classes of measures of incoherence: measures of incoherence for unsatisfiable concepts and measures of incoherence for terminologies. First, we define the scoring function for an unsatisfiable concept and use it to define a score ordering on unsatisfiable concepts. Second, we define the scoring function for a TBox and use it to define an ordering on terminology axioms and an ordering on TBoxes.

The approaches for measuring incoherence in OWL ontologies will allow us to automatically obtain context information. This context – in the form of ranking information – can be exploited in dealing with imperfect information, where there are usually several alternative solutions available and we need to explore ranking information to select the best solution.

5.1 Incoherence in OWL ontologies

We introduce the notion of incoherence in OWL ontologies defined in [FHP$^+$06].

Definition 1 (Unsatisfiable Concept) *A concept name C in a terminology $\mathcal{T}$, is unsatisfiable iff, for each interpretation $\mathcal{I}$ of $\mathcal{T}$, $C^{\mathcal{I}} = \emptyset$. The set of all unsatisfiable concept is denoted as $US(\mathcal{T})$.*

That would lead us to consider the kinds of terminologies and ontologies with unsatisfiable concepts.

Definition 2 (Incoherent Terminology) *A TBox $\mathcal{T}$ is incoherent iff there exists an unsatisfiable concept name in $\mathcal{T}$.*

Definition 3 (Incoherent Ontology) *An ontology O is incoherent iff its TBox is incoherent.*

According to the above definitions, we know that the incoherence can occur only in the terminology level. Namely, an ontology $O = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ is incoherent iff its terminology $TBox$ is incoherent. Therefore, when we talked about an axiom in an ontology, we only mean the concept inclusion axiom. Incoherence does not provide the classical sense of the inconsistency because there might exist a model for an incoherent ontology.

Definition 4 (Inconsistent Ontology) *An ontology O is inconsistent iff it has no model.*

However, incoherence and inconsistency related with each other. According to the discussion in [FHP$^+$06], incoherence is potential for the cause of inconsistency. That is, suppose C is an unsatisfiable concept in $\mathcal{T}$, if a concept assertion $C(a)$ exists in the ABox $\mathcal{A}$, then the ontology O is inconsistent.
In the following, we introduce some definitions which are useful to explain logical incoherence.

Definition 5 *[SC03] Let A be a concept name which is unsatisfiable in a TBox $\mathcal{T}$. A set $\mathcal{T}' \subseteq \mathcal{T}$ is a* minimal unsatisfiability-preserving sub-TBox (MUPS) *of $\mathcal{T}$ if A is unsatisfiable in $\mathcal{T}'$, and A is satisfiable in every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$. The set of all MUPS of $\mathcal{T}$ with respect to A is denoted as $MU_A(\mathcal{T})$*

A MUPS of $\mathcal{T}$ and A is the minimal sub-TBox of $\mathcal{T}$ in which A is unsatisfiable.

Definition 6 *[SC03] Let $\mathcal{T}$ be an incoherent TBox. A TBox $\mathcal{T}' \subseteq \mathcal{T}$ is a* minimal incoherence-preserving sub-TBox (MIPS) *of $\mathcal{T}$ if $\mathcal{T}'$ is incoherent, and every sub-TBox $\mathcal{T}'' \subset \mathcal{T}'$ is coherent. The set of all MIPSs of $\mathcal{T}$ is denoted as $MI(\mathcal{T})$.*

A MIPS of $\mathcal{T}$ is the minimal sub-TBox of $\mathcal{T}$ which is incoherent. We say a terminology axiom is *in conflict* in $\mathcal{T}$ if there exists a MIPS of $\mathcal{T}$ containing it.

5.2 Measures of Incoherence

5.2.1 Measures of incoherence for unsatisfiable concepts

If an ontology is incoherent, there is at least one unsatisfiable concept in its TBox. For these unsatisfiable concepts, some are more problematic than others. For example, given a TBox $\mathcal{T} = \{A \sqsubseteq B, B \sqsubseteq C, B \sqsubseteq \neg C\}$. A and B are both unsatisfiable concepts. However, A is unsatisfiable because of B. That is, if B becomes satisfiable, then A is also satisfiable. So in a sense, we may regard B as more incoherent than A. However, we develop an alternative (conflict-centric) characterization here.
We define an ordering between two unsatisfiable concepts based on the *scoring function*.

Definition 7 *Let $\mathcal{T}$ be an incoherent TBox, and A be an unsatisfiable concept name in $\mathcal{T}$ and $MU_A(\mathcal{T})$ be the set of all MUPSs of $\mathcal{T}$ with respect to A. The scoring function for A is a function $S_{\mathcal{T},A} : \wp(\mathcal{T}) \mapsto \mathbb{N}$ ($\wp(\mathcal{T})$ denotes the power set of $\mathcal{T}$) such that for all $\mathcal{T}' \in \wp(\mathcal{T})$*

$$S_{\mathcal{T},A}(\mathcal{T}') = |\{\mathcal{T}_i \in MU_A(\mathcal{T}) : \mathcal{T}_i \cap \mathcal{T}' \neq \emptyset\}|.$$

The scoring function $S_{\mathcal{T},A}$ for A returns for each subset $\mathcal{T}'$ of $\mathcal{T}$ the number of MUPS of $\mathcal{T}$ with respect to A that have overlap with $\mathcal{T}'$. The scoring function is originally defined in [Hun04] to compare two logical inconsistent sets of propositional formulae. It is clear that we have the following proposition.

Proposition 1 *Let $\mathcal{T}$ be an incoherent TBox, and A be an unsatisfiable concept names in $\mathcal{T}$ and $MU_A(\mathcal{T})$ be the set of all MUPSs of $\mathcal{T}$ with respect to A. Suppose $S_{\mathcal{T},A}$ is the scoring function for A, then for all $\mathcal{T}'\in\wp(\mathcal{T})$*

$$S_{\mathcal{T},A}(\mathcal{T}') = |MU_A(\mathcal{T})| - |MU_A(\mathcal{T} \setminus \mathcal{T}')|.$$

According to Proposition 1, the scoring function for $\mathcal{T}'$ gives the number of MUPS that would be eliminated from $\mathcal{T}$ if $\mathcal{T}'$ were substracted from $\mathcal{T}$.

Let $\mathcal{T}$ be an incoherent TBox, and A and B be two unsatisfiable concept names in $\mathcal{T}$. Let $M_A = \cup_{\mathcal{T}_i\in MU_A(\mathcal{T})}\mathcal{T}_i$ and $M_B = \cup_{\mathcal{T}_j\in MU_B(\mathcal{T})}\mathcal{T}_j$. Suppose $|M_A|<|M_B|$, then we add some dummy axioms to M_A such that $|M_A| = |M_B|$. We can define a score ordering as follows:

Definition 8 *Assume that $S_{\mathcal{T},A}$ and $S_{\mathcal{T},B}$ are the scoring functions for A and B respectively. $S_{\mathcal{T},A}\leq_S S_{\mathcal{T},B}$ iff there is a bijection $f : \wp(M_A) \rightarrow \wp(M_B)$ such that the following condition is satisfied:*

$$\forall\mathcal{T}' \in \wp(M_A),\ S_{\mathcal{T},A}(\mathcal{T}')\leq S_{\mathcal{T},B}(f(\mathcal{T}')).$$

As usual, $S_{\mathcal{T},A}<_S S_{\mathcal{T},B}$ denotes $S_{\mathcal{T},A}\leq_S S_{\mathcal{T},B}$ and $S_{\mathcal{T},B}\not\leq_S S_{\mathcal{T},A}$, and $S_{\mathcal{T},A}\simeq_S S_{\mathcal{T},B}$ denotes $S_{\mathcal{T},A}\leq_S S_{\mathcal{T},B}$ and $S_{\mathcal{T},B}\leq_S S_{\mathcal{T},A}$. The score ordering, denoted $\leq$, is defined as: for any two unsatisfiable concepts A and B,

$$A\leq B \ \ iff \ \ S_{\mathcal{T},A}\leq_S S_{\mathcal{T},B}.$$

Intuitively, $S_{\mathcal{T},A}\leq_S S_{\mathcal{T},B}$ means that the MUPSs in $MU_A(\mathcal{T})$ are less overlapping than those in $MU_B(\mathcal{T})$. So A is less incoherent than B with respect to the score ordering iff the MUPSs in A is less overlapping than those in B.

Example 1 *Given a TBox $\mathcal{T} = \{A\sqsubseteq B, A\sqsubseteq C, B\sqsubseteq D,\ C\sqsubseteq\neg D, E\sqsubseteq F, E\sqsubseteq\neg F, F\sqsubseteq D, E\sqsubseteq\neg D\}$, where $A, B, C,\ D, E, F$ are concept names. Clearly, A and E are two unsatisfiable concept names in $\mathcal{T}$, and $MU_A = \{\mathcal{T}_1\}$, where $\mathcal{T}_1 = \{A\sqsubseteq B, A\sqsubseteq C, B\sqsubseteq D, C\sqsubseteq\neg D\}$ and $MU_E = \{\mathcal{T}_2, \mathcal{T}_3\}$, where $\mathcal{T}_2 = \{E\sqsubseteq F, E\sqsubseteq\neg F\}$ and $\mathcal{T}_3 = \{E\sqsubseteq F, F\sqsubseteq D, E\sqsubseteq\neg D\}$. So $M_A = \{A\sqsubseteq B, A\sqsubseteq C, B\sqsubseteq D, C\sqsubseteq\neg D\}$ and $M_E = \{E\sqsubseteq F, E\sqsubseteq\neg F, F\sqsubseteq D, E\sqsubseteq\neg D\}$. Let $S_{\mathcal{T},A}$ and $S_{\mathcal{T},E}$ be the scoring function for A and E respectively, then $S_{\mathcal{T},A}(\mathcal{T}') = 1$, for all $\mathcal{T}' \in \wp(M_A)$. However, $S_{\mathcal{T},E}(\{E\sqsubseteq F\}) = 2$ and $S_{\mathcal{T},E}(\mathcal{T}')\geq 1$ for all other $\mathcal{T}'\in\wp(M_E)$. So $S_{\mathcal{T},A}<_S S_{\mathcal{T},E}$ and we have $A<E$.*

When defining the score ordering, we need to find a bijection f mapping every subset of M_A to a subset of M_B. In the following, we provide a procedure to find the bijection f. Let $|\wp(M_A)| = |\wp(M_B)| = n$.

Step 1: for each $\mathcal{T}_i\in\wp(M_A)$ and each $\mathcal{T}'_j\in\wp(M_B)$, compute $S_{\mathcal{T},A}(\mathcal{T}_i)$ and $S_{\mathcal{T},B}(\mathcal{T}'_j)$,

Step 2: rearrange $\mathcal{T}_1$,...,$\mathcal{T}_n$ as $\mathcal{T}_{i_1}$,...,$\mathcal{T}_{i_n}$ ($i_k\in\{1, ..., n\}$) such that $S_{\mathcal{T},A}(\mathcal{T}_{i_1})\geq S_{\mathcal{T},A}(\mathcal{T}_{i_2})\geq ...\geq S_{\mathcal{T},A}(\mathcal{T}_{i_n})$, and rearrange $\mathcal{T}'_1$,...,$\mathcal{T}'_n$ as $\mathcal{T}'_{j_1}$,...,$\mathcal{T}'_{j_n}$ ($j_k\in\{1, ..., n\}$) such that $S_{\mathcal{T},B}(\mathcal{T}'_{j_1})\geq S_{\mathcal{T},B}(\mathcal{T}'_{j_2})\geq ...\geq S_{\mathcal{T},B}(\mathcal{T}'_{j_n})$,

Step 3: a mapping $f_S : \wp(M_A) \rightarrow \wp(M_B)$ is defined as follows: for each $\mathcal{T}_{i_k}\in\wp(M_A)$, $f_S(\mathcal{T}_{i_k}) = \mathcal{T}'_{j_k}$.

It is clear that f_S is a bijection. We have the following proposition.

Proposition 2 *Assume that $S_{\mathcal{T},A}$ and $S_{\mathcal{T},B}$ are the scoring functions for A and B respectively. Then $S_{\mathcal{T},A} \leq_S S_{\mathcal{T},B}$ iff*

$$\forall \mathcal{T}' \in \wp(M_A),\ S_{\mathcal{T},A}(\mathcal{T}') \leq S_{\mathcal{T},B}(f_S(\mathcal{T}')).$$

Proof sketch: *"If" part is clear by the definition of score ordering. We show "only if" part.*
Suppose $S_{\mathcal{T},A} \leq_S S_{\mathcal{T},B}$, then there exists a bijection f such that for all $\mathcal{T}' \in \wp(M_A)$, $S_{\mathcal{T},A}(\mathcal{T}') \leq S_{\mathcal{T},B}(f(\mathcal{T}'))$. We shown that $S_{\mathcal{T},A}(\mathcal{T}_{i_k}) \leq S_{\mathcal{T},B}(\mathcal{T}'_{j_k})$ for all $k = 1, ..., n$ by induction over the index k.
Suppose $k = 1$. Then $S_{\mathcal{T},A}(\mathcal{T}_{i_1}) \leq S_{\mathcal{T},B}(f(\mathcal{T}_{i_1})) \leq S_{\mathcal{T},B}(\mathcal{T}'_{j_1})$.
Assume that $S_{\mathcal{T},A}(\mathcal{T}_{i_k}) \leq S_{\mathcal{T},B}(\mathcal{T}'_{j_k})$ for all $k < m$. Suppose that $S_{\mathcal{T},A}(\mathcal{T}_{i_m}) > S_{\mathcal{T},B}(\mathcal{T}'_{j_m})$. Then $S_{\mathcal{T},B}(\mathcal{T}'_{j_m}) < S_{\mathcal{T},B}(f(\mathcal{T}_{i_m}))$. This means that there exists $j_l < j_m$ such that $f(\mathcal{T}_{i_m}) = \mathcal{T}'_{j_l}$. However, since $S_{\mathcal{T},A}(\mathcal{T}_{i_m}) > S_{\mathcal{T},B}(\mathcal{T}'_{j_m})$, we have that $S_{\mathcal{T},A}(\mathcal{T}_{i_k}) > S_{\mathcal{T},B}(\mathcal{T}'_{j_m})$ for all $k < m$. Therefore, for any $k < m$, there exists $k' < m$ such that $f(\mathcal{T}_{i_k}) = \mathcal{T}'_{j_{k'}}$. Therefore, it is impossible that there exists $j_l < j_m$ such that $f(\mathcal{T}_{i_m}) = \mathcal{T}'_{j_l}$ (every such $\mathcal{T}'_{j_l}$ has corresponds to a $\mathcal{T}_{i_k}$ with $k < m$). This is a contradiction. So $S_{\mathcal{T},A}(\mathcal{T}_{i_m}) \leq S_{\mathcal{T},B}(\mathcal{T}'_{j_m})$.

We have the following proposition.

Proposition 3 *Let $\mathcal{T}$ be an incoherent TBox, and let A and B be two unsatisfiable concept names in it. If $A \sqsubseteq B \in \mathcal{T}$, then $B \leq_S A$.*
Proof sketch: *We add some dummy axioms to M_B to make $|M'_B| = |M_A|$. We can define a bijection function f as follows: for each $\phi \in M_B$, $f(\phi) = \phi$; for each $\phi \in M'_B \backslash M_B$, we map it to an arbitrary axiom in $M_A \backslash M_B$ and make sure that any two axioms are mapped to different axioms in $M_A \backslash M_B$; for any subset $\mathcal{T}'$ of M'_B, $f(\mathcal{T}') = \{f(\phi) : \phi \in \mathcal{T}'\}$. It is easy to check that for each subset $\mathcal{T}' \in \wp(M'_B)$, $S_{\mathcal{T},B}(\mathcal{T}') \leq S_{\mathcal{T},A}(f(\mathcal{T}'))$.*

Proposition 3 tells us that if A is subsumed by B then A is more incoherent than B with respect to the score ordering. If we consider the example in the beginning of this section, B is less coherent than A with respect to the score ordering. Therefore, our score ordering provides a different view on the extent of incoherence of a concept from the subsumption relation. Indeed, scoring ordering gives a conflict-centric view since the formulae involved in the conflict for concept B are a subset of those for concept A. Proposition 3 also provides us a way to improve the performance of our approach for generating the score ordering. That is, before comparing the score functions of two unsatisfiable concept, we can first check if they have subsumption relation in the ontology.

5.2.2 Measures of incoherence for terminologies

Given a TBox which may be incoherent, we propose some approaches to measuring its degree of incoherence.
The first measure is defined by the ratio of number of unsatisfiable concepts and that of all the concepts in $\mathcal{T}$.

Definition 9 *Let $\mathcal{T}$ be a TBox. Suppose $Con(\mathcal{T})$ is the set of all concept names and $US(\mathcal{T})$ be the set of all unsatisfiable concept names in $\mathcal{T}$ respectively, the unsatisfiability ratio for $\mathcal{T}$, denoted d_{UR}, is defined as follows:*

$$d_{UR}(\mathcal{T}) = \frac{|US(\mathcal{T})|}{|Con(\mathcal{T})|}.$$

The unsatisfiability ratio gives us a simple view on the incoherence of a TBox. That is, if most of the concept names are unsatisfiable in a TBox, this TBox is problematic. However, the unsatisfiability ration is misleading in some cases. For example, in an ontology such as Tamblis[1] where there are large number of unsatisfiable

[1] http://protege.cim3.net/file/pub/ontologies/tambis/tambis-full.owl.

concept name, many of the unsatisfiable concept names depend on other unsatisfiable concept names. The root unsatisfiable concept names is few (in Tamblis, 33 concepts names out of 144 unsatisfiable concept names are root unsatisfiable names) and by repairing these concept names we can get a coherent ontology. Therefore, this ontology is not "strongly" incoherent. To overcome the problem for the unsatisfiability ratio, we can only consider the root unsatisfiable concept names.

We have the following definition.

Definition 10 *Let $\mathcal{T}$ be a TBox. Suppose $Con(\mathcal{T})$ is the set of all concept names and $RU(\mathcal{T})$ be the set of all root unsatisfiable concept names in $\mathcal{T}$ respectively, the refined unsatisfiability ratio for $\mathcal{T}$, denoted d_{RU}, is defined as follows:*

$$d_{RU}(\mathcal{T}) = \frac{|RU(\mathcal{T})|}{|Con(\mathcal{T})|}.$$

Both the unsatisfiability ratio and the refined unsatisfiability ratio do not consider the amount of terminology axioms which are in conflict. We define another incoherence measure for TBoxes.

Definition 11 *Let $\mathcal{T}$ be a TBox. Suppose $MI(\mathcal{T})$ is the set of all MIPSs of $\mathcal{T}$, then the incoherence ratio for $\mathcal{T}$, denoted d_{IR}, is defined as follows:*

$$d_{IR}(\mathcal{T}) = \frac{|\cup_{\mathcal{T}_i \in MI(\mathcal{T})} \mathcal{T}_i|}{|\mathcal{T}|}.$$

The incoherence ratio measures the percentage of axioms in a TBox which are in conflict. It differentiates the root unsatisfiable concept names and derived unsatisfiable concept names. This is because any axiom whose left hand is a derived unsatisfiable concept name is not in an MIPS.

Example 2 *Let $\mathcal{T} = \{A \sqsubseteq B, A \sqsubseteq \neg B, C \sqsubseteq A\}$ and $\mathcal{T}' = \{A \sqsubseteq B, A \sqsubseteq \neg B, C \sqsubseteq \bot\}$. Then $US(\mathcal{T}) = US(\mathcal{T}') = \{A, C\}$ and $d_{UR}(\mathcal{T}') = d_{UR}(\mathcal{T}') = \frac{2}{3}$: T and T' have the same unsatisfiability ratio. However, $MI(\mathcal{T}) = \{\{A \sqsubseteq B, A \sqsubseteq \neg B\}\}$ and $MI(\mathcal{T}') = \{T'\}$. So $d_{IR}(\mathcal{T}) = \frac{2}{3}$ and $d_{IR}(\mathcal{T}') = 1$.*

The problem for the incoherence ratio is that it says nothing about to which extent the MIPSs in $MI(\mathcal{T})$ overlap.

Example 3 *Let $\mathcal{T} = \{A \sqsubseteq B, A \sqsubseteq \neg B, C \sqsubseteq D, C \sqsubseteq \neg D\}$ and $\mathcal{T}' = \{A \sqsubseteq B, A \sqsubseteq \neg B, B \sqsubseteq C, A \sqsubseteq \neg C\}$ be two coherent TBoxes, where A, B, C, D are concept names. By Definition 6, $\mathcal{T}$ has two MIPSs $\{A \sqsubseteq B, A \sqsubseteq \neg B\}$ and $\{C \sqsubseteq D, C \sqsubseteq \neg D\}$, and $\mathcal{T}'$ has two MIPSs $\{A \sqsubseteq B, A \sqsubseteq \neg B\}$ and $\{A \sqsubseteq B, B \sqsubseteq C, A \sqsubseteq \neg C\}$. According to Definition 11, we have $d_{IR}(\mathcal{T}) = d_{IR}(\mathcal{T}') = 1$. However, MIPSs in $MI(\mathcal{T})$ have no overlap whilst the MIPSs in $MI(\mathcal{T}')$ have a common axiom $A \sqsubseteq B$. Therefore, we may conclude that $\mathcal{T}$ is less coherent than $\mathcal{T}'$.*

We have defined two measures and argue that they are not fine grained enough. Next, we define an incoherence measure for TBoxes which are based on the scoring functions.

Definition 12 *Let $\mathcal{T}$ be a TBox. The scoring function for $\mathcal{T}$, is a function $S_{\mathcal{T}} : \wp(\mathcal{T}) \mapsto N$ such that for all $\mathcal{T}' \in \wp(\mathcal{T})$*

$$S_{\mathcal{T}}(\mathcal{T}') = |\{\mathcal{T}_i \in MI(\mathcal{T}) : \mathcal{T}_i \cap \mathcal{T}' \neq \emptyset\}|.$$

The scoring function $S_{\mathcal{T}}$ for $\mathcal{T}$ returns for each subset $\mathcal{T}'$ of $\mathcal{T}$ the number of MIPS of $\mathcal{T}$ that have overlap with $\mathcal{T}'$.

We have the following proposition for the scoring function.

Proposition 4 *Let $S_{\mathcal{T}}$ be the scoring function for $\mathcal{T}$. For $\mathcal{T}_i, \mathcal{T}_j \in \wp(\mathcal{T})$, we have*

$$S_{\mathcal{T}}(\mathcal{T}_i \cap \mathcal{T}_j) \leq min(S_{\mathcal{T}}(\mathcal{T}_i), S_{\mathcal{T}}(\mathcal{T}_j))$$

$$max(S_{\mathcal{T}}(\mathcal{T}_i), S_{\mathcal{T}}(\mathcal{T}_j)) \leq S_{\mathcal{T}}(\mathcal{T}_i \cup \mathcal{T}_j).$$

The scoring function can be used to define an ordering between two terminology axioms.

Definition 13 *Let $\mathcal{T}$ be a TBox and $S_{\mathcal{T}}$ be its scoring function. A score-based ordering on terminology axioms in $\mathcal{T}$, denoted $\prec_{S_{\mathcal{T}}}$, is defined as follows: for any $\phi, \psi \in \mathcal{T}$,*

$$\phi \preceq_{S_{\mathcal{T}}} \psi \;\; iff \;\; S_{\mathcal{T}}(\{\phi\}) \leq S_{\mathcal{T}}(\{\psi\}).$$

As usual, $\phi \prec_{S_{\mathcal{T}}} \psi$ denotes $\phi \preceq_{S_{\mathcal{T}}} \psi$ and $\psi \not\preceq_{S_{\mathcal{T}}} \phi$. $\phi \preceq_{S_{\mathcal{T}}} \psi$ means that ϕ is less coherent than ψ with respect to the scoring function. That is, ϕ is contained in less MIPSs of $\mathcal{T}$ than ψ. It is clear that $\preceq_{S_{\mathcal{T}}}$ is a total pre-order, i.e. a pre-order which is transitive.

Example 4 *(Example 1 Continued) There are three MIPSs of $\mathcal{T}$: $\mathcal{T}_1 = \{A \sqsubseteq B, A \sqsubseteq C, B \sqsubseteq D, C \sqsubseteq \neg D\}$, $\mathcal{T}_2 = \{E \sqsubseteq F, E \sqsubseteq \neg F\}$ and $\mathcal{T}_3 = \{E \sqsubseteq F, F \sqsubseteq D, E \sqsubseteq \neg D\}$. So $S_{\mathcal{T}}(\{E \sqsubseteq F\}) = 2$ and $S_{\mathcal{T}}(\{\phi\}) = 1$ for all other $\phi \in \mathcal{T}$. Therefore, $E \sqsubseteq F \prec_{S_{\mathcal{T}}} \phi$ for all $\phi \in \mathcal{T}$ and ϕ is not $E \sqsubseteq F$.*

Let $\mathcal{T}$ and $\mathcal{T}'$ be two TBox. Let $M_{\mathcal{T}} = \cup_{\mathcal{T}_i \in MI(\mathcal{T})} \mathcal{T}_i$ and $M_{\mathcal{T}'} = \cup_{\mathcal{T}_j \in MI(\mathcal{T})'} \mathcal{T}_j$. Suppose $|M_{\mathcal{T}}| < |M_{\mathcal{T}'}|$, then we add some dummy axioms to $M_{\mathcal{T}}$ such that $|M_{\mathcal{T}}| = |M_{\mathcal{T}'}|$. An ordering on TBoxes can be defined by the scoring functions as follows.

Definition 14 *Assume that $S_{\mathcal{T}}$ and $S_{\mathcal{T}'}$ are the scoring functions for two TBoxes $\mathcal{T}$ and $\mathcal{T}'$ respectively. $S_{\mathcal{T}} \preceq S_{\mathcal{T}'}$ iff there is a bijection $f : \wp(M_{\mathcal{T}}) \rightarrow \wp(M_{\mathcal{T}'})$ such that the following condition is satisfied:*

$$\forall \mathcal{T}' \in \wp(M_1),\; S_{\mathcal{T}}(\mathcal{T}') \leq S_{\mathcal{T}'}(f(\mathcal{T}')).$$

As usual, $S_{\mathcal{T}} prec_S S_{\mathcal{T}'}$ denotes $S_{\mathcal{T}} \preceq_S S_{\mathcal{T}'}$ and $S_{\mathcal{T}'} \not\preceq_S S_{\mathcal{T}}$, and $S_{\mathcal{T}} \equiv_S S_{\mathcal{T}'}$ denotes $S_{\mathcal{T}} \preceq_S S_{\mathcal{T}'}$ and $S_{\mathcal{T}'} \preceq_S S_{\mathcal{T}}$. The score ordering, denoted $\preceq_S$, is defined as: for any two TBoxes $\mathcal{T}$ and $\mathcal{T}'$,

$$\mathcal{T} \preceq_S \mathcal{T}' \;\; iff \;\; S_{\mathcal{T}} \preceq_S S_{\mathcal{T}'}.$$

Intuitively, $S_{\mathcal{T}} \preceq S_{\mathcal{T}'}$ means that the MIPSs of $\mathcal{T}$ are less overlapping than those of $\mathcal{T}'$. So $\mathcal{T}$ is less coherent than $\mathcal{T}'$ with respect to the score ordering iff the MIPSs of $\mathcal{T}$ is less overlapping than those of $\mathcal{T}'$.

Example 5 *Given two TBoxes $\mathcal{T} = \{A \sqsubseteq B \sqcap C, B \sqcap C \sqsubseteq \bot\}$ and $\mathcal{T}' = \{A \sqsubseteq B, A \sqsubseteq C, B \sqcap C \sqsubseteq \bot\}$, it is clear that $\mathcal{T} \equiv \mathcal{T}'$. $\mathcal{T}$ has only one MIPS which is $\mathcal{T}$ and $\mathcal{T}'$ has only one MIPS which is $\mathcal{T}'$. So $M_{\mathcal{T}} = \mathcal{T}$ and $M_{\mathcal{T}'} = \mathcal{T}'$. Since $|M_{\mathcal{T}}| < |M_{\mathcal{T}'}|$, we add a dummy axiom to $M_{\mathcal{T}}$ such that $M_{\mathcal{T}} = \{A \sqsubseteq B \sqcap C, B \sqcap C \sqsubseteq \bot, D \sqsubseteq \top\}$, where D is a new concept name. Let $S_{\mathcal{T}}$ and $S_{\mathcal{T}'}$ be the scoring functions for $\mathcal{T}$ and $\mathcal{T}'$ respectively, we then have*
$S_{\mathcal{T}}(\{A \sqsubseteq B \sqcap C\}) = 1$, $S_{\mathcal{T}}(\{B \sqcap C \sqsubseteq \bot\}) = 1$,
$S_{\mathcal{T}}(\{D \sqsubseteq \top\}) = 0$, *and*
$S_{\mathcal{T}'}(\{A \sqsubseteq B\}) = 1$, $S_{\mathcal{T}'}(\{A \sqsubseteq C\}) = 1$,
$S_{\mathcal{T}'}(\{B \sqcap C \sqsubseteq \bot\}) = 1$.
So $S_{\mathcal{T}} \prec S_{\mathcal{T}'}$ and $\mathcal{T} \prec_S \mathcal{T}'$.

According to Example 5, the scoring function defined by Definition 12 is syntax sensitive in the sense that there may exist two TBoxes $\mathcal{T}$ and $\mathcal{T}'$ where $\mathcal{T} \equiv \mathcal{T}'$ and $S_{\mathcal{T}}$ is the scoring function for $\mathcal{T}$ and $S_{\mathcal{T}'}$ is the scoring function for $\mathcal{T}'$, but $S_{\mathcal{T}} \not\equiv S_{\mathcal{T}'}$. To give a more precise measure of incoherence, we can simply split the axioms in a TBox $\mathcal{T}$ into "smaller" axioms to obtain an equivalent TBox $\mathcal{T}_s$ using the algorithm in [KPSG06]. The score ordering $\preceq_S$ is related to the incoherence ratio.

Proposition 5 *Let $\mathcal{T}$ and $\mathcal{T}'$ be two TBoxes, and $|\mathcal{T}| = |\mathcal{T}'|$. Suppose $S_{\mathcal{T}}$ and $S_{\mathcal{T}'}$ are scoring function for $\mathcal{T}$ and $\mathcal{T}'$ respectively, then*
$S_{\mathcal{T}} \preceq_S S_{\mathcal{T}'}$ *implies* $d_{IR}(\mathcal{T}) \leq d_{IR}(\mathcal{T}')$.
The converse does not hold.

Proposition 5 tells us that a score ordering takes into account the incoherence ratio. That is, if two TBoxes have the same cardinality, then a TBox $\mathcal{T}$ is less coherent than another one $\mathcal{T}'$ implies that $\mathcal{T}$ contains fewer conflicting terminology axioms.

5.3 Summary

In this chapter we have presented measures of incoherence in an incoherent ontology. These measures can provide important context information for dealing with imperfect information in an ontology. The approaches provided in the chapter have been used to illustrate how to obtain some kind of provenance context automatically.

Chapter 6

Conclusion

6.1 Summary

In this deliverable we have discussed formalisms for context representation in the NeOn project. First, we have recalled the general definition of context given in the NeOn deliverable D3.1.1. We have then instantiated the general definition by describing *provenance* and *argumentation*. This context information can be applied to deal with inconsistency and incoherence in ontologies. After that, we proposed a number of different possible groundings of the metamodel in the OWL DL language. Finally, we have given some approaches to measuring incoherence in an OWL ontology. Two kinds of measures were considered: measures of incoherence for unsatisfiable concepts and measures of incoherence for terminologies. These measures can be used to obtain ranking information automatically from an incoherent ontology and so provide important context information for maintaining and evaluating ontologies.

6.2 Roadmap

While in this deliverable we have laid the foundations for the representation of context, the next steps are the realization of solutions for obtaining, reasoning with and exploiting context. With respect to obtaining context, we will extend existing tools for the construction of ontologies with capabilities to generate context information along with the ontologies. Specific tools include: ontology learning tools (as part of task *T3.8 Context for learning networked ontologies*) and the argumentation support tools developed (as part of WP2, task *T2.3 Methods for collaborative construction, annotation, and argumentation of ontologies*).

Further, we develop approaches for reasoning with context. In the deliverable *D3.2.3 Context reasoning with imperfect information* we will provide specific methods for dealing with uncertain and inconsistent information in automatically generated or collaboratively engineered ontologies.

Other aspects of exploiting context information will include visualizing contexts as part of task *T3.7 context-sensitive visualization of networked ontologies* in collaboration with WP4.

Bibliography

[AM97] P. Smets A. Motro. *Uncertainty Management In Information Systems*. Springer, 1997.

[BCRS06] P. Buitelaar, P. Cimiano, S. Racioppa, and M. Siegel. Ontology-based information extraction with soba. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006.

[BDP92] Salem Benferhat, Didier Dubois, and Henri Prade. Representing default rules in possibilistic logic. In *KR*, pages 673–684, 1992.

[BLP04] S. Benferhat, S. Lagrue, and O. Papini. Reasoning with partially ordered information in a possibilistic logic framework. *Fuzzy Sets and Systems*, 144(1):25–41, 2004.

[BOS03] P. Buitelaar, D. Olejnik, and M. Sintek. OntoLT: A protégé plug-in for ontology extraction from text. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2003.

[CGL+06] Carola Catenacci, Aldo Gangemi, Jos Lehmann, Malvina Nissim, and Valentina Presutti. Design rationales for collaborative development of networked ontologies - state of the art and the collaborative ontology design ontology. Technical report, CNR; NeOn Deliverable D2.1.1, 2006.

[Cir01] F. Ciravegna. Adaptive information extraction from text by rule induction and generalization. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1251–1256, 2001.

[CV05] Philipp Cimiano and Johanna VËĘlker. Text2onto - a framework for ontology learning and data-driven change discovery. In Andres Montoyo, Rafael Munoz, and Elisabeth Metais, editors, *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 3513 of *Lecture Notes in Computer Science*, pages 227–238, Alicante, Spain, JUN 2005. Springer.

[DLP94] Didier Dubois, Jérôme Lang, and Henri Prade. Possibilistic logic. In *Handbook of logic in Aritificial Intelligence and Logic Programming*, pages 439–513, 1994.

[DP91] Didier Dubois and Henri Prade. Epistemic entrenchment and possibilistic logic. *Artif. Intell.*, 50(2):223–239, 1991.

[DP98] Didier Dubois and Henri Prade. Possibility theory: qualitative and quantitative aspects. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 169–226, 1998.

[DP04] Z. Ding and Y. Peng. A Probabilistic Extension to Ontology Language OWL. In *Proceedings of the 37th Hawaii International Conference On System Sciences (HICSS-37).*, Big Island, Hawaii, January 2004.

[DS05] Michael Dürig and Thomas Studer. Probabilistic abox reasoning: Preliminary results. In *Description Logics*, 2005.

[Fel98] C. Fellbaum. *WordNet, an electronic lexical database*. MIT Press, 1998.

[FHP+06] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *Proc. of AAAI'06*, 2006.

[FK00] F. Freitag and N. Kushmerick. Boosted Wrapper Induction. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, pages 577–583, 2000.

[FN98] D. Faure and C. Nedellec. A corpus-based conceptual clustering method for verb frames and ontology. In *Proceedings of the LREC Workshop on Adapting lexical and corpus resources to sublanguages and applications*, 1998.

[GL02] Rosalba Giugno and Thomas Lukasiewicz. P-shoq(d): A probabilistic extension of shoq(d) for probabilistic ontologies in the semantic web. In *JELIA*, pages 86–97, 2002.

[Hea92] M.A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[Hei94] Jochen Heinsohn. Probabilistic description logics. In *UAI'94*, pages 311–318, 1994.

[HHR+06] Peter Haase, Pascal Hitzler, Sebastian Rudolph, Guilin Qi, Marko Grobelnik, Igor Mozetič, Damjan Bojadžiev, Jérôme Euzenat, Mathieu d'Aquin, Aldo Gangemi, and Carola Catenacci. D3.1.1 context languages - state of the art. Technical Report D3.1.1, Universität Karlsruhe, August 2006.

[HRW+06] Peter Haase, Sebastian Rudolph, Yimin Wang, Saartje Brockmans, Raul Palma, Jéróme Euzenat, and Mathieu d'Aquin. D1.1.1 networked ontology model. Technical Report D1.1.1, Universität Karlsruhe, NOV 2006.

[Hun04] Anthony Hunter. Logical comparison of inconsistent perspectives using scoring functions. *Knowl. Inf. Syst.*, 6(5):528–543, 2004.

[HV05] Peter Haase and Johanna Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In Paulo C. G. da Costa, Kathryn B. Laskey, Kenneth J. Laskey, and Michael Pool, editors, *Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, pages 45–55, NOV 2005.

[Jae94] Manfred Jaeger. Probabilistic reasoning in terminological logics. In *KR'94*, pages 305–316, 1994.

[KLP97] Daphne Koller, Alon Y. Levy, and Avi Pfeffer. P-classic: A tractable probablistic description logic. In *Proc. of AAAI'97*, pages 390–397, 1997.

[KPSG06] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. Repairing unsatisfiable concepts in owl ontologies. In *ESWC'06*, pages 170–184, 2006.

[MS05] Boris Motik and Ulrike Sattler. Practical dl reasoning over large aboxes with kaon2. http://kaon2.semanticweb.org/#documentation, 2005.

[NF04] Henrik Nottelmann and Norbert Fuhr. pdaml+oil: A probabilistic extension to daml+oil based on probabilistic datalog. In *Proceedings Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 227–234, 2004.

[NVCN04] R. Navigli, P. Velardi, A. Cucchiarelli, and F. Neri. Extending and enriching WordNet with OntoLearn. In *Proc. of the GWC 2004*, pages 279–284, 2004.

[POT70] Chaim Perelman and Lucie Olbrechts-Tyteca. *TraitÂőÂű de l'argumentation: La nouvelle rhÂőÂűtorique*. Paris: Presses Universitaires de France, 1970.

[PST04] Helena Sofia Pinto, Steffen Staab, and Christoph Tempich. Diligent: Towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In *Proc. of ECAI'04*, pages 393–397, 2004.

[QHJ07] Guilin Qi, Peter Haase, and Qiu Ji. D1.2.1 consistency models for networked ontologies. Technical Report D1.2.1, Universität Karlsruhe, FEB 2007.

[SC03] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI'03*, pages 355–362, 2003.

[Sch05] Stefan Schlobach. Diagnosing terminologies. In *Proc. of AAAI'05*, pages 670–675, 2005.

[Tem06] Christoph Tempich, editor. *Ontology Engineering and Routing in Distributed Knowledge Management Applications*. 2006.

[Tou58] S. Toulmin. *The Uses of Argument*. Cambridge University Press, Cambridge, UK, 1958.

[TPSS05] Christoph Tempich, Helena Sofia Pinto, York Sure, and Steffen Staab. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (diligent). In *Proc. of ESWC'06*, pages 241–256, 2005.

[VVH+06] Denny Vrandecic, Johanna Völker, Peter Haase, Duc Thanh Tran, and Philipp Cimiano. A metamodel for annotations of ontology elements in owl dl. In York Sure, Saartje Brockmans, and Jürgen Jung, editors, *Proceedings of the 2nd Workshop on Ontologies and Meta-Modeling*, Karlsruhe, Germany, OCT 2006. GI Gesellschaft für Informatik.