NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

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

D4.4.3 Presentation of prototypes for access control in the NeOn infrastructure

Deliverable Co-ordinator: **Noam Bercovici**

Deliverable Co-ordinating Institution: **Universität Koblenz-Landau (UKO-LD)**

Other Authors: **Simon Schenk (UKO-LD)**

In this deliverable we operationalize the motivation and the functionality of the approach proposed in the previous deliverables in the context of granting access rights in the area of networked ontologies. The deliverable accompanies two software outputs: (i) a proof of concept combining the ontology customization plugin and the previous API described in the deliverable D4.4.2. (ii) a prototype using a meta knowledge reasoner to grant access.

Document Identifier:	NEON/2010/D4.4.3/v1.0	Date due:	February 28, 2010
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	January 29, 2010
Project start date	March 1, 2006	Version:	v1.0
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

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 to the writing of this document or its parts:

- Universität Koblenz Landau
- Open University

Change Log

Version	Date	Amended by	Changes
0.1	04-01-2010	Noam Bercovici	Initial setting of this deliverable
0.2	11-01-2010	Noam Bercovici	Add the motivation part
0.3	20-01-2010	Noam Bercovici	Add the chapter about authorization views
0.4	25-01-2010	Simon Schenk	Add the chapter 3
0.5	01-02-2010	Simon Schenk & Noam Bercovici	address review comments

Executive Summary

We will present in this deliverable two methods to have an access control mechnism in the NeOn Infrastructure. The first access control method is following up on what was proposed in the previous work, the functional features of the access management framework designed for the NeOn Infrastructure in the form of ontology view composer plus API. The ontology view composer is an extension of the ontology customization plugin for helping the user to define the authorization views. The second method proposes an elegant solution to ensure access rights at the runtime. It offers fine grained control on the axiom level and dynamically controls access to inference results.

Contents

List of Figures

Chapter 1

Introduction

In deliverable D4.4.1 [DKG+07] we discussed the role of access rights and access control mechanisms in the context of working with networked ontologies. While this previous deliverable focused on the clarification of terms, on the review and analysis of various access control models, and on the proposal for a new model applicable to distributed ontologies, it was left as a conceptual foundation for further work. In fact, one of its explicit objectives was "to serve as a potential foundation for a further, more formal development of access rights as a part of NeOn model and/or ontology meta-data vocabulary [... to investigate the impact of] access control on such aspects of networked ontologies as modules, collaboration workflows and processes, [...] and possibly others [...]" In the previous deliverable [Dzb09] we described and exemplified the functional features of the access management framework designed for the NeOn Infrastructure in the form of API. The API is broadly based on the existing Watson API, which in turn enables the user to query a remote ontology repository for entire ontologies, selected ontological resources and the respective meta-data. Mendelzon et al. in [RMSR04] had a proposal for database access control which consists of defining "authorization views" that specify the accessible data. We decide to adapt partially this idea for ontologies. The capability to set access rights to a part of an ontology can be seen as an instance of ontology customization. That is, a larger ontology is "partitioned" based on different criterion (meta-knowledge or/and data itself) for assigning access to certain concepts to different users. Consequently, access right definition user interface depends, at least to some extent, on the functional ontology customization plug-in. Since the work on ontology customization has been delayed, we were not able to show the full picture of our approach about granting the access to part of the ontology. In this deliverable we are able to present this full picture. We start this deliverable by reviewing a sample of conceptual motivations and situations where access control may play some role. Its main purpose is to match the generic report D4.4.1 [DKG+07] with the implementation work carried out in WP4since that deliverable.

We present two approaches for managing access rights for ontologies. First, we describe our implementation based on ontology views. This approach is analogous to access control in relational databases, where access rights also implicitly define a user specific view on the database. This approach offers reasonably fine grained access control and fast access. A disadvantage of this approach in the context of reasoning is that it only allows to limit the fragment of an ontology or ontology network used as input for reasoning. Second, we discuss access control based on meta knowledge, that is, annotations of axioms with access levels and reasoning over these annotations. This approach allows for an extremely fine grained control on the axiom level. Moreover, the ontology is not restricted a priori, but access rights can also be computed for the results of inferences, instead of only for explicit axioms. This approach, however, is based on meta knowledge reasoning as described in deliverables [QSJ09, QS09]. Hence for each subsumption check with access control, a potentially large number of subsumption checks in the underlying DL needs to be performed. Hence this approach is rather slow.

In early phases of the project, we analyzed user requirements coming from the two use cases - the pharmaceutical and agricultural domains. In the wide array of requirements, we would like to flag the following ones as directly motivating and bootstrapping the work relevant to this deliverable:

1. For the semantic nomenclature use case, D8.1.1 sees the aspect of managing access control in terms of recognizing the need of the original owners of information in the data repositories (e.g., BOTPlus) to consider some, usually detailed, information about drugs, their tests, approvals, etc. “private” - that is, available to different degree of granularity to different users. There is no explicit discussion of the need to carry out different actions on the access-controlled semantic content, the deliverable mainly discusses the “use of data” that needs to be managed for the purposes of sharing semantic nomenclature content. This requirement is described as a potential use case later in this introduction.

2. Section 5.4.5 in D7.1.1 discusses the need to tailor the user interface to different types of user (e.g., experts, editors, etc.). In addition to this customization aspect it also proposes to recognize the adaptation based on the expert’s and/or editor’s user rights that is, access to the authorized ontologies for the purposes of editing, mapping, adding label in another language, etc.

3. D8.1.1 follows on and requires a framework to deal with access rights for the users of their pharmaceutical information systems. It envisions a scenario in which ontologies are reused by other actors but some parts of the original ontology or of the data must be not public. E.g., pharmacists generate statistical information about product sales, epidemics, and diseases that could be accessed by the professional associations, but laboratories must not access to this kind of information, so access rights must be preserved.

To demonstrate the access rights plugin, we adopt a use case from the pharmaceutical industry. The pharmaceutical industry is an important element of medical assistance systems around the world; this sector is constituted by numerous public and private organizations dedicated to the research, development, manufacture and commercialization of medicines for both human and animal health. In the semantic nomenclature case study, the information of the pharmaceutical products have different providers and recipients: laboratories, government entities (Ministry of Health, Agemed, CCAA RegionsE), pharmacists associations (GSCoP), etc. Aspects like access control are of great importance for the Semantic Nomenclature, due to the fact that the case study is about aggregation of multiple and heterogeneous sources of information and spray around this information. Therefore, the BOTPlus Ontology that represents the knowledge of this source of information should have public and private parts to allow the Nomenclature’s users which have the correspondent and adequate credentials to access the private part and to obtain more information about the pharmaceutical products. The figure 1.1 describes a common situation meet by the user of the Semantic Nomenclature. The Semantic Nomenclature is called to become a standard for describing drugs, then it would be the natural choice for sharing information. The “research unit” of a pharmaceutical industry is collaborating on a specific project about the season flu 2010 with the “epidemiological department” of a public research group, they are using the Semantic Nomenclature as knowledge base. Nevertheless each of them is working on different projects related or not to this one. Let’s assume that Bob from the pharmaceutical industry is also working on the different medicines for other flu like the H1N1 or the H5N1. He may want to keep private the part of the ontology related to those viruses. In the case illustrated in figure 1.1, we need to deal with two different views; what Alice can access is represented in the figure by the yellow rectangle which includes the public ontology ABox + TBox (classes and instances) and the private part. This view excludes the private ontology part of their industrial partner. Unlike Bob’s access shown on the figure via the green rectangle, he can access the private part of the Semantic Nomenclature from his company plus the public part but not the restricted area of his partner.

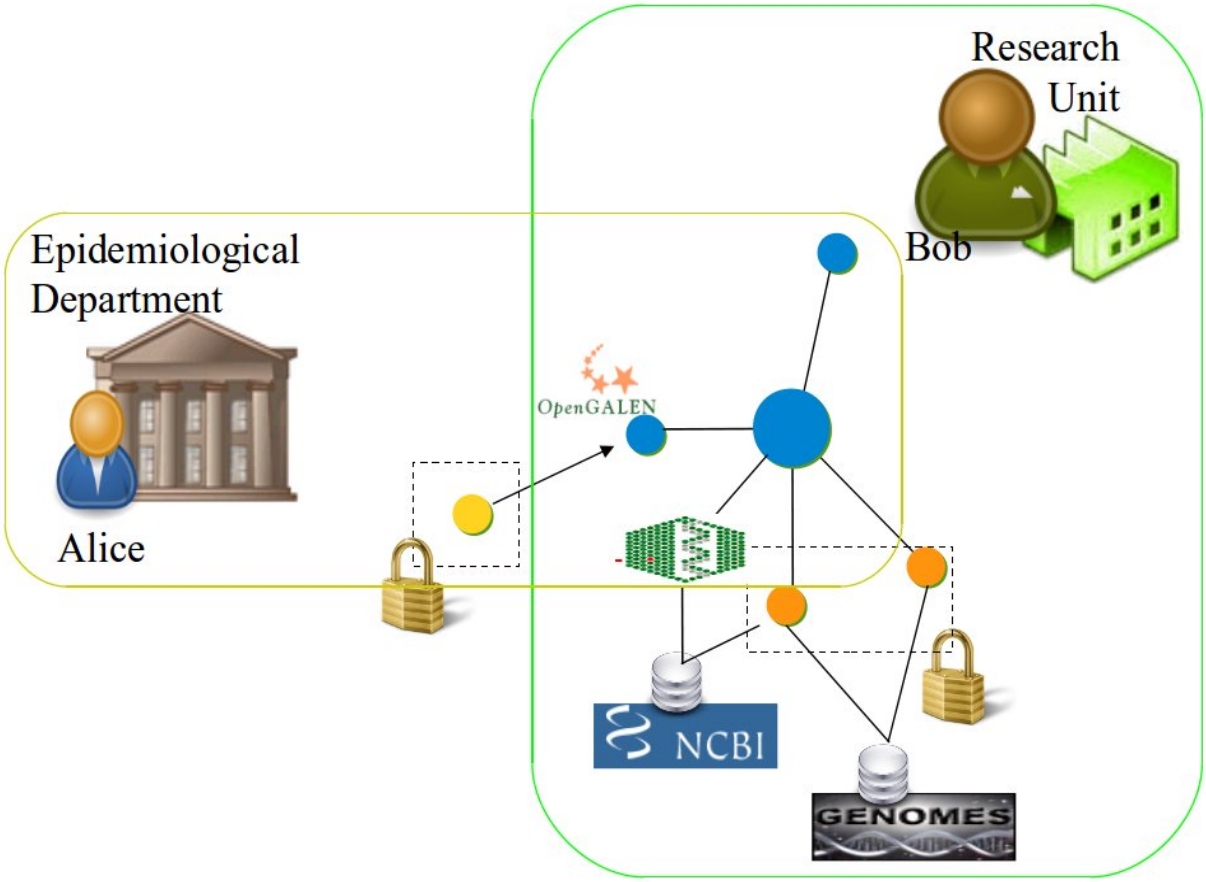

Figure 1.1: Use Case using the Semantic Nomenclature as a common knowledge sharing platform

Chapter 2

Presentation of the prototype to create authorization views

As we keep mentioning throughout this deliverable, access control has been positioned as a feature pertaining to the NeOn Infrastructure. However, we decided to make it "modular" that is, to design and develop a suite of access control components that build on top of the basic, "free to all" infrastructure and act as a kind of gatekeeper to facilitate so-called cautious knowledge sharing. This approach suits the experimental status of the access control in the world of ontologies at the moment, the research community is more concerned with actually having some ontology repositories at all, far less with making those repositories naturally compliant with the usual work practices of enterprises and enterprise information systems. In other words, our solution allows using the NeOn Infrastructure as it is as a public data store, and it offers means to also deploy access control as an enhancing but optional feature. We will describe the approach outline in [DKG+07] and refine later in [Dzb09]. Afterward, we will present the realization of the prototype in the NeOn infrastructure. As we will explain throughout this chapter, the access control process can be split into two sub processes; first, the acquisition of the module or authorization view; second, the retrievability of the authorization view depending on the access rights of a user.

2.1 Key determinants for the chosen access control method

As is common with business processes, certain information is restricted to selected individuals. Even if several people access the same information, they may see different versions of it and/or different levels of detail. Hence, it makes sense to replicate this situation and to create a number of partitions on the ontological models involved in such business processes. This can be achieved by expressing access authorities to the individual "views". Access control is governed by one or several access control models. Different access control models rely (to a different extent) on such operations as authentication or authorization to manage user access to the resources that would otherwise be accessible to potentially unauthorized users. In principle, in any access control model there are two entities:

1. subject entities / Agent : subjects are the entities permitted to perform a particular action in the access control system;

2. object entities : objects are those entities that represent resources to which access may be needed and may need to be controlled.

As further elaborated in D4.4.1 [DKG+07] access control relies at the very least on a triple comprising elements from three sets: Σ - a set of subjects (entities wanting to gain access); Ω - a set of objects (entities that may be accessed, also known as resources); and I - a set of access actions or invocations. A triple $\alpha = (s, o, i)|\Sigma.\Omega.I$ represents an access right or authority of subject s to access object o using action i.

Unlike documents, files and similar objects that usually are access-managed at the level of self-contained units, ontologies are naturally more granular. In terms of access control this means one can have an access-controlled ontology, in which primitive entities (e.g., classes, properties or instances), sub-sets of the primitive entities forming ad-hoc structures, formally modularized ontology segments and even entire ontologies may be given certain access rights.

This is a far too broad scenario, it is why we are making the following assumption; we will be treating modules as the minimal entity that can be access-controlled. An authorization view, in principle, may be as small as comprising a single class or instance, or as large as the whole ontology. More usually, module's size would be somewhere between the two extreme points of the spectrum. There are, in principle, two ways to create such a module for the sake of access control.

1. First, one may select appropriate entities from an ontology and declare them as sharing the same access privilege/right. Hence, we can define module as a customized view upon an ontology that satisfies certain meta-conditions with respect to sharing the same access right. This user-driven, personalization view of modules has been championed, e.g., in WP4 [[BDS$^+$08], [BS09] and [Ber10]].

2. Another way to define modules would be based on some structural or topological patterns appearing in the ontology. This formal, topic or structure-driven approach to module creation has been prefered in WP1.

Thus, as can be seen from this design decision, our approach is transparent as to the way modules are created. In the description of our approach in the next section we will use the ontology customization plugin.

2.2 Realization of access control in the NeOn infrastructure

As we see in the previous section, one of the key determinants is the definition of the view. In the ontology customization we choose a user-driven approach to help the user to write the view definition which we will present first. Afterwards, we will show a realization of granting access to users using our "access control API".

2.2.1 The access right acquisition

This section aims to give an overall idea about the different manner to create the view. But first we will solve the designation problem: how to refer the object the user want to access. In [DKG$^+$07] we raise this aspect and we proposed in the case of ontologies we have URL-s and URI-s at the disposal, we also ran an argument that to a great extent the ontologies and their content are about URI-s (pointers, identifiers), but access control is more about the URL (location to get content from).

In D4.4.1 we proposed using capability identifiers to act as an access handler and as an ontology designation. However, this is not practical - the URI of such an authority would look different for each user and would be hard to remember, especially if presented in the hashed form. In other words, the content of ontology (say, with URI http://foo.org/ontology) corresponds to many different access-handling URI-s (e.g., https://repo.org/ab3-1xy for user A, https://repo.org/de3-14s for user B, etc.). To resolve this issue we decided to split the designation process into several "layers". Upon creating an access-controlled view, this is given a URI, which also becomes the main access key, this value can be assigned using the GUI as shown in 2.2, a button help the user to generate the URI. This access key is then associated with the logical URI of the parent ontology and with a particular user name. Thus, user A may still use the logical URI to request the content of an ontology - let us denote it as rq_{ONT}. This logical URI is translated by our engine into possible access keys, thus the following mapping exists:

$$\mu \rightarrow \Lambda_{ONT} | \Lambda_{ONT} = \{KEY_i : viewOf(Key_i, rq_{ONT}), i = 1, 2, ...\}$$

To make a decision about granting access or otherwise, a similar mapping retrieves the keys associated with the respective user:

$$\varphi \rightarrow \Theta_{USR} | \Theta_{USR} = \{KEY_j : grantedTo(Key_j, id_{USR}), j = 1, 2, ...\}$$

Mapping μ can be interpreted as possible ways to modularize a given ontology for the purpose of a finer-grained access control. On the other hand, mapping φ is interpreted as the user's "keyring" containing valid access keys (capabilities, authorities). The actual access control decision is made on finding the most or least restrictive key satisfying the following condition:

$$rq_{ONT} | id_{USR} \Longleftrightarrow KEY_j \in \Theta_{USR} | \exists KEY_i \in \Lambda_{ONT} : KEY_j = KEY_i$$

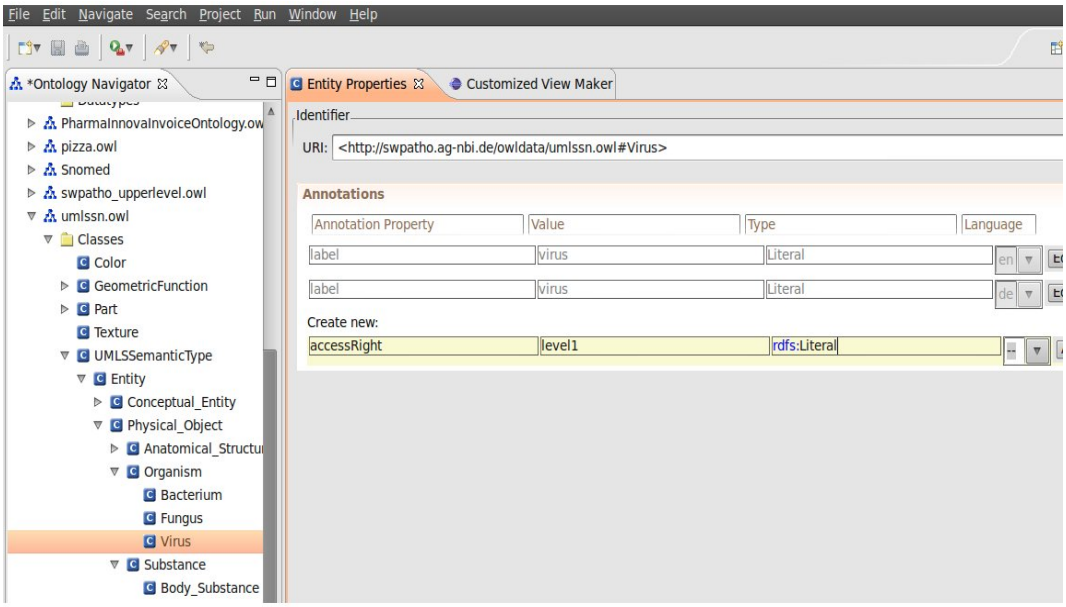

Figure 2.1: Annotate an entity to target the use of the filtering operator

The figure 2.1 shows a use of the filtering operator, first the user has to go through the ontology to annotate each entity he wants to share. We see in this figure the annotation of the class "Virus" with the annotation property "accessRight" and the value "level1". Of course the step can be done when the axion is created which makes this task less tedious for the user. Figure 2.2 shows the process of creating the view after we have annotated the needed axioms. We need to select the original ontology and then the plugin will automaticly detecte the different annotation properties used in this ontology. The user selects one property and specifies the value in this case level1. The tool allows the user to select several entites with different access right value.

We understand this method can be long and tedious for the user which is why we bring to the user another method to extract a view focus on a particular entity. In most cases, giving the access control at the entity level is too fine a granularity as the entity by itself is semanticless. What makes this entity semantic is the process of connecting it to other using properties or including it in a class expression. User wants to allow access to a specific part of the ontology this is exactly what we offer with the functionality shown in the figure2.3. In this figure we propose to create a view of Snomed focus on the class âĂIJVirusâĂİ to share this view with others. The view for our example is partially shown in the figure 2.4.

2.2.2 Using access right within the toolkit

The following section is extracted from the previous deliverable [Dzb09] and was inclued here for convenience.

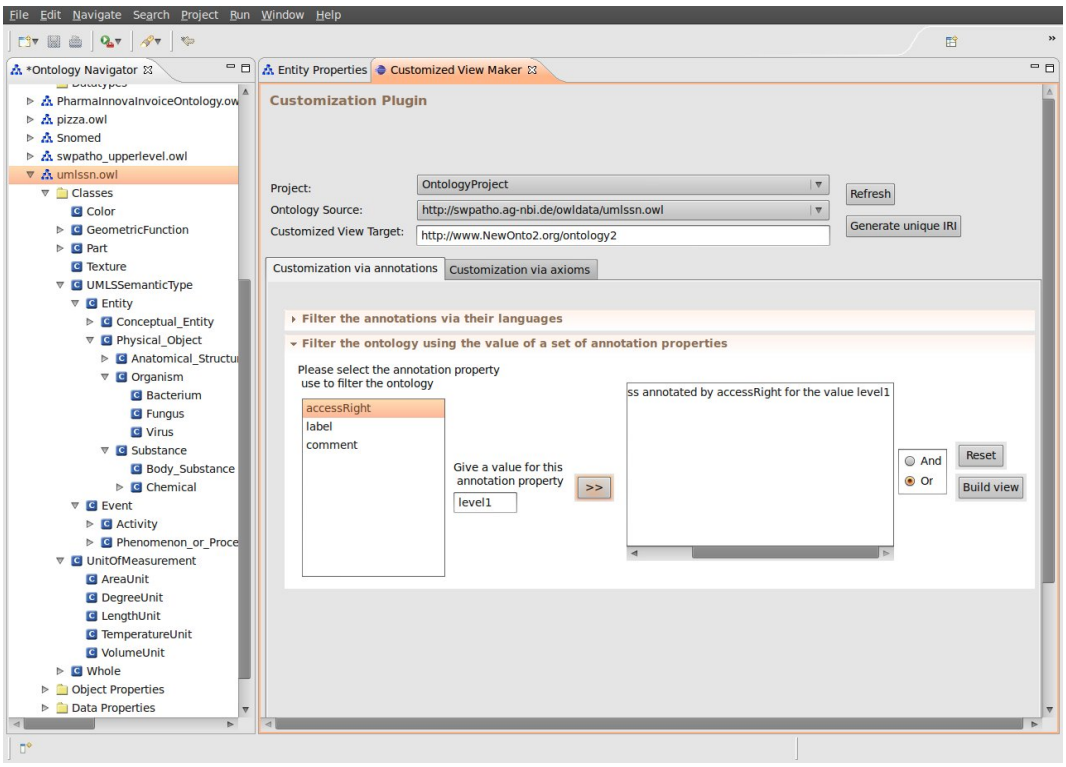

Figure 2.2: Annotation filter operator

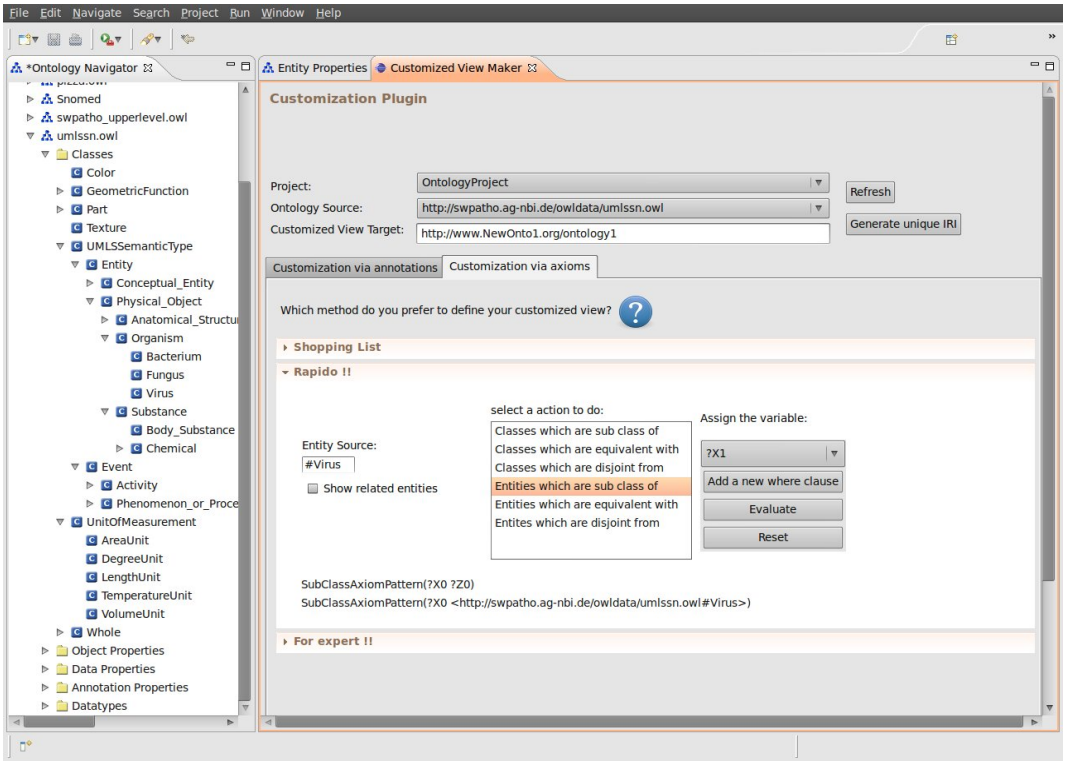

Figure 2.3: Creation of the view focus on “Virus”

Figure 2.5 represents the data flow for handling access control processing in the “access control API”. In order to explain the process of access control application the diagram is presented as data flow. The interaction starts with the user issuing either explicit or implicit query towards some ontology (1). By explicit query we

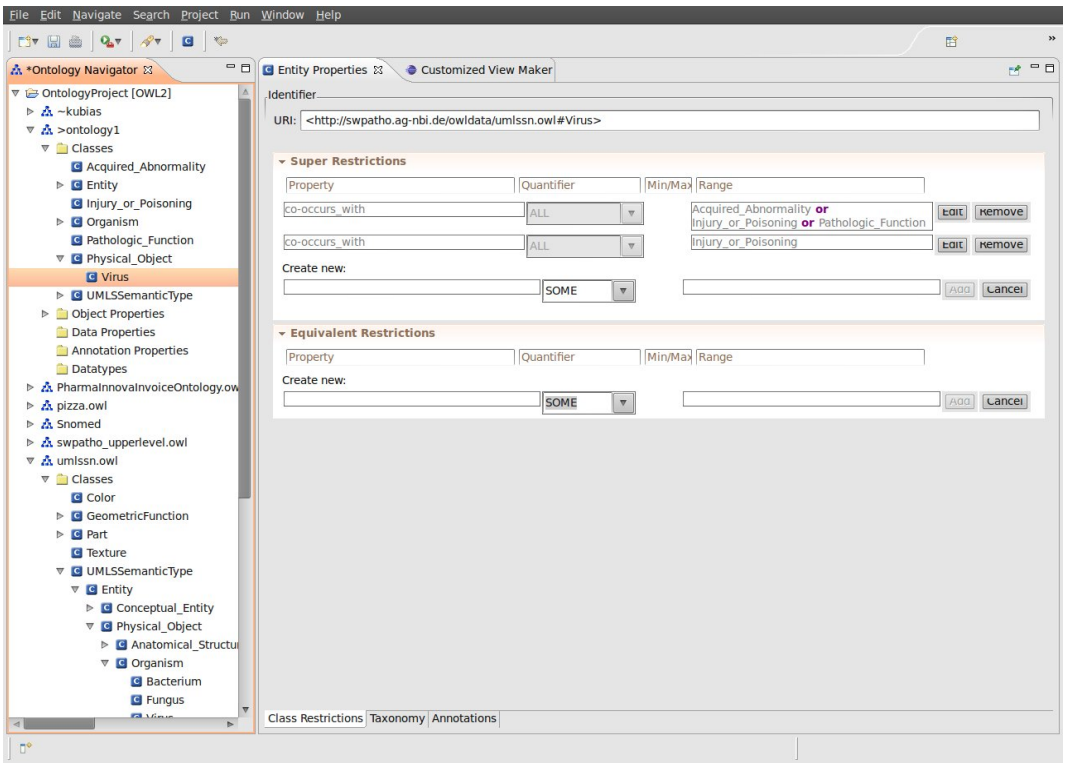

Figure 2.4: The view of Snomed centric on the class “Virus”

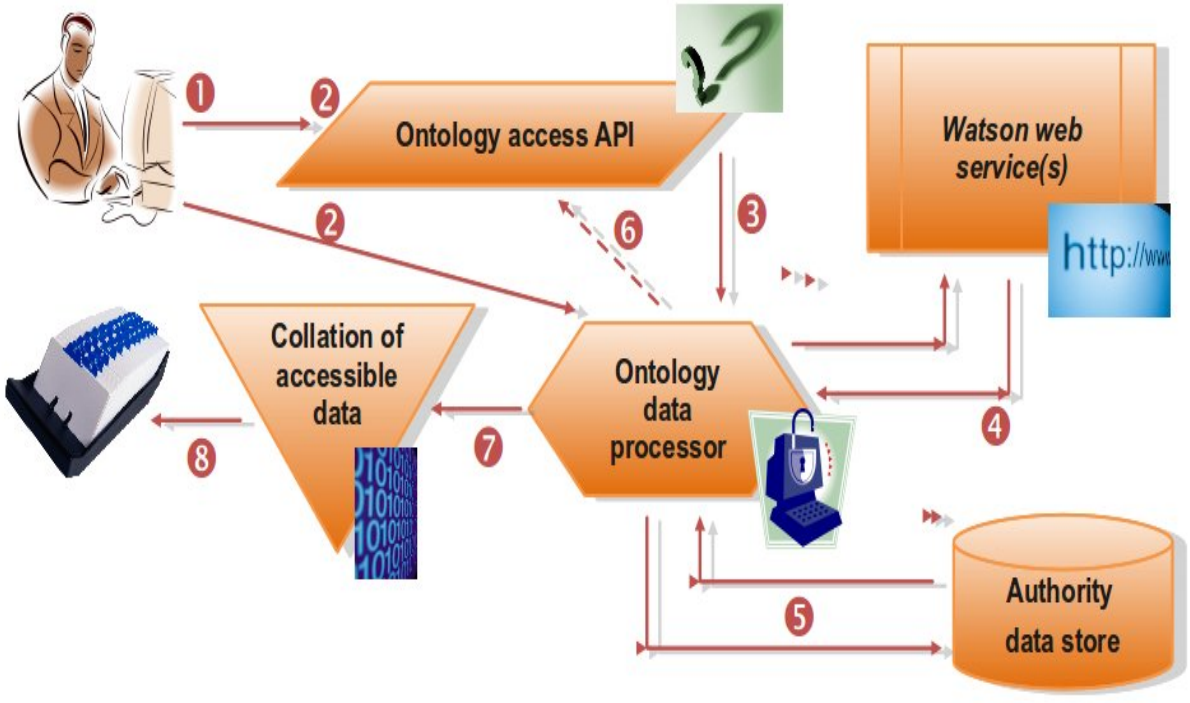

Figure 2.5: Schematic data flow for handling access control processing

mean, for example, requests to find all known entities that contain terms “dog” and “cat”; by implicit query we mean, for example, requests to describe domain and range for a given and known property in a given and known ontology. At the same time as the access to an ontology is initiated, the application passes user

identifier onto the access control engine (2) and the actual query is opaquely passed onto the appropriate Watson web service (3), but formally, it is issued by the access control engine rather than the original user.

Having processed the request, Watson replies with a result set structured according as documented for Watson API and depending on the scope of the query - term search, ontology search, or entity search (4). Upon receiving the result set, this is parsed by the ontology data processor to first identify the URI of the ontologies that are included in the result set, and then for each list of results associated with a given ontology to make a check in the authority data store whether or not ontology in question is access-controlled. The process captured by number 5 in figure 2.5 essentially corresponds to the formal transformations μ and φ that were defined in the previous section. The outcome of step 5 is basically yes/no response - either approving the inclusion of a particular entity from a particular ontology in the access-controlled data set or denying (and thus removing that particular entity from the original Watson return). Since some user requests may be implicitly nested - e.g., the request for a particular property usually assumes the description of that object property in terms of its domain and range - we allow for additional querying, whereby the engine issues clarifying or elaborating queries back to Watson web service and processes the partial results recursively until satisfied. This optional step is depicted by a dashed line and labelled as 6 in Figure 2.5. All partial results, for all retrieved ontologies that pass the authority check (and are thus accessible to a particular user) are then collated to re-create the original structure of the result as Watson would have provided it, were it not intercepted by the access control engine. The collation of data is shown as step 7 in Figure 2.5, and the whole process is concluded by delivering the accessible documents and entities to the user in step 8. The reason collation is approached recursively is to adhere to the key principle of access control, as introduced in D4.4.1 [DKG$^+$07]: ensuring that access control models operates at the level of minimal privileges.

In particular, one may imagine a situation where the user has access to object property geo:flowsTo but not to all classes that occur as domain and range. Say, the user can see geo:River as a valid domain of the above property, but has no access to get:Sea that would normally be the range of the above object property. Hence, the entire property definition needs to be adjusted prior to sending anything back to the user, in order not to inadvertently disclose or give a hint about the entities not accessible to the user. Note that the semantics of the original ontology and/or its part may thus change as a result of applying access control to certain entities. Some entities may "disappear" altogether from the ontology as it is returned to the user and others may be semantically modified. For example, assuming the user may only access rivers (i.e., class geo:River with its instances) and their respective countries (i.e., class geo:Country and its respective instances) from a geographic ontology of Europe, the ontology returned to the user will only contain these entities. It will not contain, say, classes geo:Sea or geo:City, despite them being defined in the original ontology. The latter case where the semantics is altered for some axioms corresponds to the example situation with geo:flowsTo object property - by denying access to class geo:Sea and thus not disclosing it as the property range we only have incompletely defined, unqualified object property.

In this chapter we present an elegant way to create authorization views to achieve access control. This solution allows the user to create a set of views from an ontology and restricted the access to it using a key mechanism. We also show that this solution has a weakness when the user may want to edit the view, it is for this reason we propose another solution based on the meta-knowledge to handle this issue.

Chapter 3

Using Meta Knowledge for enabling Access Control

Meta knowledge Reasoning allows to annotate ontology axioms with values from a *Meta Knowledge Dimension*, which is a set of values with two partial orders defined on it. When doing reasoning with such an ontology, we can not only do subsumption checks, but also determine which axioms are involved in the reasoning process and track and aggregate the corresponding meta knowledge assignments. This allows for example to compute an uncertainty degree of the answer based on the uncertainty degrees of the axioms in the input ontology, to compute from when an inference holds, based on modification dates of axioms, how trustworthy an answer is based on a trust hierarchy of the agents doing modifications and so on. In this deliverable, we describe how access rights can be modelled using meta knowledge dimensions, and show how these correspond to access groups and access control lists. A similar approach for access rights to DL ontologies is used in [BKP09]. While [BKP09] focuses on access control, our approach is based on the very flexible meta knowledge framework. Meta knowledge reasoning in principle is not limited to DL reasoning. Instead the blackbox algorithms used can be applied to many logical frameworks.

In the following we repeat some important definitions before modelling hierarchical ACLs in meta knowledge dimensions. We conclude by discussing applications of the resulting framework for access control.

3.1 Meta Knowledge

The meta knowledge reasoning framework used is based on Pinpointing. The blackbox algorithm employed is based on the pinpointing algorithm used by RadOn.

3.1.1 Pinpointing

As we use pinpointing as a vehicle for computing meta knowledge, we introduce pinpointing as a foundation for the rest of the section and give some information of existing algorithms for finding pinpoints.

The term pinpointing has been coined for the process of finding explanations for concluded axioms or for a discovered inconsistency. An explanation is a minimal set of axioms, which makes the concluded axiom true (or the theory inconsistent, respectively). Such an explanation is called a pinpoint. While there may be multiple ways to establish the truth or falsity of an axiom, a pinpoint describes exactly one such way.

Definition 1 *Pinpoint.*
A pinpoint for a entailed axiom A *wrt. an ontology* O *is a set of axioms* $\{A_1, ..., A_n\}$ *from* O*, such that* $\{A_1, ..., A_n\} \models A$ *and*
$\forall A_i \in \{A_1, ..., A_n\} : \{A_1, ...A_{i-1}, A_{i+1}, ..., A_n\} \not\models A$*. Analogously, a pinpoint for a refuted axiom* A *wrt. an ontology* O *is a set of axioms* $\{A_1, ..., A_n\}$ *from* O*, such that* $\{A, A_1, ..., A_n\}$ *is inconsistent and*
$\forall A_i \in \{A_1, ..., A_n\} : \{A, A_1, ...A_{i-1}, A_{i+1}, ..., A_n\}$ *is not.*

Hence, finding pinpoints for a refuted axiom corresponds to finding the Minimum Unsatisfiable Subontologies (MUPS) for this axiom [KPCGS06]. Pinpointing is the computation of all pinpoints for a given axiom and ontology. The truth of the axiom can then be computed using the *pinpointing formula* [BP07].

Definition 2 *Pinpointing Formula.*
Let A be an axiom, O an ontology and $P_1, ..., P_n$ with $P_i = \{A_{i,1}, ..., A_{i,m_i}\}$ the pinpoints of A wrt. O. Let lab be a function assigning a unique label to an axiom. Then $\bigvee_{i=1}^{n} \bigwedge_{j=i}^{m_i} lab(A_{i,j})$ is a pinpointing formula of A wrt. O.

A pinpointing formula of an axiom A describes, which (combination of) axioms need to be true in order to make A true or inconsistent respectively.

3.1.2 Semantics of Meta Knowledge

Meta knowledge can have multiple dimensions, e.g. uncertainty, a least recently modified date or a trust metric. In the next section we will model access rights using groups and access control lists as meta knowledge dimensions.

Definition 3 *Knowledge dimension. A knowledge dimension D is an algebraic structure $(B_D, \vee_D, \wedge_D)$, such that $(B_D, \vee_D)$ and $(B_D, \wedge_D)$ are complete semilattices.*

B_D represents the values the meta knowledge can take, e.g. all valid dates for the least recently modified date or a set of knowledge sources for provenance. As $(B_D, \vee_D)$ and $(B_D, \wedge_D)$ are *complete* semilattices, they are, in fact, also lattices. Hence, there are minimal elements in the corresponding orders.
As an example, let I be the meta knowledge interpretation that is a partial function mapping axioms into the allowed value range of a meta knowledge dimension, and A and B be axioms of an ontology such that $A \neq B$. Provenance, i.e. the set of knowledge sources a piece of knowledge is derived from, can be modelled as:

- $I(A \vee B) = I(A) \cup I(B)$
- $I(A \wedge B) = I(A) \cup I(B)$

The least recently modified date could be modelled as:

- $I(A \vee B) = min(I(A), I(B))$
- $I(A \wedge B) = max(I(A), I(B))$

Axioms can be assigned meta knowledge from any of the meta knowledge dimensions. Within a single assignment, the meta knowledge must be uniquely defined.

Definition 4 *Meta Knowledge Assignment.*
A meta knowledge assignment M is a set $\{(D_1, d_1 \in D_1), ..., (D_n, d_n \in D_n)\}$ of pairs of meta knowledge dimensions and corresponding truth values, such that $D_i = D_j \Rightarrow d_i = d_j$.

In our running example, the meta knowledge assignment for
H5N1 $\sqsubseteq$ *Virus* can be *{(accessGroup, epidemiologicalDepartment)}*
Without loss of generality we assume a fixed number of meta knowledge dimensions. As a default value for D_n in a meta knowledge assignment we choose the minimal element $\perp_D$.
Syntactically meta knowledge assignments are expressed using axiom annotations. We have provided a detailed grammar for meta knowledge annotations in [SDS09].

To allow for reasoning with meta knowledge, we need to formalize how meta knowledge assignments are combined. *How provenance* [GKT07] is a strategy, which describes how an axiom A can be inferred from a set of axioms $\{A_1, ..., A_n\}$, i.e. it is a boolean formula connecting the A_i. We call a logical formula expressing how provenance a *meta knowledge formula*. For example the following query for each city finds all companies:

The operators for meta knowledge dimensions extend to meta knowledge assignments, allowing us to compute meta knowledge for entailed knowledge by evaluating the corresponding meta knowledge formula.

In contrast to [SST08] we omit the $\neg$ operator in our formalization, as description logics are monotonic and $\neg$ in [SST08] allows for default negation. While axioms in the underlying description logic may contain negation, this negation is not visible and not needed on the level of the meta knowledge.

Definition 5 *Operations on Meta Knowledge Assignments.*
Let A, B *be axioms and meta*$(A) = \{(D_1, x_1), ..., (D_n, x_n)\}$ *and meta*$(B) = \{(E_1, y_1), ..., (E_m, y_m)\}$ *be meta knowledge assignments. Let dim*(A) *be the set of meta knowledge dimensions of* A*. Then meta*$(A) \vee$ *meta*$(B) = \{(D, x \vee_D y) | (D, x) \in$ *meta*(A) *and* $(D, y) \in$ *meta*$(B)\}$*.* $\wedge$ *is defined analogously.*

Having defined the operations on meta knowledge assignments, we can define how the meta knowledge of an axiom A within a meta knowledge dimension is obtained. The meta knowledge of an axiom A within a meta knowledge dimension is obtained by evaluating the corresponding meta knowledge formula in the dimension under consideration.

Definition 6 *Meta Knowledge of an Axiom.*
Let meta *be a function mapping from an axiom to a meta knowledge assignment in dimension* D*. The meta knowledge of an axiom* A *wrt.* O *in* D *is obtained by computing a pinpointing formula* ϕ *of* A *wrt.* O *and obtaining* ψ *by replacing each axiom in* ϕ *with its meta knowledge assignment in* D*. Then* meta*(A) is computed by evaluating* ψ*.*

For illustration consider the following excerpt from Snomed and modification dates:

ID	Axiom	date
1	Cell $\sqsubseteq \forall$ contains.Body_Substance	2009-10-15
2	Cell $\sqsubseteq$ Physical_Object	2009-10-12
3	Cell $\sqsubseteq \forall$ physical_part_of.(Body_Part_Organ_or_Organ_Component $\sqcap$ Cell $\sqcap$ Tissue)	2009-04-01
4	Cell $\sqsubseteq \forall$ adjacent_to.Cell	2009-12-15

Now let us answer the question: "Is something next to a cell a Physical_Object, that contains Body_Substance?" Formulated in DL as follows:

$$\exists \text{adjacent\_to}^{-}.\text{Cell} \sqcap \exists \text{contains} \sqsubseteq \text{Physical\_Object} \sqcap \exists \text{contains.Body\_Substance?}$$

From the ontology we can conclude that everything which has an adjacent Cell must be a Cell (4). If a cell contains something, it must be body_Substance (1). The last part to show is that a cell is a Physical_Object, which is the case (2). Hence, a pinpointing formula for the query axiom is $(1) \wedge (2) \wedge (4)$. Assume we want to compute the least recently modified date for the answer; we replace the axiom IDs by the corresponding modification dates and the $\wedge$ operator by the *max* operator: lmd $=$ max(2009-10-15, 2009-10-12, 2009-12-15) $=$ 2009-12-15.

3.2 Modelling ACLs using Meta Knowledge Dimensions

We model access rights through a combination of user groups and ACLs. Each user is member of one or more user groups. Groups can be hierarchical, i.e. one group can be a subgroup of another. There are two special groups: The group of all users, $\top$, and the empty group, $\bot$. Each axiom is assigned an ACL, that is a set of groups, whose members are allowed to use the axiom.

Definition 7 *Groups.*
A group *is a set of users. Let G, H be groups. We say G is a* subgroup *of H, if $\forall u \in G : u \in H$. We define an auxiliary function groups: users $\rightarrow 2^{groups}$ returning the groups a user is a member of.*

Definition 8 *ACL.*
An access control list (ACL) *wrt. a set S of groups is a subset of S.*

Based on this we model a meta knowledge dimension as follows:

Definition 9 *Access Rights Dimension.*
Let S be the set of user groups. Let A be the set of all ACLs wrt. S, i.e. the powerset of S. Let L be a partial order over A such that

- $\forall G, H \in S : \{G\} \leq_L \{H\}$ *if $G$is a subgroup of H. (subgroup inclusion)*
- $\forall B, C \in A : B \leq_L C$ *if $\forall G \in B : G \in C$. (ACL inclusion)*
- $\forall B \in A : B \leq_L \{\top\}$. *(global access)*
- $\forall B \in A : \{\bot\} \leq_L A$. *(inaccessible axioms)*

The Access Rights Dimension D is defined by A and the supremum and infimum wrt. L: $D = (A,\ sup_L,\ inf_L)$.

Depending on the desired access rights model, ACLs should default to $\{\top\}$ (Windows) or $\{\bot\}$ (Unix). Please note that while both Windows and Unix can control access on a per user basis in addition to groups, this can easily be modelled with singleton groups.
Before we can enforce access rights, we need to define how ACLs are attached to (a) axioms explicitly stated in an ontology and (b) inferred axioms. We follow the Windows strategy and use a default of $\top$.

Definition 10 *Enforcing Access Restrictions.*
Let O be an ontology and ACL_O a complete function mapping from axioms $A : O \models A$ to S. $ACL(A)$ returns the set of groups allowed to access A. A user can see an axiom only if he is in one of the groups $ACL(A)$.

- *If $A \in O$, $ACL(A)$ is included in O using a meta knowledge annotation as shown below.*
- *The access right of an inferred axiom A is computed by computing $meta_D(A)$.*

We use OWL2 annotations to express access rights in an ontology and follow the modelling of meta knowledge described in [SST08]. One such annotation expresses access granted to a single group. In order to express ACLS with multiple groups, multiple annotations on the same axiom need to be included in the ontology. For convenience reasons this model could be extended with assignments of access rights to named graphs containing multiple axioms as proposed in [SST08]. An example of how access rights are represented and associated with OWL axioms is presented below.

```
OWLAxiomAnnotation(SubClassOf(H5N1 Virus)
  MetaKnowledgeAnnotation(
    annot1 GroupAccess(epidemiologicalDepartment)))
OWLAxiomAnnotation(SubClassOf(H1N1 Virus)
  MetaKnowledgeAnnotation(
    annot2 GroupAccess(Anonymous)))
```

3.3 Prototype and Discussion

The strategy for enforcing access rights described in the previous section has been tested in a prototype using the optimized Meta Knowledge implementation described in deliverable [QS09]. We have manually created a simple access rights dimension modelling the group structure shown in figure 3.1. ACLs have been included in the ontology using meta knowledge annotations; group memberships need to be managed outside of the ontology, ideally by the operating system. In order to stay platform independent, we have omitted this step. Instead the prototype only returns the necessary access level to be allowed to see the answer to a query. Actual enforcement of access control is left to the application.

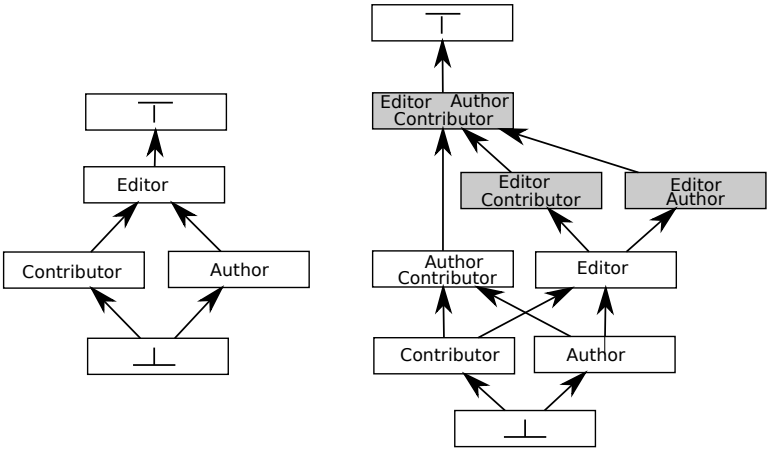

Figure 3.1: Example Groups. Left the actual groups, right the resulting lattice of ACLs. ACLs in grey are not neccessary in practice and could be omitted.

For experiments we have used the same ontologies as for the evaluation of the generic meta knowledge implementation in deliverable 1.2.5 [QS09] and extended it with access rights annotations. Figure 3.2 shows a comparison of access times for the example ontology with and without access control. As we can see, even the optimized meta knowledge algorithm is up to orders of magnitude slower than direct access to the ontology. The reason is that the meta knowledge blackbox algorithm needs multiple entailment checks for computing meta knowledge even for a query consisting of a single entailment check in the underlying DL. While this is perfectly acceptable when analysing query results or inconsistencies, it causes significant overhead when used for every single interaction with an ontology.

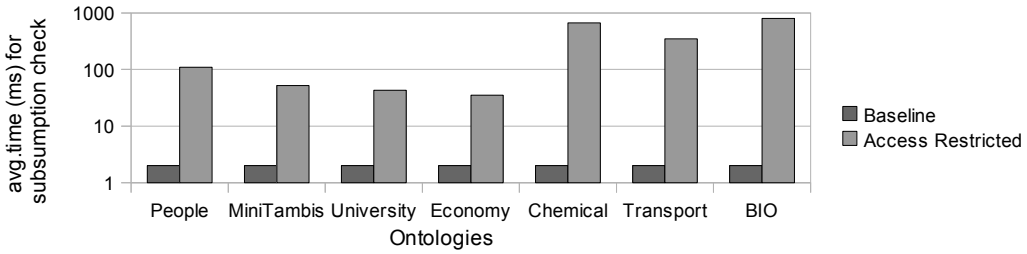

Figure 3.2: Time needed for classification

However, the approach to access rights presented in this chapter has a significant advantage: it is based on a single, enriched version of the original ontology. For access control based on customization or modularization in contrast, multiple modules or views respectively need to be managed. Of course the opposite side of the coin is that when changing access rights, the original ontology needs to be modified, while in the customization based approach only a view definition, which is declared outside of the ontology, needs to be modified.

For the time being runtime control of access rights based on meta knowledge reasoning incurs a significant

overhead. However, judging from the performance gains in DL reasoning over the last few years and from the optimizations to meta knowledge reasoning we presented in [QS09], we expect this approach to become feasible in the next years.

3.4 Comparison of the two approaches

Table 3.1 shows in a comprehensive manner the comparison between the two approaches. Method One in the table corresponds to the authorization view approach and Method Two represents the method using meta-knowledge. This table shows that those methods are complementary as the weakness from one is the strength of the other.

Table 3.1: Comparison of the two approaches

	Method One	Method Two
Advantages	propose an easy way to create the module you want to share with little interaction from the user	Work at the runtime
	pre-computed views	Work on the original ontology
Disadvantages	Make the changes in the original ontology difficult	Long process to assign the access right to each entity
Answering Requirement (cf. chapter 1)	1,3	2,3

It is important to note that the approaches are made for different purposes. The authorization view approach provides an easy tool through which the user can define the view related to part of the ontology he may want to share. This is the perfect solution if the user wants to allow read access, for example, for reasoning purposes to part of the original ontology. Nevertheless, this approach runs into one well know problem should the user want to share this view for some editing intention. The meta-knowledge approach is the better option here in the case of editing an ontology collaboratively with different expertise (e.g. expert in French language) like the fishery use case presented in requirement 2.

Chapter 4

Conclusion

In this report we described two methods for applying access to description logic ontologies. The first access control method is following up on what was proposed in the previous work, the functional features of the access management framework designed for the NeOn Infrastructure in the form of ontology view composer plus API. The ontology view composer is an extension of the ontology customization plugin for helping the user to define the views. The API is broadly based on the existing Watson API, which in turn enables the user to query a remote ontology repository for entire ontologies and selected ontological resources. We say it is only “broadly based” as due to introducing new features (user identification), one cannot implement this access control API using the straightforward “extends” keyword. However, we strive to maintain the structure of the API in terms of supporting search in the scope of entities, ontologies and semantic content in general. Also, the methods exposed by the proposed API are one to one reflecting the methods provided by the original Watson API.

The second method proposes an elegant solution to ensure access rights at the runtime. It offers fine grained control on the axiom level and dynamically controls access to inference results. The added flexibility, however, comes at the cost of a significant overhead in reasoning complexity. In fact, it is worst case exponential in the underlying DL. In order to make this second method practically usable, significant efficiency gains are needed. For this reason we have based the access control plugin on the more efficient customization based approach.

Bibliography

[BDS+08] Noam Bercovici, Martin Dzbor, Simon Schenk, Alexander Kubias, and Gerd GrÃűner. Ontology customization and module creation: query-based customization operators and model. Deliverable D4.2.2, NeOn Project, 2008.

[Ber10] Noam Bercovici. Ontology customization plugin presentation. Deliverable D4.2.4, NeOn Project, 2010.

[BKP09] Franz Baader, Martin Knechtel, and Rafael PeÃśaloza. A generic approach for large-scale ontological reasoning in the presence of access restrictions to the ontology's axioms. In *ISWC2009*. Springer, 2009.

[BP07] Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. In *TABLEAUX '07: Proceedings of the 16th international conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 11–27, Berlin, Heidelberg, 2007. Springer-Verlag.

[BS09] Noam Bercovici and Simon Schenk. Ontology customization prototype presentation. Deliverable D4.2.3, NeOn Project, 2009.

[DKG+07] Martin Dzbor, Alexander Kubias, Laurian Gridinoc, Angel Lopez-Cima, and Carlos Buil Aranda. The role of access rights in ontology customization. Deliverable D4.4.1, NeOn Project, 2007.

[Dzb09] Martin Dzbor. Realization of a prototype extension for access control in neon infrastructure. Deliverable D4.4.2, NeOn Project, 2009.

[GKT07] Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance Semirings. In *PODS*, pages 31–40, 2007.

[KPCGS06] A. Kalyanpur, B. Parsia, B. Cuenca-Grau, and E. Sirin. Axiom pinpointing: Finding (precise) justifications for arbitrary entailments in OWL-DL. Technical report, 2006.

[QS09] Guilin Qi and Simon Schenk. D1.2.5 inconsistency-tolerant reasoning with networked ontologies. Technical Report D1.2.5, Universität Karlsruhe, 2009.

[QSJ09] Guilin Qi, Simon Schenk, and Qiu Ji. D3.1.4 reasoning with meta-knowledge. Technical Report D3.1.4, Universität Karlsruhe, 2009.

[RMSR04] Shariq Rizvi, Alberto Mendelzon, S. Sudarshan, and Prasan Roy. Extending query rewriting techniques for fine-grained access control. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 551–562, New York, NY, USA, 2004. ACM.

[SDS09] Simon Schenk, Renata Dividino, and Steffen Staab. Reasoning with Provenance, Trust and all that other Meta Knowlege in OWL2. In *SWPM2009*. CEUR, 2009.

[SST08] B. Schueler, S. Sizov, and D. T. Tran. Querying for Meta Knowledge . In *WWW2008*, pages 625–634. ACM, 2008.