Open World Reasoning in Semantics-Aware Access Control: a Preliminary Study

E. Damiani[1], S. David[2]*, S. De Capitani di Vimercati[1], C. Fugazza[1], and P. Samarati[1]

[1] Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano
{damiani,decapita,fugazza,samarati}@dti.unimi.it
[2] Ontology Engineering Group
Universidad Politécnica de Madrid
sdavid@delicias.dia.fi.upm.es

Abstract. We address the relationships between theoretical foundations of Description Logics and practical applications of security-oriented Semantic Web techniques. We first describe the advantages of semantics-aware Access Control and review the state of the art; we also introduce the basics of Description Logics and the novel semantics they share. Then we translate the principle underlying the *Little House Problem* of DL into a real-world use case: by applying Open World Reasoning to the Knowledge Base modelling a Virtual Organization, we derive information not achievable with traditional Access Control methodologies. With this example, we also show that a general problem such as ontology mapping can take advantage of the enhanced semantics underlying OWL Lite and OWL DL to handle under-specified concepts.

1 Introduction

Recently, semantic Web techniques are increasingly being applied to provide advanced descriptions of users and resources in order to extend the capabilities of Access Control methodologies. The motivation for doing this is that the categorization of entities involved in authorization processes is very important for the definition of consistent rules, contributing to the overall expressiveness of a policy definition language. It is also essential for enabling rule propagation in order to relieve the system administrator's effort when managing them. The advanced built-in semantics of Semantic Web languages provide a conceptual framework for modelling these aspects with respect to widely acknowledged, object-oriented principles.

The need for a high degree of expressiveness in the representation of users and resources is even more evident in open environments such as Web Services and Virtual Organizations because it is difficult to obtain a uniform conceptualization of the entities involved. Therefore the rules regulating access to resources cannot be properly translated until correspondences are drawn between independent concepts sharing the same meaning. Exploiting the logic foundations of Semantic Web languages such as RDFS

* This work was carried out while I was a student at the Faculty of Computer Science of the Free University of Bolzano

and OWL can be of great use for reconciling with each other heterogeneous representation schemes of this kind.

Moreover, existing techniques for associating rich descriptions to users and resources often prefer the enhanced expressiveness of OWL but do not take full advantage of the underlying operational semantics of Description Logics. In this paper we try to sketch how Open World Reasoning can be useful to cross organizational boundaries and allow for a more detailed mapping of independent categorizations of entities in the composition of Virtual Organizations (VO). Essentially, our example is a security-oriented translation of the original *Little House Problem* of DL [22]. The presented example may seem very simple at a first sight. However, once shown the reasons of its importance, it reveals how subtle might be the implementation of (security related) ontologies in the Semantic Web.

The paper is organized as follows: in Sec. 2 we provide a general view on the possible applications of Semantic Web techniques for augmenting the expressiveness of Access Control frameworks and also review related work in this field. In Sec. 3 we introduce the generalities of Description Logics and their relationships with OWL Lite and OWL DL. In Sec. 4 we focus on the different semantics of Open World Reasoning w.r.t. the Closed World approach of traditional information retrieval systems such as relational databases. We also we explain the foundations and historical origins of the *Little House Problem*. In Sec. 5 we apply this problem to our simple use case. Essentially, we first provide a subgraph describing individuals within a company and define a rule to dynamically select them according to their roles and mutual dependencies. Then a second subgraph is merged with the first in order to create a VO out of both companies. We describe how far the rule can be applied to the enlarged structure with regard to different interpretations of the Knowledge Base. We show that, as soon as the second organization is integrated with the first, the under-specified environment produced by the merging of the second subgraph cannot be handled anymore by Closed World Reasoning. The original rule not only cannot take advantage of the full personnel of the enlarged VO, but it also fails to behave correctly with respect to the original one. Then we show how Open World Reasoning can solve this discrepancy and encompass the newly introduced individuals in the decision process. Finally, Sec. 6 draws the conclusions and anticipates future work.

2 Semantics-aware Access Control

The most evident advantage of integrating Access Control architectures with Semantic Web techniques is the opportunity of applying the fine-grained categorization primitives of SW languages to provide a more detailed description of the entities involved. By doing this, rules applying to a given concept can be extended to related concepts according to well defined principles such as subsumption, union, intersection. Context information (for instance users, roles, resources, and credentials) can be expressively represented with concepts and instances of an OWL ontology whose consistency can be automatically checked with existing tools. The work [3] presents rule propagation strategies which are directly driven by the semantics-aware categorization of context information.

Another possibility is to model policies themselves as an OWL ontology and try to apply the built-in semantics of the language for evaluating policy propagation. For this purpose the tools of ontology modelling does not always provide the expressiveness required by task-dependent functional properties. As an example the definition of variables (a desirable feature in a rule specification language) cannot find a proper primitive in the OWL specification. Whereas a variable can sometimes be mimicked, as for instance in our example, the lack of an infrastructure for custom rules definition is a strong limitation of pure-OWL approaches. Two mainstream implementations [1] [11] tackle this issue from different perspectives adopting hybrid approaches compared in [7]. In the meanwhile, the Semantic Web Rule Language (SWRL)[5] has been proposed to extend the expressiveness of OWL in order to apply Horn-like rules to a Knowledge Base.

A different subject is what kind of inference should be carried out on the semantics-aware Knowledge Base, as reasoning facilities can fulfil many purposes. Rule propagation on the basis of context information allows to pre-compute alternative credentials to be provided for a resource or service to be granted. Moreover, policy negotiation can take advantage of inference for filtering these alternatives according to the system administrator's preferences. With respect to these functionalities it is of foremost importance to consider the semantics underlying the reasoning process in order to avoid unexpected results in a sensitive matter such as Access Control. This paper aims at exemplifying this issue with regard to the different interpretations of a Knowledge Base provided by the Close World Assumption and Open World Assumption respectively.

3 Description Logics

The Description Logics formalism has been chosen to underlay the semantics of Semantic Web languages, like OWL DL. Hence, a thorough understanding of that formalism should be the starting point for developing Semantic Web applications. The remainder of this section introduces the basic key points and features of the Description Logics framework and how it can be used for setting up Knowledge Bases and providing inference services for the discover of possible implicit knowledge. However, this presentation is not meant to be exhaustive, therefore the interested reader shall refer to the Description Logic Handbook [4] or the bibliography for a deeper presentation of this subject.

3.1 Syntax of DL languages

A Knowledge Base is a set of assertions about a domain, defined by means of Classes and their Properties and Relationships. It can be described in Description Logics by means of a *concept language*, in which Classes are called (atomic) Concepts, whose Properties and the Relationships between them are called (atomic or primitive) Roles, and Individuals are instances of Classes. Concept, Roles and Individuals are expressed with unary predicates, binary predicates and constants, respectively.

Notation. In this paper, A and B will represent atomic Concepts, C and D complex Concepts, R and S (atomic) Roles and a, b Individuals.

The following example shows how to build a complex concept and some example of instantiation of them:

EXAMPLE: Combine the atomic Concept MAN and the atomic Role hasChild to describe the complex Concept of FATHER $\equiv$ MAN $\sqcap$ hasChild, i.e., a Father is a man that has a child. Now we can describe a domain in which there is a man called John who has a children called Paul: MAN$(John)$, hasChild$(John, Paul)$

3.2 Semantics of DL Languages

We first introduce some useful notation, used in this and in the next sections.

Notation. We will denote with Δ the domain of discourse, with $\times$ the cartesian product and with $\sqsubseteq$ the subsumption relation between Concepts or Roles.

The semantics of the languages is given in terms of an *interpretation*, defined as a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set called *domain* and the interpretation function $\cdot^{\mathcal{I}}$ is a mapping from every Concept to a subset of $\Delta^{\mathcal{I}}$, from every Role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and from every Individual to an element of $\Delta^{\mathcal{I}}$. An interpretation I is a *model* for a Concept C if the set $C^{\mathcal{I}}$ is non-empty.

One of the simplest among the concept languages is $\mathcal{AL}$ (the Attributive Language), which will be described here; it consists of the constructs shown in Figure 1 along with their syntax and semantics. The language $\mathcal{AL}$ can be extended by means of additional constructs, some of which are presented in Figure 2; each of them is uniquely assigned a letter, conventionally used for identifying the language built by augmenting $\mathcal{AL}$ with that construct, following the schema:

$$\mathcal{AL}[\mathcal{E}][\mathcal{U}][\mathcal{C}][\mathcal{Q}][\mathcal{N}][\mathcal{O}][\mathcal{I}]$$

EXAMPLE: The language $\mathcal{ALCQI}$ is $\mathcal{AL}$ augmented with full negation, qualified number restriction and inverse Roles.

Alternatively, the additional constructs of Role constructors can be written as superscripts and restrictions on the interpretation of Roles as subscript.

EXAMPLE: The languages $\mathcal{ALCI}$ and $\mathcal{ALC}^{-1}$ both identify $\mathcal{AL}$ with full negation and inverse Roles.

We sketch now the relationship between the Semantic Web languages OWL DL and OWL Lite and two concept languages of the $\mathcal{AL}$ family. We want to focus the importance of the relation between the Description Logics and the OWL DL and OWL Lite subspecies of OWL, since implementors of Semantic Web applications can take advantage from the various theoretical results (i.e., algorithm and their implementation) achieved over the years by the Description Logics community.

Notation. As a shortcut for the language $\mathcal{ALC}_{R^+}$, which is $\mathcal{AL}$ with transitive Roles and full negation, the letter $\mathcal{S}$ is used.

Using this notation, OWL DL and its Vocabulary Terms are mapped to the $\mathcal{SHOIN}(\mathbf{D})$ Description Logics, which is $\mathcal{S}$ with nominals, inverse roles, qualified number restriction and Role hierarchy, the latter denoted by the symbol $\mathcal{H}$ (see section 3.3 for a formal definition). Moreover, the symbol $\mathbf{D}$ indicates a *Datatype theory*, i.e., a mapping from a set of datatypes to a set of values. OWL Lite is equivalent to the $\mathcal{SHIF}(\mathbf{D})$ Description Logics, which is equivalent to $\mathcal{SHOIN}(\mathbf{D})$ without nominals (i.e., the `oneOf` constructor) and cardinality constraints limited to 0 and 1.

The mapping of OWL DL vocabulary terms to $\mathcal{SHOIN}(\mathbf{D})$ (and of OWL Lite to $\mathcal{SHIF}(\mathbf{D})$), along with a discussion over their complexity and the proof of reducing reasoning in OWL ontologies to Description Logics (un)satisfiability, is analyzed and presented in [15].

For both $\mathcal{SHIF}$ and $\mathcal{SHOIN}$ there are efficient sound and complete algorithms and corresponding implementations: we recall here the RACER System [12], the FaCT and iFaCT systems [13] and the Pellet reasoner [20] for OWL. Moreover, Pellet and iFaCT implement a tableaux decision procedure developed recently for the $\mathcal{SHOIQ}$ Description Logics, see [16].

Name	**Syntax**	**Semantics**
Concept name	A	$A^{\mathcal{I}} \subset \Delta^{\mathcal{I}}$
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Value restriction	$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$
Limited existential qualification	$\exists R.\top$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\}$

Fig. 1. Syntax and Semantics of $\mathcal{AL}$ constructs

3.3 TBox and ABox

When using the Description Logics formalism for describing Knowledge Bases, two categories of knowledge can be identified in them: the *intensional* knowledge, which is fixed (timeless) and does not depend from the domain of the discourse, and the *extensional* knowledge, which can change over time.

Notation. A Knowledge Base will be denoted by Σ=$(\mathcal{T}, \mathcal{A})$; where $\mathcal{T}$ is the TBox and $\mathcal{A}$ the ABox. A statement, or *axiom*, in either the TBox or the ABox, will be represented by α.

The *intensional* knowledge is contained in the TBox and is defined by Concepts and Roles, which form the terminology, i.e., the vocabulary used to describe the whole domain and to assign names to complex concepts description. An axiom $\alpha \in \mathcal{T}$ may assume one of the following forms:

Name	**Syntax**	**Semantics**	**Symbol**
Full Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$\mathcal{C}$
Existential Quantification	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$	
Unqualified	$\geq n\, R$	$\{i \in \Delta^{\mathcal{I}} \mid \lvert\{j \in \Delta^{\mathcal{I}} \mid R(i,j)^{\mathcal{I}}\}\rvert \geq n\}$	
number	$\leq n\, R$	$\{i \in \Delta^{\mathcal{I}} \mid \lvert\{j \in \Delta^{\mathcal{I}} \mid R(i,j)^{\mathcal{I}}\}\rvert \leq n\}$	$\mathcal{N}$
restriction	$=n\, R$	$\{i \in \Delta^{\mathcal{I}} \mid \lvert\{j \in \Delta^{\mathcal{I}} \mid R(i,j)^{\mathcal{I}}\}\rvert = n\}$	
Qualified	$\geq n\, R.C$	$\{i \in \Delta^{\mathcal{I}} \mid \lvert\{j \in \Delta^{\mathcal{I}} \mid R(i,j)^{\mathcal{I}} \wedge C(j)^{\mathcal{I}}\}\rvert \geq n\}$	
number	$\leq n\, R.C$	$\{i \in \Delta^{\mathcal{I}} \mid \lvert\{j \in \Delta^{\mathcal{I}} \mid R(i,j)^{\mathcal{I}} \wedge C(j)^{\mathcal{I}}\}\rvert \leq n\}$	$\mathcal{Q}$
restriction	$=n\, R.C$	$\{i \in \Delta^{\mathcal{I}} \mid \lvert\{j \in \Delta^{\mathcal{I}} \mid R(i,j)^{\mathcal{I}} \wedge C(j)^{\mathcal{I}}\}\rvert = n\}$	
Nominal[a]	I	$I^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ with $\lvert I^{\mathcal{I}}\rvert = 1$	$\mathcal{O}$
Inverse Role	R^{-1}	$\{(a,b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(b,a)\}$[b]	$\mathcal{I}$

[a] Intuitively, a Nominal is a Class that admits only one instance, i.e., a singleton.

[b] To avoid avoid the case of having Roles like R^{--} (i.e., inverse of inverse Roles), define `Inv` s.t. `Inv(`R`)`$= R^{-}$ and `Inv(`R^{-}`)`$= R$. See also [15]

Fig. 2. Additional constructs for $\mathcal{AL}$

$A \sqsubseteq C$	denotes primitive Concept definition
$A \doteq C$	denotes Concept definition
$C \sqsubseteq D$	denotes Concept inclusion
$C \doteq D$	denotes Concept equivalence
$R \sqsubseteq S$	denotes Roles inclusion

The first four are related to Concept axioms, whereas the last one refers to Role axioms and is also called Role hierarchy, i.e., a set of *role inclusion axioms*, all of the form $R \sqsubseteq S$, with both R, S primitive roles. As said above (section 3.2), the presence of a Role hierarchy is often denoted by the symbol $\mathcal{H}$.

The semantics is straightforward and we describe here only one of the cases: an interpretation $\mathcal{I}$ satisfies $C \sqsubseteq D$ if and only if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ (i.e., the interpretation of C is contained in the interpretation of D)); the other forms are defined similarly. If $\mathcal{I}$ satisfies all axioms $\alpha \in \mathcal{T}$, $\mathcal{I}$ is said to be a *model* for it and $\mathcal{T}$ is said to be *satisfiable*.

The *extensional* knowledge is contained in the ABox, which contains assertions about individuals in the domain. An axiom $\alpha \in \mathcal{A}$ has one of the forms

$C(a)$	Concept Membership assertion
$R(a,b)$	Role Membership assertion

An interpretation $\mathcal{I}$ satisfies $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and satisfies $R(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. Like in the case of the TBox, if $\mathcal{I}$ satisfies all axioms α in the ABox $\mathcal{A}$, $\mathcal{I}$ is said to be a *model* for it and $\mathcal{A}$ is said to be *satisfiable*.

Combining what has been said so far, a Knowledge Base Σ is a pair $\Sigma = (\mathcal{T}, \mathcal{A})$, which has a *model* if there exist an interpretation $\mathcal{I}$ that satisfies both $\mathcal{T}$ and $\mathcal{A}$. If an axiom α in $\mathcal{T}$ or $\mathcal{A}$ is true in every model of Σ, then we say that Σ *logically implies* α.

Notation. We write $\Sigma \models \alpha$ when Σ logically implies α.

3.4 Reasoning

A Description Logic system provides many basic inference services. Here we present some of them, along with the sketch of how they are used to build some more complex one.

1. *Subsumption* ($\Sigma \models C \sqsubseteq D$): decide whether a concept is more general than another one. That is, to decide whether $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in every model I of Σ. Upon Subsumption, the process of *Classification* is built, which is the process that receives as input a TBox $\mathcal{T}$ and a set of concepts in $\mathcal{T}$ and determines, for all pairs (C, D), whether $C \sqsubseteq D$ or $D \sqsubseteq C$. Intuitively, Classification builds the hierarchy of the concepts in $\mathcal{T}$.
2. *Concept Satisfiability* ($\Sigma \not\models C \equiv \bot$): decide whether C is satisfiable in Σ, i.e., if there is a model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$.
3. *Instance Checking* ($\Sigma \models C(a)$): the problem to decide if $C(a)$ is satisfied in every model of Σ. On Instance Checking in based the process of *Retrieval* or *Query Answering*: given a Knowledge Base Σ and a concept $C \in \Sigma$, find the set $\{a \in \Delta \mid \Sigma \models C(a)\}$. Retrieval is carried out by iterating Instance checking on every individual of the domain).
4. *Consistency Check* ($\Sigma \not\models$): decide if Σ is satisfiable, i.e., if it is coherent and admits a model.

The complexity of the inference process depends on the language in which it is applied. Intuitively, the more expressive the language, the more the process might result complex. The main properties of a reasoning process are *soundness* and *completeness*. The former is the capability of an inference process to produce only correct deductions, whereas the latter is its capability to find all possible implicit information. The importance of having sound and complete inference engines implemented by Semantic Web applications is pointed out by [2].

4 CWA Vs. OWA

The Closed World Assumption and the Open World Assumption (or Semantics) represent two different approaches in how to evaluate *implicit* knowledge in a Knowledge Base.

The difference in their behavior is usually clarified by a comparison between the structure of a Knowledge Base Σ=$(\mathcal{T}, \mathcal{A})$ and a Database, where its schema (i.e., its tables and structure) represents $\mathcal{T}$ and its tuples represent $\mathcal{A}$, and is given in terms of interpretations. On one side, we have a single Database instance, which represent the only one possible interpretation, on the other one we have one out of all possible interpretations of Σ. Hence, while in a Database if an information is not explicitly stated in a tuple, it is interpreted as "*negative*" or false knowledge, in Knowledge Bases it is considered false only if it contradicts other axioms in the domain.

The typical example that shows how these two approaches differ is described in the next section. It has originally been formulated by Andrea Schaerf in [22] and, is usually (and informally) referred to as the *Little House Problem*.

4.1 The *Little House Problem*

We present here the original formulation of the problem[3]; it stems from a discussion about complexity of Knowledge Bases. The interested reader may refer to chapter 4 of [22] or to [10] for a deeper presentation of the problem.

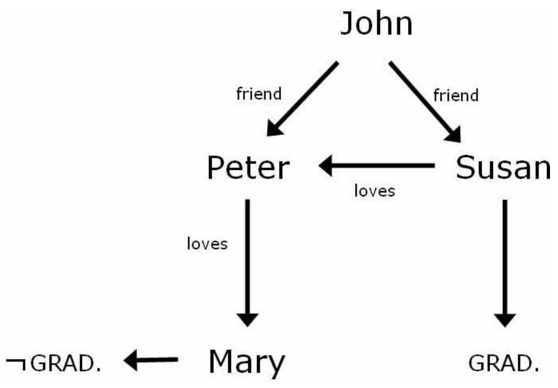

Fig. 3. The original *Little House Problem*

Let Σ and β be the following Knoledge Base and statement, respectively:

$$\begin{aligned}\Sigma = \{&\mathsf{friend}(John, Susan),\ \mathsf{friend}(John, Peter);\\ &\mathsf{loves}(Susan, Peter); \mathsf{loves}(Peter, mary);\\ &\mathsf{GRADUATE}(Susan);\ \neg\mathsf{GRADUATE}(mary); \}\\ \beta = {}&\exists\mathsf{friend}.\,(\mathsf{GRADUATE} \sqcap \exists\mathsf{loves}.\,\neg\mathsf{GRADUATE})(\mathrm{John})\end{aligned}$$

Deciding whether $\Sigma \models \beta$, amounts to query the given Knowledge Base for John having a graduate friend who loves a non-graduated person. At first glance is seems that this query has answer NO, hence $\Sigma \not\models \beta$, due principally to the lack of knowledge about Peter (as shown in the graphical representation of Σ in Figure 3): there is no evidence of him being graduate or not. Indeed fact the query has answer YES, β is TRUE, and $\Sigma \models \beta$. Without going into details, we sketch how to solve this apparent incongruity.

Informally speaking, the query asks whether, in all possible models of Σ it is possible to find an individual, say x, that satisfies at the same time all of the conditions

[3] The version presented here is expressed in the $\mathcal{ALE}$ Description Logic. However, with minor modifications, it can be expressed in other concept languages, as well as in both OWL DL and OWL Lite (see [9]).

$\mathsf{friend}(John, x)$, $\mathsf{GRADUATE}(x)$ and $\exists\mathsf{loves}.\neg\mathsf{GRADUATE}(x)$, or, in natural language, if John has a friend that is a graduate and this friend loves a third person that is not a graduate. Proving this result requires the introduction of the *case analysis* or *case reasoning* notion. With respect to the Knowledge Base in Fig. 3, in all models of Σ either $\mathsf{GRADUATE}(Peter)$ or $\neg\mathsf{GRADUATE}(Peter)$ holds, therefore we can use this remark to split all the models of Σ into two families of subproblems: one in which $\mathsf{GRADUATE}(Peter)$ holds, and another where $\neg\mathsf{GRADUATE}(Peter)$ holds, and analyze each of the two subproblems in which the original one has been split. Hence, in the former case we have that $Peter$ is an individual x that satisfies β, while in the latter case we have $Mary$ as the x that satisfies β. As a conclusion, since in both cases it is possible to satisfy β, so we conclude that $\Sigma \models \beta$.

The next sections show how these argumentations remain valid not only from a theoretical point of view, but also when trying to model ontologies involving real situations and real organizations.

5 Structuring Virtual Organizations with Ontologies

In order to provide a real-world example of how difficult the making of a Semantic Web could be, we consider a widely acknowledged practice such as the composition of Virtual Organizations. In this environment, the lack of a common framework for describing individuals and resources within companies introduces the need to integrate heterogeneous representation schemes. Since these has been conceived according to different requirements, not every concept in one model will find a corresponding concept in the other. Nevertheless, the mapping between equivalent concepts is essential to seamlessly integrate external resources into the work flow. In our example, two collaborating workforces are represented as OWL ontologies and the problem of reconciling with each other the different categorizations of human resources is translated into the problem of ontology mapping.

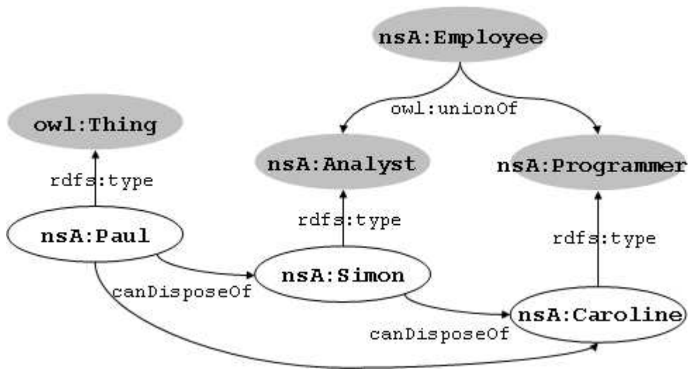

Fig. 4. A fragment of the Knowledge Base modelling the structure of company A: property $\mathsf{canDisposeOf}$ represents subordination relationships between individuals $Paul$, $Simon$, and $Caroline$.

Fig. 4 depicts a sub-graph categorizing the workforce of company A where, marked with grey background color, the concept of Employee is defined as the union of Analyst and Programmer. As a consequence, an employee has to be *either* an analyst *or* a programmer[4] and this will play an important role for determining the behavior of the reasoning process: the analysis of the possible cases carried out in the Open World can produce results not achievable with RDBMS and similar Closed World tools. Moreover, individuals created by instantiating the classes described above are linked with the canDisposeOf property to represent subordination relationships between them, as shown by the command chains linking individuals $Paul$, $Simon$, and $Caroline$, marked in Fig. 4 with no background color.

5.1 Querying the Knowledge Base

On this data structure representing the workforce of company A, we can now make queries for selecting individuals according to their roles and mutual relationships. Our sample rule selects individuals for task allocation purposes: admissible candidates must dispose of an Analyst who must in turn dispose of a Programmer. In order to avoid introducing a different formalism (such as RDQL[23] or SPARQL[8]) to query the Knowledge Base, a further class is defined by nesting two class restrictions, as shown in Fig. 5. By doing this, individuals fulfilling the rule described above will be inferred of type QueryClass. The individual $Paul$ is in this context an admissible result since he can dispose of $Simon$ who can in turn dispose of $Caroline$. This result can also be obtained with a traditional RDBMS by representing the information depicted in Fig.4 as a database schema.

As soon as company B starts to interact with the former by participating with their employees in the work flow, a different categorization of the workforce has to be integrated with the one of Fig. 4: ontology mapping is the manual or semi-automated process for achieving this. Let aside the problem of doing a correct lexical mapping between concept names, the critical issue we consider is the under-specification of concepts in either ontology. In our example, the ontology associated with company B is featuring only an Employee class (which can be distinguished by the nsB namespace) with no further specification and the only possible integration is mapping it with the Employee class of company A by means of owl:equivalentClass, as shown in Fig. 6. $Andrea$ is also inserted between $Simon$ and $Caroline$ in order to collaborate, but his status is under-specified because his company doesn't have a semantically-rich enough representation of the workforce. It is clear that, under these new conditions, the rule has no viable result in the traditional, Closed World environment of a database as $Andrea$ cannot be recognized as either an analyst or a programmer.

5.2 Reasoning in the Open World

With regard to our example, the key discrepancy between the semantics of DL and the traditional behavior of an information retrieval system is that, not knowing for sure

[4] She can also be both of them, unless of course the classes are made disjoint with each other, but doing this has no impact on the outcome of our example.

```
<owl:Class rdf:about="http://.../companyA#QueryClass">
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty
          rdf:about="http://.../companyA#canDisposeOf"/>
      </owl:onProperty>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Class rdf:about="http://.../companyA#Analyst"/>
            <owl:Restriction>
               <owl:onProperty
                     rdf:resource="http://.../companyA#canDisposeOf"/>
               <owl:someValuesFrom
                     rdf:resource="http://.../companyA#Programmer"/>
            </owl:Restriction>
          </owl:intersectionOf>
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </owl:equivalentClass>
</owl:Class>
```

Fig. 5. The OWL code defining class QueryClass.

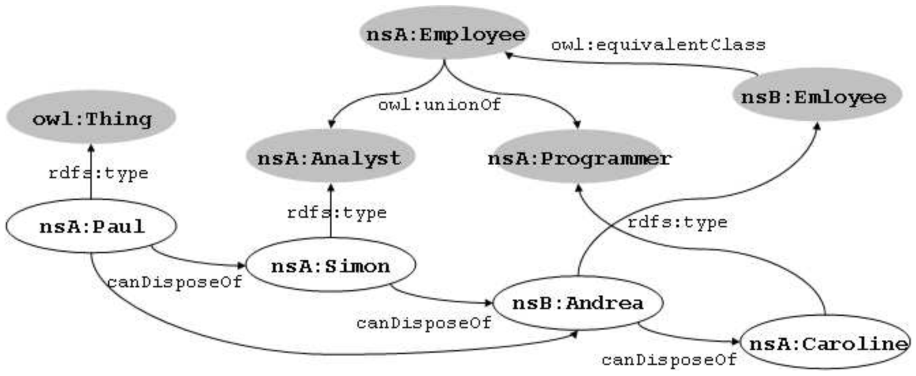

Fig. 6. The mapping of company B with company A: the Employee concepts form both ontologies are made equivalent and the individual $Andrea$ is inserted into the structure.

whether $Andrea$ is of type Analyst or Programmer, both options are examined separately to check, essentially, if this distinction really makes some difference in the evaluation of the entailment of Paul being a member of class QueryClass. The two possible cases are the following:

1. should $Andrea$ be an Analyst, $Paul$ would be a viable result for our query because he can dispose of $Caroline$, which is a Programmer, and then individuals $Paul$, $Andrea$, and $Caroline$ constitute the command chain required by the rule;
2. on the other hand, should $Andrea$ be a Programmer, $Paul$ would be a viable result as well because $Simon$ is an Analyst and can dispose of $Andrea$; this time the command chain is made by $Paul$, $Simon$, and $Andrea$.

Since in both cases $Paul$ would be inferred of type QueryClass, the under-specification of $Andrea$ does not prevent a reasoner from obtaining the correct result. We stress the importance of providing this kind of interpretation on a Knowledge Base because the problem of under-specification is very general within the context of ontology mapping. Moreover, a broad range of applications need to integrate heterogeneous descriptions in the Semantic Web and this can be more easily achieved by reasoning in the Open World. It should also be noted that OWL reasoners are not always compliant with the operational semantics of DL (i.e., implement a Closed World reasoning) and then it is not sufficient to store data as OWL ontologies to reproduce our example's behavior, as explained in [9]. We are currently using RacerPro [12] which has the capability of carrying out the necessary case analysis and produce the correct entailments.

6 Conclusions and Future Work

In this paper we applied theoretical results on the semantics associated with DL to a real-world example of ontology mapping to show the advantages of adopting an Open World approach when reasoning on incomplete information. In open environments, where uniform descriptions are not conceivable, the case analysis carried out by DL reasoners can reduce the impact of under-specification on the evaluation of rules. Whereas this work primarily considers the advantages of using an Open World approach, it is evident that the traditional approach can still be useful in any controlled portion of the Knowledge Base where under-specification is not an issue. Eventually, both visions will probably integrate with each other, as a recent work [18] suggests, and the SWRL extension to the OWL model can already represent this kind of Closed World logic.

References

1. S. Aitken, J. Bradshaw, J. Dalton, A. Uszok, R. Jeffers, M. Johnson, and Tate A. KAoS Policy Management for Semantic Web Services. *IEEE Intelligent Systems*, 19(4):32–41, 2004.
2. Grigoris Antoniou, Enrico Franconi, and Frank van Harmelen. *Introduction to Semantic Web Ontology Languages*. Springer, 2005.
3. Vijayalakshmi Atluri and Li Qin. Concept-level access control for the Semantic Web. In *Proceedings of the 2003 ACM workshop on XML security*, 2003.

4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
5. Harold Boley, Mike Dean, Benjamin Grosof, Ian Horrocks, Peter F. Patel-Schneider, and Said Tabet. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, 2004.
6. R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: when and how to use a KL-ONE-Like language. In *Principles of Semantic Networks: Explorations in the Representation of knowledge*, pages 401–456. Morgan Kaufmann Publishers, 1991.
7. Jeffrey Bradshaw, M. Lalana Kagal, Rebecca Montanari, and Alessandra Toninelli. Rule-based and ontology-based policies: Toward a hybrid approach to control agents in pervasive environments. In *Proceedings of the ISWC2005 Semantic Web and Policy Workshop*, 2005.
8. RDF Data Access Working Group. SPARQL Query Language for RDF, 2004. `http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/`.
9. Stefano David and Enrico Franconi. Semantic Web Query Tools Evaluation. In *Proceedings of Semantic Web Application and Perspective, 1st Italian Semantic Web Workshop*, 2004.
10. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Foundation of Knowledge Representation*, pages 191–236. CSLI-Publications, 1996.
11. Tim Finin, Lalana Kagal, and Anupam Joshi. Declarative policies for describing web service capabilities and constraints. In *W3C Workshop on Constraints and Capabilities for Web Services*, 2004.
12. V. Haarslev and R. Möller. Description of the racer system and its applications. In *Proceedings of International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August*, pages 131–141, 2001.
13. I. Horrocks. FaCT and iFaCT. In *Proceedings of the International Workshop on Description Logics*, pages 133–135, 1999.
14. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
15. Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. of Web Semantics*, 1(4):345–357, 2004.
16. Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.
17. Ian Horrocks and Sergio Tessaris. Querying the semantic web: a formal approach. In Ian Horrocks and James Hendler, editors, *Proc. of the 13th Int. Semantic Web Conf. (ISWC 2002)*, number 2342 in Lecture Notes in Computer Science, pages 177–191. Springer-Verlag, 2002.
18. Yann Loyer and Umberto Straccia. Any-World Assumptions in Logic Programming. *Theoretical Computer Science*, 342:351–381, 2005.
19. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview, 2004. `http://www.w3.org/TR/owl-features/`.
20. Bijan Parsia, Evren Sivrin, Mike Grove, and Ron Alford. Pellet OWL Reasoner, 2003. Maryland Information and Networks Dynamics Lab `http://www.mindswap.org/2003/pellet/`.
21. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax, 2004. `http://www.w3.org/TR/owl-semantics/`.
22. Andrea Schaerf. *Query answering in concept-based knowledge representation systems: algorithms, complexity, and sematic issues*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1994.

23. Andy Seaborne. RDQL - A Query Language for RDF, 2004. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/.

A Full Code of the Example

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:nsB="http://.../companyB#"
  xmlns:nsA="http://.../companyA#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="file://.../example.owl#"
  xml:base="file://.../example.owl">
  <owl:Ontology rdf:about="file://.../example.owl"/>
  <owl:Class rdf:about="http://.../companyA#QueryClass">
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty
            rdf:about="http://.../companyA#canDisposeOf"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
               <owl:Class rdf:about="http://.../companyA#Analyst"/>
               <owl:Restriction>
                  <owl:onProperty>
                     rdf:resource="http://.../companyA#canDisposeOf"/>
                  <owl:someValuesFrom
                     rdf:resource="http://.../companyA#Programmer"/>
               </owl:Restriction>
            </owl:intersectionOf>
          </owl:Class>
        </owl:someValuesFrom>
      </owl:Restriction>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://.../companyA#Employee">
    <owl:equivalentClass>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="http://.../companyA#Analyst"/>
          <owl:Class rdf:about="http://.../companyA#Programmer"/>
        </owl:unionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:about="http://.../companyB#Employee">
    <owl:equivalentClass rdf:resource="http://.../companyA#Employee"/>
  </owl:Class>
  <nsA:Programmer rdf:about="http://.../companyA#Caroline"/>
  <nsB:Employee rdf:about="http://.../companyB#Andrea">
    <nsA:canDisposeOf rdf:resource="http://.../companyA#Caroline"/>
  </nsB:Employee>
  <owl:Thing rdf:about="http://.../companyA#Paul">
    <nsA:canDisposeOf rdf:resource="http://.../companyB#Andrea"/>
    <nsA:canDisposeOf>
      <nsA:Analyst rdf:about="http://.../companyA#Simon">
        <nsA:canDisposeOf rdf:resource="http://.../companyB#Andrea"/>
      </nsA:Analyst>
    </nsA:canDisposeOf>
  </owl:Thing>
</rdf:RDF>
```