Proceedings

International Workshop on Ontology Dynamics (IWOD-07)

Held as part of the 4th European Semantic Web Conference (ESWC-07)

Edited by:
Giorgos Flouris, Mathieu d'Aquin

http://kmi.open.ac.uk/events/iwod

June 7, 2007
Innsbruck, Austria

Organization

The International Workshop on Ontology Dynamics (IWOD-07) was organized as part of the 4th European Semantic Web Conference (ESWC-07). It was held on June 7, 2007, in Innsbruck, Austria.

IWOD-07 website: http://kmi.open.ac.uk/events/iwod
ESWC-07 website: http://www.eswc2007.org

Organizing Committee

Giorgos Flouris (*CNR, Italy*) – co-chair
Mathieu d'Aquin (*The Open University, United Kingdom*) – co-chair
Grigoris Antoniou (*FORTH, Greece*)
Jeff Z. Pan (*University of Aberdeen, United Kingdom*)
Dimitris Plexousakis (*FORTH, Greece*)

Program Committee Members

Peter Haase (*University of Karlsruhe, Germany*)
Christian Halaschek-Wiener (*University of Maryland, USA*)
Andreas Herzig (*Université Paul Sabatier, France*)
Zhisheng Huang (*Vrije Universiteit Amsterdam, The Netherlands*)
Yarden Katz (*University of Maryland, USA*)
Thomas Meyer (*NICTA, Australia*)
Enrico Motta (*The Open University, United Kingdom*)
Wim Peters (*University of Sheffield, United Kingdom*)
Peter Plessers (*Vrije Universiteit Brussels, Belgium*)
Guilin Qi (*University of Karlsruhe, Germany*)
Heiner Stuckenschmidt (*University of Mannheim, Germany*)
Holger Wache (*University of Applied Sciences Northwestern Switzerland, Switzerland*)
Renata Wassermann (*Universidade de São Paulo, Brazil*)

External Reviewers

Richard Booth (*Mahasarakham University, Thailand*)
Kevin Lee (*NICTA, Australia*)
Johanna Völker (*University of Karlsruhe, Germany*)

Preface

One of the crucial tasks to be performed towards the realization of the vision of the Semantic Web is the encoding of human knowledge in ontologies using formal representation languages. Simply creating an ontology is not enough though; ontologies, just like any structure holding knowledge, need to be updated for several reasons, including a change in the world being modeled, a change in users' needs, the acquisition of knowledge previously unknown, classified or otherwise unavailable or a design flaw in the original conceptualization. In all these cases, the representation of our knowledge in the ontology should be modified so as to form a more accurate or adequate conceptualization of the domain. Such a modification presents several difficulties from both the practical and the theoretical point of view, as it is not always clear what the expected, or desired, result of any particular modification should be, nor how such a result can be determined.

The purpose of the International Workshop on Ontology Dynamics (IWOD-07) was to bring together researchers interested in the field of ontology dynamics in order to discuss and analyze important characteristics, open research issues and recent research developments on the field.

We received 14 submissions on various topics related to ontology dynamics; the material collected in this volume is the result of a careful evaluation process, which selected 8 contributions for presentation in the workshop and inclusion in these proceedings.

We would like to thank our fellow organizers, Grigoris Antoniou, Jeff Pan and Dimitris Plexousakis for their help in making this workshop a reality, the Program Committee members and external reviewers for their excellent reviews which ensured that the best possible material was presented in the workshop and appears in these proceedings, and, of course, the organizers of the ESWC-07 conference for supporting this workshop. Last but not least, we would like to thank the authors of all submitted papers, as well as all the participants of the workshop for their interest in IWOD-07.

Giorgos Flouris
Mathieu d'Aquin
(editors)

June 2007

Table of Contents

Ontology Evolution Analysis with OWL-MeT

Natalya Keberle[1], Yuriy Litvinenko,
Yuriy Gordeyev, Vadim Ermolayev

Dept. of IT, Zaporozhye National University,
Ukraine
nkeberle@gmail.com, jurlit@rambler.ru,
ygchaos@mail.ru, vadim@ermolayev.com

Abstract. Works on ontology versioning pay special attention to the logical analysis of ontology evolution. The paper considers extensible declarative approach to ontology change description. Metric temporal description logic with metric temporal modalities "*future n*" and "*past n*" and hybrid satisfaction operator @ is proposed as the logical basis for declarative ontology evolution analysis. Underlying time structure assumed to be linear and discrete, which is acceptable for modeling of ontology versions sequence. Introduced is OWL-MeT – metric extension of the Web ontology language OWL, which is supported with a reasoning engine under development on the basis of Pellet reasoner.

1 Introduction

Ontology versioning and change detection are one of the Semantic Web research challenges [1]. Many previous efforts at providing ontology versioning have focused on the differentiation of the conceptual and explication changes [2], on the identification of an ontology in the Semantic Web [2], [3], on ontology change operations and their effects at the instances level [3], [4]. Change detection between similar ontologies or between versions of the same ontology when there is no version log is discussed in [5], [6]. Change detection and propagation between versions of the same ontology, when there is a version log, is investigated in [7]. Special attention is paid to the change management organization for distributed and modular ontologies [8], [9], inconsistent ontologies [10].

The approach reported in the paper focuses on the analysis of ontology evolution, namely, compatibility and interchangeability of ontology versions, and proposes a formal declarative basis for such analysis. Indeed, an ontology is a formal theory [11], and for any two versions of the same ontology it is interesting to know whether these versions are compatible from the logical point of view, whether it is possible to use definition of an object in one version to access instances of that object in another version.

[1] The work of the author is partially supported in frame of PSI project. Performance Simulation Initiative (PSI) is the R&D project of Cadence Design Systems, GmbH.

Presented research aims at the development of logical means which will facilitate the various tasks of ontology evolution analysis. It is proposed to use temporal logic, and to combine temporal logic with explicit use of metric properties of time. Propositional metric temporal language (first introduced by A.Prior [12]) and calculus (investigated in [13]) serve as the basis for the presented research. Constructive proofs of soundness, completeness and decidability of propositional metric temporal calculus [13] are based on the tableau technique.

The contributions of the paper are: (i) further development of proof-theoretic approach – introduction of metric temporal description logic $\mathcal{ALCIO}(\mathcal{MT})$ with additional sort of nominals representing ontology versions; (ii) introduction of OWL-MeT – metric extension of OWL with constructs of $\mathcal{ALCIO}(\mathcal{MT})$ in a way suitable for the Semantic Web applications; (iii) introduction of reasoning engine for OWL-MeT, based on Pellet[2] – the open-source OWL-DL reasoner in Java.

The paper is organized as follows: the Section 2 presents the related work and the motivation to enhance analytical means for ontology evolution; metric temporal description logic $\mathcal{ALCIO}(\mathcal{MT})$ with metric temporal modalities and hybrid satisfaction operator is described in Section 3; Section 4 introduces OWL-MeT – the extension of OWL with the constructs of $\mathcal{ALCIO}(\mathcal{MT})$ and describes Pellet-based reasoning engine for OWL-MeT; examples of ontology evolution analysis with OWL-MeT are given in the Section 5, and Section 6 presents some conclusions made in frame of the presented research.

2 Related Work and Motivation

Approaches to ontology changes detection and storage basically work on the syntactic layer. In [2] it was proposed to use rule-based approach to change detection. The idea behind the approach is to precisely formulate what can be considered as change for every syntactical construct of ontology description language. The complete list of possible change operations for OWL-Lite was introduced in [3] in the form of "the ontology of ontology change", which is used to mark-up change logs. The framework for ontology change management, OntoView, was described in [5], [3]. Some disadvantages of the rule-based approach to change detection are investigated in [7]. It was shown that in case of multiple changes occurred in the ontology some changes may interfere, and in the worst case even cancel one another, as far as change detection rules work independently of each other.

Proof-theoretic approach to ontology evolution analysis, first introduced in MORE [14], [4] overcomes the drawbacks of other approaches. First of all, it is ontology language-independent: meta-level temporal language LTLm was used for changes analysis. Secondly, reasoning over ontology changes instead of querying changes was proposed, which allowed to rely not only on heuristics/rules of change detection, but to deduce complex changes. Finally, usage of temporal logic is natural as far as changes are characterized with time moments, when they occur.

[2] Pellet homepage is http://pellet.owldl.com

However, analysis of [14] has shown that the usage of temporal logic LTLm has some drawbacks. LTLm is past-oriented, therefore complex statements binding future and past moments can not be described. Further, statements about states of affairs in two ontology versions which are at a given distance from each other, without naming these versions are impossible. Finally, the developers of MORE [4] have outlined for the future work the convenience of explicit usage of version names in a temporal formula, i.e. hybridization of LTLm.

Interesting ideas for the detailed change analysis of both the extensional and the structural levels are proposed in [15] for evolving collections of XML documents. The authors of [15] provide a classification of query types for the analysis of structural changes occurred in the versions of an XML document: structural projection, historical queries, temporal selection, content-based selection. Use of these query types might seriously enhance the ontology versioning frameworks. The steps in this direction are made in the SEKT[3] project [4].

The review of the systems and the approaches presented above shows that detailed analysis of ontology evolution can be facilitated with the two components: (i) – there are either all versions of the ontology or the version log for that ontology, and (ii) – reasoning over the versions is performed with help of temporal logics. The motivation of the approach proposed in the paper is based on the idea of an ontology versioning system facilitating the analysis of evolution. Following [14] and [15] possible analytical queries may include:

- **Reasoning** queries:
  - (non-)derivability of a fact in one version with respect to another version, i.e. whether the fact belongs to the intersection/difference of ontology models
  - (non-)derivability of a fact in a given version
- **Meta-level ontology-specific** queries:
  - compatibility of versions (e.g. "Is the version 1 is subsumed by the version 5?")
  - compatibility of concept definitions (e.g. "Is the concept *Child* defined in the version 1 subsumed by the concept *Child* taken from the version 5?") etc.
  - most common part of concept definitions taken from different version (e.g. "What properties remain unchanged in the concept *Child* from the version 5 to the version 10") etc.
  - detection of conceptual relations [3] between the versions of ontology (e.g. "What version subsumes the version 5?")
- **Retrieval** queries
  - temporal selection (e.g. "Get the version 5")
  - historical queries (e.g. "What was added in the version 5?", "What was added in the version 5 as compare to the version 2?", "What become obsolete in the version 5 as compare to the version 10?")
  - structural projection (e.g. "Get the definition of the concept *Child* in the version 5", "Get the new/retired/unchanged instances of the concept *Child* in the version 5 as compare to the version 2")

[3] MORE is developed in frame of SEKT project, http://www.sekt-project.org

- content-based selection (e.g. "Check all the versions and get all the concept definitions where role *hasChild* is used", "Check the versions from 5 to 10 and get all the concept definitions where the role *hasChild* is used").

Presented research assumes that all ontology versions are available for analysis. Its usage in the case of presence of a version log goes beyond the scope of the paper.

Mentioned queries explore the concept of time moment (expressed in an ontology version number). Moreover, as the time moments / ontology version numbers are used explicitly in queries, there is a need to have correspondent means in underlying logic formalism for referencing time points explicitly.

Hybrid logics [16], [17] which allow naming particular possible world may provide means for referencing time moments explicitly in a formulae. Metric temporal logics [18] allow to reference time moments at a given distance to the future and to the past.

First introduced in [18], metric modalities "*future n*" and "*past n*" were then investigated by Clifford in [19], where the semantics of the metric temporal language was given. Other temporal modalities (such as Scott's unary temporal operator "next instant" [12] or Kamp's binary temporal operators "*until*" and "*since*" [12]) can be defined via metric ones. Later on metric modalities were extensively learnt in the research on real-time logic, and in the recent time, in the research on distance logics [20]. Composition of hybrid and metric temporal logic may provide means necessary for ontology evolution analysis.

3 Metric Temporal Description Logic

Highly-expressive and decidable Description Logics are of great interest in the Semantic Web applications. Known decidable temporal description logics (see, e.g. [21]) focus on topological properties of time, and provide descriptive means for both interval-based and point-based time structure. Such logics are usually seen as combination of a propositional modal logic (as far as description logics have strict correspondence with propositional multimodal logics) and temporal (therefore, also modal) logic. Expressive power of obtained composed logic depends on the degree of interaction between two modal logics.

Metric description logics are relatively new members of the description logics family. The research on metric description logics was initiated with the development of logic of distances [20]. Since then many classes of such logics are investigated for the purpose of decidability.

Metric temporal description logic $\mathcal{ALCIO}(\mathcal{MT})$ for point-based time structure is the composition of $\mathcal{ALCIO}$ – description logic with role inverses and nominals and propositional metric temporal logic. Particularly, $\mathcal{ALCIO}(\mathcal{MT})$ allows application of temporal and hybrid operators only to concepts (both non-temporal and temporal), and role restrictions are not applied to temporal concepts.

Let A, B denote atomic **non-temporal** concepts, R - atomic role, E, F - complex non-temporal concepts, P - complex role, C, D - complex **temporal** concept, $\{o\}$ - **object** nominal (denoting an individual in some possible world), $\{a\}$ - **temporal** nominal (denoting possible world, e.g. ontology version).

Then the rules presented in the Fig.1 generate complex concepts/roles.

$$
\begin{aligned}
E, F &\rightarrow A \mid top \mid bottom \mid E \sqcap F \mid E \sqcup F \mid \neg E \mid \exists R.E \mid \forall R.E \mid \{o\} \\
P &\rightarrow R \mid P^{-1} \\
C, D &\rightarrow E \mid \{a\} \mid C \text{ intersection } D \mid C \text{ union } D \mid \text{not } C \mid C@\{a\} \mid \text{future n } C \mid \\
&\quad \mid \text{past n } C \mid \text{somefuture } C \mid \text{somepast } C \mid \text{allfuture } C \mid \text{allpast } C
\end{aligned}
$$

Fig. 1. Syntax rules for $\mathcal{ALCIO}(\mathcal{MT})$ concepts/roles construction.

If C and D are temporal concepts, and $\{a\}$ is a temporal nominal, then *C equivalent D*, *C subclassof D*, $C(a)$ are temporal formulae. If φ and ψ are temporal formulae, then φ *union* ψ, φ *intersection* ψ, *not* φ are also temporal formulae.

$\mathcal{ALCIO}(\mathcal{MT})$ is interpreted over Kripke model $\mathrm{M} = < \Delta, dist, \{R_F, R_P\}, I, V >$, where $\Delta = \{\Delta^k, k \in Z\}$ is a set of possible worlds, Δ^k is a set of individuals in *k*-th possible world, $dist : \Delta \times \Delta \rightarrow N \cup \{0\}$ is a metric on Δ, R_F, R_P are accessibility relations, I is an interpretation function, and V is a hybrid valuation function.

Interpretation I associates with each Δ^k an $\mathcal{ALCIO}$-interpretation $I(k) = < \Delta^k, \cdot^{I(k)} >$.

Temporal nominals are used as identifiers of ontology versions. Satisfaction operator $@\{a\}$, where $\{a\}$ is considered as an identifier of an ontology version, is adopted from hybrid logics [17].

Additionally, let function $den : \{a\} \rightarrow Z$ be an encoding of temporal nominals into integers. Then, given temporal nominal $\{a\}$, *hybrid valuation* V assigns $\{a\}$ a unique world $\Delta^{den(a)}$ - singleton subset of Δ.

The model-theoretic semantics of $\mathcal{ALCIO}(\mathcal{MT})$-specific operators is defined as follows:

$$
\begin{aligned}
(\text{future n } C)^{I(k)} &= \{o \in \Delta^k : \exists j = k + n, o \in C^{I(j)}\} \\
(\text{past n } C)^{I(k)} &= \{o \in \Delta^k : \exists j : k = j + n, o \in C^{I(j)}\} \\
(\text{somefuture } C)^{I(k)} &= \{o \in \Delta^k : \exists j \geq k, o \in C^{I(j)}\} \\
(\text{somepast } C)^{I(k)} &= \{o \in \Delta^k : \exists j \leq k, o \in C^{I(j)}\} \\
(\text{allfuture } C)^{I(k)} &= \{o \in \Delta^k : \forall j \geq k, o \in C^{I(j)}\} \\
(\text{allpast } C)^{I(k)} &= \{o \in \Delta^k : \forall j \leq k, o \in C^{I(j)}\} \\
(C@\{a\})^{I(k)} &= \{o \in C^{I(den(a))}\}
\end{aligned}
$$

For the purposes of ontology evolution analysis we restrict the domain Δ to be a finite linear sequence of ontology versions – *time structure*, or in terms of [14] *version space*. *Time structure* is a finite sequence of temporal nominals, each of which identify particular ontology version. *Time structure* is ordered with the precedence relation, set between encodings of temporal nominals: if $den(a_1) < den(a_2)$, then the version identified with a_1 precedes the version identified with a_2.

Satisfiability problem for hybrid multi-modal tense logic with nominals, whose syntactic variant is $\mathcal{ALCIO}(\mathcal{MT})$, is EXPTIME-hard [16].

Releasing the domain Δ from being finite leads to analysis of the decidability issues for nonbranching discrete transitive and reflexive unbounded frame. Tableau rule set for hybrid and metric operators provides a ground for constructive proof of decidability of satisfiability problem for $\mathcal{ALCIO}(\mathcal{MT})$. Termination of the decision procedure is controlled with the application of looptest rules (which are applicable for transitive frames, see, e.g. [22]), and of the formulae marking rules (see, e.g. [23]). However, the satifiability problem is still EXPTIME-hard, as the results in [16] were obtained for arbitrary frame. Results obtained in [24] for NP-completeness of satisfiability problem for linear frame (irreflexive, transitive and trichotomous) seem not to be applicable here, as the behaviour of metric operators "*future n*" / "*past n*" when *n = 0* requires the accessibility relations to be reflexive.

4 OWL-MeT

OWL-MeT (abbreviation for OWL-MetricTime) is built on top of the language OWL; it has been assigned namespace *owlmet*. Main constructs for definition of temporal concepts of OWL-MET are **TClass** and **TRestriction**. **TClass**, representing named temporal concepts, is the direct subclass of *rdfs:Class*, **TRestriction**, representing unnamed temporal restrictions, is the subclass of *owlmet:TClass*. Standard OWL class *owl:Class* is defined as subclass of *owlmet:TClass*, therefore all OWL concepts are also OWL-MeT temporal concepts. Indeed, each non-temporal concept *A* (which is the instance of *owl:Class*) can be considered as equivalent to an instance of unnamed temporal restriction *future 0 A*. In OWL-MeT temporal concepts are allowed to form unions, intersections and complements, to define axioms of equivalence and subsumption between them.

OWL-MeT introduces the special sort of nominals for ontology versions – temporal nominals, or instants. **Instant** is also the subclass of *owlmet:TClass*. **TimeStructure** construct fixes the sequence of instants and it is defined with help of *rdf:sequence* construct.

Metric temporal operators are defined as instances of *rdf:property*, namely "**future**", "**allfuture**", "**past**", "**allpast**", "**somefuture**", "**somepast**"; hybrid satisfaction operator "**at**", and its supplementary operator "**happens**" are also the instances of *rdf:property*.

The abstract syntax of OWL-MeT extends the OWL abstract syntax, OWL-MeT new constructs are presented in the Fig. 2. Complete definition[4] of OWL-MeT includes abstract syntax definition, mapping to RDF graphs and usage examples.

The reasoning engine for OWL-MeT is now under development. It is grounded on the open-source Java-based OWL-DL reasoner Pellet. Jena 2.4[5] is used for the validation of OWL-MeT constructs and for correspondent RDF models building.

[4] OWL-MeT Web site: http://ermolayev.com/owl-met

axiom ::=	temporalAxiom | owlAxiom
temporalAxiom::=	'TClass(' **classID modality {annotation} {description}**')' |
	| 'DisjointClasses(' **description description { description }** ')' |
	| 'EquivalentClasses(' **description { description }** ')' |
	| 'SubClassOf(' **description description** ')' |
	| 'EnumeratedClass(' **classID { annotation } { individualID** }')' |
	| 'TimeStructure(' **instantID {instantID}** ')'
modality::=	'complete' | 'partial'
description::=	**classID** |
	| **trestriction** |
	| **resriction** |
	| 'unionOf(' **{ description }** ')' |
	| 'intersectionOf(' **{ description }** ')' |
	| 'complementOf(' **description** ')' |
	| 'oneOf(' **{ individualID }** ')'
trestriction::=	'TRestriction(' **timeProperty** ')'
timeProperty	'allfuture(' **description** ')' | 'somefuture(' **description** ')' |
	| 'allpast(' **description** ')' | 'somepast(' **description** ')' |
	| 'future(' **non-negative-integer** ')' **modality description** |
	| 'past(' **non-negative-integer** ')' **modality description** |
	| 'at(' **instantID** ')' **modality description**

Fig. 2. Abstract syntax of OWL-MeT (the added part as compare to OWL).

Extension of Pellet to reason over OWL-MeT descriptions requires also reworking of normalization and internalization procedures, and extending of the decision procedure with metric and hybrid tableau rules.

In short, hybrid extension of tableau rules creates for each temporal nominal $\{a\}$ presented in a given OWL-MeT formula a particular tableau, and establishes accessibility relations between these tableaux depending on values of $den(a)$. Metric extension of tableau rules describes as movement across the tableaux sequence using that accessibility relation, as well as creation of a particular tableau. For example, operator "*future 1 C*" checks if the tableau at the distance 1 to the future exists, if not - creates that tableau and establishes accessibility relation R_F between the initial tableau and that new. Operator "*future n C*" is considered as "*future 1 future (n-1) C*".

Operators "*allfuture C*" and "*allpast C*" due to the transitivity of time copy *C* to all tableaux accessible via accessibility relation R_F (R_P) from the given one. Operators "*somefuture C*" and "*somepast C*" create alternating tableau branches (due to the reflexivity of the model M): one branch will include *C*, and the other will include "*future 1 somefuture C*"/ "*past 1 somepast C*" respectively. Tableau termination is controlled with the help of adopted looptest rules, avoiding the introduction of both $\mathcal{ALCIO}$ -node fully contained in a predecessor $\mathcal{ALCIO}$ - node, and the introduction of a temporal nominal node fully contained in a predecessor node.

[5] Jena is the open-source Java framework for building Semantic Web applications. It provides a programmatic environment for RDF(S), OWL, SPARQL and includes a rule-based inference engine. Available at http://jena.sourceforge.net

Consider several examples of temporal concept definitions. Let concept "PerpetuumMobile" be defined as some engine that will work always in the future. Correspondent OWL-MeT definition with the non-metric operator "*allfuture*" may be as shown on the Fig. 3. Statements about a temporal concept, relating both future and past moments use metric operators "*future n*" and "*past n*": for example, the definition of "Student" as "UniversityEntrant" in one moment before is shown on the Fig. 4. Finally, statements with explicit naming of time moments use hybrid satisfaction operator "*at*". Mapping to RDF(S) due to certain limitations of RDF(S) had required the introduction of supplementary operator, which is called "*happens*" and which allows to partition temporal concepts occurring at different points. The definition of "HappyFather" (see Fig.5) is given as the union of "HappyFatherInXXCentury" and "HappyFatherInXXICentury", where the disjuncts are defined at different time moments. Time moments "XXCentury" and "XXICentury" are the individuals of the *owlmet:TInstant*.

```
<owlmet:TClass rdf:ID="PerpetuumMobile">
 <rdfs:subClassOf>
 <owlmet:TRestriction>
 <owlmet:allfuture>
  <owlmet:TClass>
    <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Engine"/>
    <owl:Restriction>
    <owl:allValuesFrom> <owl:Class rdf:about="#Thing"/> </owl:allValuesFrom>
    <owl:onProperty> <owl:ObjectProperty rdf:about="#works"/> </owl:onProperty>
    </owl:Restriction>
    </owl:intersectionOf>
 </owlmet:TClass>
 </owlmet:allfuture>
 </owlmet:TRestriction>
 </rdfs:subClassOf>
</owlmet:TClass>
```

Fig. 3. Definition of “PerpetuumMobile”.

```
<owlmet:TClass rdf:ID="Student">
 <rdfs:subClassOf>
      <owlmet:TRestriction>
      <owlmet:past rdf:datatype="&xsd;#NonNegativeInteger"> 1</owlmet:past>
      <owlmet:equivalentClass> <owlmet:TClass rdf:about="#UniversityEntrant"/>
      </owlmet:equivalentClass>
      </owlmet:TRestriction>
      </rdfs:subClassOf>
</owlmet:TClass>
```

Fig. 4. Definition of “Student”.

```
<owlmet:TClass rdf:ID="HappyFather">
<owlmet:equivalentClass>
<owlmet:TClass> <owlmet:unionOf rdf:parseType="Collection">
<owlmet:TRestriction> <owlmet:at rdf:resource="#XXthCentury"/> <owlmet:happens>
<owlmet:TClass rdf:about="#HappyFatherInXXthCentury"/>
</owlmet:happens>
</owlmet:TRestriction>
<owlmet:TRestriction> <owlmet:at rdf:resource="#XXIthCentury"/>
<owlmet:happens><owlmet:TClass rdf:about="#HappyFatherInXXIthCentury "/>
</owlmet:happens>
</owlmet:TRestriction>
</owlmet:unionOf></owlmet:TClass>
</owlmet:equivalentClass></owlmet:TClass>
```

Fig. 5. Definition of "HappyFather".

5 Ontology Evolution Analysis with OWL-MeT

Let us show how OWL-MeT may serve for the task of ontology evolution analysis. Recall that it was proposed in the Section 2 to differentiate three types of analytical queries: retrieval queries (or, queries on the structure of vocabulary), meta-level ontology queries and reasoning queries.

Reasoning queries answer the questions about derivability of a certain fact across versions. For example, to answer the question "Are individuals of concept *C* in a version *v5* derivable both in versions *v2* and *v10*?" one may check satisfiability of a temporal formula (*C* @{*v5*}) *subclassof* (*C* @{*v2*} *intersection C* @{*v10*}).

The question "What are new individuals of concept *C* in a version *v5*, which were not present two versions before" may be answered by checking satisfiability of a temporal concept (*C intersection* ((*past 2*) *not C*)) @{*v5*}.

The question "What are individuals of concept *C* in a version *v5*, which are not derivable at a distance of two versions in the future?" may be answered by checking satisfiability of correspondent temporal concept (*C intersection* ((*future 2*) *not C*)) @{*v5*}. These examples show the benefits of descriptive means of $\mathcal{ALCIO}(\mathcal{MT})$, as compare to known formalisms, e.g. [4].

To discuss the application of OWL-MeT to meta-level ontology specific queries recall that an ontology *O* is a set of terminological and assertional axioms [25], and satisfiability checking of a knowledge base is equivalent to satisfiability checking of a concept *G*, equivalent to the conjunction of definitions of all concepts and individuals taken from *O*[6]. From the other side, meta-level queries have obvious relationship with the theory of modular ontologies [9]. Indeed, ontology *versions* in their usual form are ontology *modules* without concepts, defined externally.

[6] Internalization procedure is applicable to logics allowing the definition of a universal role [25]; OWL-Lite and OWL-DL as the languages of ontology version definition possess role-forming operators – union and reflexive transitive closure – necessary to construct a universal role.

Let O_i be an ontology, actual at the version i, and let G_i be a concept, equivalent to the conjunction of definitions of all concepts and individuals in O_i. Such G_i may be considered as the concept, defined *internally* [9] in an ontology version (or module) i. Then simple checking of the fact that O_i in the version i is satisfiable may be written as G_i @{i}.

Checking satisfiability of a particular concept E taken from an ontology version i in another version, say j, may be executed in several steps. Initially, introduce a *compiled* concept $G_{E,i}$ – conjunction of all explicit and implicit definitions of concepts and individuals, used to define E in O_i . This can be done, e.g. with the help of compilation algorithm, defined in [9]. Then check satisfiability of concept (G_j *intersection* $G_{E,i}$) @{j}.

Analogously, given the whole ontology O_i, checking of the fact that in another version, j (where $j \neq i$), the ontology O_i is satisfiable may be written as (G_j *intersection* G_i) @{j}.

Retrieval queries serve for analysis of the structure of vocabulary, used to define an ontology version, and for analysis of structural changes between ontology versions. Some of the retrieval queries, such as temporal selection, may be formulated without usage of temporal logic. However, retrieval queries benefit from tight relationship with the underlying ontology language. A collection of interesting retrieval query templates, such as "NewChildren", "ObsoleteChildren" etc. in [14], or, ideally, proper ontology of ontology change operators, such as in [3] may serve as the source of various templates. More detailed analysis might require extension of $\mathcal{ALCIO}(\mathcal{MT})$ with role restrictions on temporal concepts.

6 Conclusions and Future Work

The paper proposes an enhancement of the formal apparatus for ontology evolution analysis, and introduces metric temporal description logic $\mathcal{ALCIO}(\mathcal{MT})$, which attempts to release the restrictions of known logical approaches. Reasoning support of OWL-MeT – the extension of Web Ontology language OWL with metric temporal operators proposed in $\mathcal{ALCIO}(\mathcal{MT})$ is now under development on the basis of open-source reasoner Pellet.

Future work is seen in following directions: encoding of interesting change operations, such as "RestrictRange", "ExtendRange", "RestrictDomain", "ExtendDomain", "ModifyEquivalenceToSubclass" etc., taken e.g. from [3], and adopted to OWL DL, in OWL-MeT; development and implementation of optimization strategies of reasoning; and, finally, testing the reasoner services on real cases.

7 Acknowledgements

Authors would like to thank anonymous reviewers for their helpful comments.

References

1. Research Challenges and Perspectives of the Semantic Web, EC-US NSF Strategic Research Workshop, Sophia Antipolis, France, October 3-5 (2001)
2. Klein M.C.A., Kiryakov A., Ognyanov D., Fensel D.: Finding and Characterizing Changes in Ontologies. In: Proc. of the 21st Int. Conf. on Conceptual Modeling (ER'2002), Tampere, Finland, October 7-11. LNCS 2503 (2002) 79–89
3. Klein M.C.A.: Change Management for Distributed Ontologies. Ph.D. Thesis, Aug. 2004, ISBN 90-9018400-7.
4. Huang Z., Stuckenschmidt H.: Reasoning with Multi-version Ontologies. EU-IST Integrated Project (IP) IST-2003-506826 SEKT, Deliverable D3.5.1 (2005)
5. Klein M.C.A., Noy N.F.: A Component-Based Framework For Ontology Evolution. In: Proc. of the Workshop on Ontologies and Distributed Systems, IJCAI ’03, Acapulco, Mexico.
6. Noy N.F., Musen M.: PromptDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions. In: Proc. of the 19th Nat. Conf. on Artificial Intelligence (AAAI/IAAI 2002), Edmonton, Alberta, Canada, July 28 – August 1. AAAI Press (2002) 744–750
7. Plessers P., De Troyer O.: Ontology Detection Using a Version Log. In: Proc. of the 4th International Semantic Web Conference (ISWC'2005), Galway, Ireland, November 6–10. LNCS 3729 (2005) 578–592
8. Stojanovic L., Maedche A., Motik B., Stojanovic N.: User-driven ontology evolution management. In: Proc. of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Siguenza, Spain, October 1-4. LNCS 2473 (2002) 285–300
9. Stuckenschmidt H., Klein M.C.A.: Integrity and change in modular ontologies. In: Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI'2003), Acapulco, Mexico, August 9-15. Morgan Kaufmann (2003) 900–905
10. Haase P., van Harmelen F., Huang Z., Stuckenschmidt H., Sure Y.: A Framework for Handling Inconsistency in Changing Ontologies. In: Proc. of the 4th International Semantic Web Conference (ISWC'2005), Galway, Ireland, November 6–10. LNCS 3729 (2005) 353–367
11. Guarino N.: Formal Ontology and Information Systems. In: Proc. of the 1st Int. Conf. Formal Ontology in Information Systems (FOIS'1998), Trento, Italy, June 6–8. IOS Press/Ohmsha (1998) 3–15
12. Prior A.N.: Past, Present and Future. Oxford University Press, Oxford (1967)
13. Keberle N. G.: Properties of Propositional Metric Temporal Calculus for Description of Evolving Conceptualizations. In: Theses of the 3rd Int. Conf. on Mathematical Support and Software for Artificial Intelligence (MSSAI’2005), Dnepropetrovsk, Ukraine, November 16–18 (2005) 65–66
14. Huang Z., Stuckenschmidt H.: Reasoning with Multi-version Ontologies: A Temporal Logic Approach. In: Proc. of the 4th International Semantic Web Conference (ISWC'2005), Galway, Ireland, November 6–10. LNCS 3729 (2005) 398–412
15. Chien S.-Y., Tsotras V., Zaniolo C.: Efficient Management of Multiversion Documents by Object Referencing. In: Proc. of the 27th VLDB Conference, Roma, Italy, (2001)
16. Areces C., Blackburn P., Marx M.: A roadmap on the complexity of hybrid logics. In: Flum J. and Rodriguez-Artalejo M. (eds.): Computer Science Logic, LNCS 1683. Springer (1999) 307–321
17. Blackburn P.: Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto. In: Areces C., Franconi E., Goré R., de Rijke M. and Schlingloff H. (eds.): Special Issue of the Logic Journal of the IGPL 8:3 (2000) 339–625
18. Prior A.N.: Stratified Metric Tense Logic. Theoria 33 (1967) 28–38

19. Clifford J.E.: Tense and Tense Logic. Mouton Publishers. The Hague, The Netherlands (1975)
20. Kutz O., Wolter F., Zakharyaschev M.: A Note on Concepts and Distances. Working Notes of the 2001 Intl. Description Logics Workshop (DL-2001), Stanford, CA, USA, August 1-3 (2001)113–121
21. Artale A., Franconi E.: Temporal Description Logics. In: Vila L. et al, van Beek P., Boddy M., Fisher M., Gabbay D., Galton A. and Morris R. (Eds.): Handbook of Time and Temporal Reasoning in Artificial Intelligence. MIT Press (1999)
22. Gasquet O., Herzig A., Sahade M.: Terminating Modal Tableaux with Simple Completeness Proof. In: Advances in Modal Logic, 6 (2006) 167-186
23. Horrocks I., Sattler U.: A Tableaux Decision Procedure for SHOIQ. In: Kaelbling L.P., Saffiotti A. (eds.): Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005. Professional Book Center (2005) 448–4533.
24. Franceschet M., de Rijke M., Schlingloff B.-H.: Hybrid Logics on Linear Structures: Expressivity and Complexity. In: Proc. of 10th Int. Symposium on Temporal Representation and Reasoning and 4th Int. Conf. on Temporal Logic (TIME-ICTL'2003), Cairns, Queensland, Australia, July 8-10. IEEE Computer Society (2003) 166–173
25. Baader F., Calvanese D., McGuinness D., Nardi D. and Patel-Schneider P.F.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003)

Semi-automatic Integration of Learned Ontologies into a Collaborative Framework*

Vít Nováček[1], Loredana Laera[2], and Siegfried Handschuh[1]

[1]Digital Enterprise Research Institute, National University of Ireland, Galway
IDA Business Park, Lower Dangan, Galway, Ireland
E-mail: first_name.last_name@deri.org
[2]Department of Computer Science
University of Liverpool, UK
E-mail: lori@csc.liv.ac.uk

Abstract. The paper presents a novel ontology lifecycle scenario that explicitly takes the dynamics and data-intensiveness of real world applications into account. Changing and growing knowledge is handled by semi-automatic incorporation of ontology learning results into a collaborative ontology development framework. This integration bases mainly on automatic negotiation of agreed alignments, inconsistency resolution, an ontology versioning system and support of natural language generation tools, which alleviate the end-user effort in the incorporation of new knowledge. The architecture of the respective framework and notes on its progressive implementation are presented.

1 Introduction and Motivation

Ontologies on the Semantic Web, especially in case of real world applications, are very likely subject to change given the dynamic nature of domain knowledge. Knowledge changes and evolves over time as experience accumulates – it is revised and augmented in the light of deeper understanding; new facts are getting known while some of the old ones need to be revised and/or retracted at the same time.

This holds especially for scientific domains – we have to incorporate newly discovered facts and possibly change the inappropriate old ones in the respective ontology as the scientific research evolves further. However, even virtually any industrial domain is dynamic – changes typically occur in product portfolios, personal structure or industrial processes, which can all be reflected by an ontology in a knowledge management policy.

In these domains, ontology construction is usually the result of collaboration (which involves cooperation among ontology engineers and domain experts)

* This work has been supported by the EU IST 6th framework's Network of Excellence 'Knowledge Web' (FP6-507482) and partially by Academy of Sciences of the Czech Republic, 'Information Society' national research program, the grant AV 1ET100300419.

through a manual process of the extraction of the knowledge. However, it is not always feasible to process all the relevant data and extract the knowledge from them manually, since we might not have a sufficiently large committee of ontology engineers and/or dedicated experts at hand in order to process new data anytime it occurs. This implies a need for (partial) automation of ontology extraction and management processes in dynamic and data-intensive environments. This can be achieved by ontology learning [15]. Therefore, a lifecycle of an ontology development process apt for universal application in scientific and/or industrial domains should also support appropriate mechanisms for dealing with the large amounts of knowledge that are *dynamic* in nature.

While there has been a great deal of work on ontology learning for ontology construction, e.g. [2], as well as on collaborative ontology development [18], relatively little attention has been paid to the integration of both approaches within an ontology lifecycle scenario. In this paper, we introduce our framework for practical handling of dynamic and large data-sets in an ontology lifecycle, focusing particularly on dynamic integration of learned knowledge into collaboratively developed ontologies. One of the key elements supporting our integration is the ability to reach an agreement on the semantics of the terms used in these ontologies. Since the ontologies are created under different circumstances and conditions and thus might represent different perspectives over similar knowledge, the achievement of an agreement will necessarily come through a (partially automated) negotiation process.

The dynamic nature of knowledge is one of the most challenging problems in the current Semantic Web research. Here we provide a solution for dealing with dynamics in large scale, based on properly developed connection of ontology learning and dynamic collaborative development. We do not concentrate on formal specification of respective ontology integration operators, we focus rather on implementation of them, following certain practical requirements:

1. the ability to process new knowledge (resources) automatically whenever it appears and when it is inappropriate for humans to incorporate it;
2. the ability to automatically compare the new knowledge with a "*master*" ontology that is manually and collaboratively designed and select the new knowledge accordingly;
3. the ability to resolve inconsistencies between the new and current knowledge, possibly favouring the assertions from presumably more complex and precise master ontology against the learned ones;
4. the ability to automatically sort the new knowledge according to user-defined preferences and present it to them in a very simple way, thus further alleviating human efforts in the task of final incorporation of the knowledge.

On one hand, using the automatic methods, we are able to deal with large amounts of changing data. On the other hand, the final incorporation of new knowledge is to be decided by the human users, repairing possible errors and inappropriate findings of the automatic techniques. The key to success and applicability is to let machines do most of the tedious and time-consuming work and provide people with concise and simple suggestions on ontology integration.

The rest of the paper is organized as follows: Section 2 presents a brief discussion of related work. Section 3 gives an overview of our ontology lifecycle scenario and framework, whereas Section 4 presents the integration of manually designed and learned ontologies in more detail. Finally, in Section 5 we give a simple illustrative example of concrete usage of the our integration approach. Section 6 concludes the paper and sums up our future work.

2 Related Work

Recent overviews of the state-of-the-art in ontologies and related methodologies can be found in [9]. However, none of them offers a direct solution to the previously mentioned problems. *Methontology* [8] is a methodology developed for designing ontologies to serve as a base for extending it towards evolving ontologies. The ODESeW and WebODE suite [4] projects provide an infrastructure and tools for semantic application development/management, which is also in the process of being extended for networked and evolving ontologies. However, they focus rather on the application development part of the problem than on the ontology evolution parts.

The above projects have all focused on either a single part of ontology evolution, or on a rather abstract study of the knowledge management cycle. However, mechanisms that would provide a clue on how to incorporate the dynamics into the lifecycle are typically put off only by introduction of the version management, which we find insufficient. Moreover, the need for automatic methods of ontology acquisition in data-intensive environments is acknowledged, but the place of the automatic techniques is usually not distinguished in the dynamic lifecycle settings. The work [10] describes a way of machine-assisted refinement of automatically learnt ontologies. Our approach [17] offers a broader picture of how to deal with the dynamics in a general lifecycle scenario. In this paper we concentrate on the combination of ontology learning and manual (collaborative) development in dynamic settings.

3 DINO – A Dynamic Ontology Lifecycle Scenario

DINO is an abbreviation of three key elements of our ontology lifecycle scenario and framework – *Dynamics*, *INtegration* and *Ontology*. However, the first two can also be *Data* and *INtensive*. All these features express the primary aim of our efforts – *to make the knowledge efficiently and reasonably manageable in data-intensive and dynamic domains*.

Figure 1 below depicts the scheme of the proposed dynamic and application-oriented ontology lifecycle that deals with the problems mentioned in the previous sections.

Our ontology lifecycle builds on four basic phases of an ontology lifecycle: *creation* (comprises both manual and automatic ontology development and update approaches), *versioning*, *evaluation* and *negotiation* (comprises ontology

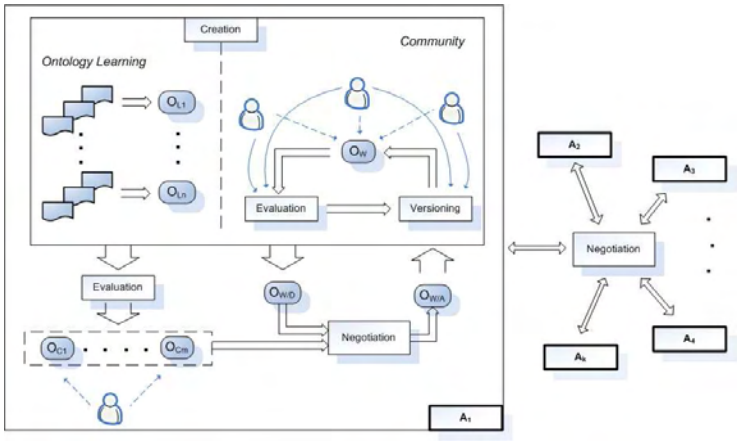

Fig. 1. Dynamics in the ontology lifecycle

alignment and merging as well as negotiation among different possible alignments). The four main phases are indicated by the boxes annotated by respective names. Ontologies or their instances in time are represented by circles, with arrows expressing various kinds of information flow. The A boxes present actors (institutions, companies, research teams etc.) involved in ontology development, where A_1 is zoomed-in in order to show the lifecycle's components in detail.

The general dynamics of the lifecycle goes as follows. The community experts (or dedicated ontology engineers) develop a (relatively precise and complex) domain ontology (the *Community* part of the *Creation* component). They use means for continuous ontology *evaluation* and *versioning* to maintain high quality and manage changes during the development process. If the amount of data suitable for knowledge extraction is too large to be managed by the community, *ontology learning* takes its place. Its results are *evaluated* and partially (we take only the results with quality above a certain threshold into account) integrated into the more precise (but typically relatively small) reference community ontology. The integration is based on alignment and merging covered by the *negotiation* component. Its proposal and implementation principles form the key contribution of this paper (see Section 4 for details). The *negotiation* component takes its place also when interchanging or sharing the knowledge with other independent actors in the field. All the phases support ontologies in the standard OWL format [1], (namely in its OWL DL flavour), although some phases may not support the full range of the syntactic structures available (e.g. natural language generation from triples, as introduced in Section 4.7 here and in [16]). In the following we concentrate on the integration component. More information on other parts of the lifecycle can be found in [17].

4 Dynamic Integration of Newly Learned Knowledge in the DINO Framework

The key novelty of the presented lifecycle scenario is its support for incorporation of changing knowledge in data-intensive domains. This is achieved by implementation of a specific integration mechanism. Its scheme is depicted in Figure 2[1]. The particular components and their connections are described in the following paragraphs.

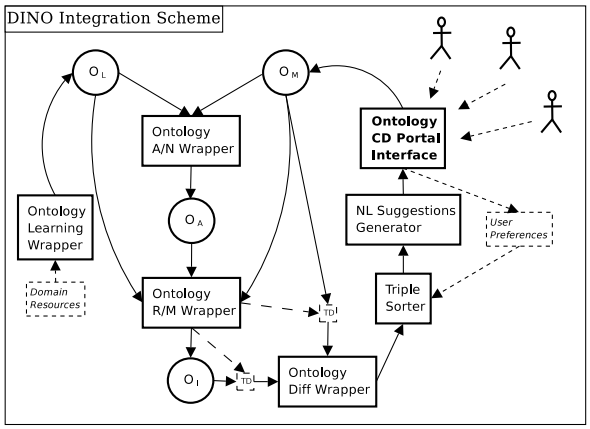

Fig. 2. Dynamic integration scheme

The integration scheme details the usage of generic lifecycle's components – mainly the *negotiation* and *versioning* – in the process of incorporation of learned ontologies into the collaboratively developed ones. However, the generic components serve only as a base for specific wrappers. Each of the phases of integration and their connections are described in the following sections.

4.1 Ontology Learning Wrapper

In this phase, machine learning and NLP methods are used for the processing of relevant resources and extracting knowledge from them (ontology learning). The ontology learning is realised using the Text2Onto framework [3]. We interface the toolbox indirectly within the collaborative ontology development portal based on MarcOnt Portal architecture (see Section 4.2). Configuration of the learning algorithms is set using a special user interface in the portal. The settings is used for batch processing of the new resources fed to the ontology learning component.

[1] The full arrows symbolise data flow in the scheme. Long-dashed arrows indicate production of a triple representation/dump of an ontology (the TD squares). The short-dashed arrows represent user involvement either in ontology production or extension preference sets' definition.

The results of one round of ontology learning – the O_L circle in Figure 2 – are optionally evaluated or refined using the Text2Onto confidence values and passed to the alignment/negotiation wrapper (see Section 4.3).

4.2 Ontology Collaborative Development Portal

The whole integration as well as the DINO framework is based on the MarcOnt Portal architecture [11] for collaborative ontology development. It is a part of a broader initiative aimed mainly at utilisation of various digital library development and maintenance efforts[2].

MarcOnt Portal offers domain-independent means for efficient distributed and collaborative ontology development. It supports features like ontology editing, ontology versioning (supported by the SemVersion system [21]), voting on ontology changes and evaluation of these votes. The elements of DINO realising various parts of the lifecycle are being implemented into the portal's core, with access provided by respective new parts of portal's user and administrative interfaces.

The ontology being developed using the portal's collaborative interfaces is the *master reference ontology* in the whole lifecycle. It is also the source for the deployment of an official version of the ontology. The O_M circle in Figure 2 represents its dump that serves as a reference to be integrated with the O_L ontology resulting from the learning process. The final suggestions (see Section 4.7) form a base for a next version of the O_M ontology submitted after the integration.

4.3 Ontology Alignment/Negotiation (A/N) Wrapper

Once the learned ontology O_L and the master ontology O_M have been created, they need to be reconciled since they cover the same domain, but might be structured differently. The reconciliation of these ontologies depends on the ability to reach an agreement on the semantics of the terms used. The agreement takes the form of an alignment between the ontologies, that is, a set of correspondences (or mappings) between the concepts, individuals and properties in the ontologies. However, the ontologies are developed in different contexts and under different conditions and thus they might represent different perspectives over similar knowledge, so the process by which to come to an agreement will necessarily only come through a negotiation process. The negotiation process is performed using argumentation-based negotiation that uses preferences over the types of correspondences in order to choose the mappings that will be used to finally merge the ontologies (see Section 4.4). The preferences depend on the context and situation. A major feature of this context is the ontology, and the structural features thereof, such as the depth of the subclass hierarchy and branching factor, ratio of properties to concepts, etc. The analysis of the components of the ontology is aligned with the approach to ontology evaluation, demonstrated in [5], and can be formalized in terms of feature metrics. Thus

[2] See `http://www.marcont.org` for details on the whole MarcOnt Initiative.

the preferences can be determined on the characteristics of the ontology. For example, we can select a preference for terminological mapping if the ontology is lacking in structure, or prefer extensional mapping if the ontology is rich in instances.

Thus, the alignment/negotiation wrapper interfaces two tools – one for the ontology alignment discovery and one for negotiation of agreed alignment. We call these tools *AKit* and *NKit*, respectively, within this section. For the former, we use the ontology alignment API [6] developed by INRIA Rhone-Alpes[3]. For the negotiation we use the framework described in [13]. Both tools are used by the wrapper in order to produce O_A – an ontology consisting of axioms[4] merging classes, individuals and properties in the O_L and O_M ontologies. It is used in consequent factual merging and refinement in the ontology reasoning and management wrapper (see Section 4.4 for details).

The wrapper itself works according to the meta-code in Algorithm 1. The

Algorithm 1 Meta-algorithm of the alignment and negotiation

Require: O_L, O_M — ontologies in OWL format
Require: $AKit, NKit$ — ontology alignment and alignment negotiation tools, respectively
Require: $ALMSET$ — a set of the alignment methods to be used
Require: $PREFSET$ — a set of alignment formal preferences corresponding to the O_L, O_M ontologies (to be used in N-kit)
1: $S_A \leftarrow \emptyset$
2: **for** $method \in ALMSET$ **do**
3: $\quad S_A \leftarrow S_A \cup AKit.getAlignment(O_L, O_M, method)$
4: **end for**
5: $A_{agreed} \leftarrow NKit.negotiateAlignment(S_A, PREFSET)$
6: $O_A \leftarrow AKit.produceBridgeAxioms(A_{agreed})$
7: **return** O_A

ontology alignment API offers several possibilities of actual alignment methods, which range from trivial lexical equality detection through more sophisticated string and edit-distance based algorithms to an iterative structural alignment by the OLA algorithm [7]. The ontology alignment API has recently been extended by a method for the calculation of a similarity metric between ontology entities, an adaptation of the SRMetric used in [20]. We also consider a set of justifications, that explain why the mappings have been generated. This information forms the basis for the negotiation framework that dynamically generates arguments, supplies the reasons for the mapping choices and negotiate an agreed alignment for both ontologies O_L and O_M.

4.4 Ontology Reasoning/Management (R/M) Wrapper

This wrapper is used for merging of the O_L and O_M ontologies. It uses Jena 2 Ontology API[5]. It merges the O_L and O_M ontologies according to the statements

[3] See `http://alignapi.gforge.inria.fr/` for up-to-date information on the API.
[4] Using constructs like *owl:equivalentClass*, *owl:sameAs*, *owl:equivalentProperty*, *rdfs:subClassOf* or *rdfs:subPropertyOf*.
[5] See `http://jena.sourceforge.net/ontology/index.html`.

in O_A, preferring the lexical labels from the master O_M ontology when two labels are said to be equivalent. Moreover, the wrapper checks for (some) possible inconsistencies caused by the merging (using Jena's simple OWL DL reasoner) and attempts to resolve them favouring the assertions in the O_M ontology, which are supposed to be more relevant, again. The resulting ontology O_I is passed to the ontology diff wrapper. As the Jena ontology model is internally based on a graph/triple (RDF) structure, it allows to easily export or transform an ontology in a triple format needed for the consequent wrapper (see Section 4.5 for details).

Algorithm 2 describes the meta-code of the process arranged by the ontology merging and reasoning wrapper. The inconsistency resolution is somewhat tricky.

Algorithm 2 Meta-algorithm of the merging and inconsistency resolution

Require: O_L, O_M, O_A — ontologies in OWL format
Require: $getEq()$ — function selecting all assertions of type *owl:equivalentClass*, *owl:sameAs*, *owl:equivalentProperty*
Require: $getRM()$ — function returning wrapper combining a generic ontology manager and (incomplete OWL Full) reasoner bound to the given ontology

1: $O_{tmp} \leftarrow copy(O_L)$
2: $O_I \leftarrow copy(O_M)$
3: $R_M \leftarrow getRM(O_M)$
4: $R_{tmp} \leftarrow getRM(O_{tmp})$
5: $R_L \leftarrow getRM(O_L)$
6: $R_A \leftarrow getRM(O_A)$
7: $R_I \leftarrow getRM(O_I)$
8: $equivalencies \leftarrow \{owl:equivalentClass, owl:sameAs, owl:equivalentProperty\}$
9: $UNIFIED \leftarrow \emptyset$
10: **for** $id \in getEq(O_A)$ **do**
11: $\quad R_{tmp}.replaceLabels(id.O_L, id.O_M)$
12: $\quad UNIFIED \leftarrow UNIFIED \cup id.O_M$
13: **end for**
14: $R_{ref} \leftarrow copy(R_{tmp})$
15: **for** $eq \in R_{tmp}.getAxiomsWithLabels(UNIFIED)$ **do**
16: $\quad R_{tmp}.retractAxioms(eq)$
17: $\quad R_I.addAxioms(eq)$
18: **end for**
19: $R_A.removeAxiomsOfType(equivalencies)$
20: $R_I.addAxioms(R_{tmp}.getAllAxioms())$
21: $R_I.addAxioms(R_A.getAllAxioms())$
22: $R_I.resolveInconsistencies(R_{ref})$
23: $R_I.augmentStructure()$
24: **return** O_I

However, we can apply a sort of "greedy" heuristic, considering the assertions in the master O_M ontology to be more valid. Therefore we query the R_{ref} structure (with the axioms of learned ontology, possibly with replaced labels) in the resolution process. We currently handle the following inconsistencies:

- **sub-class** hierarchy **cycles**: these are resolved by cutting the cycle by removing an *owl:subClassOf* statement present in R_{ref};
- **disjointness-subsumption** conflicts: if classes are said to be disjoint and a sub-class relationship holds between them or if they have common sub-classes at the same time, the conflicting assertion indicated by R_{ref} is removed;

– **disjointness-instantiation** conflicts: if an individual is said to be an instance of classes that are disjoint, the assertion indicated by R_{ref} is removed.

When there are several removal candidate axioms involved in one inconsistency, we sort them according to the confidence provided by the Text2Onto learning algorithms [3], which is stored in the R_{ref} reference structure. Similarly to [10], we start removing the axioms with least overall confidence, until we do not resolve the inconsistency (thus keeping the more "relevant" discoveries intact). We keep the conflicting assertions when they all originate from the O_M master ontology and let the users to cope with this fact. Note that the sources of inconsistencies are provided by simple natural language description and recorded for further examinations by human users – they can eventually decide to favour the learned assertions if appropriate for the given task in the given context.

The function *augmentStructure*() attempts to complete the structure of learned axioms using the more precise and complex knowledge in the O_M master ontology. Currently, augmentation of *owl:subClassOf* and instantiation relations using *rdfs:domain* and *rdfs:range* assertions in property definitions from O_M ontology is taken into account (see Section 5 for an example). More sophisticated extensions are possible in the future.

If we want to include even the "equal" labels from the learned ontology, we can omit the renaming and subtractions in lines 10-16 and 19 and include the respective equality statements from O_A into O_I, together with respective axioms from O_L. The decision depends on users – whether they want to prefer the labels from master ontology or not (e.g. when looking for possible unknown synonyms of important terms from O_M in domain resources; this could be useful for example in the medicine domain in task of identification of different names for the same drugs and/or proteins).

4.5 Ontology Diff Wrapper

Possible extension of a master ontology O_M by elements contained in the merged and refined ontology O_I naturally corresponds to the differences between them. These are discovered by means of the SemVersion library [21], which is interfaced within this wrapper. In particular, the possible extensions are equal to the additions O_I brings into O_M. We compute the additions from the triple-based representation[6] of O_I and O_M ontologies. The additions are passed to the triple sorter then (see Section 4.6 for details).

4.6 Triple Sorter

The addition triples passed to this component form a base to the eventual extension suggestions for the domain experts. However, the number of additions

[6] Since SemVersion does not currently support full OWL diff computations. The triple representation is provided by the ontology R/M wrapper, as indicated by the *TD* (triple dump) squares in Figure 2.

can generally be quite large, so an ordering that takes a relevance measure of possible suggestions into account is needed. Thus we can for example eliminate suggestions with low relevance level when presenting the final set to the users (without overwhelming them with a large number of possibly irrelevant suggestions).

As a possible solution to this task, we have proposed and implemented a method based on string subsumption and Levenshtein distance [14]. These two measures are used within relevance computation by comparing the predicate, subject and object lexical labels of a triple to two sets (S_p, S_n) of words, provided by users. The S_p and S_n sets contain preferred and unwanted words, respectively, concerning the lexical level of optimal extensions.

Relevance score of a triple T with respect to the wanted/unwanted sets of words S_p/S_n ($relScore(T, S_p, S_n)$ below) is given by the formula:

$$relScore(T, S_p, S_n) = rel(T, S_p) - rel(T, S_n),$$

where $rel(T, S)$ is a function measuring the relevance of the triple T with respect to the words in the set S. The higher the value, the more relevant the triple is. The function[7] naturally measures the "closeness" of the T triple's predicate, subject and object labels to the set of terms in S_w (S_w stands for S_p or S_n). The value of 1 is achieved when the label is a direct substring of or equal to any word in S_w or vice versa. When the Levenshtein distance between the label and a word in S_w is lower than or equal to the defined threshold t, the relevance decreases from 1 by a value proportional to the fraction of the distance and t. If this is not the case (i.e. the label's distance is greater than t for each word in S_W), a similar principle is applied for words of the possibly multi-word label and the relevance is further proportionally decreased (the minimal possible value being 0).

4.7 Mapping Triples to Natural Language Suggestions

The DINO framework is supposed to be used primarily by users who are not experts in ontology engineering. Although the MarcOnt Portal [11] already offers a simple ontology editing interface, we would like to further help the user in ontology augmentation by the learned knowledge. Therefore the suggestions are produced in the form of very simple natural language statements. These are obtained directly from the sorted triples passed to this component, using a minor modification of the generation process in CLIE described in [19]. Examples of this final form of suggestions can be found in the next section. The suggestions are still bound to the underlying triples, therefore the user can very easily add the respective OWL axioms into the new version of the O_M master ontology without actually dealing with the intricate OWL syntax itself.

[7] We described the relevance function in more detail in [17, 16], together with complexity analysis (which is in feasible class of $O(m \log(m))$ with respect to the number of triples).

5 Evaluation and Usage Example

The DINO framework is still a work in progress, and thus no proper evaluation has been carried out yet. However, preliminary evaluation of the core negotiation and preference-based suggestion sorting techniques has been made. The implemented sorting algorithm placed 80.7% of triples from a test sample into an order intuitively prepared by a human user. Details on the sorting evaluation are in [17, 16]. The negotiation component has been evaluated using the Ontology Alignment Evaluation Initiative test suite[8]. Experiments on the impact of the argumentation approach over a set of mappings and a comparison wrt. current alignment tools is presented in [12]. The preliminary results of these experiments are promising and suggest that the argumentation approach can be beneficial and an effective solution to the problem of dynamically aligning heterogeneous ontologies.

In the following we provide a simple illustrative example of concrete usage of the DINO integration mechanism. Imagine a medical institution that has developed an ontology O_M covering the basic concepts in clinical practice and research, possibly with help of ontology engineering experts when deploying the DINO framework. The ontology may need to be extended by new information in research (e.g. when new treatments or diagnosis methods are developed and published). Related information can be found in respective documents (research papers, industry white-papers, etc.). Figure 3 presents a sample text fragment with the respective learned OWL O_L ontology (we omit the namespace for simplicity).

... while **cerebellar astrocytoma** *is usually* **discovered by** *means of* **CT**... *using a* **diagnostic procedure** *of* **scanning**... **GVHD**, *an* **immune dysfunction**... *GVHD, a* **disease** *being a type of* **dysfunction**...

```
...
<owl:ObjectProperty rdf:ID="discovered-by"/>
<owl:Thing rdf:ID="CT"/>
<owl:Thing rdf:ID="cerebellar-astrocytoma">
  <discovered-by rdf:resource="#CT"/>
</owl:Thing>
<owl:Class rdf:ID="diagnostic-procedure"/>
<owl:Class rdf:ID="immune-dysfunction"/>
<owl:Class rdf:ID="dysfunction"/>
<owl:Class rdf:ID="scanning">
  <rdfs:subClassOf rdf:resource="#diagnostic-procedure"/>
</owl:Class>
<immune-dysfunction rdf:ID="GVHD"/>
<owl:Class rdf:ID="disease">
  <rdfs:subClassOf rdf:resource="#dysfunction"/>
</owl:Class>
...
```

Fig. 3. A text sample and the learned ontology

The ontologies O_L and O_M are aligned and negotiated (see Figure 4). The preferences have been chosen on the basis of the ontological information of O_L and O_M (see Section 4.3 for details). The O_M ontology and the ontology O_A, consisting of axioms produced from the negotiated mappings are shown in Figure 5. When trying to merge the O_M and O_L ontologies into O_I according

[8] See `http://oaei.ontologymatching.org/`.

Fig. 4. Negotiated mappings

```
...
<owl:ObjectProperty rdf:ID="InstrumentalProperty"/>
<owl:ObjectProperty rdf:ID="DiscoveredUsing">
  <rdfs:subPropertyOf rdf:resource="#InstrumentalProperty"/>
  <rdfs:range rdf:resource="#Manifestation"/>
  <rdfs:domain rdf:resource="#DiagnosisProcedure"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Manifestation"/>
<owl:Class rdf:ID="Procedure"/>
<owl:Class rdf:ID="DiagnosisProcedure">
  <rdfs:subClassOf rdf:resource="#Procedure"/>
</owl:Class>
<owl:Class rdf:ID="SoftTissueCytoma"/>
<owl:Class rdf:ID="AstroCytoma">
  <rdfs:subClassOf rdf:resource="#SoftTissueCytoma"/>
</owl:Class>
<owl:Class rdf:ID="Disease">
<owl:Class rdf:ID="Dysfunction">
  <rdfs:subClassOf rdf:resource="#Disease"/>
</owl:Class>
...
```

```
...
<owl:ObjectProperty rdf:ID="DiscoveredUsing">
  <owl:equivalentProperty rdf:resource="#discovered-by"/>
</owl:ObjectProperty>
<AstroCytoma rdf:ID="cerebellar-astrocytoma"/>
<owl:Class rdf:ID="DiagnosisProcedure">
  <owl:equivalentClass rdf:resource="#diagnostic-procedure"/>
</owl:Class>
<owl:Class rdf:ID="immune-dysfunction">
  <owl:subClassOf rdf:resource="#Dysfunction"/>
</owl:Class>
<owl:Class rdf:ID="Dysfunction">
  <owl:equivalentClass rdf:resource="#dysfunction"/>
</owl:Class>
...
```

Fig. 5. A master ontology sample and the respective mapping

to the technique described in Section 4.4, we find out that there is one inconsistency – "*disease*" is said to be a subclass of "*dysfunction*" and vice versa, which creates a cycle in the taxonomy. Therefore we remove the respective "invalid" assertion that originated from the O_L ontology. On the other hand, we can extend the learned knowledge based on range and domain of the "*DiscoveredUsing*" property. We can infer new assertions on the instantiation of "*cerebellar astrocytoma*" (instance of "*Manifestation*") and "*CT*" (instance of "*DiagnosisProcedure*").

Now we can produce the triples (with O_L equivalent labels replaced by those from O_M) from the O_I merge, together with respective suggestions based on the differences between O_I and O_M. We present the sorted triples and their transformations into natural language statements[9] in Table 1.

Note that the above example may be also used if we just need to align and possibly extend the ontology with another institution's knowledge base – the only difference is that we do not perform the ontology learning and also omit retractions in the integration process, as noted in Section 4.4. This can be applied in the critical task of inter-mediation of medicine information, for example.

[9] They are preceded by sample relevance values, corresponding to {`Scanning, discover, cytoma`} and {`subclass, disease, dysfunction`} sets of preferred and unwanted labels, respectively.

`<AstroCytoma rdf:ID="cerebellar-astrocytoma"/>`	+0.667: CEREBELLAR ASTROCYTOMA is a *new instance* of ASTROCYTOMA.
`<Manifestation rdf:ID="cerebellar-astrocytoma"/>`	+0.667: CEREBELLAR ASTROCYTOMA is a *new instance* of MANIFESTATION.
`<DiagnosisProcedure rdf:ID="CT"/>`	+0.389: CT is a *new instance* of DIAGNOSIS PROCEDURE.
`<immune-dysfunction rdf:ID="GVHD"/>`	+0.333: GVHD is a *new instance* of IMMUNE DYSFUNCTION.

`<owl:Class rdf:ID="scanning">`
`  <rdfs:subClassOf rdf:resource="#DiagnosisProcedure"/>`
`</owl:Class>`
	-0.444: A *new class* SCANNING is a *sub-class* of DIAGNOSIS PROCEDURE.

`<owl:Thing rdf:ID="cerebellar-astrocytoma">`
`  <DiscoveredUsing rdf:resource="#CT"/>`
`</owl:Thing>`
	-0.667: CEREBELLAR ASTROCYTOMA is DISCOVERED USING CT.

`<owl:Class rdf:ID="immune-dysfunction">`
`  <rdfs:subClassOf rdf:resource="#Dysfunction"/>`
`</owl:Class>`
	-0.833: A *new class* IMMUNE DYSFUNCTION is a *sub-class* of DYSFUNCTION.

Table 1. Extension triples and the respective NL suggestions

6 Conclusions and Future Work

We have presented the basic principles of DINO – a novel framework for dynamic ontology development in data-intensive domains. As a core contribution of the paper, we have described the mechanism of integration of learned and collaboratively developed knowledge. It covers all the requirements specified in Section 1. The proposed combination of automatic and collaborative tools in knowledge acquisition, integration and inconsistency resolution ensures production of reliable, broad and precise ontologies when using DINO in dynamic settings.

Our present and future work concentrates mainly on full implementation of the DINO framework by the respective extensions of the MarcOnt Portal architecture. We also plan to continuously evaluate the framework in line with demands of medicine industry (which we are currently specifying with help of e-Health experts) and possibly other domains.

References

1. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. *OWL Web Ontology Language Reference*, 2004. Available at (February 2006): `http://www.w3.org/TR/owl-ref/`.
2. C. Brewster, F. Ciravegna, and Y. Wilks. User-centred ontology learning for knowledge management. In *In Proceedings 7th International Workshop on Applications of Natural Language to Information Systems, Stockholm.*, 2002.
3. P. Cimiano and J. Völker. Text2Onto - a framework for ontology learning and data-driven change discovery. In *Proceedings of the NLDB 2005 Conference*, pages 227–238. Springer-Verlag, 2005.
4. O. Corcho, A. Lopez-Cima, and A. Gomez-Perez. The ODESeW 2.0 semantic web application framework. In *Proceedings of WWW 2006*, pages 1049–1050, New York, 2006. ACM Press.
5. K. Dellschaft and S. Staab. On how to perform a gold standard based evaluation of ontology learning. In *Proceedings of the International Semantic Web Conference. Athens, GA, USA.*, 2006.
6. J. Euzenat. An API for ontology alignment. In *ISWC 2004: Third International Semantic Web Conference. Proceedings*, pages 698–712. Springer-Verlag, 2004.
7. J. Euzenat, D. Loup, M. Touzani, and P. Valtchev. Ontology alignment with ola. In *Proceedings of the 3rd International Workshop on Evaluation of Ontology based Tools (EON)*, Hiroshima, Japan, 2004. CEUR-WS.

8. M. Fernandez-Lopez, A. Gomez-Perez, and N. Juristo. Methontology: from ontological art towards ontological engineering. In *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, pages 33–40, Stanford, USA, March 1997.
9. A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowledge Processing. Springer-Verlag, 2004.
10. P. Haase and J. Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In P. C. G. da Costa, K. B. Laskey, K. J. Laskey, and M. Pool, editors, *Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, pages 45–55, NOV 2005.
11. S. Kruk, J. Breslin, and S. Decker. MarcOnt initiative. Líon Deliverable 3.01, DERI, Galway, 2005.
12. L. Laera, I. Blacoe, V. Tamma, T. Payne, J. Euzenat, and T. Bench-Capon. Argumentation over ontology correspondences in mas. In *In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007), Honolulu, Hawaii, USA. To Appear*, 2007.
13. L. Laera, V. Tamma, J. Euzenat, T. Bench-Capon, and T. R. Payne. Reaching agreement over ontology alignments. In *Proceedings of 5th International Semantic Web Conference (ISWC 2006)*. Springer-Verlag, 2006.
14. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics Control Theory*, 10:707–710, 1966.
15. A. Maedche and S. Staab. Ontology learning. *Handbook on Ontologies*, 2004.
16. V. Nováček, M. Dabrowski, S. R. Kruk, and S. Handschuh. Extending community ontology using automatically generated suggestions. In *Proceedings of FLAIRS 2007*. AAAI Press, 2007. In press.
17. V. Nováček, S. Handschuh, L. Laera, D. Maynard, M. Völkel, T. Groza, V. Tamma, and S. R. Kruk. Report and prototype of dynamics in the ontology lifecycle (D2.3.8v1). Deliverable 238v1, Knowledge Web, 2006.
18. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative Ontology Development for the Semantic Web. In *1st International Semantic Web Conference (ISWC2002)*, Sardinia, 2002. Springer.
19. V. Tablan, T. Polajnar, H. Cunningham, and K. Bontcheva. User–friendly ontology authoring using a controlled language. In *Proceedings of LREC 2006 - 5th International Conference on Language Resources and Evaluation*. ELRA/ELDA Paris, 2006.
20. V. Tamma, I. Blacoe, B. L. Smith, and M. Wooldridge. Introducing autonomic behaviour in semantic web agents. In *In Proceedings of the Fourth International Semantic Web Conference (ISWC 2005), Galway, Ireland, November.*, 2005.
21. M. Völkel and T. Groza. SemVersion: RDF-based ontology versioning system. In *Proceedings of the IADIS International Conference WWW/Internet 2006 (ICWI 2006)*, 2006.

Change Management for Metadata Evolution

Diana Maynard[1], Wim Peters[1], Mathieu d'Aquin[2], and Marta Sabou[2]

[1] Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street, Sheffield, UK
{diana, wim}@dcs.shef.ac.uk
[2] Knowledge Media Institute, Open University,
Walton Hall, Milton Keynes, UK
{m.daquin, r.m.sabou}@open.ac.uk

Abstract. In order to meet the demands of the Semantic Web, today's ontologies need to be dynamic, networked structures. One important challenge, therefore, is to develop an integrated approach to the evolution process of ontologies and related metadata. Within this context, the specific goal of this work is to capture the evolution of metadata due to changed concepts, relations or metadata in one of the ontologies, and to capture changes to the ontology caused by changes to the metadata. After a short discussion of the nature of metadata, we propose a methodology to capture (1) the evolution of metadata induced by changes to the ontologies, and (2) the evolution of the ontology induced by changes to the underlying metadata. This will lead to the implementation of an approach for evolution of metadata related to ontologies.

1 Introduction

Support for metadata evolution becomes extremely important in a distributed, dynamic environment. Change management should warrant the continuity of data access, i.e. all data previously associated with an older version of an ontology should be accessible and interpretable through the new version. When ontologies evolve, these changes should be propagated to all the information objects that are dependent on them, such as local copies of ontologies and annotated texts, although this may not always be possible in practice, particularly where networked ontologies are concerned. After the detection of changes in conceptual structure between two versions of an ontology, the ontology management system must enable the update of metadata affected by the changes in the ontology, in order to maintain a consistent link between ontology and metadata. This link is defined by a number of attributes, amongst which are the URLs for ontology, annotated text and concept. On the other hand, the system must also be capable of enabling updates to the ontology which may be necessary when the metadata changes: for example, the processing of new documents might show the emergence of new concepts and new relations, reflecting an evolution of the domain itself, or some concepts may become less significant. Similarly, the mismatch between the results of two different annotators in a collaborative annotation

environment might require the merging of two subconcepts in an ontology. In these scenarios, the evolution of ontologies should be guided by changes in the metadata, to keep the knowledge they contain up-to-date with respect to the considered domain.

With respect to ontology evolution, we therefore distinguish two types of changes: top-down and bottom-up [12]. Top-down changes are explicit changes in the ontology, to which the metadata needs to be adapted. These will be discussed in Section 3. Bottom-up changes occur when distributed metadata need to be reflected by changes to an ontology, and will be discussed in Section 4. First, however, in Section 2 we take a brief look at semantic metadata, in order to fully understand the role of its creation, existence and need for maintenance.

Finally, we would like to clarify some terminological ambiguity when talking about metadata. This is due to the use of the term in various areas of knowledge engineering. First, there is the notion of metadata as ontology metadata. Ontology metadata provides information about the ontology, e.g. who created it, how many concepts it contains, etc. A proposal exists for a standard description for this type of metadata information, the Ontology Metadata Vocabulary (OMV) [5]. Changes in the ontology may affect the value of these standardised attributes: for instance, the addition of another natural language for the labels will lead to a change in the OMV language attribute. The second notion of metadata stems from the area of natural language processing, and is often called semantic metadata or annotations. This type of metadata concerns concept instantiation in the form of the annotation of textual (or other forms of) data, and therefore contains information about the linguistic content of the ontology. Note that within this work, we are concerned primarily with text rather than other forms of media such as images or videos, so where the question of media is left unspecified, we shall be referring to textual forms of media. In this sense of metadata, data annotation concerns the task of adding semantic metadata to text. In this context it concerns the linking of instances in the text to concepts in the ontology, and potentially also finding relations between such concepts. This is known as semantic metadata creation. It is applied to ontology evolution in the form of bottom-up change discovery and ontology population, which concerns adding instances from the text to concepts in the ontology. In this paper, we will be discussing the dynamics of semantic textual metadata: hereafter when we refer to metadata, unless otherwise stated, we mean semantic metadata.

2 Creation of semantic metadata

The Semantic Web aims to add a machine tractable, repurposable layer to complement the existing web of natural language hypertext. In order to realise this vision, the creation of semantic annotation, the linking of web pages to ontologies, and the creation, evolution and interrelation of ontologies must become automatic or semi-automatic processes. An important aspect of the Semantic Web revolution is that it is based largely on human language materials, and in making the shift to the next generation knowledge-based web, human language

will remain crucial. In the context of new work on distributed computation, Semantic Web Services go beyond current services by adding ontologies and formal knowledge to support description, discovery, negotiation, mediation and composition. This formal knowledge is often strongly related to informal materials. To make these types of services cost-effective, we need automatic knowledge harvesting from all forms of content that contain natural language text or spoken data.

Semantic annotation is essentially the task of assigning to the entities in the text links to their semantic descriptions. This kind of metadata provides both class and instance information about the entities. Semantic annotations enable many new applications to be performed, such as highlighting, indexing and retrieval, categorisation, generation of more advanced metadata, and a smooth traversal between unstructured text and available relevant knowledge. Semantic annotation can be applied to any kind of text - web pages, regular (non-web-based) documents, text fields in databases, etc. - or even to non-textual forms of data (although, as mentioned earlier, we shall restrict ourselves here to textual content). Furthermore, knowledge acquisition can be performed on the basis of the extraction of more complex dependencies, such as analysis of the relationships between entities, event and situation descriptions, and so on.

Automatic semantic annotation of textual data is generally carried out by means of some kind of ontology-based information extraction (OBIE). While semantic annotation can of course be performed manually, it is a time-consuming and laborious task, which certainly cannot scale to the demands of real world applications on the web. Therefore at the least, a semi-automatic, if not fully automatic, process is required. Ontology-based IE differs from traditional IE in a number of ways. First, it makes uses of a formal ontology rather than a flat lexicon or gazetteer, which may also require reasoning to be carried out. Second, it not only finds the (most specific) type of the extracted entity, but it also identifies it, by linking it to its semantic description in the instance base. This allows entities to be traced across documents and their descriptions to be enriched through the IE process. This more semantic form of IE is therefore a much harder task than the traditional one: see for example [7], which describes the extension of a traditional IE system into one which performs a more semantic extraction, comparing the two tasks, systems and results.

The automatic population of ontologies with instances from the text requires the existence of an ontology and a corpus. From this, an OBIE application identifies instances in the text belonging to concepts in the ontology, and adds these instances to the ontology in the correct location. It is important to note that instances may appear in more than one location in the ontology, because of the multidimensional nature of many ontologies and/or ambiguities which cannot or should not be resolved at this level. For examples of OBIE applications, see for example [9, 2].

3 Top-Down metadata evolution

As metadata creation is an expensive task, it is important that sets of ontology metadata and document annotations are kept in sync with an evolving ontology. As far as possible, we do not want to have to reannotate a whole corpus every time the ontology changes in some way (although in some cases this is inevitable). The evolution of the related metadata has to be synchronised with the evolution of the ontologies for the purpose of preserving instance data and compatibility between ontology versions . Therefore, methods for evolving metadata automatically and in parallel with the networked ontologies are required. In the presence of networked ontologies, this includes the synchronisation of distributed metadata.

Consequently, changes in ontology structure need to be captured by means of evolution operations, and described in some standardised fashion. One solution to this problem is to keep the metadata static and keep track of the (specific) version of the ontology used for the text annotation. In this case, we assume that annotations are stable but contextual, and thereby manage the evolution of the ontology only at the ontological level by means of links between the old and new versions of the ontology, or the link between different ontologies. In this case we could study how the annotations according to one ontology can be imported into a new ontology (linked to the other ontology in a formal way). However, this approach leads to a high level of complexity if there are many versions of the ontology and also makes automatic processing more difficult (for example, using the populated ontology for tasks such as information retrieval and question answering). Furthermore, it is not evident how this could be achieved when material is added to or deleted from the ontology, as links cannot exist to items which are non-existent. So it would only be useful for certain types of change and not as a complete framework for change (assuming that we wish to avoid data loss).

3.1 Related Work

The main candidate for the change capture and description phases is [6], who propose a framework for ontology evolution that integrates all sources of ontology change information. A so-called transformation set encompasses all changes that have occurred between an old and a new version of the ontology. The changes that can occur in OWL ontologies have been gathered into a change typology and made available on the web. The change typology is an ontology of change operations, and covers basic change operations such as `delete superclass`, and complex operations, e.g. `add an entire subtree`. The ontology of basic change operations contains `add` and `delete` operations for each feature of the OWL knowledge model. Complex operations consist of a combination of basic operations. We have evaluated this ontology in the light of the requirements for capturing the dynamics of metadata.

The main element missing from Klein and Noy's typology is the networked nature of ontologies. For example, the addition of a new ontology to the network

would require the addition of new instances and possibly other changes such as relational information linking instances to different parts of the network. We do not deal here specifically with network-related changes, preferring to focus in the first instance on the changes that stem from one ontology; networked ontology evolution will be part of future work.

The other difference between Klein and Noy's typology and our proposed framework is that they basically developed the change ontology for their own metamodel of OWL, which was built with different assumptions and design considerations that are partially incompatible with ours, since our work is within the scope of the NeOn project[3], which relies on a particular metamodel for OWL. For example, the references to slots are more concerned with a frame-based model and are not really appropriate here, although many of them can be translated into changes that affect instances.

Finally, we also aim to provide a different kind of categorisation of the changes: for example, distinguishing between changes that stem from or affect the network, ontology, concept or instance, because these may have different causes and different effects. We also need to characterise the actions associated with the changes, rather than just specifying the information loss: for example, addition of a new concept in the ontology may require additional annotation in order to find the relevant metadata (instances); deletion of a concept may involve an automatic deletion of its relevant instances; merging two concepts requires merging of their instances, and so on.

For this purpose we need to establish:

1. which change classes are relevant for metadata evolution;
2. what effect such changes have on the metadata
3. what action should follow these changes.

For instance, if a concept is moved, references to the concept should be found and modified appropriately; if a concept is deleted, annotations referencing the concept should be changed to reference its superclass, etc.

3.2 A framework for capturing metadata evolution

We classify the changes according to those which stem principally from the concept, instance or property level. First we look at the effect on the instances caused by changes to the ontology / concepts. We describe the change and the effect on the data (largely as established by Klein and Noy, though with some differences related to the frame-based vs OWL implementation), and propose the actions that should be taken. Note that our aim here is to attempt to specify the changes and actions that should take place – some of these will most likely occur as a matter of course (such as merging associated instances when their respective classes are merged).

[3] http://www.neon-project.org

It must be borne in mind that the change typology represents an initial framework for capturing changes in ontologies. Its envisaged role is to serve as the basis for further discussion and a more formal specification.

Most changes originating at the property level do not really affect the metadata as such: for example, changing a property means that a new property name will be attached to the instance, but does not affect the instance per se. However, if a new property is added, reannotation may be necessary to acquire new instances. On the other hand, if a property is deleted, the instance should automatically inherit the property of its superclass.

Changes occurring at the concept level are the ones most likely to influence the metadata: in some cases, the instances may have to be moved to new classes; in other cases, reannotation may be required in order to acquire new instances (e.g. when new classes are added to the ontology).

Thus we see that for each change, a set of 3 possible actions exists:

1. do nothing;
2. some manual action is required;
3. some automatic action is required.

Do Nothing: The first action is self-explanatory and requires no further discussion. In some cases, a degree of data loss is inevitable.

Manual Action: The second action almost always requires some kind of reannotation of the corpus. This can be for several reasons:

First, existing information in the ontology needs to be reclassified. This could be the case where a new subclass is added. Imagine we have an ontology which contains the class `comestible` and the subclass `food`. Now imagine we add a new subclass of `comestible`, `drink`. Before this subclass was added, all instances of `drink` that were not also instances of `food` would have been classified simply as `comestible`, because that was the most specific class to which they could belong in the ontology. Once `drink` is added to the ontology, such instances need reclassifying under the `drink` class. However, this is almost impossible to do automatically because we have no way of knowing which instances of `comestible` should be moved and which should not, unless we return to the text for further analysis, or unless the ontology provides us with further information.

Second, information may be missing from the ontology. This could be the case when a new top class is added to the ontology.

Finally, it may also be a combination of the two factors. For example, in the case where a new superclass is added to the ontology, some information may be missing and some may need reclassification.

While missing instances can simply be added to the appropriate place in the ontology when found in the text, reclassification is a little more tricky because it consists of a two-stage process: first the system must find the instances in the text and recognise that they are currently not classified in the most appropriate way in the ontology, and second, it must follow the automatic procedure for reclassification as specified below.

Automatic Action: The third action requires a set of procedures to be followed for automating the reclassification of instances in the ontology. Below we give an example of some such possible procedures according to the GATE ontology API: naturally the exact procedures will be implementation-specific according to the ontology model used. Note that some of the actions could be defined simply as natural consequences rather than explicit actions.

- When a class is added
  - It will automatically inherit from its superclasses a set of properties
- When an equivalent class is added
  - It will inherit all the instances from its equivalent class
  - These instances will all have a sameIndividualAs statement added
- When an instance is added
  - It will automatically inherit from its superclasses a set of properties.
- When a class is deleted
  - A list of all its superclasses is obtained. For each class in this list, a list of its subclasses is obtained and the deleted class is removed from it.
  - All subclasses of the deleted class are moved to subclasses of the parent of the deleted superclass. A list of all its disjoint classes is obtained. For each subclass in this list, a list of its disjoint classes is obtained and the deleted class is removed from it.
  - All instances of the deleted class are moved to its direct superclass in the ontology.

4 Bottom-Up ontology evolution

Textual resources and the associated metadata generally evolve faster than the related ontologies. In that sense, they reflect more accurately the evolution of the domain itself, e.g. progresses realised in scientific fields, new trends or obsolete elements. Even if ontologies are supposed to be a stable conceptualisation of the domain, the evolution of the metadata has to be reflected in the related ontologies, to keep the relation between ontologies and metadata up-to-date, and so that the ontologies evolve in accordance with the current state of the domain they represent. Therefore, the dynamics of metadata have to be captured in a way that it can be related to the adequate ontologies and guide their evolution by suggesting corresponding ontology changes, leading to a metadata-driven maintenance of ontologies. We shall therefore investigate which operations are necessary to cover a number of evolution strategies for bottom-up change discovery, e.g. in the case of a required extension or refinement of the ontological structure of a particular ontology component, or the suggestion of the merge of two classes for a particular application if an automatic classifier is unable to distinguish between the two (or if human agreement is not reached) and the concepts in question are considered relatively similar.

In this section, we consider the notion of metadata according to a general definition: information about the content of a resource or a document. Resources

and documents can take different forms, (texts, images, etc.), and the associated metadata can be either manual annotations (e.g. folksonomy tags given by the author of an image) or automatically extracted (e.g. using tools for information extraction from texts). In the case of semantic annotations, metadata is represented according to ontology elements. The goal of this section is to show how ontologies can evolve in accordance with the evolution of metadata associated with documents of the domain. The basis of the proposed mechanism consists of relating evolving (non-semantic) metadata with the considered ontologies, in order to assess the insufficiencies of these ontologies in representing the considered metadata as semantic annotations. The process of suggesting changes in an ontology based on changes in the underlying annotated dataset was defined as data-driven change in [8]. It can be argued that the methodology described here is data-driven rather than metadata-driven, in the sense that, in most of the cases, metadata evolves as a consequence of the evolution of the underlying data. However, in the case of manual annotation, for example, the annotator may change the metadata associated with a document, without changing the document itself.

The proposed methodology can be considered as a bottom-up approach for two reasons. First, the basic principle is that changes in metadata would guide the evolution of the ontologies. Second, the definition of the methodology itself is based on experimentations using real life datasets: by studying the evolution of these datasets through the associated metadata, we aim at defining general principles for a metadata-driven ontology evolution. Therefore, we plan to apply this methodology to two concrete case studies using two different datasets and two different forms of metadata: the evolution of ontologies depending on changes in folksonomies and depending on changes in textual documents in a given domain. We first give a brief outline of the methodology we plan to follow, followed by descriptions of two case studies.

4.1 Methodology

This section presents our approach for studying changes in metadata and deriving suggestions of changes in the corresponding ontologies. More precisely, we describe a method to capture (1) the evolution of different types of metadata and (2) the implied changes in the related (networked) ontologies. The proposed methodology relies on a bottom-up approach for capturing the relations between metadata evolution and ontology changes. First, the data contained in a set of existing documents/resources has to be automatically related to the considered ontologies, in order to form an exploitable corpus of ontology-based metadata. Metadata changes can then be captured on the basis of the generated structure, and the study of the implications of these changes on ontologies can be carried out, with the aim of deriving suggestions of corresponding ontology evolutions. We therefore aim to establish what effect the evolution of metadata has on the ontologies, in terms of ontology changes. For instance, if a new prominent term appears from a set of textual resources, a concept should be added in the on-

tology; if two terms appear to be related in the metadata (e.g. through frequent co-occurrence) then the ontology should be extended with a new relation.

Obtaining this exploitable corpus of metadata, linking the resources to the ontologies, leads to several common, generic or domain-specific issues:

Pre-processing: First, the considered documents, resources and metadata can take different forms, and may contain noise, redundancies, etc. A preprocessing step is generally required in order to obtain an exploitable corpus, including domain (or data)-dependent tasks such as filtering, transformation, etc.

Conceptual Organisation: Once a set of descriptive terms is obtained for each of the considered documents, we need to identify from among these terms those that represent important domain concepts, which may be contained in the ontologies. More importantly, potential relations between these terms, or more precisely between the corresponding concepts, have to be detected. One possibility is to group (or cluster) the terms according to their relations in the documents, suggesting in this way potential relations in the ontologies.

Ontology Matching: Finally, the links between the obtained conceptual structure and the considered ontologies have to be established. This can be realised by using generic ontology matching techniques [1], mapping the organised terms to ontology concepts, and relating them according to ontology relations. It is worth noting that these mappings already provide indications of missing knowledge in the ontologies: the terms and relations for which there is no mapping suggest missing concepts and relations.

These three tasks provide the basic structure which we rely on for capturing metadata changes and the related ontology evolutions.

4.2 Capturing metadata changes

The evolution of the metadata can be captured through the changes occurring in the corresponding set of terms, conceptual structures, and mappings. Documents can be added or removed, leading to a different (either extended or reduced) set of terms. The comparison of the results of the conceptual organisation of the terms obtained at different times allows the consideration of changes from a more abstract (conceptual) point of view, indicating for example previously unknown relations between terms, or ones that have become obsolete. It is worth noticing here that these conceptual structures are not ontologies, but rather intermediary descriptions of the metadata, used to guide the integration (matching) with the considered ontologies, and facilitating the study of metadata evolution. Finally, new mappings to the ontologies may appear from the conceptual structure, and some may be modified or reconsidered. Basically, by comparing the processed metadata at different levels and different time points, we can trace, capture and study these evolutions in an abstract, ontology-related representation. The outcome here is a characterisation of the changes in the metadata that may lead to particular evolution of ontologies.

4.3 Suggesting ontology changes

Capturing the evolution of the metadata is only the first part of the problem. Each evolution may lead to an extension of the existing gap between the metadata and the ontology coverage. For example, if some newly added terms have no correspondence in the ontologies, an extension with additional concepts and relations may be required. On the other hand, if the set of terms generated at any moment in time is smaller than the set of terms generated previously (e.g., because some data has been deleted or because some terms are less significantly represented) then this can lead to the pruning of the ontologies in such a way that they do not contain obsolete concepts/relations. Our goal is to provide general principles for suggesting changes in ontologies in order to fill this additional gap. More precisely, studying the evolution of real-life datasets would allow us to associate suggestions of changes in the ontology to typical changes in the captured metadata, thus providing a basis for guiding the maintenance of ontologies according to the related metadata.

The proposed methodology for the bottom-up, metadata-driven evolution of ontologies is summarized in Figure 1. It is illustrated in the next two sections (4.4 and 4.5) by the application to two real life case studies.

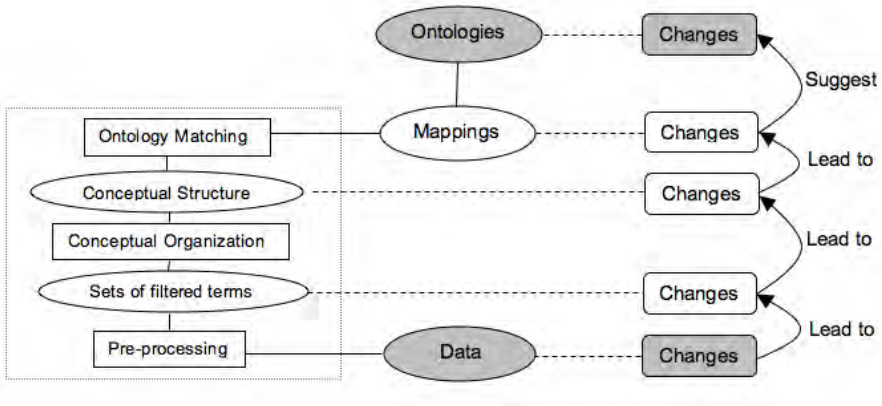

Fig. 1. Schematic view of the bottom-up approach for ontology evolution

4.4 Case Study 1: Evolving ontologies through linking them to folksonomies

Social tagging systems such as Flickr (http://www.flickr.com), for photo-sharing, and Del.icio.us (http://del.icio.us), for social bookmarking, are becoming more and more popular, nowadays covering a wide range of resources and communities, with a huge number of participants sharing and tagging a large number of

resources. Due to their popularity, folksonomies are changing rapidly as users add new resources and tags. Because they are updated continuously, folksonomies are up-to-date with respect to the vocabulary used by a wide range of people, thus reflecting new terms that appear. Unlike folksonomies, ontologies are built at a much slower rate and therefore they often lag behind the novel terminology in a given domain. A solution for automatically enriching (and hence evolving) ontologies is to align them to folksonomies and modify them so that they reflect the changes in the folksonomies. This alignment is also beneficial for folksonomies. Current tag sets lack any semantic relations and therefore are hard to use during searching. The alignment to ontologies allows the enrichment of folksonomies with semantic relations so that more semantic searches can be performed.

In this first case study, we investigate how ontology changes can be derived from changes in folksonomies. The process of linking folksonomies to ontologies has been described in [11] and will be used in the first part of the case study (as it covers the first three major steps of our methodology). We investigate the tag sets in Flickr and Del.icio.us, due both to their popularity (with a large number of resources, users, and tags) and availability. In our experiments, we use the same Del.ici.ous tags as [11] and Flickr tags for photos posted between 01-02-2004 and 01-03-2006.

Having derived the metadata following the work of [11], we need to capture the way metadata changes. Our plan is to re-run the process described in their work at different points in time and compare the output (i.e., see how the obtained clusters differ). We will identify clusters that have been added or which have disappeared. At a more fine-grained level, for each cluster we will monitor internal changes, namely additions or elimination of some tags. All these types of changes have important influences on the evolution of the ontology. Our task will be to derive a wide range of metadata changes taking into account the metadata derived in our experiments.

Once a set of changes in the metadata has been identified, these changes can be used to suggest corresponding updates to the ontologies. While this typology of changes will be derived when the experiments are run and therefore will be grounded in actual data, we can already predict some typical changes. For example, if a new cluster is added, then find an ontology that contains elements (concepts/relations) with the same label as the tags in the cluster. If the ontology only covers a subset of the terms, then find methods to extend the ontology with concepts corresponding to the missing tags. If a new tag is added to a cluster, then if no corresponding concept exists in the ontology to which the cluster was aligned, insert the tag in the ontology.

4.5 Case Study 2: Data Driven Ontology Learning on FAO data

The second case study is performed on a different type of data set from the previous case study. In this case, we investigate how ontologies can be evolved through applying ontology learning methods to textual data. The data set used is

a collection of textual data from the FAO[4] (United Nations Food and Agriculture Organisation, Fisheries Department). There are several collections, including news items, fact sheets on species and internal documentation.

We envisage that pre-processing will be performed using existing software packages. In particular, we will experiment with three tools: TermExtractor [10] – a web service for the extraction of domain relevant terminology; GATE [4] – a suite of natural language applications for document annotation; Text2Onto [3] – an ontology learning tool that provides a variety of algorithms needed for the entire process of ontology learning. Some of these algorithms deal with the simplest task in ontology learning, that of extracting relevant terms from a corpus.

The next step is one of conceptual organisation, which aims at deriving some meaningful structure between the identified terms. Again, we envisage the possibility to use two different kinds of packages to perform this task: clustering algorithms for when TermExtractor is used, leading to clusters similar to those obtained in case study 1; and the ontology learning algorithms from Text2Onto to derive an ontological structure between the identified terms (including subsumption relations, mereological relations, general relations).

Depending on the conceptual structure derived previously, we can use a variety of ontology matching approaches to align these structures to ontologies. Simple, string-similarity based methods can be used to find correspondences between terms in a cluster and the labels of ontology elements. If the derived structures are richer than sets of terms, we can also employ structural matching methods to find alignments between the derived ontologies and the ontologies (ontology networks) that need to be updated.

Capturing metadata changes and linking them to updates in the corresponding ontologies will be the topic of our investigation after running the above mentioned experiments. We can already say, however, that these types of changes will somehow depend on the kind of conceptual structure that the metadata takes. If we derive clusters of terms, then we can provide a similar typology of changes as in case study 1. If the conceptual structures derived are ontologies, then we can re-use and extend Klein's typology of changes to capture changes between ontologies derived at different moments in time and then suggest adequate changes in the ontologies.

5 Conclusion and Future Directions

In this paper we have described a framework and methodology to capture the dynamics of metadata, consisting of two main aspects: the evolution of metadata when an ontology changes, and changes to the ontology resulting from metadata evolution. The first aspect involves using the Klein and Noy ontology as a starting point, and proposing changes in order to make it more suitable for our needs. We then discuss how this will be implemented. For the second aspect, we have

[4] http://www.fao.org

described a methodology for capturing changes in metadata with the aim of suggesting changes in the related ontologies. We adopt a bottom-up approach, studying the evolution of real-life datasets to come up with general principles for metadata-driven ontology evolution. In this line of approach, we propose to apply this methodology to two concrete case studies, using two different datasets, and two different forms of metadata: tags of folksonomies, and texts related to the agricultural domain. Therefore, the obvious next step of this work is to run the experiments corresponding to these case studies, capturing the evolution of the metadata by relating the considered resources to the considered ontologies at different points in time. We believe that the study of these ontology-based metadata evolutions will allow us to understand and potentially capture the implications of metadata changes on the related ontologies. The concrete outcome of this work should be general rules for suggesting ontology changes to reflect metadata changes, providing a basis for a metadata-driven ontology evolution.

Since metadata changes lead to ontology changes, and ontology changes lead to metadata changes, the interactions between these two processes have to be considered. We can therefore imagine an integrated process that would alternatively suggest changes in ontologies and metadata until a stable version of both elements is obtained.

6 Acknowledgements

This work was partially supported by the EU project NeOn (IST-2005-027595).

References

1. J. Banerjee, W. Kim, H. Kim, and H. F. Korth. Semantics and implementation of schema evolution in object-oriented databases. In *Proceedings of SIGMOD*, 1987.
2. K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349—373, 2004.
3. P. Cimiano and J. Voelker. Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, Alicante, Spain, 2005.
4. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.
5. J. Hartmann, Y. Sure, P. Haase, and M. del Carmen Suarez-Figueroa. OMV – Ontology Metadata Vocabulary. In *ISWC 2005*, 2005.
6. M. Klein and N. F. Noy. A Component-Based Framework for Ontology Evolution. In *Workshop on Ontologies and Distributed Systems, IJCAI*, 2003.
7. D. Maynard, K. Bontcheva, and H. Cunningham. Towards a semantic extraction of named entities. In G. Angelova, K. Bontcheva, R. Mitkov, N. Nicolov, and N. Nikolov, editors, *Proceedings of Recent Advances in Natural Language Processing (RANLP'03)*, pages 255–261, Borovets, Bulgaria, Sep 2003. `http://gate.ac.uk/sale/ranlp03/ranlp03.pdf`.

8. D. Maynard, W. Peters, and Y. Li. Metrics for evaluation of ontology-based information extraction. In *WWW 2006 Workshop on "Evaluation of Ontologies for the Web" (EON)*, Edinburgh, Scotland, 2006.
9. D. Maynard, M. Yankova, A. Kourakis, and A. Kokossis. Ontology-based information extraction for market monitoring and technology watch. In *ESWC Workshop "End User Apects of the Semantic Web")*, Heraklion, Crete, 2005.
10. R. Navigli and P. Velardi. Learning Domain Ontologies from Document Warehouses and Dedicated Websites. *Computational Linguistics*, 4(2), 2004.
11. L. Specia and E. Motta. Integrating Folksonomies with the Semantic Web. In *ESWC*, Innsbruck, Austria, 2007.
12. L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. User-driven ontology evolution management. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, pages 285–300, Siguenza, Spain, 2002.

Ontology Dynamics with Multimedia Information: The BOEMIE Evolution Methodology ⋆

Silvana Castano[1], Sofia Espinosa[2], Alfio Ferrara[1], Vangelis Karkaletsis[3], Atila Kaya[2], Sylvia Melzer[2], Ralf Möller[2], Stefano Montanelli[1], Georgios Petasis[3]

[1] Università degli Studi di Milano,
DICo, 10235 Milano, Italy,
{castano, ferrara, montanelli}@dico.unimi.it
[2] Hamburg University of Technology,
Institute of Software Systems,
21079 Hamburg, Germany,
{sofia.espinosa,at.kaya,sylvia.melzer,r.f.moeller}@tuhh.de
[3] Software and Knowledge Engineering Laboratory,
Institute of Informatics and Telecommunications,
National Centre for Scientific Research (N.C.S.R.) "Demokritos",
P.O. BOX 60228, Aghia Paraskevi, GR-153 10, Athens, Greece.
{vangelis, petasis}@iit.demokritos.gr

Abstract. In this paper, we present the ontology evolution methodology developed in the context of the BOEMIE [1] project. Ontology evolution in BOEMIE relies on the results obtained through reasoning for the interpretation of multimedia resources in order to evolve (enhance) the ontology, through population of the ontology with new instances, or through enrichment of the ontology with new concepts and new semantic relations.

1 Introduction

Ontology learning is a wide domain of research, involving methods and techniques for the acquisition of an ontology from semantic information/conceptual knowledge extracted from a domain. Being closely related to the field of knowledge acquisition, a significant amount of the work has been presented in the bibliography that concentrates on the task of knowledge acquisition from text, through the re-use of widely adopted natural language processing and machine learning techniques [1, 2]. The methodology proposed by the BOEMIE project tries to extend existing approaches by considering modalities beyond text, such as still image, video and audio. The proposed methodology can be separated into three major components:

– *A multimodal information extraction engine.* This information extraction engine is responsible for extracting instances of "primitive" concepts as well as instances of relations between concept instances, from documents belonging to the textual, still

⋆ This paper has been partially funded by the BOEMIE Project, FP6-027538, 6th EU Framework Programme.

[1] http://www.boemie.org

image, video and audio modalities. "Primitive" concepts (known as "mid-level" concepts within BOEMIE) are concepts whose instances can be directly identified in corpora of a specific modality. For example, in the textual modality the name or the age of a person is a mid-level (or "primitive") concept, as instances of these concepts can be associated with relevant text portions. On the other hand, the concept person is not a mid-level concept, as it is a "compound" concept that has other concepts as properties, like the person name, age, gender, etc. "Compound" concepts are referred as high-level concepts within the BOEMIE project, and cannot be directly identified in a corpus and thus associated with a corpus segment.

- *A semantic interpretation engine*, responsible for producing one or more explanations of an event described in a document. Semantic interpretation operates on the instances of mid-level concepts and relations between them extracted by the information extraction engine, in order to create instances of high-level concepts that explain the event, always according to the domain ontology. Semantic interpretation is performed through standard (i.e. induction) but also non-standard (i.e. abduction) reasoning services and is formalised as a two-level process. During the first level, semantic interpretation is performed on the extracted information (mid-level concept instances/relations) from a single modality only, in order to form modality specific high-level concept instances. At a second level, the modality specific high-level instances are fused, in order to produce high level concept instances that are not modality specific, and contain information extracted from all involved modalities.
- *An ontology evolution toolkit*, which uses the results obtained through reasoning at the interpretation phase in order to evolve (enhance) the ontology, through population of the ontology with instances, or through enrichment of the ontology with new concepts and new relation types.

In this paper, we focus on the BOEMIE evolution methodology and the role of semantic interpretation for ontology evolution. Then, we describe how ontology population and enrichment are articulated, by discussing the main design principles and by describing samples evolution scenarios. The paper is organized as follows: a description of the ontology evolution methodology in BOEMIE is given in Section 2. Methods and techniques for multimedia interpretation by reasoning are discussed in Section 3. The evolution activities of population and enrichment are presented in Sections 4 and 5, respectively. In Section 6, we discuss the original contributions of the present work. Finally, we give our concluding remarks in Section 7.

2 Ontology Evolution in BOEMIE

Ontology evolution within BOEMIE uses as input the results of the semantic representation performed upon information extracted from multiple modalities (combined through fusion). In order to be able to deal with all possible situations requiring evolution, a pattern-driven approach is adopted. Typical input to the evolution toolkit is in the form of ABoxes, containing the results of the semantic representation of

the fused extracted information. These results typically include instances of mid-level concepts, relation instances between mid-level concept instances, high-level instances, relation instances between instances of high-level concepts, and possibly instances of the "unknown" mid-level concept. According to the information contained in ABoxes constituting the input, the evolution pattern selector selects the most prominent evolution pattern, triggering either ontology population or ontology enrichment.

The BOEMIE ontology evolution methodology is shown in Figure 1.

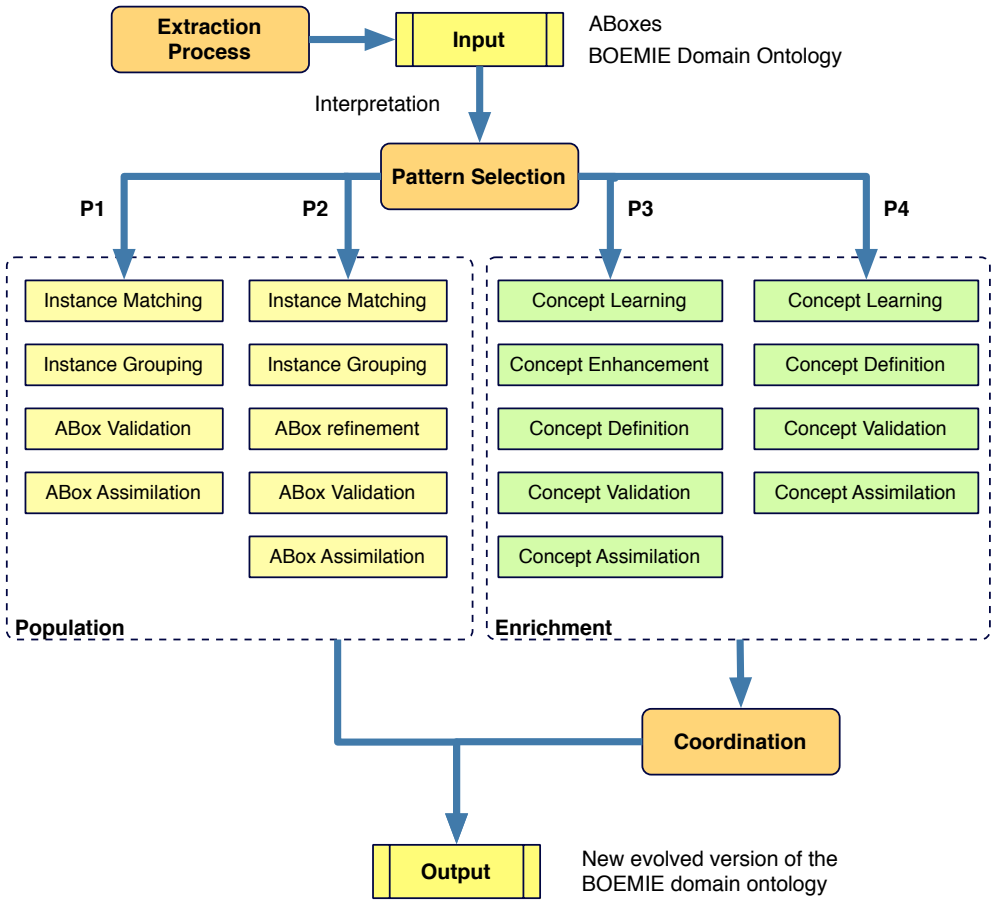

Fig. 1. The BOEMIE evolution methodology

Ontology population is the activity of adding new instances into the ontology and it is performed every time at least one explanation can be found for a multimedia resource.

Ontology enrichment is the activity of extending the ontology, through the addition of new elements (e.g., concepts, relations, properties). Ontology enrichment is performed every time the background knowledge is not sufficient to explain the extracted information from the processed multimedia documents. Thus, the ontology enrichment activity is expected to extend this background knowledge through the addition of new ontology elements.

Coordination is the activity of producing a log of the changes introduced into the new evolved version of the ontology with respect to the initial version and of defining and updating the mapping knowledge between the domain ontology and other related external knowledge sources supporting the enrichment. Since this activity is affected by the changes of the background knowledge, it is executed after the enrichment activity.

In the remainder of the paper, we focus on the reasoning activity for resource interpretation and on the activities of ontology population and enrichment, by providing some relevant example.

2.1 Evolution Patterns

The BOEMIE evolution approach is featured by two main requirements:

- the capability of classifying the different situations that trigger ontology evolution by characterizing the results of the semantic interpretation process (i.e., the information specified in the incoming A-Box) with respect to the background knowledge;
- the definition of an appropriate activity articulation to correctly modify the ontology in each specific evolution situation.

The expected result of the semantic interpretation is a single explanation for a multimedia resource, that is, the resource is instance of only one high-level concept. However, other situations can occur when the background knowledge is not sufficient, characterized either by more than one possible explanation for the same multimedia resource or by the absence of explanations, meaning that no high-level concept can be found in the ontology for describing such a resource. Finally, we can also have the case where not only the high-level concept describing the resource is missing, but also for one or more elements identified in the resource a mid-level concept can not be assigned.

To take these requirements into account, four different *evolution patterns* (see Figure 2) have been identified for ontology evolution in BOEMIE. An evolution pattern determines the characteristics of the input ABox it deals with, defines the kind of evolution to be performed over the ontology (i.e., population or enrichment), and is articulated into a set of activities for implementing all the required changes. Population patterns (P1 and P2) describe the situation where the interpretation has found one or more high-level concepts explaining the resource and, thus, the ontology population activity is performed. Enrichment patterns (P3 and P4) describe the situation where no high-level concepts explaining the resource are found in the ontology, thus triggering ontology enrichment to acquire this missing knowledge. Pattern P4 has been conceived to deal with situations where not only the high-level concept is missing (like

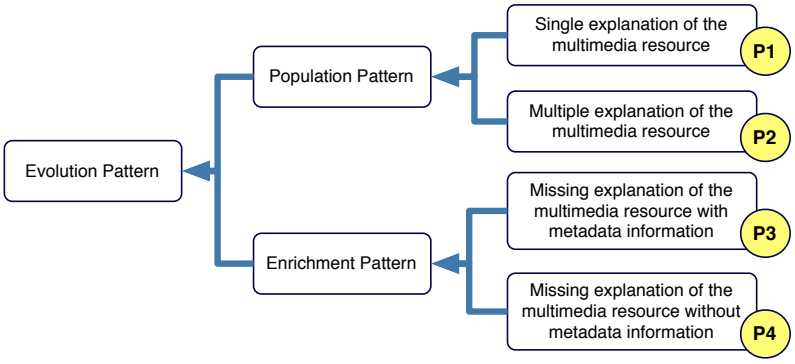

Fig. 2. BOEMIE evolution patterns

P3) but also one or more mid-level concepts are missing for the interpretation of the incoming resource. In case of missing explanations for mid-level concept instances, pattern P4 is always selected as prominent, to first enrich the ontology with missing mid-level concepts, thus enabling the interpretation of all mid-level concept instances. Then, in a subsequent cycle, the most suitable pattern will be chosen for handling the new situation appropriately.

3 Reasoning for Multimedia Interpretation

An ontology in a description logic framework is seen as a tuple consisting of a TBox and an ABox. In order to construct a high-level interpretation, the ABox part of the ontology is extended with some new assertions describing individuals and their relations. These descriptions are derived by media interpretation processes using the ontology (we assume the ontology axioms are denoted in a set Σ).

Interpretation processes are set up for different modalities, still images, videos, audio, and texts. In this section we discuss the interpretation process using an example interpretation for still images. The output is a symbolic description represented as an ABox. This ABox is the result of an abduction process (see [3] for a general introduction). In this process a solution for the following equation is computed: $\Sigma \cup \Delta \models \Gamma$. The solution Δ must satisfy certain side conditions.

The *abduction* inference service aims to construct a set of (minimal) ABox assertions Δ for a given set of assertions Γ such that Δ is consistent w.r.t. to the knowledge base $(\mathcal{T}, \mathcal{A})$ and satisfies [4]:

1. $T \cup A \cup \Delta \models \Gamma$
2. If $\Delta^{'}$ is an $ABox$ satisfying $T \cup A \cup \Delta^{'} \models \Gamma$, then $\Delta^{'} \models \Delta$ (Minimality)
3. $\Delta \not\models \Gamma$ (Relevance)

In Figure 3 an example from the athletics domain is presented. Assuming that it is possible to detect a horizontal bar bar_1, and a human $human_1$ by image analysis processes, the output of the analysis phase is represented as an ABox Γ. Assertions for the individuals and (some of) their relations detected by analysing Figure 3 are shown in Figure 4.

Fig. 3. Image displaying a high jump or a pole vault event.

$$\begin{aligned} bar_1 &: Horizontal\_Bar \\ human_1 &: Human \\ (bar_1, human_1) &: near \end{aligned}$$

Fig. 4. ABox Γ representing the result of the image analysis phase.

In order to continue the interpretation example, we assume that the ontology contains the axioms shown in Figure 5 (the ABox of the ontology is assumed to be empty). For some purposes, description logics are not expressive enough. Thus, some additional mechanism is required without jeopardizing decidability. In order to capture constraints among aggregate parts, we assume that the ontology is extended with DL-safe rules (rules that are applied to ABox individuals only). In Figure 6 a set of rules for the athletics example is specified. Note that the spatial constraints *touches* and *near* for the parts of a $Pole\_Vault$ event (or a $High\_Jump$ event) are not imposed by the TBox in Figure 5. Thus, rules are used to represent additional knowledge. Since spatial relations depend on the specific "subphases" of the events, corresponding clauses are included on the right-hand sides of the rules. For instance, a jumper as part of a $High\_Jump$ is near the bar if the image shows a $High\_Jump$ in the jump phase.

In the following we assume that rules such as those shown in Figure 6 are part of the TBox Σ. In order to provide a high-level interpretation, i.e. to provide a description of the image content in the form of high-level aggregates, we assume that spatial relations between certain objects detected by low-level analysis processes are not arbitrary.

In order to construct an interpretation for the image in Figure 3, an explanation is computed to answer why it is the case that a human is near a horizontal bar. Such explanations are considered the results of image interpretation processes. As mentioned above, the idea is to use the abduction inference service for deriving these kinds of (minimal) explanations (in the sense of interpretations). Minimal explanations can be extended appropriately in order to match expectations and task context.

We start with the computation of a minimal explanation in the athletics scenario. For this purpose, we slightly modify the abduction equation by taking into consideration that initially the ABox does not need to be empty. Thus, we divide Γ (see Figure 4)

$$
\begin{aligned}
Athlete &\equiv Human \sqcap \exists hasProfession.Sport \\
Jumper &\sqsubseteq Athlete \\
Pole &\sqsubseteq SportEquipment \\
Horizontal\_Bar &\sqsubseteq SportEquipment \\
Foam\_Mat &\sqsubseteq SportEquipment \\
Jumping\_Event &\sqsubseteq Event \sqcap \\
&\quad \exists_{\leq 1} hasParticipant.Jumper \\
Pole\_Vault &\sqsubseteq Jumping\_Event \sqcap \\
&\quad \exists hasPart.Pole \sqcap \\
&\quad \exists hasPart.Horizontal\_Bar \\
&\quad \exists hasPart.Foam\_Mat \\
High\_Jump &\sqsubseteq Jumping\_Event \sqcap \\
&\quad \exists hasPart.Horizontal\_Bar \\
&\quad \exists hasPart.Foam\_Mat \\
PV\_InStartPhase &\sqsubseteq \top \\
PV\_InEndStartPhase &\sqsubseteq \top \\
HJ\_InJumpPhase &\sqsubseteq \top
\end{aligned}
$$

Fig. 5. A tiny example TBox Σ for the athletics domain.

$$
\begin{aligned}
touches(Y,Z) \leftarrow\ & Pole\_Vault(X), \\
& PV\_InStartPhase(X), \\
& hasParticipant(X,Y), Jumper(Y), \\
& hasPart(X,Z), Pole(Z). \\
near(Y,Z) \leftarrow\ & Pole\_Vault(X), \\
& PV\_InEndStartPhase(X), \\
& hasPart(X,Y), Horizontal\_Bar(Y), \\
& hasParticipant(X,Z), Jumper(Z). \\
near(Y,Z) \leftarrow\ & High\_Jump(X), \\
& HJ\_InJumpPhase(X), \\
& hasPart(X,Y), Horizontal\_Bar(Y), \\
& hasParticipant(X,Z), Jumper(Z).
\end{aligned}
$$

Fig. 6. Additional restrictions for $Pole\_Vault$ and $High\_Jump$ in the form of rules.

into a part Γ_2 that the agent would like to have explained, and a part Γ_1 that the interpretation agent takes for granted. In our case Γ_2 is $\{(bar_1, human_1) : near\}$ and Γ_1 is $\{human_1 : Human, \quad bar_1 : Horizontal\_Bar\}$.

Coming back to the abduction problem specified above, we need solution(s) for the equation $\Sigma \cup \Delta \cup \Gamma_1 \models \Gamma_2$. In other words, given the background ontology Σ from Figures 5 and 6, the query

$$Q_1 := \{() \mid near(bar_1, human_1)\}$$

derived from Γ_2 should return *true*.

Obviously, this is not the case if Δ is empty. In order to see how an appropriate Δ could be derived, let us have a look at the rules in Figure 6. In particular, let us focus on the rules for $Pole\_Vault$ first. If we apply the rules to the query in a backward chaining way (i.e. from left to right) and unify corresponding terms we get variable bindings

for Y and Z. The "unbound" variable X of the corresponding rules is instantiated with fresh individuals (e.g. pv_1). It is easy to see that two possible solutions for the abduction equation can be derived. For this example the output of the interpretation process are two interpretation ABoxes representing two possible interpretations of the same image (see Figures 7 and 8). As a result, this example illustrates the situation where evolution pattern P2 can be triggered.

$$
\begin{aligned}
human_1 &: Human \\
bar_1 &: Horizontal\_Bar \\
(bar_1, human_1) &: near \\
hj_1 &: High\_jump \\
hj_1 &: HJ\_InJumpPhase \\
human_1 &: Jumper \\
(hj_1, human_1) &: hasParticipant \\
(hj_1, bar_1) &: hasPart
\end{aligned}
$$

Fig. 7. ABox representing the first result of the abduction process.

$$
\begin{aligned}
human_1 &: Human \\
bar_1 &: Horizontal\_Bar \\
(bar_1, human_1) &: near \\
pv_1 &: Pole\_Vault \\
pv_1 &: PV\_InEndStartPhase \\
human_1 &: Jumper \\
(pv_1, human_1) &: hasParticipant \\
(pv_1, bar_1) &: hasPart
\end{aligned}
$$

Fig. 8. ABox representing the second result of the abduction process.

Note that due to the involvement of $human_1$ in the pole vault event in Figure 7, $human_1$ is now seen as an instance of $Jumper$, and, due to the TBox, also as an $Athlete$. Thus, information from high-level events also influences information that is available about the related parts. With queries for $Jumper$s the corresponding media objects would not have been found otherwise. Thus, recognizing high-level events is of utmost importance in information retrieval systems (and pure content-based retrieval does not help).

The example discussed here covers the interpretation of still images. It is necessary, however, to keep in mind that each media object might consist of multiple modalities, each of which will be the basis of modality-specific interpretation results (ABoxes). In order to provide for an integrated representation of the interpretation of media objects as a whole, these modality-specific interpretation results must be appropriately integrated. A cornerstone of this integration process will be to determine which modality-specific names refer to the same domain object. This will be discussed in later sections.

Continuing the example, it might be the case that for some images the ontology does not contain relevant axioms or rules. In this case, the interpretation result, i.e. the result of solving the abduction problem $\Sigma \cup \Delta \cup \Gamma_1 \models \Gamma_2$ will be degenerated because, due to missing axioms or rules in Σ, Δ must necessarily be equal to Γ_2 in order to solve the equation. As an example of such a situation we can discuss an interpretation of Figure 3 without the rules from Figure 6 and the GCIs for $Pole\_Vault$ and $High\_Jump$ in Figure 5. The degenerate interpretation result is shown (as Γ) in Figure 4. This result illustrates the situation where evolution pattern P4 can be triggered.

4 Ontology Population

Ontology population is the process of inserting concept instances and relation instances into an existing ontology. The proposed by BOEMIE approach for ontology population is built upon two axes: entity disambiguation and consistency maintenance. With entity disambiguation we refer to the process of identifying instances that refer to the same real object or event. If an ontology is populated with an instance without checking if the real object or event represented by the instance already exists in the ontology (as an instance that has populated the ontology at an earlier population step), then redundant information (in the best case) will be inserted into the ontology. A worst case scenario is the redundant instances to contain contradicting information, which may lead to an inconsistent ontology. At the same time maintaining the consistency of an ontology is crucial (mainly through the elimination of contradictory information), as an inconsistent ontology can not be used to reason with.

The ontology population activity can be decomposed into the following tasks:

- *Instance matching:* The first task of the population activity is the identification of similar instances contained in the ontology for each HLC instance (HLCi) in the set of explanations. Having a single HLCi as input, instance matching is expected to return a set of instances that populate the HLC and are similar to the incoming HLCi. Each returned matching instance is also expected to have a similarity figure, which measures the similarity of the two HLCis. The results of instance matching can be used to group instances that represent the same real object or event (as HLCis that are similar are assumed to represent the same real object or event) and possibly help in disambiguating multiple explanations in the case of evolution pattern P2 (during the ABox refinement task).
- *Instance grouping:* This task is responsible for grouping all the instances that represent the same real object or event, by exploiting the results of the instance matching task, where every incoming HLCi has been matched with a set of other instances that populate this HLC. Instance grouping is responsible to decide which of these matching instances clusters will be kept and grouped together to form a group that represents the same real object or event.
- *ABox refinement (evolution pattern P2 only):* In case of multiple explanations the most suitable explanation is selected by exploiting the results of the instance matching/grouping tasks. Assertions related to the rest of the explanations are removed from the ABox, thus leading to a refined version of it.
- *ABox validation:* This task performs consistency checking, to detect possible inconsistencies due to the additions that will be performed to the ontology. Two tasks must be validated: the addition of the incoming ABox (i.e., the one that originally triggered the population activity) into the ontology and the addition of a new instance of a grouping concept or the modification of an existing one. Both types of validation can be performed through standard reasoning (inference) services.
- *ABox assimilation:* The final task is responsible for performing the needed changes in the ontology (by creating all instances/relations in all ontological modules), in order to incorporate the information in the new ABox into the ontology.

Continuing the example presented in Section 3 where the ABox obtained by the semantic interpretation process contains two explanations (Figures 7 and 8), if this ABox is processed by the evolution pattern selector, then evolution pattern P2 will be selected, triggering a population operation. The first task of population (instance matching) will try to measure the similarity of each of the explanations found in the input ABox with instances already in the ontology, explained by the same concept. For example, in the case of the first explanation which is explained by an instance of the $High\_Jump$ concept (hj_2), instance matching will employ matching techniques in order to measure the similarity of the hj_2 instance with all instances that already populate the $High\_Jump$ concept. The same will happen for the pv_2 instance, and all other instances of HLCs that may be found in the input ABox (such as $human_2$). During instance grouping, all instances that were found similar during instance matching (i.e. they had a similarity above a certain threshold) will be checked whether they represent the same real object or event, and instances that do represent the same real object or event will be grouped by associating them with an instance that represents this real object or event (and no instances will be merged or eliminated from the ontology). For example, if there was enough information to identify that instance $human_2$ refers to the same person as another instance $human_x$ already in the ontology, then both $human_2$ and $human_x$ will be retained and associated with a new instance $human_{real}$ that represents the real person (while $human_2$ and $human_x$ are seen as "participations" of this person in specific event, location and time, which may even have different properties like age). Returning to our example, due to the very limiting information available, no grouping can be performed for any of the instances, as usually grouping requires information from more than one modality (i.e. the name of the athlete usually provided by the text modality is important to identify whether two instances refer to the same athlete).

Assuming that enough instances of both $High\_Jump$ and $Pole\_Vault$ concepts exist in the ontology, the ABox refinement task may be able to disambiguate multiple explanations, by selecting one instance as most "prominant". For example, the overal similarity of the hj_2 with other instances of the $High\_Jump$ concept may be better than the overal similarity of pv_2 with other instances of $Pole\_Vault$ (i.e. due to a missing pole instance). In such a case, a single explanation (i.e. hj_2) will be selected from the available explanations. In such a case, all instances not associated with hj_2, such as pv_2, will be discarded from the input ABox. Finally, during ABox validation the resulting ABox will be checked for consistency and if it is found to be consistent, it will be assimilated to the ontology.

5 Ontology Enrichment

Ontology enrichment is the activity of extending an ontology, through the addition of new concepts and relations. Ontology enrichment is performed every time the background knowledge is not sufficient to explain the extracted information from the processed multimedia documents. Thus, the ontology enrichment activity is expected to extend this background knowledge through the addition of new concepts/relations, in order to better explain extracted information in the future.

The ontology enrichment activity is triggered by either P3 or P4 evolution patterns. Evolution pattern P3 is selected when no explanation (i.e., an HLCi) has be found for a given ABox, and can lead to the insertion of a new HLC or a new relation into the ontology, or in the accumulation of the ABox in a "waiting" queue if available evidence cannot justify the addition of a new concept/relation. On the other hand, evolution pattern P4 is selected when the background knowledge is not sufficient to even assign MLCs to all of the extracted elements of a multimedia resource, thus inserting instances of the "unknown" MLC in the ABox. In this case, pattern P4 can result in the addition of a new MLC in the ontology. In fact, the detection of new MLCs is considered to have priority over the identification of new HLCs, because knowledge about a new MLC can lead to different semantic interpretation results about the same resources. As a result, when instances of the "unknown" MLC are found in ABoxes that contain no explanations, evolution pattern P4 is selected instead of P3.

Ontology enrichment is decomposed into the following tasks:

- *Concept learning:* The goal of this task is to propose new concepts (either HLCs or MLCs) and relations by exploiting similarities found through clustering, either in unexplained documents (evolution pattern P3) or in unknown objects recognised by the information extraction engine (evolution pattern P4). It can be decomposed into two main sub-tasks, clustering and concept formation.
  - *Clustering:* The main objective of the clustering task is to provide evidence that can support the creation of new concepts or relations.
  - *Concept formation:* This task is applicable only if a new HLC has been proposed by clustering. Exploiting the results of clustering, concept formation examines the clustered elements in order to extract common information (such as concepts/properties and relations) and use this common information to form a new concept, which is the result of this task.
- *Concept enhancement (evolution pattern P3 only):* This task is responsible for improving a concept identified by concept learning, through knowledge acquired from external sources, such as external domain ontologies or taxonomies.
- *Concept definition:* This task receives the new concept (either a new MLC or HLC) or relation as defined through the previous tasks, and shows the concept/relation definition to the ontology expert. The ontology expert must approve the new concept/relation in order to be assimilated into the ontology and additionally can revise the definition of the new concept/relation.
- *Concept validation:* This task performs consistency checking, by trying to detect possible inconsistencies due to the addition of the new concept relation to the ontology.
- *Concept assimilation:* The last task of ontology enrichment is responsible for performing the needed changes in the ontology in order to incorporate the newly formed concept/relation into the ontology.

As an example, we can assume an ABox where the semantic interpretation activity was unable to found an explanation, and in addition instances of the "unknown" MLC have been extracted by the information extraction toolkit. When such an ABox is processed by the evolution pattern selector, pattern P4 will be selected. In such a case,

the ABox will be placed in a "waiting" stage. Once a significant number of instances of the "unknown" MLC have been assimilated, then clustering will be performed, during concept formation. If enough (modality specific) information is available that can lead to the formation of clusters, the concept definition task is responsible to present the results of the clustering to the ontology expert. The ontology expert must decide if the presented information is enough to justify the addition of a new MLC. In such a case, the expert must define the new concept(s), and associate all presented instances represented by the new concept with it.

In case of an ABox that has no explanation and also no instances of the "unknown" MLC, evolution pattern P3 will be selected. In such a case, the ABox will be also placed in a "waiting" stage, until enough ABoxes have been gathered. The ABox gathered are clustered in order to obtain clusters of ABoxes that are similar and all ABoxes in a cluster can possibly be explained by a single concept, which is not contained into the current version of the ontology. Once clusters have been identified, a new concept is formed for each found cluster by using all common information among all ABoxes of the cluster. Each newly formed concept will be further enhanced during concept enhancement, by trying to locate the formed concept in external knowledge sources through coordination and exploiting information from these knowledge sources, like concept/relation names and properties. Once new concepts/relations have been formed and possibly enhanced, they must be approved and reviewed by the ontology expert during the concept definition task: the expert must decide which of these proposed concepts/relations will be kept, what their definition will be and which ABoxes can be associated with them. Each concept/relation definition must be checked for consistency and assimilated into the ontology, if no inconcistencies have been found.

6 Original Contributions

The recent success of distributed and dynamic infrastructures for knowledge sharing has raised the need of semiautomatic/automatic ontology evolution strategies [5, 6]. An overview of some proposed approaches in this direction is presented in [7], even if limited concrete results have appeared in the literature. In most recent work, formal and logic-based approaches to ontology evolution are also being proposed. In [8], the authors provide a formal model for handling the semantics of change phase embedded in the evolution process of an OWL ontology. The proposed formalization allows to define and to preserve arbitrary consistency conditions (i.e., structural, logical, and user-defined).

A six-phase evolution methodology has been implemented within the KAON [9] infrastructure for business-oriented ontology management. The ontology evolution process starts with the capturing phase, that identifies the ontology modifications to apply either from the explicit business requirements or from the results of a change discovery activity. In the representation phase, the identified changes are described in a suitable format according to the specification language of the ontology to modify (e.g., OWL). The effects of the changes are evaluated in the semantics of change phase, where the ontology consistency check is also performed. Due to the fact that an ontology can reuse or extend other ontologies (e.g., through inclusion or mapping), the propaga-

tion phase ensures that any ontology change is propagated to the possible dependent artifacts in order to preserve the overall consistency. The subsequent implementation phase has the role to log all the performed changes in order to support the recovery facilities that in the final validation phase are provided to reverse an ontology change in case that an undesired effect occurs.

With respect to the state of the art literature on ontology evolution, original contributions of BOEMIE can be seen at two different levels, the whole methodology and the specific activities. The methodology as a whole proposes a new conceptualization of the problem of evolving multimedia ontologies, by presenting a pattern-driven evolution approach, where the most prominent evolution pattern for a specific evolution scenario is automatically identified on the basis of the results of the semantic interpretation activity against the background knowledge. Moreover, the methodology aims to minimize the human involvement by providing a set of learning, matching, and reasoning techniques that offer support in the various evolution activities, to allow the ontology expert to refine proposed working knowledge and/or to validate/choose among proposed alternative choices. Concerning novel contributions at the level of specific activities, the methodology for ontology population uses an innovative approach for the detection of instances which refer to the same real object or event, based on instance matching and non-standard clustering techniques. For ontology enrichment, clustering is used for detecting enough information to support the introduction of a new concept/relation and this supporting information is enhanced though information retrieved from external knowledge sources. Thus, the involvement of the ontology expert is reduced, as the expert is required to revise an already formed concept/relation than define this concept/relation from scratch. Matching techniques are used in combination with reasoning and clustering techniques in BOEMIE, thus leading to the development of a more flexible and comprehensive approach to concept/instance matching for evolution, by enforcing both structural matching and semantic matching.

The ontology evolution approach proposed by BOEMIE puts significant effort in maintaining the consistency of the ontology while trying on the same time to identify and eliminate redundant information. With respect to the state of the art in the field of knowledge representation and reasoning, the novel contribution of the evolution methodology regards the following issues: i) formalization of high-level multimedia interpretation as a logical decision problem and its implementation as a non-standard inference service, namely abduction; ii) extension of the knowledge representation formalism (SWRL) in order to support adequate knowledge representation for abduction tasks; iii) development of new optimization techniques for ABox consistency checking and query answering and its integration into a state-of-the art DL reasoner; iv) first approach for using non-standard description logic inference problems for formalizing the learning problem in BOEMIE (e.g., LCS, MSC, rewriting).

7 Concluding Remarks

In this paper, we have presented the methodology for multimedia ontology evolution developed in the context of the BOEMIE project. We have specified how the semantic interpretation of the information extracted from multimedia resources can be used to

achieve the coordinated and consistent evolution of the ontology. We have described different evolution scenarios triggering population or enrichment patterns over the ontology. Current and future work within BOEMIE is devoted to: i) the investigation of instance matching techniques to support ontology population and to complement the role of reasoning techniques in the resource interpretation activity; ii) the investigation of clustering techniques to extract common information (such as concepts and relations) from ABoxes and use this common information to form new concepts; iii) the implementation of an ontology evolution toolkit providing an interactive environment where all the proposed techniques will be integrated into a coherent whole according to an open architecture.

References

1. Alani, H., Kim, S., Millard, D., Weal, M., Hall, W., Lewis, P., Shadbolt, N.: Automatic ontology-based knowledge extraction from web documents. IEEE Intelligent Systems **18** (2003) 14–21
2. Buitelaar, P., Cimiano, P., Racioppa, S., Siegel, M.: Ontology-based information extraction with soba. In: Proceedings of the International Conference on Language Resources and Evaluation. (2006) 2321–2324
3. Shanahan, M.: Perception as Abduction: Turning Sensor Data into Meaningful Representation. Cognitive Science **29** (2005) 103–134
4. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies. In: Proc. OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11. (2006)
5. Haase, P., Sure, Y.: State-of-the-art on ontology evolution. SEKT informal deliverable 3.1.1.b, Institute AIFB, University of Karlsruhe (2004)
6. Klein, M., Noy, N.: A component-based framework for ontology evolution (2003)
7. Ding, Y., Foo, S.: Ontology research and development. part 2 - a review of ontology mapping and evolving. Journal of Information Science **28** (2002) 375–388
8. Haase, P., Stojanovic, L.: Consistent evolution of owl ontologies. In Gómez-Pérez, A., Euzenat, J., eds.: ESWC. Volume 3532 of Lecture Notes in Computer Science., Springer (2005) 182–197
9. Oberle, D., Volz, R., Motik, B., Staab, S.: An extensible ontology software environment. In Staab, S., Studer, R., eds.: Handbook on Ontologies. International Handbooks on Information Systems. Springer (2004) 311–333
10. Kim, S., Alani, H., Hall, W., Lewis, P., Millard, D., Shadbolt, N., Weal, M.: Artequakt: Generating tailored biographies from automatically annotated fragments from the web. In: Proceedings of Workshop on Semantic Authoring, Annotation & Knowledge Markup (SAAKM02), the 15th European Conference on Artificial Intelligence, (ECAI02), Lyon, France (2002) 1–6

Text-based ontology construction using relational concept analysis

Rokia Bendaoud, Mohamed Rouane Hacene, Yannick Toussaint, Bertrand Delecroix, and Amedeo Napoli

UMR 7503 LORIA, BP 239, 54506 Vandœuvre-lès-Nancy, FRANCE

Abstract. We present a semi-automated process that constructs an ontology based on a collection of document abstracts for a given domain. The proposed process relies on the formal concept analysis (FCA), an algebraic method for the derivation of a conceptual hierarchy, namely '*concept lattice*', starting from data context, i.e., set of individuals provided with their properties. First, we show how various contexts are extracted and then how concepts of the corresponding lattices are turned into ontological concepts. In order to refine the obtained ontology with transversal relations, the links between individuals that appear in the text are considered by the means of a richer data format. Indeed, Relational Concept Analysis (RCA), a framework that helps FCA in mining relational data is used to model these links and then inferring relations between formal concepts whose semantic is similar to roles between concepts in ontologies. The process describes how the final ontology is mapped to logical formulae which can be expressed in the Description Logics (DL) language $\mathcal{FLE}$. To illustrate the process, the construction of a sample ontology on the astronomical field is considered.

1 Introduction

Knowledge systems are of great importance in many fields, since they allow knowledge representation, sharing and reasoning. However, the knowledge acquisition process is complex and can be seen as a "*bottleneck*" [12]. The difficulty is to acquire knowledge (especially from experts) and then to maintain knowledge in a given domain. For example, in the area of astronomy, assigning classes to the growing number of celestial objects is a difficult task and leads to a large number of classes. Traditionally, this classification task is performed manually according to the object properties appearing in the astronomy documents. The task consists in reading articles of various sources that deal with a given celestial objects and finding the corresponding class. At present, more than three million celestial objects were classified in this way and made available through the SIMBAD database[1], but considerable work has to be done in order to classify the billion remaining objects. Moreover, human experts are not confident with the resulting classification as the classes lack precise definitions to be examined when a new object must be classified.

[1] http://simbad.u-strasbg.fr/simbad/sim-fid

The spread of languages and frameworks for building ontologies, mainly within the Semantic Web initiative, has turned current trends in classification towards the construction of classification in the form of ontologies [15]. Ontologies are an explicit specification of a domain conceptualization, developed for the purpose of sharing and reuse. It comprises a set of concepts and a set of taxonomic and transversal relations. In attempt to bring a formal representation to the ontology components (concepts, roles, etc.), several studies [8] have documented the mapping of an ontology into DL formulae. Such translation is crucial as it makes the domain knowledge encoded by the means of ontology at the disposal of DL reasoners which in turn enables sharing and reasoning on a clear semantic basis.

The aim of this paper is to introduce a semi-automated process for the construction of classifications in the form of ontologies [15] and the derivation of expressions in Description Logics (DL) that formally describes the resulting classes. Several approaches were proposed for ontology construction, such those relying on Formal Concept Analysis (FCA) [3]. FCA is a mathematical approach for abstracting conceptual hierarchies from set of individuals (e.g., celestial objects, telescopes, etc.) and the set of their properties (e.g., emitting, collimated, mass, etc). These individuals and their properties are extracted from text corpora using NLP tools. Applying FCA with the aim of ontology construction brings forward two main benefits. First, the formal characterization of the FCA-powered concept hierarchy provides a basis for a formal specification to the derived ontology. Moreover, many efficient operations have been designed in FCA to maintain the concept hierarchy over data evaluation, such as those performing an incremental update of the hierarchy by adding either a formal object or a formal attribute and those operations for lattice assembly from parts [13]. These various operations could be used to solve the 'bottleneck' problem in knowledge acquisition. Indeed, when the concept hierarchy changes, the ontology will evolve and still be correct and consistent.

However, in order to deal with complex descriptions of individuals that go beyond a mere conjunction of properties, an extended FCA framework, namely 'Relational Concept Analysis' (RCA) is used to derive conceptual hierarchies where, beside property sharing, formed concepts reflect commonalties in object links [5]. RCA approach lifts up links between individuals to the rank of relations between concepts whose meaning is similar to roles in ontologies. RCA output — concepts organized by a partial order relation — is translated in a very obvious way to an ontology components [9]. Moreover, recent advances in combining RCA and DL languages have shown how RCA output, in particular concepts provided with relational descriptions, can be expressed in the form of DL formulae ranging in the $\mathcal{FLE}^2$ language family [7].

The proposed process is fed with astronomy data to classify celestial objects. The translation of the ontology into a DL knowledge base (KB) allows querying the KB through a DL reasoner and thus answering to '*competency questions*'.

[2] DL language that comprises the following constructors: conjunction $\sqcap$, universal quantification $\forall$ and existential quantification $\exists$.

These questions are first written in natural language and then translated into the DL language. Competency questions look like '*do objects* `M87` *and* `PSRA` *belong to the same class*?', '*Which objects can be observed with an Xray telescope*?', or '*What are the objects that* `MXX-Newton` *observes*?', etc.

The paper starts with an overview of the proposed methodology that builds a domain ontology based on free text. The next section introduces the processing texts with NLP tools that are used to collect RCA data. Section 4 recalls the FCA method, its extended framework RCA, and their application to the domain of astronomy. Section 5 presents the translation of the RCA output into DL KB. First, general rules are listed and then applied to the result of the previous step. We present in the section 6 the related work and conclude with brief discussion on the learned facts and the remaining open issues.

2 Methodology

Our methodology (described in figure 1) is based on "Methontology" [1]. The "Methontology" is a semi-automatic methodology, that builds an ontology from a set of terms extracted from resources (the resources are not specified). The objective is to find the exhaustive definition for each concept and each relation of the ontology in DL language. The four steps of the "Methontology" are adapted on proposed methodology.

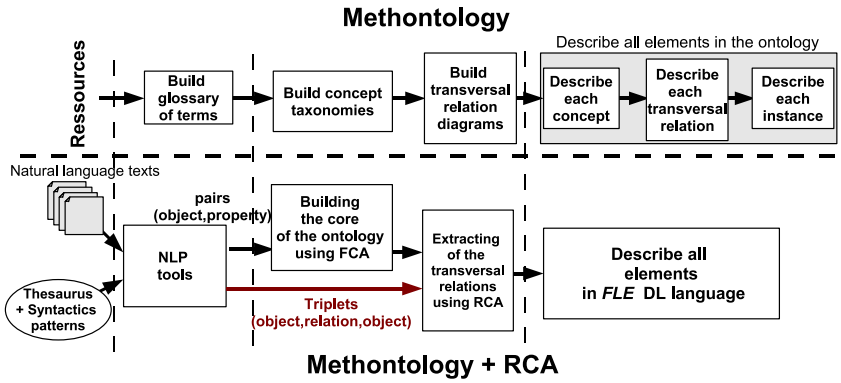

Fig. 1. Mapping between the "Methontology" and Methodology + RCA

Resources: They are represented by the texts corpora, the thesaurus of astronomy[3] and the syntactic patterns[4] such as: all `NGC nnnn` where n is a number represents one celestial object.

[3] http://msowww.anu.edu.au/library/thesaurus/
[4] http://simbad.u-strasbg.fr/simbad/sim-fid

Build glossary of terms: The extraction of the terms is done from the texts corpora using the existing resources in the astronomical domain. We extract also in this step the pairs (object,property) and the tuples (object,relation,object) using Natural language processing (NLP) tools.

Build concept taxonomies: We propose in this step to use the FCA. The FCA is the mathematical tool (presented in the section 4) that builds the hierarchy of concepts by grouping the terms sharing the same properties.

Build transversal binary relation diagrams: The extraction of the transversal relations is done in the same time as the construction of new hierarchy of concepts taking into account their properties and also their links with other objects. This step is done with RCA (see the section 4).

Describe all elements of the ontology: The representation of all concepts, relations and instances is done with $\mathcal{FLE}$ language. The representation in a DL language is done to support reasoning, i.e. classification, instantiation and consistency checking (see the section 5).

3 Processing texts with NLP tools

We want to extract the pairs (object,property) and the tuples ($object_1$,link,$object_2$) from the text corpora. The links, in the tuples, are used to define the set of the relations in the ontology (see section 4.2). We choose to use the Faure's approach [4] based on the Harris hypothesis [16]. This hypothesis studies the syntactic regularities in the text corpora of sub-languages (or specific languages), allowing to identify the syntactic schema to build classes. There classes are grouping the terms (celestial objects) that are arguments of the same set of verbs, i.e., the subject of the same set of verbs and the complement of the same set of verbs. For example: The set {HR5223, PRSA, SS433} are in the same class because they are appearing as subject with the verb {to emit} and as complement with the set of verbs {to observe,to locate}. The set of verbs is translated to the set of properties, like for example if one term are subject of the verb "to emit", it has a property "emitting" and if one term are complement of the verb "to observe", it has the property "observed". We use the same approach to extract the set of links, if $object_1$ is the subject of the verb V and the $object_2$ the complement of the verb V then we extract the tuple ($object_1$,VP,$object_2$) where VP is the verb phrase which represent the link between ($object_1$,$object_2$).

The parsing of the corpus is done with the shallow parser "Stanford Parser"[5] [6]. We give two examples in the astronomic domain:

1. *"One HR2 candidate was* `detected` *and* `regrouped` *in each of the galaxies NGC 3507 and CygnusA"*. We extract the pairs: (HR2, `regrouped`), (HR2, `detected`), (NGC 3507, `regrouping`), (CygnusA, `regrouping`).

[5] http://nlp.stanford.edu/software/lex-parser.shtml

2. *'The XMM-Newton X-ray telescope observed the bursting pulsar M87"*, the extraction process will first identify `XMM-Newton X-ray` as a Telescope, and `M87` as a `celestial object`. We extract the tuple : (`M87`, Observed-ByXRay,`XMM-Newton X-ray`).

4 Background on concept lattices

4.1 Basics of FCA

FCA is a mathematical approach to data analysis based on lattice theory. The basic data format in FCA [3] is a binary table $\mathcal{K} = (G, M, I)$ called *formal context*, where G is a set of individuals (called *objects*), M a set of properties (called *attributes*) and I the relation "has" on $G \times M$. Table in the left-hand side of Fig. 2 represents an example of context. Here, G is the set of `celestial objects` and M the set of their properties. A pair (X, Y) where X is a maximal set of individuals (called *extent*) and Y is a maximal set of shared properties (called *intent*), is called a *formal concept*. For instance, $(\{Andromeda, NGC3507\}, \{observed, grouping\})$ is a concept (see diagram in the right hand side of Fig. 2).

Celestial objects	emitting	accreting	collimated	observed	located	grouping
PSRA	X			X	X	
NGC3507				X		X
Andromeda		X		X		X
M87			X	X	X	
HR2				X	X	
NGC2018			X	X	X	
HR5223	X			X	X	
SS433	X			X	X	

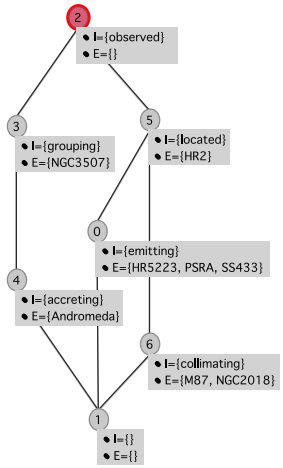

Fig. 2. The binary context of celestial objects and the corresponding concept lattice.

Furthermore, the set $\mathcal{C}_\mathcal{K}$ of all concepts of the context $\mathcal{K} = (G, M, I)$ is partially ordered by extent inclusion also called the *specialization* (denoted $\leq_\mathcal{K}$) between concepts. $\mathcal{L} = \langle \mathcal{C}_\mathcal{K}, \leq_\mathcal{K} \rangle$ is a complete lattice, called the *concept lattice*. Fig. 2 illustrates a context and its corresponding lattice. A simplified (or reduced) labeling schema is often used where each object and each attribute appear only once on the diagram. The full extent of a concept is made up of all objects

whose labels can be reached along a descending path from the concept while its full intent can be recovered in a dual way (ascending path). For details on the construction of concept lattices, see [3].

As many practical applications involve non-binary data, *many-valued contexts* has been introduced in FCA where individuals have value associated to properties. The construction of a lattice for this kind of contexts requires a preprocessing step, called *conceptual scaling* [3], that derives a binary context out of many-valued one. Scaling turns a non-binary attribute into a set of binary ones representing abstractions of values on the domain of the underlying non-binary attribute. For instance, the values of non-binary attribute *orbitalPeriod* in the context illustrated in Tab. 1 could be distributed on the ranges *short* and *long*, each of them expressed as a predicate (e.g., *orbital period* $\leq$ 24 *hours* for short one). Observe that the definition of the predicates precedes the scaling task and is usually in charge of a domain expert.

4.2 From FCA to RCA

Relational Concept Analysis (RCA)[5] was introduced as an extended FCA framework for extracting formal concepts from sets of individuals described by '*local*' properties and links. In RCA data are organized within a structure called '*relational context family*' (RCF). RCF comprises a set of contexts $\mathcal{K}_i = (G_i, M_i, I_i)$ and a set of binary relations $r_k \subseteq G_i \times G_j$, where G_i and G_j are the object sets of the contexts $\mathcal{K}_i$ and $\mathcal{K}_j$, called *domain* and *range*, respectively. For instance, table in Fig. 2 and Tab. 1 depict a sample RCF made of two contexts, `celestial objects` context and `telescopes` context.Two inter-context relations, '*Observed By Xray*' (OBXray) and '*Observed By Infrared*' (OBInfrared) indicate the observation links between telescopes and objects.

The relationnal and non relationnal attributes in both contexts list the features of objects such as the orbit height (perigee) and the orbital period for telescopes and emitting or grouping faculty for the `celestial objects`.

Telescopes			
	perigee	orbitalPeriod	mass
BeppoSAX	600 *km*	96 *min*	1400 *kg*
XMM-Newton	114000 *km*	48 *hours*	3800 *kg*
Chandra	26300 *km*	66 *hours*	1790 *kg*

OBXray			
	BeppoSAX	XMM-Newton	Chandra
M87		X	
NGC2018			X

OBInfrared			
	BeppoSAX	XMM-Newton	Chandra
HR5223	X		
SS433	X		

Table 1. Sample RCF encoding astronomy data.

RCA uses the mechanism of '*relational scaling*' which translates domain structures (concept lattices) into binary predicates describing individual subsets. Thus, for a given relation r which links formal objects from $\mathcal{K}_i = (G_i, M_i, I_i)$

to those from $\mathcal{K}_j = (G_j, M_j, I_j)$, new kind of attributes, called '*relational attributes* are created and denoted by $r{:}c$, where c is concept in $\mathcal{K}_j$. For a given object $g \in G_i$, relational attribute $r : c$ characterizes the correlation of $r(g)$ and the extent of $c = (X, Y)$. Many levels of correlation can be considered such as the 'universal' correlation $r(g) \subseteq X$ and the 'existential' correlation $r(g) \cap X$. Due to correlation constraint, existential encoding of object links yields to richer link sharing among objects and thus a wider conceptual structure to explore when mining relevant concepts. In the present work, we consider only existential scaling.

	perigeeIsLow	perigeeIsHigh	shortOrbitalPeriod	longOrbitalPeriod	light	heavy
BeppoSAX	X		X		X	
XMM-Newton		X		X		X
Chandra		X		X	X	

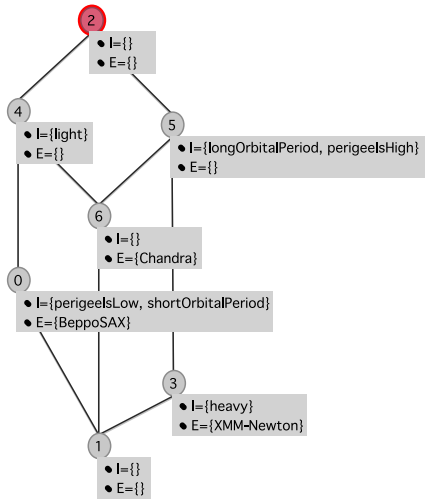

Fig. 3. The derived context of telescopes and the corresponding lattice.

For example, suppose that the context of celestial objects has to be scaled along the relation OBXray with respect to the lattice given in Fig. 3. As OBXray(M87) = {XMM_Newton} and the telescope XMM_Newton is present in the extent of concepts c_2, c_3 and c_5 (see Fig. 3), the celestial objects context is extended by relational attributes of the form $r : c_i$, where $i = \{2, 3, 5\}$. Tab. 2 depicts the extended context of celestial objects after the scaling of both relations OBXray and OBInfrared. It can be noticed that beside local attributes, new relational attributes encode object links that have been assigned to objects. For instance, in Figure 4, objects HR5223 and SS433 in the concept c_9 share the attribute OBInfrared:c0 which is interpreted as a common link with telescope BeppoSAX (the only object in the extent of concept c_0 of Figure 3).

	Local attributes						Relational attributes													
	emitting	accreting	collimated	observed	located	grouping	OBxray:c_0	OBxray:c_1	OBxray:c_2	OBxray:c_3	OBxray:c_4	OBxray:c_5	OBxray:c_6	OBinfrared:c_0	OBinfrared:c_1	OBinfrared:c_2	OBinfrared:c_3	OBinfrared:c_4	OBinfrared:c_5	OBinfrared:c_6
HR5223	X			X	X									X		X		X		
M87			X	X	X				X	X		X								
SS433	X			X	X									X		X		X		
NGC2018			X	X	X				X		X	X	X							

Table 2. The result of scaling of celestial objects context along its relations. Formal objects that are not affected by relational scaling are not displayed.

4.3 Qualitative interpretation of RCA

The relational scaling is the key step in a process which, given an RCF, derives a relational lattice family (RLF), one lattice by context. A relational attribute is interpreted as a relation between two concepts, on the first side the concept whose intent owns this attribute (i.e. the domain), and, on the other side, the concept indicated in the relational attribute expression (i.e. the range). The RLF extraction process is iterative since relational scaling modifies contexts and thereby the corresponding lattices, which in turn, implies a re-scaling of all the relations that use these lattices as source of predicates. This iterative process stops when a fixed point is reached, i.e., additional scaling steps do not involve any more context extension.

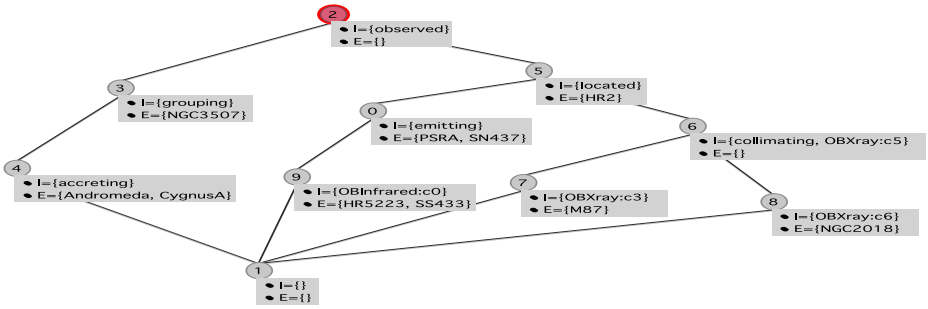

Fig. 4. The final relational lattice of celestial objects context

The analysis of the sample RCF using RCA process yields to the concept lattices illustrated in Fig. 3 and Fig. 4. Relational attributes in concept intents are associated to the most specific concepts in the corresponding lattice. Telescope

context is not a domain of relation in the running RCF. Therefore, the final lattice corresponds to the initial one shown in Fig. 3. By contrast, the lattice of celestial objects context has changed. The resulting concepts trigger yet further sharing, at the object links level. Indeed, the intents of various formal objects are enriched with relational attributes encoding inter-object links. These attributes lift up object link to relations between concepts. For example, the concept c_6 in Fig. 2 represents the celestial objects `M87` and `NGC2018`, that are both binary stars as they are observed, located and collimated. The intent of the former concept is encoded with the relational attribute `OBXray:`c_5, meaning binary stars are also observable by XRay telescopes. Moreover, new concept are discovered. For example, even if the two celestial objects `HR5223` and `SS433` have already composed a formal concept in the initial lattice (concept c_0 in Fig. 2) with an additional object, namely `PSRA` they let a new concept emerge in the final lattice (concept c_9 in Fig. 4), due to the common link they share with the telescope `BeppoSAX`. The new concept represents the stars that are observable with an Infrared telescope such as `BeppoSAX`.

5 Ontology derivation

The ontology resulting from the RCA process is represented with the DL $\mathcal{FLE}$.

The TBox		
RCA entity	Ontology	Example
Context $\mathcal{K}$	Atomic concept c $\equiv \alpha(\mathcal{K})$	α(`Telescope`)$\equiv$ Telescope
Formal attribute $m \in M$	Defined concept c $\equiv \alpha(m) \equiv \exists m.\top$	α(`observed`)=Object $\equiv$ $\exists$observed.$\top$
Concept $c = (X, Y) \in \mathcal{C}$	Defined concept $\alpha(c)$, i.e. $\alpha(c) \equiv \sqcap_{m \in Y} \alpha(m)$	$\alpha(\mathsf{C}_5) \equiv \exists$observed.$\top \sqcap \exists$located.$\top$
$\forall (c, \bar{c}) \in \mathcal{C} \times \mathcal{C}$, i.e. $c \prec \bar{c}$	Inclusion axiom $\alpha(c) \sqsubseteq \alpha(\bar{c})$	$\alpha(\mathsf{C}_8) \sqsubseteq \alpha(\mathsf{C}_6)$
Relation $r \in R$	primitive role $\alpha(r)$	OBXray is a primitive role in the TBox
Relational attribute $r.C$	Atomic concept c $\equiv \alpha(r) \equiv \exists r.\alpha(c)$	α(OBXray.XMM-Newton)$\equiv$ $\exists$OBXray.XMM-Newton
The ABox		
RCA entity	Ontology	Example
Formal object $g \in G$	Instance $\alpha(g)$	Andromeda is an instance
Element $(g, m) \in I$	Assertion $\alpha(m)(\alpha(g))$	Object(HR2)
Let $c = (X, Y)$, $\forall g \in X$	Concept instantiation $\alpha(c)(\alpha(g))$	HR2 is an instance of the concept `Star`

Table 3. Mapping between lattice and DL knowledge base

The translation between the RCA formal concepts and relations and the DL $\mathcal{FLE}$ is carried on using a function α defined as follows: $\alpha : (\mathbf{K}, \mathbf{R}) \rightarrow$ TBox $\sqcup$

ABox, where: $(\mathbf{K}, \mathbf{R})$ is a family RCF, TBox and ABox being the components of the ontology. The function α is presented in the Tab. 3. The application of the function α in the two lattices (Fig. 3 and Fig. 4) results in the ontology in the Fig.5.

5.1 The translation of the concepts lattice into the ontology

The translation of each context represents an atomic concept, that express the top $\top$ of the hierarchy in this context. Each formal attribute is translate in defined concept. For example, attribute `observed` is translated into the concept $\mathsf{c} \equiv \exists\, \mathsf{observed}.\top$. Each relational attribute r.C is translated in defined concept in the TBox. For example, the relational attribute if the form OBXray.BeppoSAX is translated into $\mathsf{c} \equiv \exists\, \mathsf{OBXray.BeppoSAX}$, etc.

The design of the ontology is carried out in collaboration with astronomers. The astronomers have to give a label to each concept in the ontology according to the properties and the links associated to the instances of a concept. For example, the class of objects having the set of properties {`observed,located,collimating`} and the link {`Observed-By-Xray`} with the range `X-Ray-Telescope` is labeled by `Binary-Star`. The class of objects having the set of properties {`observed,located,emitting`} and the relation {`Observed-By-Infra-Red`} with the range `Infra-Red-Telescope` is labeled by `Pulsing-Variable-Star`: `Infra-Red-Telescope` observes `Young-Star` that has a large emission compared with the `X-Ray-Telescope` that observes older stars like `Binary-Star`. This representation is done only to give one label for each set of celestial objects and to help the experts to read the ontology.

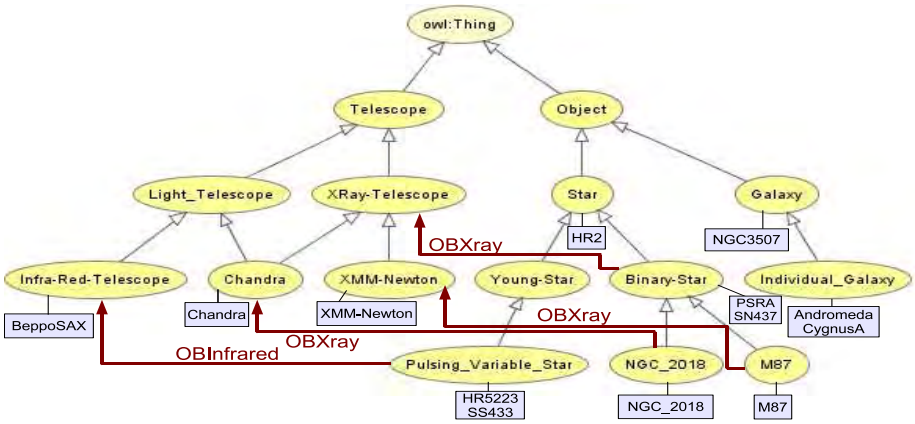

Fig. 5. Complete Ontology

5.2 Representation of the concepts in the DL language $\mathcal{FLE}$

The ontology is represented within the $\mathcal{FLE}$ language. Tab. 4 presents the definition of each concept in the ontology presented in the figure (Fig. 5). The ontology can be used for three kinds of tasks :

N° in the lattice	Concept Name	Defined Concept
C_2	Object	$\exists$observed.$\top$
C_5	Star	$\exists$observed.$\top \sqcap \exists$located.$\top$
C_0	Young-Star	$\exists$observed.$\top \sqcap \exists$located.$\top \sqcap \exists$emitting.$\top$
C_9	Pulsing-Variable-Star	$\exists$observed.$\top \sqcap \exists$located.$\top \sqcap \exists$emitting.$\top \sqcap \exists$OBInfrared. Infra-Red-Telescope
C_6	Binary-Star	$\exists$observed.$\top \sqcap \exists$located.$\top \sqcap \exists$collimated.$\top \sqcap \exists$OBXray.Xray_Telescope
C_7	M87	$\exists$observed.$\top \sqcap \exists$located.$\top \sqcap \exists$collimated.$\top \sqcap \exists$OBXray.XMM-Newton
C_8	NGC_2018	$\exists$observed.$\top \sqcap \exists$located.$\top \sqcap \exists$collimated.$\top \sqcap \exists$OBXray.Chandra
C_3	Galaxy	$\exists$observed.$\top \sqcap \exists$grouping.$\top$
C_4	Individual-Galaxy	$\exists$observed.$\top \sqcap \exists$grouping.$\top \sqcap \exists$accreting.$\top$
T_2	Telescope	Telescope
T_4	light_Telescope	$\exists$light.$\top$
T_5	XRay-Telescope	$\exists$longOrbitalPeriod.$\top \sqcap \exists$perigeeIsHight.$\top$
T_0	Infra-Red-Telescope	$\exists$shortOrbitalPeriod.$\top \sqcap \exists$perigeeIsLow.$\top$
T_6	Chandra	$\exists$longOrbitalPeriod.$\top \sqcap \exists$perigeeIsHight.$\top \sqcap \exists$light.$\top$
T_3	XMM-Newton	$\exists$longOrbitalPeriod.$\top \sqcap \exists$perigeeIsHight.$\top \sqcap \exists$heavy.$\top$

Table 4. Definition of each concept of the Fig 5 in $\mathcal{FLE}$

1. **Ontology population:** Let o_1 an object with the properties {a,b}, and the relations {r_1.c_1,r_2.c_2}. A first task is instantiation,i.e. to find the class of an object such as o_1. The class of o_1 is the most general class X such that X $\sqsubseteq \exists$a.$\top \sqcap \exists$b.$\top \sqcap \exists r_1.c_1 \sqcap \exists r_2.c_2$. For example, let us consider the question *"What is the class of the object GRO, that has the properties* {observed,located,emitting} and the relation OBInfrared with the range Infra-red-Telescope? The answer is: the most general class X $\sqsubseteq \exists$observed.$\top \sqcap \exists$located.$\top \sqcap \exists$emitting.$\top \sqcap \exists$OBInfrared. Infra-red-Telescope. This class in the ontology is the concept Pulsing-Variable-Star.
2. **Comparison of celestial objects:** Let us consider two objects o_1 and o_2. A second task consists in comparing o_1 and o_2 and determining whether o_1 and o_2 have the same class. One way for checking that is to find the

class of o_1, then the class of o_2, and then to test whether the two classes are equivalent. For example, let us consider the two objects `M87` and `PSRA`. `M87` is an instance of the class `M87` and `PSRA` is an instance of the class `Young-Star`. Knowing that `M87` $\sqcap$ `Young-Star` $= \bot$, it can be inferred that both objects do not belong to the same class.

3. **Detection of the domain or the range of relation:** Let us consider the relation r_1 with the range C_1. A third task consists in finding the domain of the relation r_1. The domain of r_1 is the most specific class X such that X is the most specific class, union of all the classes linked to the class C_1 by the relation r_1. For example *Which objects can be observed by Xray with a Xray telescope*? The most specific class domain of the relation `observed by Xray` where `Xray telescope` is the range, is the concept `Binary-star`.

6 Related work

6.1 Building the core ontology

There are two main approaches for building ontologies from text corpora. The first one is based on the co-occurrence of terms in text and on the use of similarity measures for building the hierarchy of the objects classes [10]. This approach can not satisfy our needs to give a definition to each concept of the hierarchy, because every concept is represented by numeric vector and it is difficult to find an interpretation for each vector. The second approach is symbolic, and is based on the use of a syntactic structure to describe an object by the verb with which it appears. Faure uses this structure for building the object classes and the statistic measures for building the hierarchy of the classes [4]. Cimiano uses the same approach but builds the hierarchy of classes using FCA, without taking into account the relations between objects [12].

6.2 Extracting the transversal relations

The extraction of transversal relations allows us to have a better definition of each concept. The concepts are not only defined by their properties but also by their relations with other concepts. We cite two related approaches in the extraction of relations. The first one is the work of Aussenac-Gilles [11], who proposes to use a learning method to extract syntactic patterns. Tuples manually extracted from the texts (`term`$_1$`, relation`$_1$`, term`$_2$`)` are the inputs. All the tuples (`term`$_1$`, relation`$_k$`, term`$_2$`)` are searched to build a general relation R, such that $R = relation_1 \sqcup \ldots \sqcup relation_n$. Then, tuples of the form $(term_i, R, term_j)$ are extracted. This method groups the set of objects according to the relations that they share, and extracts the general relations between two concepts. It does not use the hierarchy of the concepts to make a generalization. A second approach by Maedche and Staab [2] consists in extracting the association rules [14] ($term_1 \Rightarrow term_2$) and in keeping only those rules having a given support and frequency. This method finds all the pairs (C_1, C_2) linked by one relation but does not specify the name of the relation between these pairs.

7 Conclusion

A method for building an ontology from text corpora was proposed. The method uses the RCA framework that extends standard FCA for mining relational data. RCA derives a structure that is compatible with an ontology. We have shown how RCA output could be represented in terms of DL expressions ranging in the $\mathcal{FLE}$ DL family. The proposed method was applied to the astronomy domain in order to extract knowledge about celestial objects that can be used through a DL reasoner for problem-solving such as celestial objects classification and comparison. The construction of a first prototype ontology from astronomy data proved that RCA-based ontology construction is a promising method allowing to data mining and knowledge representation techniques.

On going work consists in improving the RCA input data gathering process by considering alternate syntactic patterns in the extraction of object pairs such as (subject, verb), (complement, verb), (subject, adjective), etc. These new sorts of pairs will provide a contexts with additional formal attributes that make formal object descriptions richer as well as a new inter-context relations. Eventually, the construction of hierarchy of relations need to be addressed. The principle consists of using once again the RCA abstraction process to introduce abstract relations between concepts based on the transversal relations —originally inferred from instances links— that hold among their subsumers. Once the derived relation hierarchy merged with concept hierarchy, the resulting structure forms a complete ontology that fully captures the domain knowledge.

References

1. Gómez-Pérez A., M. Fernàndez-López, and O. Corcho. *Ontological Engineering*. Springer Verlag, 2004.
2. Maedche A. and S. Staab. Discovering conceptual relation from text. In *Proceeding of the 14th European Conference on artifical intelligence*, pages 321–325, Berlin, Germany, 2000.
3. Ganter B. and R. Wille. *Formal Concept Analysis Mathematical Foundations*. Springer Verlag, 1999.
4. Faure D. and C. Nedellec. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In *The LREC workshop on Adapting lexical and corpus reesources to sublanguages and applications*, Granada, Spain, 1998.
5. M. Dao, M. Huchard, M. Hacene Rouane, C. Roume, and P. Valtchev. Improving generalization level in uml models: Iterative cross generalization in practice. In *Proceedings of the 12th International Conference on Conceptual Structures (ICCS'04)*, volume 3127 of *Lecture Notes in Computer Science*, pages 346–360, Huntsville, AL, July 2004. Springer-Verlag.
6. Marneffe M.C. de., B. MacCartney, and C.D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*, GENOA, ITALY, 2006.
7. Baader F. Description logic terminology. In Baader F., D. Calvanese, D. McGuinness, N. Daniele, and P.F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 485–495. Cambridge University Press, 2003.

8. Baader F., I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In Hutter D. and W. Stephan, editors, *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, volume 2605 of *Lecture Notes in Artificial Intelligence*, pages 228–248. Springer-Verlag, 2005.
9. Rouane M. H., M. Huchard, A. Napoli, and P. Valtchev. Proposal for combining formal concept analysis and description logics for mining relational data. In *Int. Conference on Formal Concept Analysis, ICFCA 2007, Clermont-Ferrand, France*, Lecture Notes in Computer Science. Springer Verlag, 2007.
10. Sanderson M. and B. Croft. Deriving concept hierarchies from text. In *Research and Development in Information Retrieval*, pages 206–213, 1999.
11. Aussenac-Gilles N., B. Biébow, and S. Szulman. Revisiting ontology design: A method based on corpus analysis. In Dieng R. and O. Corby, editors, *12th Int. Conference in Knowlegde Engineering and knowledge Management (EKAW'00)*, volume 1937, pages 172–188, 2000.
12. Cimiano P., A. Hotho, and S. Staab. Learning concept hierarchies from text corpora using formal concept analysis. In *Journal of Artificial Intelligence Research (JAIR)*, volume Volume 24, pages 305–339, 2005.
13. Valtchev P., M. Rouane Hacene, and R. Missaoui. A generic scheme for the design of efficient on-line algorithms for lattices. In A. de Moor, W. Lex, and B. Ganter, editors, *Proceedings of the 11th Intl. Conference on Conceptual Structures (ICCS'03)*, volume 2746 of *Lecture Notes in Computer Science*, pages 282–295, Berlin - Germany, 2003. Springer.
14. Agrawal R. and R. Srikant. Mining generalized association rules. In *21st VLDB Conference*, Zurich, Swizerland, 1995.
15. Gruber T.R. Toward principales for the design of ontologies used for knowledge sharing. In *Formal Analysis in Concepttual Analysis and Knowledge Representation*, 1993.
16. Harris Z. *Mathematical Structure of Language*. Wiley J. and Sons, 1968.

Base Revision in Description Logics - Preliminary Results

Márcio Moretto Ribeiro and Renata Wassermann

Institute of Mathematics and Statistics
University of São Paulo – Brazil,
{marciomr,renata}@ime.usp.br

Abstract. Belief Revision deals with the problem of adding new information to a knowledge base in a consistent way. The theory has been developed having in mind classical logics. In this paper, we show some problems of applying belief revision methods directly to ontologies represented in description logics and propose new operations that overcome these problems.

1 Introduction

We have seen in recent years more and more attention devoted to the issue of representing a domain of application by means of ontologies. After a period when the main interest was on building new ontologies, lately there have been some efforts towards tools for changing, repairing and maintaining ontologies.

Since knowledge is not static, there is a necessity to deal with the evolution of ontologies. When ontologies evolve, inconsistency may arise. There are many approaches concerning how to deal with inconsistencies. In [1], a unifying framework was described that accommodates four different approaches:

- *Consistent evolution* prevents the introduction of inconsistencies in a consistent ontology.
- *Repairing* makes an inconsistent ontology consistent.
- *Reasoning with inconsistency* tries to derive meaningful conclusions from an inconsistent ontology.
- *Versioning* keeps track of changes and compatibility issues between different versions of the ontology.

Belief revision theory [2, 3] addresses the first two approaches for dealing with inconsistency: preventing the introduction of inconsistency in a knowledge base and repairing an inconsistent knowledge base. In addition to studying constructions for operations of revision in knowledge bases, belief revision studies postulates that this constructions must satisfy. There are some proposals for constructing revision operators for ontologies, however almost none which takes care of the formal properties that the operations satisfy. For this reason it became important to study how to apply belief revision techniques in ontologies.

Alchourrón, Gärdenfors and Makinson in [4] proposed a set of postulates that every operation of belief revision should satisfy. This set of postulates,

together with some proposed constructions, is know as AGM paradigm due to the authors' initials. Although this is the most influential work in the area, when we try to apply it to ontologies we have some problems: first the beliefs of an agent are represented by belief sets, sets of sentences closed under the consequence operator. Dealing with closed sets is a problem because they are very often infinite. The second problem was presented in [5]. The author showed in this work that some description logics can not satisfy the AGM postulates, they are not AGM-compliant. In particular SHOIN and SHIFF, the description logics behind OWL-DL and OWL-Lite, are not AGM-compliant. Some works [6, 7] present rational sets of postulates that could replace the AGM paradigm, but they use belief sets too.

Although less intuitive, belief base revision [3] became a good alternative because it relies on finite sets and moreover, it can be used with any compact and monotonic logic as showed in [8]. In particular it can be used with any description logic that receives just finite inputs. Although this works in theory, the constructions for revision rely on the existence of negation in the logic and many description logics do not admit the negation of all kinds of axioms. In this paper, we propose two new options for belief base revision operations and discuss the relation between them and the ones in the literature.

The paper proceeds as follows: in Section 2 we introduce the concepts of belief revision that we will use and describe briefly some previous attempts to apply belief revision to description logics. In Section 3, we present two new operations characterized by sets of postulates and constructions. Then in Section 4, we compare our operations to two other existing operations and show the relation between them. Finally, in Section 5 we conclude and point towards future work.

2 Belief Revision and Description Logics

In this section, we briefly introduce the area of Belief Revision and some previous proposals to apply it to Description Logics. We first introduce the most widely used theory for belief revision, known as the AGM paradigm due to the authors of the seminal paper [4]. Then we discuss two proposals of applying the AGM paradigm to description logics and their shortcomings. We next show an alternative to AGM theory that is more suitable for computational needs and extensively studied in the literature.

2.1 AGM theory and Description Logics

Traditionally, in AGM theory [4, 2, 9], the beliefs of an agent are represented by a set of formulas closed under logical consequence, the *belief set*. Although the logic used is not specified, there are several assumptions made which hold for classical logic.

Three operations can be performed on belief sets: contraction, expansion and revision. Contraction consists in giving up as many beliefs as needed so that the new belief set does not imply a specified sentence. Expansion consists in adding

information to the belief set. The result of an expansion can be a inconsistent set. Revision is consistent incorporation of new information, i.e., if the input sentence is consistent then the new belief set will be consistent (even if the old belief set was not). If necessary, consistence is obtained by deleting some parts of the original belief set. The operations are characterized by a set of axioms (the rationality postulates) and several constructions have been proposed together with representation theorems with respect to the postulates.

There are two main problems when one wishes to apply the original AGM theory to practical problems. The first is the fact that the belief sets are assumed to be closed under a consequence operator Cn, i.e., if K is a belief set, then $K = Cn(K)$. This usually means dealing with infinite sets. The second problem is that the consequence operator Cn is assumed to be tarskian, compact, satisfy the deduction theorem and supraclassicality. Following [10], we will refer to these properties as the AGM-assumptions. The AGM-assumptions exclude many interesting logics, such as many description logics.

Flouris et al. [10] have defined a class of *AGM-compliant logics*, that is, logics in which a contraction operator can be defined satisfying the AGM postulates. They have shown that all logics satisfying the AGM-assumptions (but not only them) are AGM-compliant. As a negative result, he has shown that some important description logics, such as SHOIN and SHIFF are not AGM-compliant.

Later works [7, 6] presented sets of postulates that could replace the AGM postulates for contraction and be used with a wider class of logics. Using the set of postulates proposed in [6], any tarskian and compact logic (in particular SHOIN and SHIFF) allows for a contraction operator satisfying the postulates.

However, if we want to use belief revision in real applications we should not use infinite belief sets. For this reason, in the rest of this paper, we concentrate on belief revision operations on belief bases, i.e., sets of formulas not necessarily closed under the consequence operator.

Flouris [5] has studied the applicability of base operations to description logics. He follows the work by Fuhrmann [11], where the AGM postulates have been adapted to deal with bases. It was already noted in the literature that these postulates were not suitable for belief base operations, due to the recovery postulate, but Flouris has shown the properties needed in order for a logic to be *base-AGM-compliant*. The conditions are stronger than the ones for the belief set case, which exclude the logics we are interested in. This is why we chose to follow Hansson's approach, where new sets of postulates were designed specific for the belief base case.

2.2 Belief Base Operations

Belief bases have been introduced in the literature [12, 13] as an alternative for representing the beliefs of an agents. Belief bases are (usually finite) sets of formulas not necessarily closed under logical consequence. The three operations for belief change in the AGM paradigm can be adapted for belief bases. We follow here the formalization in [3].

Of the three AGM operations only expansion is characterized in an unique way. When a belief base B is expanded with a proposition α, the resulting set $B + \alpha$ is obtained by simply adding the new belief to the old belief set:

$B + \alpha = B \cup \{\alpha\}$

Contraction and revision for belief bases are not uniquely defined, but as in AGM theory, constrained by sets of postulates. Unlike AGM theory, the different constructions proposed for contraction of belief bases are not equivalent (similarly for revision), i.e., there is not a single set of postulates that characterizes all constructions.

We will present in this section the operation of kernel contraction proposed by Hansson in [14] and the set of postulates that characterizes this construction.

The construction makes use of the concept of kernel, the set of minimal subsets of a given set that imply a given sentence:

Definition 1 *Let B be a set of formulas and α a formula. The kernel $B \perp\!\!\!\perp \alpha$ of B and α is defined as follows. For any set Y, $Y \in B \perp\!\!\!\perp \alpha$ if and only if:*

- $Y \subseteq B$
- $\alpha \in Cn(Y)$
- *For all Y' such that $Y' \subset Y$, $\alpha \notin Cn(Y')$*

An incision function σ selects at least one element of each kernel set. The idea is that σ picks up enough elements from the kernel so that if these elements were taken out of the belief base, then the new set would not imply the given sentence.

Definition 2 *An incision function σ for B is a function that for all α:*

- $\sigma(B \perp\!\!\!\perp \alpha) \subseteq \bigcup(B \perp\!\!\!\perp \alpha)$
- *If $\emptyset \neq X \in B \perp\!\!\!\perp \alpha$, then $X \cap \sigma(B \perp\!\!\!\perp \alpha) \neq \emptyset$*

Now we can define a kernel contraction for B:

Definition 3 *Let B be a belief base, α a formula and σ an incision function for B. The kernel contraction of B by α is defined as:*
$B -_\sigma \alpha = B \setminus \sigma(B \perp\!\!\!\perp \alpha)$

Hansson has shown that:

Theorem 1 *[14] The operator $-$ for B is a kernel contraction if and only if it satisfies the following postulates:*

[success] *If $\alpha \notin Cn(\emptyset)$, then $\alpha \notin Cn(B - \alpha)$.*
[inclusion] *$B - \alpha \subseteq B$.*
[core-retainment] *If $\beta \in B$ and $\beta \notin B - \alpha$, then there is a set B' such that $B' \subseteq B$ and $\alpha \notin Cn(B')$, but $\alpha \in Cn(B' \cup \{\beta\})$.*
[uniformity] *If it holds that for all subsets B' of B that $\alpha \in Cn(B')$ if and only if $\beta \in Cn(B')$, then $B - \alpha = B - \beta$.*

The first postulate says that the result of a contraction should not imply the contracted sentence. Inclusion says that the new belief base should not contain anything that was not already in the original set. The postulate of core-retainment tries to capture the intuition that if a sentence has to be removed, then this sentence is relevant to imply α. The last postulate says that if every subset of B that implies α implies β then contracting α should be the same as contracting β and vice-versa.

The representation theorem for this construction was shown in [8] to hold for any compact and monotonic logic. So it holds for a large variety of description logics, in particular for SHOIN and SHIFF.

In AGM theory and in most works deriving from it, revision operators (*) are related to contraction operators (-) via the *Levi identity*:

$B * \alpha = (B - \neg\alpha) + \alpha$

Hansson [15] has proposed an operation of *external revision* using the *reversed Levi identity*:

$B * \alpha = (B + \alpha) - \neg\alpha$

There are several possibilities for constructing an external revision operator. We will present here one based on kernel contraction.

Definition 4 *Let - be a kernel contraction operator. The operator of external kernel revision is defined by:*

$B * \alpha = (B + \alpha) - \neg\alpha$

Hansson has shown that:

Theorem 2 *[8] The operator $*$ for B is an external kernel revision if and only if it satisfies the following postulates:*

[**non-contradiction**] *If $\neg\alpha \notin Cn(\emptyset)$ then $\neg\alpha \notin Cn(B * \alpha)$*
[**inclusion**] $B * \alpha \subseteq B + \alpha$
[**core-retainment**] *If $\beta \in B$ and $\beta \notin B * \alpha$, then there is some B' such that $B' \subseteq B \cup \{\alpha\}$ and $\neg\alpha \notin Cn(B')$, but $\neg\alpha \in Cn(B' \cup \{\beta\})$*
[**success**] $\alpha \in B * \alpha$
[**weak-uniformity**] *If $\alpha, \beta \in B$ and for all $B' \subseteq B$ we have that $\neg\alpha \in Cn(B')$ if and only if $\neg\beta \in Cn(B')$ then $B \cap (B * \alpha) = B \cap (B * \beta)$.*
[**pre-expansion**] $B + \alpha * \alpha = B * \alpha$

In [8] it was shown that the theorem above holds for any monotonic and compact logic satisfying a property called α-local non-contravention:

Definition 5 *A consequence operator Cn satisfies α-local non-contravention if and only if, if $\neg\alpha \in Cn(B \cup \{\alpha\})$, then $\neg\alpha \in Cn(B)$.*

This characterization of revision, as well as the construction, relies on the existence of negation in the logic. This is not the case for several description logics. In particular, SHIFF and SHOIN do not admit negation of all kinds of axioms.

We can re-write the postulates of external kernel revision in a way such they do not use negation:

[**weak consistency**] If $\bot \notin Cn(\alpha)$ then $\bot \notin Cn(B * \alpha)$
[**inclusion**] $B * \alpha \subseteq B + \alpha$
[**core-retainment**] If $\beta \in B$ and $\beta \notin B * \alpha$, then there is some B' such that $B' \subseteq B$ and $\bot \notin Cn(B' \cup \{\alpha\})$, but $\bot \in Cn(B' \cup \{\beta, \alpha\})$
[**success**] $\alpha \in B * \alpha$
[**weak uniformity**] If $\alpha, \beta \in B$ and for all $B' \subseteq B$ we have that $\bot \in Cn(B' \cup \{\alpha\})$ if and only if $\bot \in Cn(B' \cup \{\beta\})$ then $B \cap (B * \alpha) = B \cap (B * \beta)$.
[**pre-expansion**] $B + \alpha * \alpha = B * \alpha$

It is not difficult to see that in logics where negation has a classical behavior (i.e., satisfies explosiveness[1] and α-local-non-contravention), this set of postulates is equivalent to the previous one.

Another idea would be to define the negation of an axiom in an abstract way and then try to find the negation of every axiom that could be built. This approach was followed in [16], but for belief sets. The authors define consistency negation of an axiom in a generic way. Their goal was to find a definition of negation that: exists in (almost) every DL, the definition coincides with classical negation if applied to classical logic and checking if an axiom is the negation of another should be decidable. We could try to adapt this approach for belief bases, but this will be left for a future work. In this work we will follow a different approach.

In the next section, we propose a new construction for revision, together with an axiomatic characterization, that does not depend directly on the notion of negation.

3 Revision without negation

In the last Section, we have seen constructions for belief base operators of contraction and revision which are characterized by sets of postulates. We have seen that the constructions and postulates can be used with a large class of logics. Usually revision is built using contraction and the Levi identity, or its reverse. This is not easy when we are dealing with description logics. In order to use the Levi identity we have to add an axiom and then contract by the negation of it. The difficulty arises when we try to find the negation of an axiom. In many description logics the negation of some axioms is not defined. This problem was already mentioned in [10] and [5].

Our construction is based on an alternative operation proposed by Hansson, *semi-revision* [17]. A usual operator of revision guarantees, through the success postulate, that after the revision a given formula is added to the belief base. An operator of semi-revision does not satisfy the success postulate. So the result of a semi-revision is always a consistent set such that the formula by which we have revised does not necessarily belong to it.

[1] A consequence operator Cn satisfies explosiveness if and only if, for all α and β, $\beta \in Cn(\{\alpha, \neg\alpha\})$.

First a particular construction for an operator of semi-revision will be presented with the associated set of postulates and representation theorem. Then this operator will be adapted to an operator of belief base revision, i.e., an operator that satisfies the success postulate.

The semi-revision operator is built by adding the formula and then contracting the inconsistency:

Definition 6 *[17] Let B be a belief base, α a formula and $-$ a kernel contraction for B. The kernel semi-revision of B by α is defined as:*
$B?\alpha = (B + \alpha) - \bot$

Theorem 3 *[17]*
The operator $?$ for B is a kernel semi-revision if and only if it satisfies the following postulates:

[consistency] $\bot \notin Cn(B?\alpha)$
[inclusion] $B?\alpha \subseteq B + \alpha$
[core-retainment] *If $\beta \in B$ and $\beta \notin B?\alpha$ then there is at least one B' such that $B' \subseteq B + \alpha$ and $\bot \notin Cn(B')$, but $\bot \in Cn(B' \cup \{\beta\})$*
[pre-expansion] $(B + \alpha)?\alpha = B?\alpha$
[internal exchange] *If $\alpha, \beta \in B$ then $B?\alpha = B?\beta$*

It has been shown in [8] that the representation theorem above holds for any compact and monotonic logic such that $\bot \notin Cn(\emptyset)$.

In an operation of semi-revision, if the new formula is involved in an inconsistency, it may be given up when contracting by $\bot$. In order to transform it in a revision (satisfying the success postulate), we need to find a way to "protect" the new formula and make sure it stays in the revised belief base.

We present two different constructions for revision without negation, both based on semi-revision, with different properties. The first one assures that the revised base is always consistent, but success does not hold in case the formula to be added is inconsistent. In the second construction, success always holds, but if the formula to be added is inconsistent, the resulting revised base is inconsistent. This is more in line with the AGM paradigm.

3.1 Weak success

For the first construction, we have to restrict the incision functions that can be used:

Definition 7 *An incision function that protects consistent inputs is a function $\overline{\sigma}$ such that:*

- $\overline{\sigma}(\alpha, B \perp\!\!\!\perp \bot) \subseteq \bigcup(B \perp\!\!\!\perp \bot)$
- *If $\emptyset \neq X \in B \perp\!\!\!\perp \bot$, then $X \cap \overline{\sigma}(\alpha, B \perp\!\!\!\perp \bot) \neq \emptyset$*
- *If $\bot \notin Cn(\{\alpha\})$, then $\alpha \notin \overline{\sigma}(\alpha, B \perp\!\!\!\perp \bot)$*

The idea is that the incision function preserves α whenever it is possible, i.e., whenever α is not a contradiction by itself. If α is consistent, then there is no set in $B \perp\!\!\!\perp \bot$ containing only α, so it is always possible to choose other elements for the incision.

We can now define a semi-revision that protects consistent inputs:

Definition 8 *A kernel semi-revision with weak success is defined as:*
$B?_{\overline{\sigma}}\alpha = (B + \alpha) \setminus \overline{\sigma}(\alpha, B + \alpha \perp\!\!\!\perp \bot)$

A semi-revision protecting the input adds a formula and then removes all the inconsistencies, like the usual semi-revision, but it never chooses α to be removed, unless α is inconsistent.

Theorem 4 *Let Cn be a compact and monotonic consequence operator. The operator $?$ for B is a kernel semi-revision with weak success if and only if it satisfies the following postulates:*

[**weak success**] *If $\bot \notin Cn(\{\alpha\})$, then $\alpha \in B?\alpha$*
[**consistency**] $\bot \notin Cn(B?\alpha)$
[**inclusion**] $B?\alpha \subseteq B + \alpha$
[**core-retainment**] *If $\beta \in B$ and $\beta \notin B?\alpha$ then there is at least one B' such that $B' \subseteq B + \alpha$ and $\bot \notin Cn(B')$, but $\bot \in Cn(B' \cup \{\beta\})$*
[**pre-expansion**] $(B + \alpha)?\alpha = B?\alpha$

Proof: (i) construction $\Rightarrow$ postulates:

Let $?_{\overline{\sigma}}$ be an operator of kernel semi-revision with weak success based on an incision function that almost protects the input, $\overline{\sigma}$. It follows directly from the construction that *inclusion* and *pre-expansion* are satisfied. From the definition of an incision function that almost protects the input, it follows that $?_{\overline{\sigma}}$ satisfies *weak success* and *consistency*. Finally, for *core-retainment*, let $\beta \in B \setminus B?_{\overline{\sigma}}\alpha$. Then by construction $\beta \in \overline{\sigma}(\alpha, (B \cup \{\alpha\}) \perp\!\!\!\perp \bot)$. This means that for some set $X \in (B \cup \{\alpha\}) \perp\!\!\!\perp \bot$, $\beta \in X$. Let $B' = X \setminus \{\beta\}$. We have $B' \subseteq B \cup \{\alpha\}$, $\bot \notin Cn(B')$ and $\bot \in Cn(B' \cup \{\beta\})$.

(ii) postulates $\Rightarrow$ construction: Let $?$ be an operator satisfying the postulates above and let σ be such that for every formula α:

$\sigma(\alpha, B \perp\!\!\!\perp \bot) = B \setminus (B?\alpha)$

We have to show (1) that σ is an incision function that almost protects the input for the given domain and (2) that $B?\alpha = B?_\sigma\alpha$.

(1) We have to show that the three conditions of Definition 7 are satisfied. For the first condition, let $\beta \in \sigma(\alpha, B \perp\!\!\!\perp \bot)$. Then it holds that $\beta \in B \setminus (B?\alpha)$ and it follows from *core-retainment* that there is some $B' \subseteq B + \alpha$ such that $\bot \notin Cn(B')$ and $\bot \in Cn(B' \cup \{\beta\})$. It follows that there is some subset B'' of B' such that $B'' \cup \{\beta\} \in B \perp\!\!\!\perp \bot$ and hence, $\beta \in \bigcup(B \perp\!\!\!\perp \bot)$.

For the second condition, let $\emptyset \neq X \in B \perp\!\!\!\perp \bot$. Suppose that $X \cap \sigma(\alpha, B \perp\!\!\!\perp \bot) = \emptyset$. Then $X \subseteq B?\alpha$. Since $\bot \in Cn(X)$, it follows from monotony that $\bot \in Cn(B?\alpha)$, contrary to *consistency*. This contradiction is sufficient to prove that $X \cap \sigma(\alpha, B \perp\!\!\!\perp \bot) \neq \emptyset$.

For the third condition, suppose $\bot \notin Cn(\{\alpha\})$. By *weak success*, $\alpha \in B?\alpha$, and hence, $\alpha \notin \sigma(\alpha, B \perp\!\!\!\perp \bot)$.

(2) By definition, $\sigma(\alpha, (B \cup \{\alpha\}) \perp\!\!\!\perp \bot) = (B \cup \{\alpha\}) \setminus ((B \cup \{\alpha\})?\alpha) = (B \cup \{\alpha\}) \setminus B?\alpha$ (*pre-expansion*). Hence, $B?_\sigma\alpha = (B \cup \{\alpha\}) \setminus \sigma((B \cup \{\alpha\}) \perp\!\!\!\perp \bot) = (B \cup \{\alpha\}) \setminus ((B \cup \{\alpha\}) \setminus B?\alpha) = B?\alpha$ (*inclusion*). □

3.2 Success

We now define an operation of belief base revision that does not rely on negation and has the property of success.

Definition 9 *An incision function that protects the input is a function σ such that:*

- $\sigma(\alpha, B \perp\!\!\!\perp \bot) \subseteq \bigcup(B \perp\!\!\!\perp \bot)$
- *If* $\bot \notin Cn(\{\alpha\})$ *and* $\emptyset \neq X \in B \perp\!\!\!\perp \alpha$, *then* $X \cap \sigma(\alpha, B \perp\!\!\!\perp \bot) \neq \emptyset$
- $\alpha \notin \sigma(\alpha, B \perp\!\!\!\perp \bot)$

The idea here is that the incision function always preserves α. In the case where α is a contradiction, not enough will be chosen to make the resulting belief base consistent. In particular, when α is inconsistent we could have an incision function that protects α that returns the empty set and in this case, $B * \alpha = B + \alpha$.

Definition 10 *A kernel revision without negation is defined as:*
$B *_\sigma \alpha = (B + \alpha) \setminus \sigma(\alpha, B + \alpha \perp\!\!\!\perp \bot)$

Theorem 5 *Let Cn be a compact and monotonic consequence operator. The operator $*$ for B is a kernel revision without negation if and only if it satisfies the following postulates:*

[**success**] $\alpha \in B * \alpha$
[**weak consistency**] *If* $\bot \notin Cn(\{\alpha\})$, *then* $\bot \notin Cn(B * \alpha)$
[**inclusion**] $B * \alpha \subseteq B + \alpha$
[**core-retainment**] *If* $\beta \in B$ *and* $\beta \notin B * \alpha$ *then there is at least one* B' *such that* $B' \subseteq B + \alpha$ *and* $\bot \notin Cn(B')$, *but* $\bot \in Cn(B' \cup \{\beta\})$
[**pre-expansion**] $(B + \alpha) * \alpha = B * \alpha$

Proof: (i) construction ⇒ postulates:

Let $*_\sigma$ be an operator of kernel revision without negation based on an incision function σ that protects the input. It follows directly from the construction that *inclusion* and *pre-expansion* are satisfied. From the definition of an incision function that protects the input it follows that $*_\sigma$ satisfies *success* and *weak consistency*. Finally, for *core-retainment*, let $\beta \in B \setminus B *_\sigma \alpha$. Then by construction $\beta \in \sigma(\alpha, (B \cup \{\alpha\}) \perp\!\!\!\perp \bot)$. This means that for some set $X \in (B \cup \{\alpha\}) \perp\!\!\!\perp \bot$, $\beta \in X$. Let $B' = X \setminus \{\beta\}$. We have $B' \subseteq B \cup \{\alpha\}$, $\bot \notin Cn(B')$ and $\bot \in Cn(B' \cup \{\beta\})$.

(ii) postulates ⇒ construction: Let $*$ be an operator satisfying the postulates above and let σ be such that for every formula α:

$\sigma(\alpha, B \perp\!\!\!\perp \bot) = B \setminus (B * \alpha)$

We have to show (1) that σ is an incision function that protects the input for the given domain and (2) that $B * \alpha = B *_\sigma \alpha$.

(1) We have to show that the three conditions of Definition 9 are satisfied. For the first condition, let $\beta \in \sigma(\alpha, B \perp\!\!\!\perp \bot)$. Then it holds that $\beta \in B \setminus (B * \alpha)$ and it follows from *core-retainment* that there is some $B' \subseteq B + \alpha$ such that $\bot \notin Cn(B')$ and $\bot \in Cn(B' \cup \{\beta\})$. It follows that there is some subset B'' of B' such that $B'' \cup \{\beta\} \in B \perp\!\!\!\perp \bot$ and hence, $\beta \in \bigcup(B \perp\!\!\!\perp \bot)$.

For the second condition, let $\bot \notin Cn(\{\alpha\})$ and $\emptyset \neq X \in B \perp\!\!\!\perp \bot$. Suppose that $X \cap \sigma(\alpha, B \perp\!\!\!\perp \bot) = \emptyset$. Then $X \subseteq B * \alpha$. Since $\bot \in Cn(X)$, it follows from monotony that $\bot \in Cn(B * \alpha)$, contrary to *weak consistency*. This contradiction is sufficient to prove that $X \cap \sigma(\alpha, B \perp\!\!\!\perp \bot) \neq \emptyset$.

For the third condition, it suffices to note that by *success*, $\alpha \in B * \alpha$, and hence, $\alpha \notin \sigma(\alpha, B \perp\!\!\!\perp \bot)$.

(2) By definition, $\sigma(\alpha, (B \cup \{\alpha\}) \perp\!\!\!\perp \bot) = (B \cup \{\alpha\}) \setminus ((B \cup \{\alpha\}) * \alpha) = (B \cup \{\alpha\}) \setminus B * \alpha$ (*pre-expansion*). Hence, $B *_\sigma \alpha = (B \cup \{\alpha\}) \setminus \sigma((B \cup \{\alpha\}) \perp\!\!\!\perp \bot) = (B \cup \{\alpha\}) \setminus ((B \cup \{\alpha\}) \setminus B * \alpha) = B * \alpha$ (*inclusion*). □

4 Comparing the operations

We have defined two different operations that can be used to revise ontologies in description logics. In this section we compare them to other proposals according to their properties.

In [18], the authors proposed the use of semi-revision for revising ontologies. The problem with semi-revision is that there is no guaranty that the new formula will be in the revised ontology, or even implied by it. We can see the operations described in the previous section as a link between semi-revision and revision.

Starting from semi-revision, when we switch to the operation defined in Section 3.1, we get some guaranty of success, through the *weak success* postulate. Whenever the new formula is consistent, it will be incorporated to the belief base. The price we pay for weak success is the loss of *internal exchange*. This means that, even if α and β are elements of B, the result of revising by them may be different.

From the operation defined in Section 3.1 to the one defined in Section 3.2, we get the *success* postulate, but loose unconditional *consistency*. This means that the resulting revised base may end up being inconsistent, but this only happens if the formula being added is inconsistent. This operation is closer to AGM-style intuitions. What is missing from Hansson's external revision is some form of uniformity. This is due to the syntactical nature of "protecting" α. One could think about other definitions of protection that would take into account logically equivalent formulas, or set of formulas, but this is left for future work.

In the rest of this section, we present a short example to illustrate the use of the revision operations. The example is described in a simple description logic.

Let us consider the following knowledge base B about Tweety:

$$Bird \sqsubseteq Fly \tag{1}$$
$$Bird(Tw) \tag{2}$$
$$Peng \sqsubseteq \neg Fly \tag{3}$$
$$\neg Peng(Tw) \tag{4}$$

The base contains thus information about birds (that they fly), penguins (that they do not fly) and an individual, Tweety, that is a bird and not a penguin.

Now assume that we receive information that we were wrong and that Tweety is a penguin, i.e., we want to add a new formula α to B:

$\alpha = Peng(Tw)$

4.1 Semi-Revision

In order to find the resulting knowledge base $B?\alpha$, we first have to compute the kernel set $B \cup \{\alpha\} \perp\!\!\!\perp \bot$.

$B \cup \{\alpha\} \perp\!\!\!\perp \bot = \{\{\neg Peng(Tw), Peng(Tw)\},$
$\{Bird \sqsubseteq Fly, Bird(Tw), Peng \sqsubseteq \neg Fly, Peng(Tw)\}\}$

There are several different possibilities for an incision function, we could have, for example:

$\sigma(B \cup \{\alpha\} \perp\!\!\!\perp \bot) = \{Peng(Tw)\}$

This would mean that the new information is not accepted and the resulting base is equal to the original one:

$B?\alpha = B + \alpha \setminus \{\alpha\}$

4.2 Semi-revision with weak success

In this case, we need an incision function that almost protects α, i.e., if α is consistent, it is never chosen. Since $Peng(Tw)$ is consistent, the incision function has to choose at least one element different from α of each set in $B \cup \{\alpha\} \perp\!\!\!\perp \bot$. One possible choice is:

$\overline{\sigma}(B \cup \{\alpha\} \perp\!\!\!\perp \bot) = \{\neg Peng(Tw), Bird \sqsubseteq Fly\}$

For this example the resulting set is:

$K?_{\overline{\sigma}}\alpha = \{Bird(Tw), Peng \sqsubseteq \neg Fly, Peng(Tw)\}$

4.3 Revision without negation

This operation only differs from semi-revision with weak success in the case where the new formula is inconsistent. Since in this example, the formula being added is consistent, an incision function that almost protects α is also an incision function that protects α and the result may be the same with the two operations.

4.4 External kernel revision

For external kernel revision, we have to compute the kernel set $B \cup \{\alpha\} \perp\!\!\!\perp \neg\alpha$:

$$B \cup \{\alpha\} \perp\!\!\!\perp \neg\alpha = \{\{\neg Peng(Tw)\},$$
$$\{Bird \sqsubseteq Fly, Bird(Tw), Peng \sqsubseteq \neg Fly\}\}$$

We could obtain the same result as in the two previous operations, by taking an incision function such that:

$$\sigma(B \cup \{\alpha\} \perp\!\!\!\perp \neg\alpha) = \{\neg Peng(Tw), Bird \sqsubseteq Fly\}$$

Note however, that this operation and revision without negation are not equivalent, the postulate of core-retainment is not identical in both cases. In fact we can show that if an operation satisfies the core-retainment of external kernel revision then it satisfies the core-retainment of revision without negation, but this is not true in the other way. Intuitively in the second any formula that is responsible for the deriving the inconsistency can be removed, but in the first one only the ones that are responsible for deriving $\neg\alpha$ can be removed.

Moreover, this operation depends on the existence of negation: if we want to revise by $Peng \sqsubseteq Bird$, for example, in several description logics we cannot form the negation of this axiom and the traditional operation of revision cannot be used.

5 Conclusions and Future Work

In [8], several operations of belief base change were generalized and shown to hold for a wide range of logics. The operations for contraction can be used with any compact and monotonic logic, and are thus applicable to ontologies represented in most description logics. Revision, on the other side, is usually defined based on contraction using negation. In several description logics, there is no negation for all kinds of axioms.

In this paper we have proposed two new operations of belief base revision, characterized by sets of postulates and constructions. The idea was to get rid of the dependence on negation.

In [16], the authors propose two different possible definitions for negation in ontologies. Future works include testing this notions of negation with the (reversed) Levi identity in order to see whether we get a meaningful revision operator.

We also plan to implement and test the ideas presented here. When trying to implement the operations defined in the previous sections, we are confronted with the problem of finding the kernel sets, which is the computational bottleneck of these constructions. Fortunately this problem has already been addressed for description logics. Description logic systems typically offer a set of inference services, such as: consistency checking, classification and concept subsumption. In [19] the authors proposed that the systems should provide debugging services. One of these debugging services was called axiom pinpointing. Axiom pinpointing is an inference service that given a sentence provides the set of all justifications for this sentence. By justification the author mean the smallest sets of formulas from the knowledge base (KB) that imply the sentence. In other words, axiom pinpointing returns the kernel of the KB given a sentence.

In [19] the authors show two ways of computing axiom pinpointing. One way was called black-box because it is reasoner-independent, the reasoner is used as an oracle that tells if a concept is satisfiable. The second way was called glass-box because it is reasoner-dependent. Though the first way is more general, the second way is much more efficient. The glass-box algorithm showed in [19] was based on the tableaux decision procedure for concept satisfiability in SHOIN.

Both services return only one justification for the sentence (one element of the kernel). To find all justifications, an algorithm based on Reiter's Hitting Set Tree Algorithm [20] could be used, which would also find the possible incision functions. The use of Reiter's algorithm for implementing belief change operators was suggested in [21] and used in [18].

References

1. Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., Sure, Y.: A framework for handling inconsistency in changing ontologies. In: Proceedings of the Fourth Internation Semantic Web Conference. Volume 3729 of LNCS., Springer (2005) 353–367
2. Gärdenfors, P.: Knowledge in Flux: Modeling the Dynamics of Epistemic States. MIT Press, Cambridge (1988)
3. Hansson, S.O.: A Textbook of Belief Dynamics. Kluwer Academic Publishers (1997)
4. Alchourrón, C., Gärdenfors, P., Makinson, D.: On the logic of theory change. Journal of Symbolic Logic **50**(2) (1985) 510–530
5. Flouris, G.: On Belief Change and Ontology Evolution. PhD thesis, University of Crete (2006)
6. Ribeiro, M.M., Wassermann, R.: First steps towards revising ontologies. In: Proceedings of the Second Workshop on Ontologies and their Applications (WONTO 2006). (2006)
7. Flouris, G., Plexousakis, D., Antoniou, G.: On generalizing the agm postulates. In: Proceedings of the 3rd European Starting AI Researcher Symposium (STAIRS-06). (2006)
8. Hansson, S.O., Wassermann, R.: Local change. Studia Logica **70**(1) (2002) 49–76
9. Gärdenfors, P., Rott, H.: Belief revision. Handbook of Logic in Artificial Intelligence and Logic Programming (1995)

10. Flouris, G., Plexousakis, D., Antoniou, G.: On applying the AGM theory to DLs and OWL. In: Proceedings of the International Semantic Web Conference. (2005) 216–231
11. Fuhrmann, A.: Theory contraction through base contraction. Journal of Philosophical Logic **20** (1991) 175–203
12. Nebel, B.: A knowledge level analysis of belief revision. In Brachman, R., Levesque, H., Reiter, R., eds.: First International Conference on Principles of Knowledge Representation and Reasoning - KR'89, Toronto, ON, Morgan Kaufmann (May 1989) 301–311
13. Hansson, S.O.: Belief Base Dynamics. PhD thesis, Uppsala University (1991)
14. Hansson, S.O.: Kernel contraction. Journal of Symbolic Logic **59** (1994) 845–859
15. Hansson, S.O.: Reversing the Levi identity. Journal of Philosophical Logic **22** (1992) 637–639
16. Flouris, G., Huang, Z., Pan, J.Z., Plexousakis, D., Wache, H.: Inconsistencies, negations and changes in ontologies. In: AAAI, AAAI Press (2006)
17. Hansson, S.O.: Semi-revision. Journal of Applied Non-Classical Logic **7**(1-2) (1997) 151–175
18. Halaschek-Wiener, C., Katz, Y., Parsia, B.: Belief base revision for expressive description logics. In: OWL: Experiences and Directions 2006. (2006)
19. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.: Debugging unsatisfiable classes in OWL ontologies. Journal of Web Semantics - Special Issue of the Semantic Web Track of WWW2005 **3**(4) (2005)
20. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence **32** (1987) 57–95
21. Wassermann, R.: An algorithm for belief revision. In: Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000), Morgan Kaufmann (2000)

A Stratification-based Approach for Inconsistency Handling in Description Logics

Guilin Qi[1], Jeff Z. Pan[2]

[1] Institute AIFB
Universität Karlsruhe
D-76128 Karlsruhe, Germany
gqi@aifb.uni-karlsruhe.de
[2] Department of Computing Science, The University of Aberdeen
Aberdeen AB24 3FX
jpan@csd.abdn.ac.uk

Abstract. Inconsistency handling is a central problem in many knowledge representation fields, such as belief revision, belief merging. Many approaches have been proposed to handle inconsistency in ontologies. In this paper, we propose a stratification-based approach for inconsistency handling in description logics (DLs), a family of ontology languages. This approach consists of two steps. In the first step, we obtain a preference relation on the axioms in the *DL knowledge base* using an algorithm. Then two existing approaches in first-order logic are adapted to resolve conflicting information in the stratified DL knowledge base.

1 Introduction

Ontologies play a crucial role for the success of the Semantic Web [12]. There are many representation languages for ontologies, such as description logics (or DLs for short) [4]. Inconsistency may occur because of several reasons, such as modelling errors, migration or merging ontologies, and ontology evolution. Current DL reasoners, such as RACER [14], can detect logical inconsistency. But they only provide lists of unsatisfiable classes and the process of *resolving* inconsistency is left to the user or ontology engineers. The need to improve DL reasoners to reasoning with inconsistency is becoming urgent to make them more applicable. Many approaches were proposed to handle inconsistency in ontologies based on existing techniques for inconsistency management in traditional logics, such as propositional logic and nonmonotonic logics [24, 21, 18].

It is well-known that priority or preference plays an important role in inconsistency handling [2, 7, 20]. In [2], the authors introduced priority to default terminological logic such that more specific defaults are preferred to more general ones. When conflicts occur in reasoning with defaults, defaults which are more specific should be applied before more general ones. In [20], an algorithm, called *refined conjunctive maxi-adjustment* (RCMA for short) was proposed to weaken conflicting information in a *stratified* DL knowledge base and some consistent DL knowledge bases were obtained. To weaken a terminological axiom,

they introduced a DL expression, called *cardinality restrictions* on concepts [3]. In [26], a revision-based approach was given to resolve inconsistency in a stratified DL knowledge base. Instead of using cardinality restrictions on concepts, this approach weakens DL axioms (both terminological axioms and assertional axioms) by removing those instances which are responsible for inconsistency.

In this paper, we propose a stratification-based approach for inconsistency handling in DLs. First, we give an algorithm to obtain a preference relation on the axioms of an inconsistent DL knowledge base. The knowledge base associated with this preference relation is a *stratified DL knowledge base*. We then apply two existing approaches in first-order logic to resolve conflicting information in the stratified DL knowledge bases. The first approach is called a possibilistic logic approach and the second approach is called a lexicographic-based approach. We analyze the pros and cons of both approaches.

This paper is organized as follows. Section 2 gives a brief review of description logics. In Section 3, we provide background knowledge on stratified knowledge bases and two inconsistency handling approaches. An algorithm to stratify a DL knowledge base is proposed in Section 4. We then adapt the existing inconsistency handling approaches to DLs in Section 5. Before conclusion, we have a brief discussion on related work.

2 Description logics

In this section, we introduce some basic notions of Description Logics (DLs), a family of well-known knowledge representation formalisms [4]. We consider $\mathcal{ALC}$ [25], which is a simple yet relatively expressive DL. Let N_C and N_R be pairwise disjoint and countably infinite sets of *concept names* and *role names* respectively. We use the letters A and B for concept names, the letter R for role names, and the letters C and D for concepts. The set of $\mathcal{ALC}$ concepts is the smallest set such that: (1) every concept name is a concept; (2) if C and D are concepts, R is a role name, then the following expressions are also concepts: $\neg C$ (full negation), $C \sqcap D$ (concept conjunction), $C \sqcup D$ (concept disjunction), $\forall R.C$ (value restriction on role names) and $\exists R.C$ (existential restriction on role names). An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$, called the *domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ which maps every concept C to a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and every role R to a subset $R^{\mathcal{I}}$ of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that, for all concepts C, D, role R, the following properties are satisfied:

(1) $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$,
(2) $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$,
(3) $(\exists R.C)^{\mathcal{I}} = \{x | \exists\ y s.t. (x, y) \in R^{\mathcal{I}}$ and $y \in C^{\mathcal{I}}\}$,
(4) $(\forall R.C)^{\mathcal{I}} = \{x | \forall y (x, y) \in R^{\mathcal{I}}$ implies $y \in C^{\mathcal{I}}\}$.

A DL knowledge base consists of two components, the *terminological box (TBox)* T and the *assertional box (ABox)* A. A TBox is a finite set of terminological axioms of the form $C \sqsubseteq D$ (general concept inclusion or GCI for short) or $C \equiv D$ (equalities), where C and D are two (possibly complex) $\mathcal{ALC}$ concepts. An interpretation $\mathcal{I}$ satisfies a GCI $C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies an equality

$C \equiv D$ iff $C^{\mathcal{I}} = D^{\mathcal{I}}$. It is clear that $C \equiv D$ can be seen as an abbreviation for the two GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$. Therefore, we take a TBox to contain only GCIs. We can also formulate statements about individuals. We denote individual names as $a,\ b,\ c$. A *concept* (*role*) assertion axiom has the form $C(a)$ ($R(a, b)$), where C is a concept description, R is a role name, and $a,\ b$ are *individual names*. To give a semantics to ABoxs, we need to extend interpretations to individual names. For each individual name a, $\cdot^{\mathcal{I}}$ maps it to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies a concept axiom $C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$, it satisfies a role axiom $R(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An *ABox* contains a finite set of concept and role axioms. A DL knowledge base K consists of a TBox and an ABox, i.e. it is a set of GCIs and assertion axioms. An interpretation $\mathcal{I}$ is a *model* of a DL (TBox or ABox) axiom iff it satisfies this axiom, and it is a model of a DL knowledge base K if it satisfies every axiom in K. In the following, we use $M(\phi)$ (or $M(K)$) to denote the set of models of an axiom ϕ (or DL knowledge base K). K is consistent iff $M(K) \neq \emptyset$. Let K be an inconsistent DL knowledge base. A set $K' \subseteq K$ is a *conflict* [1] of K if K' is inconsistent, and any sub-knowledge base $K'' \subset K'$ is consistent. Given a DL knowledge base K and a DL axiom ϕ, we say K *entails* ϕ, denoted as $K \models \phi$, iff $M(K) \subseteq M(\phi)$.

3 Stratified Knowledge Bases

In this section, we first provide some background knowledge on stratified knowledge bases. Then some inconsistency handling approaches in classical logic are introduced.

3.1 Stratified knowledge base

We consider a first order language $\mathcal{L}$ determined by a set of variable symbols and a set of predicate and function symbols. 0-ary functions are constants. We use uppercase letters like P and R for predicate symbols, lowercase letters like a, b, c for constant symbols, and x, y for variable symbols. The classical consequence relation is denoted as $\vdash$. We denote formulae in $\mathcal{L}$ by ϕ, ψ, γ,... A *classical knowledge base* K is a finite set of first-order formulae. K is inconsistent iff $K \vdash \phi$ and $K \vdash \neg\phi$ for some formula ϕ.

A *stratified* knowledge base, sometimes also called prioritized knowledge base [5], is a set K of (finite) propositional formulas together with a total preorder $\leq$ on K (a preorder is a transitive and reflexive relation, and $\leq$ is a total preorder if either $\phi \leq \psi$ or $\psi \leq \phi$ holds for any $\phi, \psi \in K$)[2]. Intuitively, if $\phi \leq \psi$, then ϕ is considered to be at least less or equally important than ψ. K can be equivalently defined as a sequence $K = (S_1, ..., S_n)$, such that formulas in S_i have the same

[1] The notion of conflict is different from the notion of minimal unsatisfiability-preserving sub-TBox of a DL knowledge base defined in [24] in that it concerns inconsistency instead of incoherence.

[2] For simplicity, we use K to denote a stratified knowledge base and ignore the total preorder $\leq$.

level of priority and have higher priority than the ones in S_j where $j < i$. Each subset S_i is called a stratum of K and i the priority level of each formula of S_i. Therefore, the higher the stratum, the higher the priority level of a formula in it. A subbase A of K is also stratified, that is, $A = (A_1, ..., A_n)$ such that $A_i \subseteq S_i$, $i = 1, ..., n$.

3.2 Reasoning with inconsistent stratified knowledge bases

Many approaches have been introduced to reasoning with inconsistent stratified knowledge bases [5, 7–9]. We consider two approaches, one is the possibilistic logic approach [9] and the other is the adapted lexicographic-based approach [7].

Possibilistic logic approach Possibilistic logic inference [9] is based on a suitable consistent stratified subbase of K. Suppose $K = \{S_1, ..., S_n\}$. Let $Inc(K) = max\{i : S_i \cup ... \cup S_n \ is \ inconsistent\}$ be the inconsistency degree of K. There are two possibilistic consequence relations.

Definition 1. *Let $K = \{S_1, ..., S_n\}$ be a stratified knowledge base. A formula ϕ is said to be a possibilistic consequence of K, denoted by $K \vdash_\pi \phi$ if and only if $S_{Inc(K)+1} \cup ... \cup S_n \vdash \phi$.*

A formula ϕ is a possibilistic consequence of K if and only if it can inferred by the set of formulas whose priority levels are greater than $Inc(K)$, the inconsistency degree of K.

Definition 2. *Let $K = \{S_1, ..., S_n\}$ be a stratified knowledge base. A formula ϕ is said to be a i-consequence of K, denoted by $K \vdash_i \phi$ if and only if the following conditions are satisfied:*

(1) $i > Inc(K)$
(2) $S_i \cup ... \cup S_n \vdash \phi$
(3) for any $j > i$, $S_j \cup ... \cup S_n \nvdash \phi$.

In Definition 2, Condition (1) ensures that the i-consequence is not trivial. Condition (2) says that ϕ can be inferred from the set of formulas whose priority levels are greater than i and Condition (3) means that i is the highest priority level which can be attached to ϕ.

To check whether a formula ϕ is a possibilistic consequence or an i-consequence of K, we first need to compute the inconsistency degree of K, which is a hard task.

Proposition 1. *[19] Computing $Inc(K)$ requires $\lceil log_2 n \rceil$ satisfiability checks, where n is the number of different strata of K.*

According to Proposition 1, it requires $\lceil log_2 n \rceil + 1$ satisfiability checks to decide whether a formula ϕ is a possibilistic consequence of K or not.

Adapted lexicographic-based approach In [7], a stratified first-order logic approach for handling inconsistency was proposed to adapt the lexicographic-based approach in propositional logic [5]. When a formula of the form $\forall x\phi(x)$ is involved in a conflict, then it is simply deleted by the lexicographic-based approach to restore consistency. In contrast, the adapted lexicographic-based approach weakens the conflicting formula by dropping only instances of this formula which are responsible of a conflict. For example, if a formula of the form $\forall x\phi(x)$ is conflicting for $x = a$, then this formula is weakened as $\forall x\neg(x = a)\rightarrow\phi(x)$. Let us explain the approach in more detail.

Let ϕ be a formula which is universally quantified with a set of variable $X = \{x_1, ..., x_n\}$. Let $I = \{i_1, ..., i_n\}$ be such that i_k $(k = 1, ..., n)$ are instances of x_k respectively. Let us denote the formula $\neg(\bigwedge_{k=1,...,n}(x_k = i_k))$ as $Different(I, X)$. The following definitions can be found in [7].

Definition 3. *Let ϕ be a first-order formula which is universally quantified with a set of variable $X = \{x_1, ..., x_n\}$ where n is finite. ϕ_{weak} is called a weakened formula of ϕ if it has the form: $A\rightarrow\phi$, where $A = \{Different(I_j, X) : j = 1, ..., m\}$ (or A can be seen as the conjunction of formulas in it). The degree of a weakened formula ϕ_{weak} is defined as $degree(\phi_{weak}) = |A|$, i.e., it is the cardinality of A.*

The degree of a weakened formula is used to count the number of instances that cannot be applied, i.e. instances that are ignored.

The weakened base of a first-order knowledge base is defined as follows.

Definition 4. *Let $K = \{S_1, ..., S_n\}$ be a stratified knowledge base, where S_n contains formulas that are completely certain (that is, they cannot be deleted or weakened if they are involved in a conflict). A first-order knowledge base $K' = \{S'_1, ..., S'_n\}$ is said to be a weakened base of K if* 1) *K' is consistent, and* 2) *K' is only obtained by replacing some formula ϕ of $\{S_1, ..., S_{n-1}\}$ by their weakened counterpart ϕ_{weak}.*

The degree of a stratum S'_i of a weakened base K' is defined as: $degree(S'_i) = \Sigma_{\phi_{weak}\in S'_i} degree(\phi_{weak})$. We then can define the ranking between weakened bases as follows.

Definition 5. *Let K' and K'' be two weakened bases of K. K' is said to be lexicographically preferred to K'', denoted by $K' >_{lex} K''$, if $\exists i$, $1\leq i\leq n$ such that i) $degree(S'_i) < degree(S''_i)$, and ii) $\forall j > i$, $degree(S'_j) = degree(S''_j)$. K' is said to be lexicographically preferred weakened base of K if there is no consistent weakened base K'' such that $K''>_{lex}K'$. A formula ψ is said to be a lex conclusion of K, denoted $K \vdash_{lex} \psi$, if ψ is a consequence of all lexicographically preferred weakened bases of K.*

4 Stratification of DL Knowledge Bases

In this section, we define an algorithm transform an inconsistent DL knowledge base into a stratified DL knowledge base, i.e. each element of the base is assigned a rank, based on the weakening-based revision operator. More precisely,

a stratified DL knowledge base is of the form $K = S_1 \cup ... \cup S_n$, where for each $i \in \{1, ..., n\}$, S_i is a finite multi-set of DL sentences. Sentences in each stratum S_i have the same rank or reliability, while sentences contained in S_j such that $j > i$ are seen as more reliable.

There are many ways to obtain a stratified DL knowledge base. For example, the stratification can be given by an expert or by ontology learning [15]. The stratification can also be computed automatically. In this section, we propose an algorithm to stratify a DL knowledge base. We assume that a TBox T consists of two adjoint subsets: a set of completely sure terminology axioms T_c, i.e., axioms which will not be involved in any conflict, and a set of default terminology axioms T_d. We also assume that the information in an ABox is completely sure. The knowledge base is called a default DL knowledge base. That is, a default DL knowledge base K is already stratified as $K = \{T_d, A \cup T_c\}$, where T_c contains completely sure terminology axioms, A contains assertion axioms, and T_d contains default terminology axioms. This assumption is often adopted in default theories [1, 7, 23]. In default theories, *specificity* is a commonly used criterion for ranking a set of default rules [22, 23, 2]. Many methods have been proposed to compute specificity in default theories. In [22], Pearl gives a method to rank a set of default rules such that a more specific default is preferred to a more general one. This method is then revised and applied to stratify a knowledge base consisting of a set of default and hard rules in [6]. In this section, we propose a stratification algorithm based on the stratification method in [6]. Given a set of terminology axioms $T = T_c \cup T_d$, where T_c contains completely sure terminology axioms, and T_d contains default terminology axioms, we say that a default terminology axiom $C_1 \sqsubseteq D_1$ is more specific than another one $C_2 \sqsubseteq D_2$ iff $T \models C_1 \sqsubseteq C_2$ but $T \not\models C_2 \sqsubseteq C_1$. Note that the ordering relation defined by specificity is not necessarily a total preorder.

Stratification Algorithm

Input: Default terminology axioms base T_d, completely sure terminology axioms base T_c

Output: Stratified default terminology axiom base T_s

begin

m=1;

while $T_d \neq \emptyset$ **do**

begin

$S_m = \{C_i \sqsubseteq D_i | C_i \sqsubseteq D_i \in T_d,$ *and* $T_c \cup T_d \cup \{C_i(a)\}$ *is consistent,* a *is a new instance*$\}$;

If $S_m = \emptyset$ **then** stop (inconsistent terminology axioms).

$T_d = T_d \setminus S_m$; $m = m + 1$;

end begin

end while

Return $T_s = \{S_1, S_2, ..., S_m\}$.

end

In the stratification algorithm, when there exists m such that S_m is empty, then we say that T_d is inconsistent with T_c and we end the algorithm (because all

the other element in T_d are blocked to be stratified). In the following, we assume that T_c is always consistent with T_d. Given a default DL knowledge base $K = \{T_d, A \cup T_c\}$, suppose T_d is stratified as $T_s = \{S_1, ..., S_m\}$ using the stratification algorithm, we get a stratified DL knowledge base $K' = \{S_1, ..., S_{m+1}\}$, where $S_{m+1} = A \cup T_c$.

Let us look at an example.

Example 1. Let $K = \{T_d, A \cup T_c\}$, where $T_d = \{bird \sqsubseteq flies, penguin \sqsubseteq \neg flies\}$, $T_c = \{penguin \sqsubseteq bird\}$ and $A = \{penguin(Cheeky)\}$. We now apply the stratification algorithm to stratify T_d. First, since $T_d \cup T_c \cup \{bird(a)\}$ is consistent and $T_d \cup T_c \cup \{penguin(a)\}$ is inconsistent, where a is an arbitrary bird name, we have $S_1 = \{bird \sqsubseteq flies\}$. There is only one element left in T_d, so $S_2 = \{penguin \sqsubseteq \neg flies\}$. That is, T_d is stratified as $T_s = \{S_1, S_2\}$. Note that $penguin \sqsubseteq \neg flies$ is more specific than $bird \sqsubseteq flies$ because we have $penguin \sqsubseteq bird$ in T_c. K is then further stratified as $K' = \{S_1, S_2, A \cup T_c\}$.

In Example 1, the ranking obtained by the stratification algorithm agrees with the notion of specificity. More generally, suppose $C_i \sqsubseteq D_i$ is a terminology axiom in T_d such that $T_d \cup T_c \cup \{C_i(a)\}$ is inconsistent. Then the assertion $C_i(a)$ triggers a more general default terminology axiom in T_d which is responsible for the inconsistency. Therefore, the higher the rank is, the more specific the default terminology axiom is.

5 Inconsistency Handling in Stratified DL Knowledge Bases

5.1 Possibilistic logic approach

We apply the possibilistic logic approach to deal with inconsistency in a stratified DL knowledge base K. We have the following two definitions.

Definition 6. *Let $K = \{S_1, ..., S_n\}$ be a stratified DL knowledge base. Let $Inc(K) = max\{i : S_i \cup ... \cup S_n$ is inconsistent$\}$ be the inconsistency degree of K. For any DL statement ϕ, ϕ is a possibilistic consequence of K, denoted $K \models_\pi \phi$, if and only if, $S_{Inc(K)+1} \cup ... \cup S_n \models_\pi \phi$.*

Definition 7. *Let $K = \{S_1, ..., S_n\}$ be a stratified DL knowledge base. Let $Inc(K)$ be the inconsistency degree of K. For any DL statement ϕ, ϕ is a i-consequence of K, denoted $K \models_i \phi$, if and only if the following conditions are satisfied:*

(1) $i > Inc(K)$
(2) $S_i \cup ... \cup S_n \models \phi$
(3) for any $j > i$, $S_j \cup ... \cup S_n \not\models \phi$.

By Definition 6 and Definition 7, both the possibilistic consequence and the i-consequence relation are independent of DL reasoners, i.e., we can treat the DL reasoner as a black box and use it to check knowledge base consistency. Another advantage of the possibilistic approaches is that they are independent of DL languages, although we restrict our discussion to DL $\mathcal{ALC}$ in this paper.

The main task of possibilistic inferences defined above is to compute the inconsistency degree of K, which requires $\lceil log_2 n \rceil$ DL consistency checks, where n is the number of different strata of K.

Let us go back to Example 1.

Example 2. (Continuing Example 1) Suppose that K is stratified as $K' = \{S_1, S_2, S_3\}$, where $S_1 = \{bird \sqsubseteq flies\}$, $S_2 = \{penguin \sqsubseteq \neg flies\}$ and $S_3 = \{penguin(Cheeky), penguin \sqsubseteq bird\}$. Let us check if $Cheeky$ can fly. First, we compute the inconsistency degree of K'. Since $\cup_{i=1}^{3} S_i$ is inconsistent and $S_2 \cup S_3$ is consistent, $Inc(K') = 1$. It is clear that $S_2 \cup S_3 \models_\pi \neg flies(Cheeky)$. So we can conclude that $Cheeky$ cannot fly. Furthermore, we can conclude that $K' \models_2 \neg flies(Cheeky)$, that is, the priority level of the proposition that $Cheeky$ cannot fly is two.

Possibilistic logic approach simply blocks a default terminology axiom if it is responsible for the conflict and its priority level is not larger than the inconsistency degree. This may result in unnecessary loss of information. Let us continue to consider Example 3. Suppose we are told that $Kelly$ is a bird. We add $bird(Kelly)$ to S_3, that is, $S_3 = \{penguin(Cheeky), bird(Kelly), penguin \sqsubseteq bird\}$. Since $S_2 \cup S_3 \not\models flies(Kelly)$. So we cannot conclude that $Kelly$ can fly. This is because $bird \sqsubseteq flies$ is blocked and cannot be used to infer that $flies(Kelly)$.

5.2 Lexicographic-based approach

In this section, we apply the adapted lexicographic-based approach to the description logic setting. To do this, we need to extend the logic $\mathcal{ALC}$ with *cardinality restrictions* on concepts, which was proposed in [3]. Cardinality restrictions on a concept C are of the form $\geq m\ C$ and $\leq n\ C$, which express that the concept C has at least m elements and at most n elements respectively. We only consider cardinality restriction of the form $\leq n\ C$. An interpretation $\mathcal{I}$ is said to satisfy $\leq\ n\ C$ iff $|C^{\mathcal{I}}| \leq n$. Each GCI $C \sqsubseteq D$ can be equivalently transformed into a cardinality restriction of the form $\leq\ 0\ C \sqcap \neg D$, which says that the concept $C \sqcap \neg D$ is empty.

When a GCI is debugged to be erroneous, it is generally deleted to restore consistency in current methods [18, 21, 24]. However, as we can see from Example 1, this can result in unnecessary loss of information. In [20], a method is proposed to weaken a conflicting GCI rather than delete it. The idea is that we first transform every GCI $C \sqsubseteq D$ into an equivalent cardinality restriction $\leq\ 0\ C \sqcap \neg D$. For a cardinality restriction which is involved in conflict, we simply weaken it as $\leq\ n\ C \sqcap \neg D$, where $n \geq 1$. We adopt this method to weaken a GCI.

Definition 8. *Let $C \sqsubseteq D$ be a GCI. A weakening $(C \sqsubseteq D)_{weak}$ of $C \sqsubseteq D$ has the form $\leq\ n\ C \sqcap \neg D$, where $n \geq 0$. We use $d((C \sqsubseteq D)_{weak}) = n$ to denote the degree of $(C \sqsubseteq D)_{weak}$.*

It is clear that $d((C \sqsubseteq D)_{weak}) = 0$ if $(C \sqsubseteq D)_{weak} = \leq\ 0\ C \sqcap \neg D$.

We now consider the weakening of a stratified DL knowledge base.

Definition 9. *Let $K = \{S_1, ..., S_n\}$ be a stratified DL knowledge base, where S_n contains completely sure terminology axioms and assertion axioms. A stratified DL knowledge base $K' = \{S'_1, ..., S'_n\}$ is said to be a weakened base of K if it satisfies the following conditions:*

- *K' is consistent,*
- *There is a bijection from $S_1 \cup ... \cup S_{n-1}$ to $S'_1 \cup ... \cup S'_{n-1}$ such that for each $\phi \in K$, $f(\phi)$ is a weakening of ϕ,*
- *$S'_n = S_n$.*

The degree of a stratum S'_i of a weakened base K' is defined as: $degree(S'_i) = \Sigma_{\phi_{weak} \in S'_i} degree(\phi_{weak})$.

The ranking between weakened bases is defined as follows.

Definition 10. *Let K be a stratified DL knowledge base. Let $K' = \{S'_1, ..., S'_n\}$ and $K'' = \{S''_1, ..., S''_n\}$ be two weakened bases of K. K' is said to be lex-preferred to K'', denoted by $K' >_{lex} K''$, if $\exists i$, $1 \leq i \leq n$ such that 1) $degree(S'_i) < degree(S''_i)$, and 2) $\forall j > i$, $degree(S'_j) = degree(S''_j)$.*

Similar to the adapted lexicographic-based approach, we can define the following inference.

Definition 11. *Let K' be a weakened base of K. K' is a lex-preferred weakened base of K if there is no consistent weakened base K'' of K such that $K'' >_{lex} K'$. A formula ψ is said to be a lexicographic conclusion of K, denoted $K \models_{lex} \psi$, if ψ is a consequence of all lex-preferred weakened bases of K.*

We illustrate the lexicographic-based approach by the following example.

Example 3. (Continuing Example 2) K' has three weakened bases: $K_1 = \{S_{11}, S_{12}, S_{13}\}$, where $S_{11} = \{\leq 1\ bird \sqcap \neg flies\}$, $S_{12} = S_2$ and $S_{13} = S_3$; $K_2 = \{S_{21}, S_{22}, S_{23}\}$, where $S_{21} = S_1$, $S_{22} = \{\leq 1\ penguin \sqcap \neg flies\}$ and $S_{23} = S_3$; $K_3 = \{S_{31}, S_{32}, S_{33}\}$, where $S_{31} = S_1$, $S_{32} = S_2$ and $S_{33} = \{penguin(Cheeky), \leq 1\ penguin \sqcap \neg bird\}$. It is easy to check that K_1 is the only lex-preferred weakened base of K. Since $K_1 \models bird(Kelly)$, we have $K \models_{lex} bird(Kelly)$.

Next, we consider the semantic computation of the lexicographic-based approach.

Definition 12. *Let $\mathcal{W}$ be a non-empty set of interpretations and $\mathcal{I} \in \mathcal{W}$, ϕ a terminology axiom of the form $C \sqsubseteq D$, and K be a DL knowledge base (K is not stratified here). The number of ϕ-exceptions for $\mathcal{I}$ is:*

$$e^{\phi}(\mathcal{I}) = \begin{cases} |C^{\mathcal{I}} \cap (\neg D^{\mathcal{I}})| & \text{if } C^{\mathcal{I}} \cap (\neg D^{\mathcal{I}}) \text{ is finite} \\ \infty & \text{otherwise.} \end{cases} \quad (1)$$

The number of K-exceptions for $\mathcal{I}$ is $e^K(\mathcal{I}) = \Sigma_{\phi \in K} e^{\phi}(\mathcal{I})$. The ordering $\preceq_K$ on $\mathcal{W}$ is: $\mathcal{I} \preceq_K \mathcal{I}'$ iff $e^K(\mathcal{I}) \leq e^K(\mathcal{I}')$, for $\mathcal{I}' \in \mathcal{W}$. $\mathcal{I} \equiv_K \mathcal{I}'$ denotes $\mathcal{I} \preceq_K \mathcal{I}'$ and $\mathcal{I}' \preceq_K \mathcal{I}$

The definition of ϕ-exception originates from Definition 6 in [20]. However, in [20], it is used to define an ordering $\preceq_K^{\pi}$ on a set of interpretations with the same *pre-interpretation* $\pi = (\Delta^{\pi}, d^{\pi})$, where Δ^{π} is a domain and d^{π} is a *denotation function* which maps every individual name a to a different element in Δ^{π}.

We define the lexicographical preference ordering as follows.

Definition 13. *Let $K = (S_1, ..., S_n)$ be a stratified DL knowledge base, where S_n contains completely sure terminology axioms and assertion axioms, and Ω be the set of models of S_n. The lexicographical preference ordering $\preceq_{lex,K}$ is defined as $\mathcal{I} \preceq_{lex,K} \mathcal{I}'$ iff $\forall i \in \{1, ..., n-1\}$, $\mathcal{I} \equiv_{S_i} \mathcal{I}'$ or $\exists i$ such that $\mathcal{I} \prec_{S_i} \mathcal{I}'$, and $\mathcal{I} \equiv_{S_j} \mathcal{I}'$ for all $n > j > i$. The set of minimal models of K w.r.t $\preceq_{lex,K}$ is denoted as $min(\Omega, \preceq_{lex,K})$.*

The following results give semantic interpretation of the lexicographic-based inference. We first prove a lemma.

Definition 14. *Let K and K' be two consistent DL knowledge bases (K and K' are not stratified), where K consists of terminology axioms. A DL knowledge base $K_{weak,K'}$ is a weakened knowledge base of K w.r.t K' if it satisfies:*

- *$K_{weak,K'} \cup K'$ is consistent, and*
- *There is a bijection f from K to $K_{weak,K'}$ such that for each $\phi \in K$, $f(\phi)$ is a weakening of ϕ.*

The set of all weakened bases of K w.r.t K' is denoted by $Weak_{K'}(K)$.

Lemma 1. *Let K and K' be two consistent DL knowledge bases, where K consists of terminology axioms, and $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models K'$. Let $l = min(d(K_{weak,K'}) : K_{weak,K'} \in Weak_{K'}(K), \mathcal{I} \models K_{weak,K'})$. Then $e^K(\mathcal{I}) = l$.*

Proof. Suppose $K_{weak,K'} \in Weak_{K'}(K)$ such that $d(K_{weak,K'}) = l$ and $\mathcal{I} \models K_{weak,K'}$. Let $\phi = C \sqsubseteq D \in K$ and $\phi_{weak} \in K_{weak,K'}$. Suppose $d(\phi_{weak}) = n$, that is, $\phi_{weak} = \leq_n C \sqcap \neg D$. Since $\mathcal{I} \models K_{weak,K'}$, $\mathcal{I} \models \phi_{weak}$. Moreover, for any other weakening ϕ'_{weak} of ϕ, if $d(\phi'_{weak}) < n$, then $\mathcal{I} \not\models \phi'_{weak}$ (because otherwise, we find another weakening $K'_{weak,K'} = (K_{weak,K'} \setminus \{\phi_{weak}\}) \cup \{\phi'_{weak}\}$ such that $d(K'_{weak,K'}) < d(K_{weak,K'})$ and $\mathcal{I} \models K'_{weak,K'}$). So $|C^{\mathcal{I}} \cap \neg D^{\mathcal{I}}| \leq n$. We further have $|C^{\mathcal{I}} \cap \neg D^{\mathcal{I}}| \geq n$. Otherwise, suppose $|C^{\mathcal{I}} \cap \neg D^{\mathcal{I}}| < n$. Then there exists ϕ_{weak} of ϕ such that $d(\phi'_{weak}) < n$, this is a contradiction. Therefore, $e^{\phi}(\mathcal{I}) = |C^{\mathcal{I}} \cap \neg D^{\mathcal{I}}| = n = d(\phi_{weak})$. That is, $e^K(\mathcal{I}) = l$.

Proposition 2. *Let $K = (S_1, ..., S_n)$ be a stratified DL knowledge base, where S_n contains completely sure terminology axioms and assertion axioms. ϕ is a DL statement and Ω is the set of models of S_n. Then $K \models_{lex} \phi$ iff $\mathcal{I} \models \phi$, for all $\mathcal{I} \in min(\Omega, \preceq_{lex,K})$.*

Proof. Suppose $\mathcal{K}$ *contains all the lex-preferred weakened bases of* K*. We need to prove that for every interpretation* $\mathcal{I}$*,* $\mathcal{I} \models \mathcal{K}$ *iff* $\mathcal{I} \in min(\Omega, \preceq_{lex,K})$*, where* $\mathcal{I} \models \mathcal{K}$ *iff* $\mathcal{I} \models K_i$ *for all* $K_i \in \mathcal{K}$*.*

"Only if part"

Suppose $\mathcal{I} \models \mathcal{K}$ *and* $\mathcal{I} \notin min(\Omega, \preceq_{lex,K})$*. Then* $\exists \mathcal{I}'$ *such that* $\mathcal{I}' \prec_{lex,K} \mathcal{I}$*. That is, there exists some* i *such that* $\mathcal{I}' \prec_{S_i} \mathcal{I}$ *and* $\mathcal{I}' \equiv_{S_j} \mathcal{I}$ *for all* $n > j > i$*. Suppose* $K' = \{S'_1, ..., S'_n\} \in \mathcal{K}$*, then* $\mathcal{I} \models K'$*. Since* $\mathcal{I}' \equiv_{S_j} \mathcal{I}$ *for all* $n > j > i$*, by Lemma 1, there exists a weakened base* S''_j *of* S_j *such that* $\mathcal{I}' \models S''_j$ *and* $degree(S'_j) = degree(S''_j)$*. This can be proved by induction over priority level* k *of* K*.*

For $k = n-1$*. Since* $\mathcal{I}' \equiv_{S_{n-1}} \mathcal{I}$*, we have* $e^{S_{n-1}}(\mathcal{I}) = e^{S_{n-1}}(\mathcal{I}')$*. By Lemma 1, we have* $degree(S'_{n-1}) = e^{S_{n-1}}(\mathcal{I})$*. Moreover, there exists a weakened base* S''_{n-1} *of* S_{n-1} *such that* $degree(S''_{n-1}) = e^{S_{n-1}}(\mathcal{I}')$*. So* $degree(S'_{n-1}) = degree(S''_{n-1})$*.*

Suppose for all $k \geq l$*, where* $l > i+1$*, there exists a weakened base* S''_k *of* S_k *such that* $degree(S''_k) = degree(S'_k)$ *and* $\mathcal{I}' \models S_k$*. Since* $k-1 > i$*, we have* $\mathcal{I} \equiv_{S_{k-1}} \mathcal{I}'$*. That is,* $e^{S_{k-1}}(\mathcal{I}) = e^{S_{k-1}}(\mathcal{I}')$*. Similarly, by Lemma 1, there exists a weakened base* S''_{k-1} *of* S_{k-1} *such that* $\mathcal{I}' \models S''_{k-1}$ *and* $degree(S''_{k-1}) = degree(S'_{k-1})$*.*

Since $\mathcal{I}' \prec_{S_i} \mathcal{I}$*, we have* $e^{S_i}(\mathcal{I}') < e^{S_i}(\mathcal{I})$*. By Lemma 1, there exists a weakened base* S''_i *of* S_i *such that* $\mathcal{I}' \models S''_i$ *and* $degree(S''_i) < degree(S'_i)$*. This is a contradiction because we then can find a weakened base* $K'' = \{S''_1, ..., S''_n\}$ *such that* $K'' >_{lex} K'$*. Therefore, if* $\mathcal{I} \models \mathcal{K}$*, then* $\mathcal{I} \in min(\Omega, \preceq_{lex,K})$*.*

"If part"

Suppose $\mathcal{I} \in min(\Omega, \preceq_{lex,K})$*. Let us assume that* $\mathcal{I} \not\models \mathcal{K}$*. Suppose* K' *is a weakened base of* K *such that* $\mathcal{I} \models K'$*, and for there does not exist a weakened base* K'' *of* K *such that* $\mathcal{I} \models K''$ *and* $K'' >_{lex} K'$*. Since* $\mathcal{I} \not\models \mathcal{K}$*, we have* $degree(K'') < degree(K')$ *for all* $K'' \in \mathcal{K}$*. Let* $K'' \in \mathcal{K}$ *and there exists an interpretation* $\mathcal{I}'$ *such that* $\mathcal{I}' \models K''$*. By Definition 10, there exists* i *such that* $degree(S''_i) < degree(S'_i)$ *and for all* $n > j > i$*,* $degree(S''_j) = degree(S'_j)$*. By Lemma 1, it is easy to show that* $e^{S_j}(\mathcal{I}') = e^{S_j}(\mathcal{I})$ *for all* $n > j > i$ *and* $e^{S_i}(\mathcal{I}') < e^{S_i}(\mathcal{I})$*. So* $\mathcal{I}' \prec_{lex,K} \mathcal{I}$*, which is a contradiction.*

This completes the proof.

According to Proposition 2, we can define the lexicographic-based inference in a semantic way.

Definition 15. *Let* $K = (S_1, ..., S_n)$ *be a stratified DL knowledge base.* ϕ *is a DL statement. Then* K *lexicographically entails* ϕ*, denoted* $K \models_{lex} \phi$*, iff* $\omega \models \phi$*, for all* $\omega \in min(\Omega, \preceq_{lex,K})$*.*

Compared with possibilistic approaches, the lexicographic-based approach is more fine-grained and can keep more original information. However, it is based on cardinality restrictions on concepts, so it cannot be used to deal with inconsistency in DLs which disallow cardinality restrictions on concepts. Furthermore, to implement the lexicographic-based approach, we need to pinpoint the instances which are responsible for the inconsistency, which is usually a hard task.

6 Related Work

A lot of work has been done on handling inconsistency in DLs [1, 2, 21, 16, 24, 18, 20]. In [1], Reiter's default logic is embedded into terminological representation formalisms. In their paper, conflicting information is treated as *exceptions*. To deal with conflicting default rules, they instantiated each rule using individuals appearing in the ABox and applied two existing default reasoning methods to compute all extensions. Then, in [2], priorities were introduced to default terminological logic such that more specific defaults are preferred to more general ones. In our stratification algorithm, we also give priority to a more specific default terminology. However, when handling inconsistency, we do not need the instantiation step. Furthermore, in [1, 2], the resolution of conflicting ABox assertions was not considered. Recently, some methods for repairing inconsistencies [24, 21] or reasoning with inconsistent ontologies [16, 18] have been proposed. A common problem with these methods is that they do not take advantage of DL expressions. If a terminological axiom is detected to be erroneous (that is, it is involved in a conflict), then it is simply deleted. In contrast, we introduce an important DL expression, i.e. cardinality restrictions, to deal with an erroneous terminological axiom. Our lexicographical-based approach is closely related to the adaptive lexicographic-based approach in [8]. However, our approach is more general than the adaptive lexicographic-based approach. The later can only deal with inconsistencies arising due to instances (or individual names in DLs) explicitly introduced in the facts (or ABox assertions), while our approach is also applicable when inconsistencies result from TBox axioms. In [20], the authors proposed an algorithm, called refined conjunctive maxi-adjustment (RCMA), for inconsistency handling in a stratified knowledge base based on cardinality restrictions. Our second inconsistency handling method is also based on cardinality restrictions. However, our method differs from RCMA method in that we only weaken those GCIs which are involved in conflict and RCMA method weakens not only conflicting GCIs but also GCIs not involved in conflict. This work is also related to some other approaches to extend DLs with nonmonotonic theories, such as defeasible description logics [13, 17, 27] and belief change in DLs [10, 11]. Defeasible description logics combines defeasible logic and description logics by adding a layer of rules from defeasible logic on top of ontologies in description logics. As in defeasible logic, an acyclic relation on the set of rules is assumed to deal with conflicting rules. This preference relation may not be a total preorder as we have assumed in the paper. The default terminology axioms are similar to the defeasible rules in defeasible description logics. However, rules are not terminology axioms. In [10, 11], AGM's theory of belief change has been applied to description logics. However, they only studied the feasibility of applying the generalized AGM postulates for belief change to DLs. No explicit belief change operators were proposed in their papers.

7 Conclusions

In this paper, we first proposed an approach to stratifying a DL knowledge base such that a more specific conflicting terminology axiom is preferred to a more general one. Then two inconsistency handling approaches first-order logic are adapted to deal with inconsistency in a stratified DL knowledge base. The first approach is the possibilistic logic approach, which drops formulas whose priority level is not larger than the inconsistency degree. The deficiency of this approach is that it suffers form the drowning problem and it will result in undesirable conclusions. In contrast, the second approach weakens the conflicting terminology axioms instead of deleting them. The semantics of the approach is also discussed.

Our weakening method is based on cardinality restrictions. However, from the implementation point of view, the cardinality restriction is not very promising as no main-stream DL reasoners supports it yet. In a future work, we will explore other DL constructors such as nominals to weaken terminology axioms. Finally, to implement our approaches, an important problem is to detect GCIs and and assertions which are responsible for the conflict. Some existing techniques on debugging of unsatisfiable classes, such as debugging methods in [24, 21], may be adapted to pinpoint the conflicting axioms in a stratified DL knowledge base.

Acknowledgements Research presented in this paper was partially supported by the European Commission NeOn (IST-2006-027595, http://www.neon-project.org/) and the Knowledge Web (IST-2004-507842, http://knowledgeweb.semanticweb.org/) projects.

References

1. F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms, *Journal of Automated Reasoning*, 14(1):149-180, 1995.
2. F. Baader and B. Hollunder. Priorities on defaults with prerequisites, and their application in treating specificity in terminological default logic, *Journal of Automated Reasoning*, 15(1): 41-68, 1995.
3. F. Baader, M. Buchheit, and B. Hollander. Cardinality restrictions on concepts. *Artificial Intelligence*, 88: 195-213, 1996.
4. F. Baader, D.L. McGuiness, D. Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, implementation and application*, Cambridge University Press, 2003.
5. S. Benferhat, C. Cayrol, D. Dubois, L. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Proc. of IJCAI'93*, 640-645, 1993.
6. S. Benferhat, R.E. Baida, and F. Cuppens. A stratification-based approach for handling conflicts in access control. *Proceedings of 8th ACM Symposium on Access Control Models and Technologies*, 189-195, 2003.
7. S. Benferhat, and R.E. Baida. A stratified first order logic approach for access control. *International Journal of Intelligent Systems*, 19:817-836, 2004.
8. S. Benferhat, S. Kaci, D.L. Berre, and M.A. Williams. Weakening conflicting information for iterated revision and knowledge integration. *Artificail Intelligence*, vol. 153(1-2):339-371, 2004.

9. Dubois, D.; Lang, J.; and Prade, H. 1994. Possibilistic logic. In *Handbook of logic in Aritificial Intelligence and Logic Programming*, Volume 3. Oxford University Press, 439-513.
10. G. Flouris, D. Plexousakis and G. Antoniou. On applying the AGM theory to DLs and OWL, In *Proc. of ISWC'05*, 216-231, 2005.
11. G. Flouris, Z. Huang, J.Z. Pan, D. Plexousakis, and H. Wache. Inconsistencies, negation and changes in ontologies. In *Proc. of AAAI'06*, to appear, 2006.
12. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web, *Scientific American*, 284(5):3443, 2001.
13. G. Governotori. Defeasible description logic, In G. Antoniou and H. Boley, editors, *Rules and Rule Markup Languages for the Semantic Web*, LNCS 3323, pages 98-112, 2004. Springer Verlag.
14. V. Haarslev and R. Möller, RACER System Description, In *Proc. of IJCAR'01*, 701-706, 2001.
15. P. Haase and J. Völker, Ontology Learning and Reasoning - Dealing with Uncertainty and Inconsistency In *Proc. of URSW'05*, 45-55. November 2005.
16. P. Haase, F. van Harmelen, Z. Huang, H. Stuckenschmidt, and Y. Sure. A framework for handling inconsistency in changing ontologies, In *Proc. of ISWC'05, LNCA3729*, 353-367, 2005.
17. S. Heymans and D. Vermeir. A defeasible ontology language, In R. Meersman and Z. Tari, editors, *Confederated International Conferences: CoopIS, DOA and ODBASE 2002*, number 2519 in LNCS, page 1033-1046, Berlin, 2002. Springer Verlag.
18. Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies, In *Proc. of IJCAI'05*, 254-259, 2005.
19. J. Lang, "Possibilistic logic: complexity and algorithm" D. Gabbay, Ph. Smets, ed., *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, vol. 5, pp: 179-220, 2000.
20. T. Meyer, K. Lee, and R. Booth. Knowledge integration for description logics, In *Proc. of AAAI'05*, 645-650, 2005.
21. B. Parsia, E. Sirin and A. Kalyanpur. Debugging OWL ontologies, In *Proc. of WWW'05*, 633-640, 2005.
22. J. Pearl. System Z: A Natural Ordering of Defaults with Tractable Applications to Nonmonotonic Reasoning. *Proc. of TARK'90*, 121-135, 1990.
23. J. Quantz and V. Royer. A Preference Semantics for Defaults in Terminological Logics, In *Proc. of KR'92*, 294-305, 1992.
24. S. Schlobach, and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies, In *Proc. of IJCAI'03*, 355-360, 2003.
25. M. Schmidt-Schauß, and G. Smolka. Attributive Concept descriptions with complements, *Artificial Intelligence*, 48:1-26, 1991.
26. G. Qi, W. Liu and D.A. Bell. A revision-based approach to handling inconsistency in description logics. In *Proc. of NMR06*, 2006.
27. K. Wang, D. Billington, J. Blee, and G. Antoniou. Combining description logic and defeasible logic for the semantic web, In Grigoris Antoniou and Harlod Boley, editors, *RuleML 2004*, LNCS, Berlin, 2004. Springer Verlag.

An Analysis of Approaches to Resolving Inconsistencies in DL-based Ontologies

Peter Haase, Guilin Qi

Institute AIFB, University of Karlsruhe,
D-76128 Karlsruhe, Germany
{haase, gqi}@aifb.uni-karlsruhe.de

Abstract. In this paper we provide an overview and analysis of approaches for dealing with inconsistencies in DL-based ontologies. We propose criteria for the comparison of the different approaches. These criteria facilitate the users to choose an appropriate approach to dealing with inconsistencies for their purpose.

1 Introduction

The problem of inconsistency (or incoherence) handling in ontologies is recently attracting a lot of attention. Inconsistency can occur due to several reasons, such as modeling errors, migration from other formalisms, merging ontologies, and ontology evolution.

One way to deal with logical contradictions is to resolve logical modeling errors whenever a logical problem is encountered[1]. Several methods have been proposed to debug erroneous terminologies and have them repaired when inconsistencies are detected [SC03,Sch05,PSK05,FS05].

Considering the varieties of approaches to resolving inconsistency, we require criteria for comparing these approaches to facilitate the selection of an appropriate approach. Three criteria are proposed in [HvHH+05]: (1) is the approach applied at development or at runtime, (2) is the input a consistent or an inconsistent ontology, (3) is the output a consistent or an inconsistent ontology (or an answer). While these criteria already provide a useful classification, they are not sufficient for the selection of an adequate approach for a given application scenario.

In this paper, we first propose some criteria for comparing approaches for dealing with inconsistencies. We then give an overview of approaches for resolving inconsistency and compare them against the proposed criteria.

The paper is organized as follows. Section 2 provides some basic notions of terminology debugging. Some evaluation criteria are proposed in Section 3. We then give an overview of approaches for repairing inconsistency in Section 4. In Section 5, we compare existing approaches against the proposed criteria. Finally, we conclude the paper in Section 6.

[1] Another way to deal with inconsistencies is to reason with inconsistent ontologies (such as the approach given in [HvHtT05]). However, the analysis of them is out of the scope of this paper.

2 Preliminaries

We assume that the reader is familiar with Description Logics (DLs) and refer to Chapter 2 of the DL handbook [BCM+03] for an excellent introduction. A DL knowledge base (or local ontology) O consists of a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. A TBox contains intensional knowledge such as concept definitions of the form $C \sqsubseteq D$, where C and D are concepts. An ABox contains extensional knowledge and is used to describe individuals. Throughout this paper, let $\mathcal{T} = \{Ax_1, ..., Ax_n\}$ be a set of (terminological) axioms, where Ax_i is of the form $C_i \sqsubseteq D_i$ for each $1 \leq i \leq n$ and arbitrary concepts C_i and D_i. A TBox is called *unfoldable* if the left-hand sides of the axioms (the defined concepts) are atomic, and if the right-hand sides (the definitions) contain no direct or indirect reference to the defined concept [Neb90].

We introduce the notion of incoherence in DLs defined in [FHP+06].

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

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 $O = \langle \mathcal{T}, \mathcal{A} \rangle$ is incoherent iff its terminology $TBox$ is incoherent. 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.*

Let us consider an ontology $O = \{C_1(a), C_2(b), C_3 \sqsubseteq C_1, C_3 \sqsubseteq C_2, C_1 \sqsubseteq \neq C_2\}$. It is clear that O is incoherent because C_3 is an unsatisfiable concept in O and it is consistent. 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 $O = \langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent.

Current DL reasoners, such as RACER, can detect logical incoherence and return unsatisfiable concepts in OWL ontologies. However, typically they do not support the diagnosis and incoherence resolution. To explain logical incoherence, it is important to debug *relevant* axioms which are responsible for the contradiction.

Definition 5. *[SC03] Let A be a named concept 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 $\mathcal{T}$ w.r.t A is denoted as $MU_A(\mathcal{T})$*

A MUPS of $\mathcal{T}$ *w.r.t* A is the minimal sub-TBox of $\mathcal{T}$ in which A is unsatisfiable. We will abbreviate the set of MUPS of $\mathcal{T}$ *w.r.t* a concept name A by $mups(\mathcal{T}, A)$. Let us consider an example from [SC03]. Suppose $\mathcal{T}$ contains the following axioms:

$$ax_1 : A_1 \sqsubseteq \neg A \sqcap A_2 \sqcap A_3 \qquad ax_2 : A_2 \sqsubseteq A \sqcap A_4$$
$$ax_3 : A_3 \sqsubseteq A_4 \sqcap A_5 \qquad ax_4 : A_4 \sqsubseteq \forall s.B \sqcap C$$
$$ax_5 : A_5 \sqsubseteq \exists s.\neg B \qquad ax_6 : A_6 \sqsubseteq A_1 \sqcup \exists r.(A_3 \sqcap \neg C \sqcap A_4)$$
$$ax_7 : A_7 \sqsubseteq A_4 \sqcap \exists s.\neg B$$

where A, B and C are atomic concept names and A_i $(i = 1, ..., 7)$ are defined concept names, and r and s are atomic roles. In this example, the unsatisfiable concept names are A_1, A_3, A_6, A_7 and MUPS of $\mathcal{T}$ w.r.t A_i $(i = 1, 3, 6, 7)$ are:

$mups(\mathcal{T}, A_1) : \{\{ax_1, ax_2\}, \{ax_1, ax_3, ax_4, ax_5\}\}$
$mups(\mathcal{T}, A_3) : \{ax_3, ax_4, ax_5\}$
$mups(\mathcal{T}, A_6) : \{\{ax_1, ax_2, ax_4, ax_6\}, \{ax_1, ax_3, ax_4, ax_5, ax_6\}\}$
$mups(\mathcal{T}, A_7) : \{ax_4, ax_7\}$

MUPS are useful for relating sets of axioms to the unsatisfiability of specific concepts, but they can also be used to calculate a minimal incoherence preserving sub-TBox, which relates sets of axioms to the incoherence of a TBox in general and is defined as follows.

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. The set of MIPS for a TBox $\mathcal{T}$ is abbreviated with $mips(\mathcal{T})$. For $\mathcal{T}$ in the above example, we get 3 MIPS:

$mips(\mathcal{T}) = \{\{ax_1, ax_2\}, \{ax_3, ax_4, ax_5\}, \{ax_4, ax_7\}\}$

3 Criteria

In this section we discuss a number of criteria for the comparison of approaches to dealing with inconsistency in distributed knowledge bases.

Applications Resolving inconsistencies is an important issue to address for a number of different tasks in ontology management. While some approaches are general and may be applied for different tasks, other approaches are developed to support particular applications such as *Repair* of inconsistent ontologies,

Evolution of ontologies, or *Merging* of potentially mutually inconsistent ontologies.Depending on the assumptions, particular approaches may be applicable to one or many of these application scenarios.

Granularity of Repair Dealing with and resolving inconsistencies can be performed on different levels of granularity. Most approaches consider a knowledge base as a set of axioms. In a trivial way, knowledge bases may be repaired by simply removing problematic axioms completely. However, often only parts of an axiom may be the true cause of an inconsistency. In such cases, more fine granular approaches (e.g. on the level of concepts) would be desirable. Some approaches are more fine granular in the sense that they allow weaken axioms by changes on to the substructure of the axioms.

Preservation of Structure Often the algorithms for diagnosing and repairing inconsistencies require some transformation of the knowledge base to some normal form (e.g. negation normal form.) While the thus obtained knowledge bases are logically equivalent, they may be un-intuitive to the user. If a repair is performed based on that normal form, the effect on the original structure of the knowledge base may be hard to reconstruct. However, the traceability of effects in terms of the original knowledge base is often important for the user. It is therefore desirable to preserve the structure of the original axioms wherever possible.

Inconsistency vs. Incoherence: Inconsistency is often used as a term to refer to a number of different types of conflicts in a knowledge base. On the level of the TBox, typically the notions of *unsatisfiability* and *incoherence* are relevant. A concept is *unsatisfiable* w.r.t. a terminology if, and only if its interpretation is empty in every model of the terminology. A TBox is incoherent if it contains an unsatisfiable concept. On the other hand, *inconsistency* of an ontology means that there exists no model at all for the ontology. Inconsistency may occur both in the TBox and the ABox.

Support for terminological and assertional knowledge When dealing with inconsistencies it often is important to consider whether an inconsistency occurs in the TBox (as part of the terminological knowledge) or whether the ABox (assertional knowledge, i.e. the data corresponding to some TBox) is inconsistent. In Description Logics, the problem of diagnosis has classically focused on dealing with coherence on the terminological level. In many applications it is however important to deal with various forms of inconsistencies and incoherence in ABoxes and TBoxes in an integrated way. Some approaches provide solutions to dealing with either one type of inconsistency and disregard the other. Others do not make the distinction at all.

Complexity Reasoning with expressive Description Logics typically is already intractable for standard reasoning tasks. Often the approaches for dealing with inconsistencies introduce an additional level of complexity. Complexity results

thus depend on the complexity on the supported logic on the one hand, and the properties of the algorithms for handling inconsistencies on the other hand. In order to assure practicability, these complexity issues need to be taken into account.

Support for Multiple/networked knowledge bases Many approaches to dealing with inconsistencies have been developed for dealing with single, isolated ontologies. Few approaches have been developed specifically for dealing with multiple ontologies that are networked or distributed in a certain way. Other approaches may be applied to multiple knowledge base scenarios by simply considering the union of the individual knowledge bases as a single knowledge base. We evaluate what kind of networking relationships are supported or how the approach can be extended to operate with multiple ontologies.

Exploitation of background / context knowledge Typical approaches for dealing with inconsistencies only consider the content of the knowledge base itself as input for diagnosis and repair. However, often it is useful to consider additional information about the relevance and importance of particular parts of the knowledge base as background knowledge. Such context information may be captured as provenance (e.g., indicating the trustworthiness of the source), in the form arguments (e.g., why certain axioms have been introduced), etc.

Interactivity, user involvement Many approaches to dealing with inconsistencies aim at a completely automated procedure. Others rely on the user to decide how to deal with particular situations. For example, it may be possible that the diagnosis of the problem is performed automatically, but the decision about how to fix a problem may be left to the user.

Availability of implementations Finally we discuss whether the approach is implemented and available for use or whether it would be feasible and desirable to implement it.

4 Overview of Approaches

In this section, we give an overview of the approaches for resolving inconsistency. When resolving inconsistency, we can either delete some erroneous axioms or weaken them. In any case, we often expect that minimal information is dropped to restore inconsistency. To achieve this requirement of minimal change, we often need a technique called debugging, which we will introduce in the following. There are mainly two important groups working on debugging and repairing incoherent ontologies. The first group comes from the Vrije Universiteit Amsterdam and the second group is the MindSwap group at University of Maryland. There are other work on resolving inconsistency which discuss specific scenario such as ontology revision and ontology integration. In the following, we first introduce the approaches for debugging and diagnosis. After that, we give a brief review of the application of AGM's belief revision theory to DLs. Finally, we introduce the approaches for knowledge integration in DLs.

4.1 Debugging and Diagnose DL-based Ontologies in MUPSter

We recapitulate two approaches given in [SHC06] which are proposed to debug an ontology, i.e. to calculate MUPS and MIPS: a top-down approach and an informed bottom-up approach.

A top-down approach to explanation: The first approach is originally proposed in [SC03]. Their debugging approach is restricted to unfoldable $\mathcal{ALC}$ TBoxes. Suppose $\mathcal{T}$ is an incoherent unfoldable TBox and A is an unsatisfiable in it. To calculate a MUPS of $\mathcal{T}$ *w.r.t* A, we can construct a tableaux from a branch B initially containing only *labelled formula* $(a \ : \ A)^{\emptyset}$ (for a new individual name a) by applying the tableaux rules as long as possible. The rules are standard $\mathcal{ALC}$-tableaux rules with lazy unfolding, and have to be read as follows: assume that there is a tableaux $T = \{B, B1, ..., B_n\}$ with $n+1$ branches. After applying one of the rules on B, we get a tableaux $T' = \{B', B_1, ..., B_n\}$ or $T'' = \{B', B'', B_1, ..., B_n\}$.

Once no more rules can be applied, we know which atoms are needed to close a saturated branch and can construct a minimization function for A and $\mathcal{T}$ according to the tableaux rules. A propositional formula ϕ is called a *minimization function for* A *and* $\mathcal{T}$ if A is unsatisfiable in every subset of $\mathcal{T}$ containing the axioms which are true in an assignment making ϕ true. Here axioms are used as propositional variable in ϕ. As we can identify unsatisfiability of A *w.r.t* a set S of axioms with a closed tableaux using only the axioms in S for unfolding, branching on a disjunctive rule implies that we need to join the functions of the appropriate sub-branches conjunctively. If an existential rule has been applied, the new branch B' might not necessarily be closed on formulas for both individuals. Assume that B' closes on the individual a but not on b. In this case $min\_function(a, B, \mathcal{T}) = \bot$, which means that the related disjunct does not influence the calculation of the minimal incoherent TBox.

Based on the minimisation function $min\_function(a, \{(a \ : \ A)^{\emptyset}\}, \mathcal{T})$, denoted ϕ, which is calculated using some rules, we then can calculate the MUPS of $\mathcal{T}$ *w.r.t* A.

From MUPS we can easily calculate MIPS based on an additional operation on sets of TBoxes, called *subset-reduction*. Let $M = \{\mathcal{T}_1, ..., \mathcal{T}_m\}$ be a set of TBoxes. The *subset-reduction* of M is the smallest subset $sr(M) \subseteq M$ such that for all $\mathcal{T} \in M$ there is a set $\mathcal{T}' \in sr(M)$ such that $\mathcal{T}' \subseteq \mathcal{T}$.

Let $\mathcal{T}$ be an incoherent TBox with unsatisfiable concepts $\Delta^{\mathcal{T}}$. The set of all MIPSs of $\mathcal{T}$, denoted $mips(\mathcal{T})$, is obtained by the following equation: $mips(\mathcal{T}) = sr(\bigcup_{A \in \Delta^{\mathcal{T}}} mups(\mathcal{T}, A))$, where $mups(\mathcal{T}, A))$ is the set of all MUPSs of $\mathcal{T}$ *w.r.t* A.

A bottom-up approach to explanation: The top-down approach is based on modifying the internals of a DL reasoner. This approach is computationally very hard in the worst-case. In [SHC06], a bottom-up approach is proposed to calculate MUPS with the support of an external DL reasoner. The main advantage of this approach is that it can deal with any DL-based ontology supported by an external reasoner. Unlike the top-down approach, they support various DL-based ontology languages, including OWL-DL.

Given an unsatisfiable concept A and a terminology $\mathcal{T}$, MUPS can be systematically calculated by checking whether A is unsatisfiable in subsets $\mathcal{T}'$ of $\mathcal{T}$ of increasing size. Such a procedure is complete and easy to implement, but infeasible in practice because the number of the subsets of $\mathcal{T}$ is exponential to the number of axioms in $\mathcal{T}$. To solve this problem, a *selection function* is introduced to control the subsets of $\mathcal{T}$ that are checked for satisfiability of A. Such a selection function selects increasingly large subsets which are heuristically chosen to be relevant additions to the currently selected subset. Although this approach is not guaranteed to give us the complete solution set of MUPS, it provides an efficient approach for debugging inconsistent terminologies.

Calculating terminological diagnoses: Terminological diagnosis, as defined in [Sch05], is an instance Reiter's diagnosis from first principles. Therefore, we can use Reiter's algorithms to calculate terminological diagnoses. An important notion in diagnosis is called a *conflict set*, which is an incoherent subset of a TBox. Given a TBox $\mathcal{T}$, a subset $\mathcal{T}'$ of $\mathcal{T}$ is a diagnosis for an incoherent $\mathcal{T}$ if $\mathcal{T}'$ is a minimal set such that $\mathcal{T} \setminus \mathcal{T}'$ is not a conflict set for $\mathcal{T}$.

Reiter introduced a hitting set tree algorithm to calculate diagnoses from conflict sets [Rei87]. Given a collection of sets $\mathcal{C}$, a *hitting set* for $\mathcal{C}$ is a set $H \subseteq \cup_{S \in \mathcal{C}} S$ such that $H \cap S \neq \emptyset$ for each $S \in \mathcal{C}$. A hitting set H for $\mathcal{C}$ is minimal if and only if the following conditions hold: (1) H is a hitting set; (2) for any $H' \in \mathcal{C}$, if $H \subset H'$, then H' is a hitting set for $\mathcal{C}$. It has been shown in [SHC06] that a subset $\mathcal{T}'$ of $\mathcal{T}$ is a diagnosis for an incoherent TBox $\mathcal{T}$ if and only if $\mathcal{T}'$ is a minimal hitting set for the collection of conflict sets of $\mathcal{T}$.

To calculate minimal hitting sets, we can adapt Reiter's hitting set tree (HS-tree) algorithm. Given a collection $\mathcal{C}$ of sets, a HS-tree T is the smallest edge-labeled and node-labeled tree, such that the root is labeled by ✓ if $\mathcal{C}$ is empty. Otherwise it is labeled with any set in $\mathcal{C}$. For each node n in T, let $H(n)$ be the set of edge labels on the path in T from the root to n. The label for n is any set $S \in \mathcal{C}$ such that $S \sqcap H(n) = \emptyset$, if such a set exists. If n is labeled by a set S, then for each $\sigma \in S$, n has a successor, n_σ joined to n by an edge labeled by σ. For any node labeled by ✓, $H(n)$, i.e. the labels of its path from the root, is a hitting set for $\mathcal{C}$.

Figure 1 shows a HS-tree T for the collection $\mathcal{C} = \{\{1,2,3,4,5,6\}, \{3,4,5\}, \{1,2,4,6\}, \{1,2\}, \{4,7\}\}$ of sets. T is created breadth first, starting with root node n_0 labeled with $\{1,2,3,4,5,6\}$. For diagnostic problems the sets in the collection are conflict sets which are created on demand. In our case, conflict sets for a terminological diagnosis problem can be calculated by a standard DL engine (by definition each incoherent subset of $\mathcal{T}$ is a conflict set).

4.2 Debugging and Repairing OWL Ontologies in SWOOP

A drawback of the debugging approach in [SC03] is that it is restricted to unfoldable ALC TBoxes. Furthermore, it is based on the tableaux algorithms for DLs. Therefore, it is dependent on the tableaux reasoner. In [PSK05,KPGS06], two

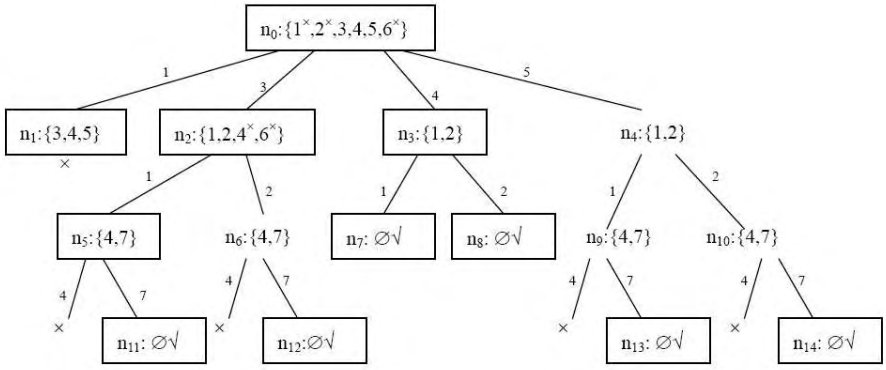

Fig. 1. HS-Tree with small conflict sets

orthogonal debugging approaches are proposed to detect the clash/sets of support axioms responsible for an unsatisfiable classes, and to identify root/derived unsatisfiable classes. The first one is a glass box approach which is based on description logic tableaux reasoner-Pellet. This approach is closely related to the top-down approach to explanation in [SHC06]. However, the approach proposed in [PSK05] is not limited to DL $\mathcal{ALC}$ and is designed for OWL DL. The second one is a black box approach [KPGS06] which is better suitable to identify dependencies in a large number of unsatisfiable classes. The approach is reasoner-independent, in the sense that the DL reasoner is solely used as an oracle to determine concept satisfiability with respect to a TBox. It consists of two main steps. In the first step, it computes a *single* MUPS of the concept and then it utilizes the Hitting Set algorithm to retrieve the remaining ones. This approach is closely related to the bottom up approach to explanation. Based on the debugging approach, in [KPSG06], the authors give a tool to repair unsatisfiable concepts in OWL ontologies. The basic idea is to rank erroneous axioms and then to generate a plan to resolve the errors in a given set of unsatisfiable concepts by taking into account the axiom ranks.

4.3 Consistent Ontology Evolution

[HS05] describes a process to support the consistent evolution of OWL DL based ontologies, which ensures that the consistency of an ontology is preserved when changes are applied to the ontology. The process consists of two main phases: (1) *Inconsistency Detection*, which is responsible for checking the consistency of an ontology with the respect to the ontology consistency definition and identifying parts in the ontology that do not meet consistency conditions; (2) *Change Generation*, which is responsible for ensuring the consistency of the ontology by generating additional changes that resolve detected inconsistencies. The authors define methods for detecting and resolving inconsistencies in an OWL ontology

after the application of a change. As for some changes there may be several different consistent states of the ontology, *resolution strategies* allow the user to control the evolution.

The methods for detecting inconsistencies rely on the idea of a selection function are to identify the *relevant* axioms that contribute to the inconsistency. In the most simple case, syntactic relevance – considering how the axioms of the ontology are structurally connected with the change – is used. Based the selection function, algorithms to find *minimal inconsistent subontologies* and *maximal consistent subontologies* are presented.

The approach only supports repairs by removing complete axioms from the ontology, a weakening based on a finer granularity as future extension is mentioned, but no algorithms are proposed. The approach does not make a distinction between ABox and TBox axioms, as such both ABox and TBox inconsistencies are trivially supported. Further, the approach does not provide any explicit support for dealing with networked ontologies.

4.4 AGM's Postulates for Belief Change and Description Logics

AGM's theory of belief change [Gar88] has been widely used to deal with logical inconsistency resulting from revising a knowledge base by newly received information. There are three types of belief change, i.e. *expansion*, *contraction* and *revision*. Expansion is simply to add a sentence to a knowledge base; contraction requires to consistently remove a sentence from a knowledge base and revision is the problem of accommodating a new sentence to a knowledge base consistently. Alchourrón, Gardenfors and Markinson proposed a set of postulates to characterize each belief change operator. The application of AGM' theory to description logics is not trivial because it is based on the assumptions that generally fail for DLs [FPA04]. For example, a DL is not necessarily closed under the usual operators such as $\neg$ and $\wedge$. In [FPA05,FPA06], the basic AGM postulates for contraction were generalized to DLs and the feasibility of applying the generalized AGM theory of contraction to DLs and OWL was studied. There work is based on the *coherence model*[2]. That is, the knowledge base is closed under consequence operation, i.e., $K = Cn(K)$, where K is a knowledge base and Cn is the consequence operation of the underlying language. They showed that in many important DLs, such as $\mathcal{SHOIN}(\mathcal{D})$ and $\mathcal{SHIQ}$, it is impossible to define a contraction operator that satisfies the generalized AGM postulates. In [FHP$^+$06], the authors first defined the notions of axiom negation and discussed postulates for revision. However, explicit construction of a revision operator was not considered in these papers.

4.5 Knowledge Base Revision in Description Logics

The work in [FPA04,FPA05,FPA06] presumes that the original DL knowledge base is closed under logical consequence relation and the result of revision is

[2] The notion of coherence model has nothing to do the notion of incoherence in DLs defined in Section 2.

still a DL knowledge base which is closed under logical consequence relation. In [QLB06a], the revised AGM postulates for belief revision in [KM92] were generalized and two revision operators which satisfy the generalized postulates were given. One operator is the weakening-based revision operator which is defined by weakening of statements in a DL knowledge base. The weakening-based revision operator may result in counterintuitive results in some cases, so another operator was proposed to refine it. It was shown that both operators capture some notions of minimal change.

4.6 Knowledge integration for description logics

In [MLB05], an algorithm, called *refined conjunctive maxi-adjustment* (RCMA for short) was proposed to weaken conflicting information in a *stratified* DL knowledge base and some consistent DL knowledge bases were obtained. To weaken a terminological axiom, they introduced a DL expression, called *cardinality restrictions* on concepts. However, to weaken an assertional axiom, they simply delete it. In [QLB06b], the authors first define two revision operators in description logics, one is called a weakening-based revision operator and the other is its refinement. The revision operators are defined by introducing a DL constructor called *nominals*. The idea is that when a terminology axiom or a value restriction is in conflict, they simply add explicit exceptions to weaken it and assume that the number of exceptions is minimal. Based on the revision operators, they then propose an algorithm to handle inconsistency in a *stratified* description logic knowledge base. It was shown that when the weakening-based revision operator is chosen, the resulting knowledge base of their algorithm is semantically equivalent to that of the RCMA algorithm. However, their syntactical forms are different.

5 Discussion

We have introduced some existing approaches for resolving inconsistency and incoherence in DLs. In this section, we compare these approaches with respect to the evaluation criteria proposed in Section 3.1. The results of comparison are compactly summarized in Table 1 and Table 2.

According to Table 1 and Table 2, MUPSter and SWOOP are mainly applied to debug and repair incoherence, whilst other approaches are applied to deal with inconsistency. When resolving incoherence, MUPSter will delete axioms in the TBox and SWOOP deletes either axioms or concepts in the TBox. To resolve inconsistency, the knowledge base revision approaches and knowledge integration approaches either delete axioms in a DL knowledge base or remove some instances which are responsible for inconsistency. However, the AGM-based approach for revision in DLs and the approach on consistent ontology evolution simply delete the whole axioms. Before dealing with incoherence, both MUPster and SWOOP may split the axioms into smaller axioms, so the structure will be lost. One of the knowledge base revision approaches given in [QLB06a], called

Table 1. Evaluation results

Criteria	MUPSter	SWOOP	AGM-based approaches
Application	debugging, repair	debuging, repair	revision
Granularity	axiom	axiom or concept	axiom
Preservation of structure	partially	partially	no
Support for ABox, TBox	TBox	TBox	TBox and ABox
Inconsistency vs. Incoherence	incoherence	incoherence	inconsistency
Complexity	PSPACE-complete	PSPACE-hard	PSPACE-hard
User involvement	no	yes	no
Availability of implementation	yes	yes	no
Support for networked ontologies	no	no	no
Exploitation of context	no	partially	no

Table 2. Evaluation results

Criteria	consistent ontology evolution	knowledge base revision	knowledge integration
Application	ontology evolution	revision	merging
Granularity	axiom	axiom or instance	axiom or instance
Preservation of structure	yes	partially	partially
Support for ABox, TBox	TBox and ABox	TBox and ABox	TBox and ABox
Inconsistency vs. Incoherence	inconsistency	inconsistency	inconsistency
Complexity	PSPACE-hard	PSPACE-hard	PSPACE-hard
User involvement	yes	no	no
Availability of implementation	yes	no	no
Support for networked ontologies	no	no	yes
Exploitation of context	no	no	yes

refined weakening-based revision approach, also requires to split axioms, so it does not preserve the structure of the axioms. The weakening-based revision approach in [QLB06a] does not change the structure of the axioms. The knowledge integration approach proposed in [MLB05] needs to transform all terminology axioms into the cardinality restrictions on concepts. So the structure of the axiom is lost. It has been shown that debugging in DL $\mathcal{ALC}$ is PSPACE-complete in [SC03] because it is based on tableaux algorithm. The glass box approach is also based on tableaux algorithm, so it is at least PSPACE-hard. Other approaches are at least PSPACE-hard because they need to checking inconsistency, which is PSPACE-hard. Among all the approaches, only MUPSter and SWOOP are implemented and only SWOOP has user interface. The MUPSter, AGM-based approaches and the knowledge base revision approaches do not explore context information to deal with incoherence or inconsistency. Whilst SWOOP and knowledge integration approaches explore the ranking information to resolve incoherence or inconsistency.

6 Conclusion

In this paper, we have provided a survey of existing work on resolving inconsistency in DL-based ontologies. We then proposed a set of criteria for comparing between different approaches.

It does not come as a surprise that none of the surveyed approaches is universally applicable for any application scenario, instead different approaches are good for different purposes. Our comparison aims at supporting the selection of the right tools for the job.

As a general observation we see that the use of context information in the process of repairing ontologies as well as the support for multiple, networked ontologies is still underdeveloped. As part of our future work, we plan to advance the techniques for dealing with inconsistencies in these directions.

Acknowledgements Research presented in this paper was supported by the European Commission under contract IST-2006-027595 NeOn (http://www.neon-project.org/).

References

[BCM+03] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[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.

[FPA04] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. Generalizing the agm postulates: preliminary results and applications. In *NMR*, pages 171–179, 2004.

[FPA05] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. On applying the AGM theory to DLs and OWL. In *Proc. of 4th International Conference on Semantic Web (ISWC'05)*, pages 216–231. 2005.

[FPA06] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. Evolving ontology evolution. In *SOFSEM*, pages 14–29, 2006.

[FS05] Gerhard Friedrich and Kostyantyn M. Shchekotykhin. A general diagnosis method for ontologies. In *Proc. of 4th International Conference on Semantic Web (ISWC'05)*, pages 232–246, 2005.

[Gar88] Peter Gardenfors. *Knowledge in Flux-Modeling the Dynamic of Epistemic States*. The MIT Press, Cambridge, Mass, 1988.

[HS05] Peter Haase and Ljiljana Stojanovic. Consistent evolution of owl ontologies. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 182–197. Springer, 2005.

[HvHH+05] Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure:. A framework for handling inconsistency in changing ontologies. In *Proc. of 4th International Semantic Web Conference (ISWC'05)*, pages 353–367. Springer, 2005.

[HvHtT05] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proc. of 19th International Joint Conference on Artificial Intelligence(IJCAI'05)*, pages 254–259. Morgan Kaufmann, 2005.

[KM92] Hirofumi Katsuno and Alberto O. Mendelzon. Propositional knowledge base revision and minimal change. *Artif. Intell.*, 52(3):263–294, 1992.

[KPGS06] Aditya Kalyanpur, Bijan Parsia, Bernardo Cuenca Grau, and Evren Sirin. Justifications for entailments in expressive description logics. Technical report, University of Maryland Institute for Advanced Computer Studies (UMIACS), 2006.

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

[MLB05] Thomas Meyer, Kevin Lee, and Richard Booth. Knowledge integration for description logics. In *Proc. of 20th National Conference on Artificial Intelligence (AAAI'05)*, pages 645–650. AAAI Press, 2005.

[Neb90] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artif. Intell.*, 43(2):235–249, 1990.

[PSK05] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In *Proc. of WWW'05*, pages 633–640, 2005.

[QLB06a] Guilin Qi, Weiru Liu, and David A. Bell. Knowledge base revision in description logics. In *Proc. of JELIA'06*, pages 386–398. Springer Verlag, 2006.

[QLB06b] Guilin Qi, Weiru Liu, and David A. Bell. A revision-based algorithm for handling inconsistency in description logics. In *Proc. of NMR'06*, pages 124–132, 2006.

[Rei87] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[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.

[SHC06] Stefan Schlobach, Zhisheng Huang, and Ronald Cornet. Inconsistent ontology diagnosis: Evaluation. Technical report, Department of Artificial Intelligence, Vrije University Amsterdam; SEKT Deliverable D3.6.2, 2006.