ISWC 2008

The 7th International Semantic Web Conference

Dimitris Plexousakis
Jeff Z. Pan
Giorgos Flouris
Mathieu d'Aquin
Grigoris Antoniou

Ontology Dynamics (IWOD 2008)

October 27, 2008

The 7th International Semantic Web Conference
October 26 – 30, 2008
Congress Center, Karlsruhe, Germany

ISWC 2008

Platinum Sponsors

Ontoprise

Gold Sponsors

BBN
eyeworkers
Microsoft
NeOn
SAP Research
Vulcan

Silver Sponsors

ACTIVE
ADUNA
Saltlux
SUPER
X-Media
Yahoo

The 7th International Semantic Web Conference
October 26 – 30, 2008
Congress Center, Karlsruhe, Germany

ISWC 2008

Organizing Committee

General Chair
Tim Finin (University of Maryland, Baltimore County)

Local Chair
Rudi Studer (Universität Karlsruhe (TH), FZI Forschungszentrum Informatik)

Local Organizing Committee
Anne Eberhardt (Universität Karlsruhe)
Holger Lewen (Universität Karlsruhe)
York Sure (SAP Research Karlsruhe)

Program Chairs
Amit Sheth (Wright State University)
Steffen Staab (Universität Koblenz Landau)

Semantic Web in Use Chairs
Mike Dean (BBN)
Massimo Paolucci (DoCoMo Euro-labs)

Semantic Web Challenge Chairs
Jim Hendler (RPI, USA)
Peter Mika (Yahoo, ES)

Workshop chairs
Melliyal Annamalai (Oracle, USA)
Daniel Olmedilla (Leibniz Universität Hannover, DE)

Tutorial Chairs
Lalana Kagal (MIT)
David Martin (SRI)

Poster and Demos Chairs
Chris Bizer (Freie Universität Berlin)
Anupam Joshi (UMBC)

Doctoral Consortium Chairs
Diana Maynard (Sheffield)

Sponsor Chairs
John Domingue (The Open University)
Benjamin Grosof (Vulcan Inc.)

Metadata Chairs
Richard Cyganiak (DERI/Freie Universität Berlin)
Knud Möller (DERI)

Publicity Chair
Li Ding (RPI)

Proceedings Chair
Krishnaprasad Thirunarayan (Wright State University)

Fellowship Chair
Joel Sachs (UMBC)

Organization

The 2nd International Workshop on Ontology Dynamics (IWOD-08) was organized as part of the 7th International Semantic Web Conference (ISWC-08). It was held on October 27, 2008, in Karlsruhe, Germany.

IWOD-08 website: http://www.abdn.ac.uk/~r01srt7/iwod2008/index.php
ISWC-08 website: http://iswc2008.semanticweb.org/

Organizing Committee

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

Program Committee Members

Mathieu d'Aquin (*The Open University, United Kingdom*)
Grigoris Antoniou (*FORTH, Greece*)
Alessandro Artale (*Free University of Bozen-Bolzano, Italy*)
James Delgrande (*Simon Fraser University, Canada*)
Giorgos Flouris (*FORTH, Greece*)
Peter Haase (*University of Karlsruhe, Germany*)
Zhisheng Huang (*Vrije Universiteit Amsterdam, The Netherlands*)
Thomas Meyer (*Meraka Institute, South Africa*)
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*)

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. Creating ontologies is simply the first step; ontologies, just like any structure holding knowledge, need to be updated as well. There are several reasons why an ontology should change: changes could be initiated because of a change in the world being modeled; the users needs may change, requiring a different conceptualization; knowledge previously unknown, classified or otherwise unavailable may become known; or a design flaw may have been noticed 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 2nd International Workshop on Ontology Dynamics (IWOD-08) 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 9 submissions on various topics related to ontology dynamics; the material collected in this volume is the result of a careful evaluation process, which selected 5 contributions for presentation in the workshop and inclusion in these proceedings.

We would like to thank the Program Committee members for their excellent reviews which ensured that the best possible material was presented in the workshop and appears in these proceedings, the organizers of the ISWC-08 conference for supporting this workshop and, of course, the authors of all submitted papers, as well as all the participants of the workshop for their interest in IWOD-08. Last but not least, we would like to thank Thomas Meyer from Meraka Institute for agreeing to give an invited talk (titled: “Ontology Dynamics Meets Belief Change”) at the workshop.

Jeff Z. Pan
Dimitris Plexousakis
Mathieu d’Aquin
Grigoris Antoniou
Giorgos Flouris
(editors)

October 2008

Table of Contents

Controlled Ontology Evolution through Semiotic-based Ontology Evaluation

Renata Dividino[1] and Daniel Sonntag[2]

[1] Fraunhofer Institute for Computer Graphics Research (IGD), Germany
renata.dividino@igd.fraunhofer.de
[2] German Research Center for Artificial Intelligence (DFKI), Germany
daniel.sonntag@dfki.de

Abstract. Ontology evaluation is an important issue that must be addressed during the lifecycle of an ontology because it assures that the ontology reflects the desired requirements during selection, design, population, evolution, usage, or other processes of the ontology lifecycle. Ontology evaluation aims at assessing the quality and the adequacy of an ontology from the point of view of a particular predefined criterion, for use in a specific context, for a specific purpose. Here, we demonstrate that evaluation is an important issue for ontology evolution. To capture the different nature of evaluation and evolution criteria, we use a tool (S-OntoEval) aiming at assessing ontology quality by implementing existing quality metrics that draw upon semiotic theory. An evaluation of the SWIntO ontology of the SmartWeb project shows its controlled evolution by quality measurement and improvement.

1 Introduction

Ontology learning concerns the development of automatic methods for the extraction of a domain model from a domain-specific data set. The domain model itself is often built up in terms of populating ontology instances [3]. In the ontology learning domain, it became obvious that benchmarks for evaluation of information extraction methods are missing, separately and in combination with the resulting populated ontologies. We argue that the scope of evaluating ontologies is much broader; only recently, ontology evaluation tools have been developed to tackle the ontology evaluation problems [24, 10, 1]. How does this relate to the development of an integrated approach to the evolution of ontologies?

Interestingly, the *evolution* of ontology (of which we provided an instance by ontology learning and population) can be said to be much related to the *evaluation* of ontologies. For example, if the evaluation of the ontologies shows improved measurements in terms of quality tests when a new version is distributed (cf. change management), the ontology really has evolved. In fact, one could also think of an evaluation based definition of ontology evolution. In this paper, we describe our work toward the controlled evolution of ontologies through semiotic-based evaluation methods. Rather than speaking of ontology evolution when new concepts are added or instances populated (only), we follow a semiotic-based ap-

proach which addresses structural, semantic, and pragmatic aspects of ontology evaluation (and evolution) in a theoretical framework.

In Section 2 we present a semiotic-based ontology evaluation approach and examine how it can be used for the controlled evolution of ontologies. Section 3 introduces the semiotic evaluation framework, as well as existing metrics and measures, followed by implementation details of the tool. In Section 4 we discuss a case study; Section 5 concludes.

2 Semiotic based Ontology Evaluation and Evolution

The ontology evaluation is a basis for defining what a good ontology is. What is a good ontology? This question, although quite crucial for developing ontology-based applications from a practical perspective, cannot be answered clearly, due to the fact that no commonly agreed upon definition of what "good" means exists for ontologies. Fundamentally, ontologies are determined to be good or bad based on correlations between success histories, user satisfaction feedback and measures. Related work on ontology evaluation presents various perspectives on the quality of an ontology such as reusability, inference speed, logical consistency, application performance support, and domain coverage. Basically, the decision on which assessment method is to be used depends on which dimension of the ontology will be evaluated.

At first place, an ontology is a fairly complex graph structure; it is composed of concepts and relations, some of which are explicitly defined, others which follow from a set of axioms. Additionally, an ontology's structure defines meaning explicitly by using a formal language, i.e. ontologies are graphs having formal semantics. Several criteria for evaluating ontologies are focused on structural aspects (topological and logical adequacy), i.e. on evaluating the internal structure of the ontology such as connectivity, modularity and logical consistencies.

Other dimensions become apparent in an ontology depending on the context, which in turn is given by the way the ontology is chosen, built, exploited, etc. Ontologies express an intended meaning (or conceptualization) and meanings vary more or less depending on the context in which they are used. According to this, a good ontology should be a close approximation to the conceptualization that is supposed to be described. A context-specific ontology evaluation, for instance, is applied when the quality of an ontology directly influences to the output, or performance of an application that uses it.

Finally, considering that not all ontology users are ontology experts, but must gather information from an ontology based on its annotations (metadata), one might also assess the ontology quality depending on the level of annotation. A good ontology is the one which allows the users to obtain the relevant information. A well annotated ontology allows users to easily recognize its properties; to find out, given a task, how economically or computationally suitable it is.

By analyzing the heterogeneous nature of works with respect to the assessment of the ontology, we could note three groups of evaluation methods: those assessing the graph structure and formal semantics of an ontology; the second

group assessing an ontology's intended use; and the last group addressing the quality level of the ontology's annotations. The three assessment approaches are directly analogous to a semiotic[3] assessment. Next section shows that the semiotic-based evaluation provides a metric framework to perform a complete ontology evaluation.

2.1 Semiotic based Evaluation

Following Saussure [4], semiotics embraces the traditional branches of linguistics: syntactics (deals with the *structure* of signs and sign systems), semantics (deals with the *meanings* of signs and sign systems; that is, the meanings of words, sentences, gestures, paintings, mathematical symbols, etc.), and pragmatics (deals with *inferential meaning*, aspects of communication expressed through social context).

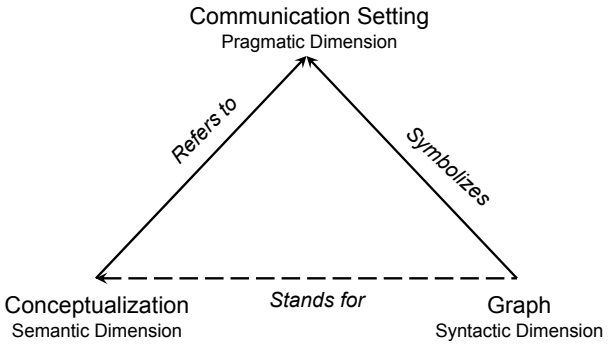


Fig. 1. Ontology as a semiotic object

An ontology is a semiotic object [1], and its quality may be assessed with respect to its structure (syntax and formal semantics), its intended conceptualization (cognitive semantics), and its communication setting (pragmatics), see Figure 1. Thus, semiotic theory is used as an "umbrella" framework to determine the quality of an ontology. Anticipatorily, what concerns the bridge to ontology evolution, the evolution of metadata (e.g., see [23, 16]) for ontologies directly corresponds to structural, functional, and pragmatic ontology quality improvement, respectively.

[3] Semiotic, from the greek word *seemeiootikee*, denotes the study of signs, what they represent and signify, and how we act and think in their universe. Semiotic Theory was developed independently by the logician and philosopher Charler Sanders Peirce and the linguist Ferndinand de Saussure. Saussure's approach was a generalization of formal, structuralist linguistics; Peirce's was an extension of reasoning and logic in the natural sciences.

2.2 Semiotic based Evolution

The lifecycle management of an ontology is mainly composed of activities of the ontology engineering process, ontology usage, and the interplay between usage and engineering activities [26]. Methodologies for ontology engineering [17] agree that the main ontology engineering steps are: requirement analysis, development, evaluation, and maintenance. Ontology usage, though, encompasses all activities performed with an ontology after it has been engineered such as retrieval, reasoning, population, etc. The interplay between usage and engineering activities is realized by processes such as population and fusion which lead to ontology changes. Ontology changes over time and these changes have to be considered during the ontology lifecyle with ontology evolution.

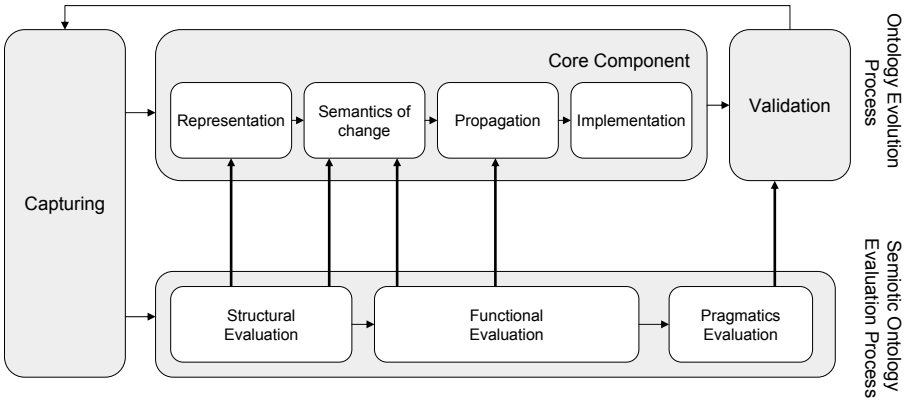


Fig. 2. Semiotic evaluation implements certain stages of the evolution process

The process of ontology evolution [2], the loop from usage back to engineering activities, consists of six phases: (1)capture changes, (2)change representation, (3)semantics of changes, (4)change propagation, (5)change implementation, and (6)change validation. Semiotic ontology evaluation can be used to support and control evolution at specific steps of the ontology evolution process:

- *Capture of Change*: capture changes can be seen as the process of re-definition of design requirements (which leads back to requirement analysis activity of the ontology engineering process). Explicit requirements induce changes on the ontology structure. Changes on ontologies basically reflect new interest of users and are motivated by the use of the ontology in a context (usage-driven changes), and/or ontological changes from the instances by applying techniques such as data-mining or various heuristics (data-driven changes) [12]. Fundamentally, the *Capture of Change* step characterizes the first step

of any evaluation process: the definition of evaluation criteria (reasons of changes).

- *Change Representation*: change representation is about making the ontology changes visible in a form of an adequate representation [12]. This process leads back to the development activity of the ontology engineering process, which is followed by the evaluation activity. However, an evaluation step done at this point (e.g., checking for the adequacy of the changes' representation with respect to the ontology language and constructs, i.e., analysis at the ontology's syntax level), may avoid ontology errors that follow the executions of the evolution process.
- *Semantics of Change*: ontology changes need to be managed such that the ontology remains consistent. The consistency of an ontology can be defined in terms of structural, logical, and user-defined consistency conditions [23]. *Structural Consistency* ensures that the ontology obeys the constraints of the ontology language with respect to how the constructs of the ontology language are used. *Logical Consistency* regards the formal semantics of the ontology: viewing the ontology as a logical theory, an ontology is logically consistent if it is satisfiable, meaning that it does not contain contradicting information. *User-defined Consistency* verifies if there are definitions which explicitly contradict the ones defined by the user and they must be met for the ontology in order to be consistent. This analysis is the most subjective one because the evaluation should check definitions of consistency that are not captured by the underlying ontology language itself, but rather given by some application or usage context.

  Consistency checking defined for ontology evolution (defined in [23]) is analogous to the semiotic assessment. The semiotic ontology evaluation approach aims at checking ontology inconsistencies, i.e., assessing the ontology quality and adequacy from the point of view of a particular predefined criterion such as its formal semantics (structural level) and its use in a specific context (functional level).
- *Change Propagation*: ontologies often reuse and extend other ontologies. Therefore, an ontology change might potentially corrupt ontologies depending on the updated ontology [2]. The change propagation step aims to verify consistency of dependent ontologies after an ontology update has been performed. In this phase of the ontology evolution process, the semiotic ontology evaluation approach plays an important role. The evaluation at the functional level, which evaluates the adequacy of the ontology for the purpose it was develop, enables engineers to measure the approximation of the updated ontology's intended conceptualization (or intended meaning) to the dependent ontologies' intended conceptualization.[4]
- *Change Validation*: in this phase the decision to perform changes on the on-

[4] Others structural metrics for checking consistency in distributed environments such as the *Replication Ontology Consistency* metric developed from [13] can be incorporated to the semiotic evaluation framework in order to assure more accurate for the specific problem.

tology is thought over; the user is able to approve the changes applied or to reverse them. Evaluation results can contribute to the change validation stage of the ontology evolution process by guiding or giving controlled direction on how one can evolve an ontology. To engineers, in turn, the results present an overview of the adequacy of the changes with respect to the ontology structure, ontology use, etc. for a change validation. Furthermore, the evaluation at the pragmatic level of the ontology can assure documentation quality in terms of adequate ontology metadata.
Evaluating an ontology at its structural and functional level already enhances its pragmatics quality since evaluation results at these semiotic levels can be attached to the ontology as documentation, and thus pragmatics analysis can be helpful to provide better visibility in the process of ontology versioning.

From the above it can be said that ontology evaluation is an important process of the ontology lifecycle that can guide engineers in making incremental improvements; ontology evaluation is not just an activity to be executed during the early phases of the ontology lifecycle (during the ontology engineering process) but to be executed in all usage and engineering activities for ontology evolution, due to continual changes.

3 Semiotic Ontology Evaluation Tool and Current Metrics

From an ontology engineer's point of view, the quality of ontologies is critical for the success of their improvements, cost reduction and maintenance. Likewise, from the user's point of view, it is critical to the success of the final applications that reuse and share these ontologies. In this work, we re-use our own implementation of the semiotic-based evaluation approach (also cf. [1]), the S-OntoEval tool (described in [6, 7]).

As introduced, an ontology is a semiotic object [1] (see Figure 1). Figure 3 presents the three steps of the semiotic evaluation approach. Figure 3(a) shows that the semiotic evaluation starts by assessing the quality of the ontology structure, i.e., its graph (G). Figure 3(b) presents the assessment of the relation between ontology structure and its conceptualization (C), i.e., what the ontology aims to conceptualize through its structure. The last step is illustrated in Figure 3(c), the assessment of the relation between ontology structure and conceptualization, and the user who uses the ontology, i.e., its communication setting (CS). The evaluation steps show the inter-dependency between the semiotic levels, i.e., the evaluation on a higher semiotic level depends on the analysis on the previous level for its applicability.

3.1 Structural Evaluation

The evaluation on the structural level of an ontology focuses on the evaluation of its syntax (its internal structure) and formal semantics, thus assessing the

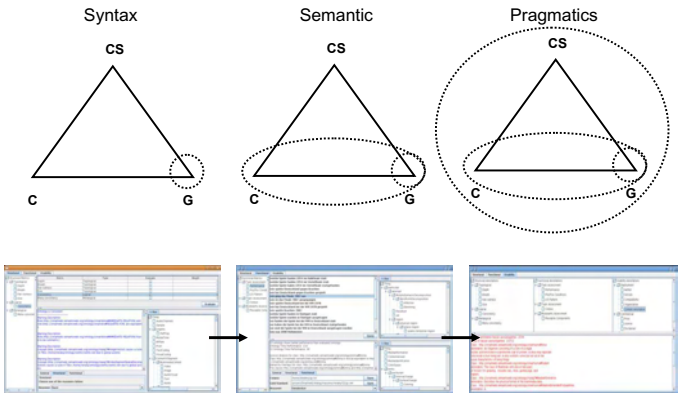


Fig. 3. Steps of the semiotic evaluation approach and the S-OntoEval tool [6, 7]

ontology's topological and logical dimensions. An ontology's structural dimension is represented by its graph structure. With this in mind, the topological dimension of an ontology may be measured by means of (graphical) structure metrics without taking into account the semantics. Although topology-related measures are certainly relevant to the problem of ontology evaluation, assuring a well structured ontology, their level of abstraction makes them unsuitable for a complete ontology evaluation. Basically, structure-based measures are appropriate for preventing that errors in the graph structure elements propagate to the other evaluation levels. An ontology's formal semantics is represented by its logical and meta-logical properties. Formal semantics of an ontology define meaning associated with the representation language used to represent or interpret data. Consequently, the evaluation of the logical and meta-logical adequacy of an ontology depends heavily on the choice of the ontology language's formal semantics (as Description Logics). Some measures related to logical dimensions are described in [1, 11, 27, 9].

The S-OntoEval implements, with help of reasoners, *consistency checking*. The *consistency checking* evaluation checks logical-adequacy of the ontology model. The consistency of an ontology model refers to its lack of contradiction, i.e., none of the facts deducible from the model contradicts others. Therefore, the ontology consistency can be considered as an agreement among ontology entities with respect to the semantics of the underlying ontology language. The use of an inconsistent ontology with respect to its theory may, for example, lead to erroneous results in the evolution process.

3.2 Functional Evaluation

The functional dimension coincides with the main purpose of an ontology, i.e., specifying a given conceptualization, or a set of contextual assumptions about

a domain [1]. Thus, an evaluation on the functional level is intended to assess the accuracy of an ontology with respect to its conceptualization. Measures for evaluating the functional dimension are based on the relation between the ontology graph and its intended meaning (its cognitive semantics). This evaluation level is the most subjective one, since the relationship between an ontology and a conceptualization is not straightforward and always dependent on an agent.

The ontology graph and its meaning are kept together by the agent that conceives the conceptualization (the "cognitive" semantics) and on the semantic space that formally encodes the conceptualization (the "formal" semantics) (see Figure 1). Formal semantics can help to define the context by providing an interpretation of the graph, but interpretations depend on the commitment of (one or more) agent(s) that are motivated by a typical expertise in that context. In order to deal with the problem of subjectiveness, an appropriate evaluation strategy should involve ways to measure the ontology, extended to a given expertise, competency or task. The matching problem requires us to find ways to measure the extent to which an ontology mirrors something that is "in the experience" of a given community and that includes not only a corpus of documents, but also theories, practices and know-how that are not necessarily represented in their entirety in the available documents.

In the S-OntoEval tool, the functional dimension is assessed through the *task-based* evaluation measure [21]. A task-based evaluation approach deals with measuring the quality of an ontology based on the adequacy of the ontological model for a specific task. The evaluation points out how effective a given ontology is in light of a well-defined task. A task-based evaluation is a quantitative evaluation for measuring the performance of an ontology for a given task, which, first of all, provides empirical means for comparing the performance of a system with and without ontological support, and secondly, provides a straightforward way of comparing ontologies competing for the same task by comparing their performance. The main idea is to observe the improvement or degradation of the performance of the task which depends on the ontology used. The task-based evaluation approach is a gold standard-based method [14] which involves the creation of a validate corpus of answers for a certain task (a gold standard corpus as evolved version to test the evolution process). The gold standard is used as a reference for checking the comparative performance.

Within the ontology evolution process, the *task-based* evaluation is used to observe the improvement or degradation of the task's performance when using an ontology and later its evolved version. Consequently this assessment is used to verify whether the evolved version remains consistent with respect to the purpose it is used for, i.e. whether it (still) provides the best performance to the given task or not.

3.3 Usability Evaluation

Evaluation on the usability (profile) level is intended to assess the ontology's profile. An ontology profile is a set of ontology annotations, i.e., the metadata about

an ontology and its elements. Ontology annotations address the communication purpose of the ontology, hence its pragmatics.

Metadata has become important as the number of existing ontologies and ontology libraries grows, as well as for ontology versioning. Users need ways to find out among existing ontologies, the one which is more suitable for the intended use, as well as to be sure that the ontology chosen for a project, according to some specific formal criteria, is the most appropriate one. An ontology's profile is intended to guide users in making such decisions (provides users' recognition). The work of [19] accentuates the importance of an evaluation at this usability level (which we put in context of the evolution of annotations).

Therefore, ontology recognition requires an adequate set of annotations, which can be preliminarily classified in structural, functional and user-oriented annotations containing all relevant information about the ontology, from the point of view of a specific user or group (ontology engineers and ontology consumers). Unfortunately, none from the existent annotation tags provided by ontology languages follows the semiotic approach. An extension of the OWL syntax annotation following the semiotic approach is proposed in [6]. In the S-OntoEval tool, the usability-profile is assessed by the *annotation analysis* metric which quantifies the total amount of metadata linked to the ontology. Roughly speaking, evolving the annotations means to control the increase of the total amount of annotations and their controlled language use, e.g., using a subset of English terms and a simple grammar.

4 Case Study - Evaluation

The SmartWeb project [28, 22] combines multimodal and mobile user interface technology with question answering technology. The goal of the project is to lay the foundations for multimodal user interfaces in distributed and composable Semantic Web services on mobile devices. The SmartWeb project relies on the representation of knowledge by means of the SWIntO [20] ontology. SWIntO integrates a selection of domain and task specific ontologies with general/foundational ontologies DOLCE [15] and SUMO [18], which results in a relatively deep hierarchical structure with foundational DOLCE classes on top, followed by general SUMO classes and finally classes in the soccer domain, besides task-specific classes on discourse and navigation.

The assessment of the SWIntO using the S-OntoEval tool starts by *consistency checking*. The main goal is to assure formal consistency, and to avoid that formal anomalies propagate to the other evaluation levels. *Consistency checking* was done twice by plugging the RACER (Renamed ABox and Concept Expression Reasoner)[5] and Pellet[6] reasoners to the tool. The *evaluation notification* consists of a list of ontology errors and warnings which guides engineers to correct and remove inconsistencies until they obtain a consistent ontology.

[5] See *http://www.racer-systems.com* for more information.
[6] See *http://www.mindswap.org/2003/pellet* for more information.

In the SmartWeb system, the SWIntO is used to support the execution of different tasks. These tasks' results, and/or performance, might depend on the ontology being used. The *question answering* task from the SmartWeb system, for instance, enables users to pose open-domain multimodal questions and delivers answers relying on a multitude of SWIntO resources. One of the main goals of the SmartWeb project is to assure good performance during the user-system dialogue process, and for that reason it is of great importance to use the ontology which is the most suitable one for the application (supporting better performance for the given task by this evolution). Basically, the task-based evaluation process is demonstrated by comparing the time-performance of the *question-answering* task to an "ideal time-performance". The ideal time-performance is obtained by plugging a *Gold Standard* ontology (the evolved version) into the system and running the task; the outputs of this process are the *gold standard answers* and their time-performance. The Gold Standard ontology is used as a reference ontology for checking the performance of an ontology driven system.

The Gold Standard ontology used here consist of an evolved version of the SWIntO. The Gold Standard ontology includes ontology changes in one of the SWIntO's domain ontology, the Sport Event ontology. The Sport Event ontology features a large set of instances that primarily model facts of the Football World Cup 2002, serving as a basis for SmartWebs ontology-based search components. Modeling decisions were basically made taking into account the conscientious performance optimization effort, i.e. the performance of the task supported by the Gold Standard should be considered optimal. Additionally, the process of building the Gold Standard ontology has involved different techniques, the On-to-Knowledge methodology [25] which provides solid guidelines for building ontologies, and design patterns [8] which provide means of encoding the ontology conceptualization in a formal ontology language by formal patterns.

Basically, the evaluation is composed of two sub-steps: *gold standard answers* generation, and comparison between *gold standard answers* and *task output answers*. The *evaluation notification* consists of a list of queries associated with their expected time-performance and the real one; and an analysis of the differences and similarities between the *gold standard answer* model and the real (un-evolved) one. The comparison is performed at two different levels, lexical and conceptual (taxonomy and semantic relations), based on algorithms proposed by [14, 5]. The evaluation is performed iteratively until the engineer has evolved the ontology to support the desired performance for the given task. At each step of the evaluation process, a version of the evaluated ontology and its *evaluation notification* is versioned (see Figure 4). Within the *evaluation notification* engineers are able to detect and resolve performance conflicts in the ontology files. Furthermore, by maintaining the version history of the ontology files, engineers are able to keep track of the changes done during the evaluation process.

The evaluation results provided by the evaluation of the ontology at its structural and functional levels can be attached to the ontology's version as documentation, and thus enhancing its pragmatics quality. Additional documentation attached to the ontology languages' construct "rdf:comments" are quantified by

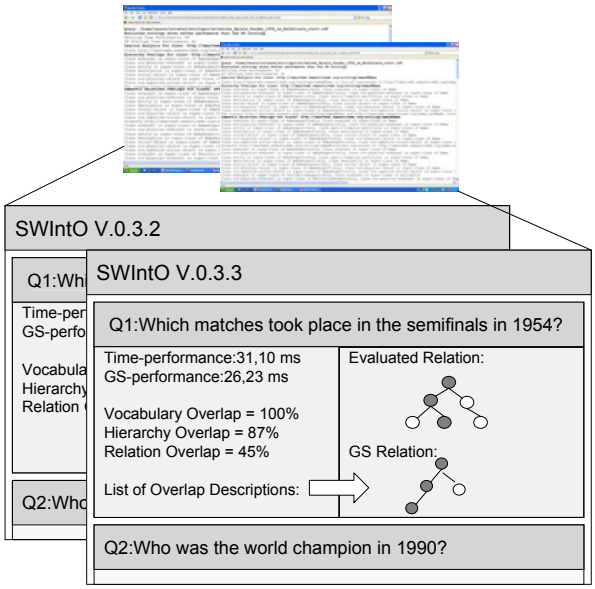


Fig. 4. Example of the functional evaluation outputs

the *annotation analysis*. The pragmatics analysis is helpful to provide better visibility for users in the process of ontology versioning as part of its evolution.

5 Conclusion

Ontology evaluation is an important issue that must be addressed during the lifecycle of an ontology because it assures that the ontology reflects the desired criteria during the different ontology lifecycle processes. One of those processes is ontology evolution, and we demonstrated that evaluation is an important issue for ontology evolution since quality measurement can guide a controlled evolution. The question, however, "What is a good ontology?" (or correspondingly "What is a good ontology evolution?") cannot be answered to perfect satisfaction; but we think, when criteria of quality with respect to the purpose of the ontology (or the purpose of the ontology changes) are considered, a semiotic based approach serves both the evaluation and controlled evolution: we proposed a controlled evolution approach by evaluation methods (i.e., quality measurement and improvement).

In order to capture the different nature of evaluation and evolution criteria, we implemented a semiotic based evaluation tool (S-OntoEval). The tool realizes a complete evaluation combining several evaluation metrics categorized into

the semiotic levels. The main goal is to show its controlled evolution by quality measurement and improvement considering different evaluation methods and theories but based on a semiotic evaluation framework. In the evaluation example, we chose four metrics among others (*depth*, *consistency checking*, *task-based*, and *annotation analysis*) which we believe to be representative in any ontology evaluation process.

Although a lot of the research has been successfully performed in ontology evaluation field, there are still open issues that should be resolved. The fact that structural, functional, and pragmatics measures have different nature may lead to incomparable measurement results, and thus makes a challenge to combine different evaluation scores reasonably. Hence, the evolution of the ontology can only be controlled at a specific level of granularity and quality. Therefore, their semi-automatic application and integration should come into the fore in future research.

Acknowledgment

This research is supported by the German Federal Ministry of Economics and Technology under the grant number 01MQ07016 (THESEUS). The responsibility for this publication lies with the authors.

References

1. M. Ciaramita A. Gangemi, C. Catenacci and J. Lehmann. Qood grid: A metaontology-based framework for ontology evaluation and selection. 2006.
2. Stephan Bloehdorn, Peter Haase, York Sure, and Johanna Völker. Ontology evolution. In John Davies, Rudi Studer, and Paul Warren, editors, *Semantic Web Technologies - Trends and Research in Ontology-based Systems*, chapter 4, pages 51–70. JUN 2006.
3. P. Buitelaar and P. Cimiano, editors. *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, volume 167 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, Amsterdam, 2008.
4. F. de Saussure. *Cours de Linguistique Generale (1916)*. Phylosophical Library, 1965. translated by W. Baskin as Course in General Linguistics.
5. K. Dellschaft and S. Staab. On how to perform a gold standard based evaluation of ontology learning. *In In: Proc. of ISWC-2006 International Semantic Web Conference*, 2006.
6. R. Dividino. Semiotic-based ontology evaluation tool. Master's thesis, Universität des Saarlandes, 2007.
7. R. Dividino, M. Romanelli, and D. Sonntag. Semiotic-based ontology evaluation tool (s-ontoeval). In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008.
8. Aldo Gangemi. Ontology design patterns for semantic web content. In *International Semantic Web Conference*, pages 262–276, 2005.
9. N. Guarino and C. Welty. Evaluating ontological decisions with ontoclean. *Commun. ACM*, 45(2):61–65, 2002.

10. J. Hartmann, Y. Sure, A. Giboin, D. Maynard, M. Suarez-Figueroa, and R. Cuel. Methods for ontology evaluation. KWeb Deliverable D1.2.3, University of Karlsruhe, DEC 2004.
11. N. Huang and S. Diao. *Structure-Based Ontology Evaluation*, pages 132–137. IEEE Computer Society, Washington, DC, USA, 2006.
12. J. Kleb. Ontology evolution. In Raphael Volz, editor, *Semantics At Work*. 2007.
13. A. Maedche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies on the semantic web. *The VLDB Journal*, 12(4):286–302, 2003.
14. A. Maedche and S. Staab. *Measuring similarity between ontologies*. Springer, Madrid, Spain, 2002.
15. C. Masolo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider. Wonderweb deliverable d18. In *The WonderWeb Library of Foundational Ontologies*, 2004.
16. D. Maynard, W. Peters, M. dAquin, and M. Sabou. Change management for metadata evolution. In *International Workshop on Ontology Dynamics (IWOD-07)*, pages 27–40, 2007.
17. G. Nagypal and W. Müller. Ontology design. In Raphael Volz, editor, *Semantics At Work*. 2007.
18. I. Niles and A. Pease. Towards a standard upper ontology, 2001.
19. N. Noy. Evaluation by ontology consumers. pages 1541–1672. IEEE Intelligent Systems, 2004.
20. D. Oberle, A. Ankolekar, P. Hitzler, P. Cimiano, C. Schmidt, M. Weiten, B. Loos, R. Porzel, H. Zorn, V. Micelli, M. Sintek, M. Kiesel, B. Mougouie, S. Vembu, S. Baumann, M. Romanelli, P. Buitelaar, R. Engel, D. Sonntag, N. Reithinger, F. Burkhardt, and J. Zhou. DOLCE ergo SUMO: On Foundational and Domain Models in the Smartweb Integrated Ontology (SWIntO). *Web Semant.*, 5(3):156–174, 2007.
21. R. Porzel and R. Malaka. A task-based approach for ontology evaluation. *ECAI Workshop on Ontology Learning and Population*, 2004.
22. D. Sonntag, R. Engel, G. Herzog, A. Pfalzgraf, N. Pfleger, M. Romanelli, and N. Reithinger. SmartWeb Handheld - Multimodal Interaction with Ontological Knowledge Bases and Semantic Web Services. In *Artifical Intelligence for Human Computing*, pages 272–295, 2007.
23. L. Stojanovic, N. Stojanovic, and S. Handschuh. Evolution of the metadata in the ontology-based knowledge management systems. In *Proceedings of the 1st German Workshop on Experience Management*, pages 65–77. GI, 2002.
24. Y. Sure. Why evaluate ontology technologies? because it works! *IEEE Intelligent Systems*, pages 1541–1672, 2004.
25. Y. Sure and R. Studer. On-to-knowledge methodology - final version. Technical Report Deliverable 18, Institute AIFB, University of Karlsruhe, 2002.
26. D. Thanh Tran, P. Haase, H. Lewen, O. Munoz-Garcia, A. Gomez-Perez, and R. Studer. Lifecycle-support in architectures for ontology-based information systems. In *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, pages 508–522, Busan, Korea, 2007.
27. D. Vrandecic and Y. Sure. How to design better ontology metrics. In *ESWC*, pages 311–325, 2007.
28. W. Wolfgang. SmartWeb: Ein multimodales Dialogsystem für das semantische Web. In Bernd Reuse, editor, *40 Jahre Informatikforschung in Deutschland*. Springer, Heidelberg, Berlin, 2007.

This article was processed using the LaTeX macro package with LLNCS style

An Ontology Creation Methodology: A Phased Approach

Jon Atle Gulla[1], Vijayan Sugumaran[2]

[1] Computer Science Department
Norwegian University of Science and Technology
NO-7491 Trondheim, Norway
jag@idi.ntnu.no
[2] Department of Decision and Information Sciences
Oakland University
Rochester, MI 48309
sugumara@oakland.edu

Abstract. Existing ontology construction workbenches have severe limitations in creating ontologies from representative text collections. This paper presents a new workbench-supported ontology creation methodology that is specially designed for non-ontology experts with little knowledge of semantic markup languages. The workbench generates OWL models and is implemented as part of a search project for the entertainment domain. New techniques are gradually being added to the workbench, and early testing indicates that the ontology creation approach can be used by domain experts without much guidance from ontology experts.

Keywords: Semantic Web, Ontologies, Ontology Creation, Text Mining.

1 Introduction

Ontology creation – also referred to as ontology learning – is the process of automatically or semi-automatically constructing ontologies from representative domain text. The assumption is that textual domain descriptions reflect the terminology that should go into an ontology, and that appropriate linguistic and statistical methods should be able to extract the appropriate concept candidates and their relationships from these texts. Numerous approaches for ontology creation have been proposed in recent years [9, 10, 11, 12, 13], and they tend to allow ontologies to be generated relatively faster and with less costs than traditional modeling environments. However, the approaches still rest on manual expertise, and no satisfactory automatic solutions have been found. Ontology creation toolsets generate candidate ontology elements, but human labor is needed to verify the suggestions and complete the ontologies.

In industrial ontology engineering projects there are still very few ontology creation toolsets in use. Most ontologies are constructed using traditional modeling approaches with teams of ontology experts and domain experts working together [5,

7]. This may be explained by the strategic issues involved in ontology engineering, but it also seems that current ontology creation tools do not have the reliability or credibility needed in large-scale ontology engineering projects. A challenge is also the incremental updating of ontologies, which has so far not been dealt with properly by current ontology creation toolsets. In industrial contexts, the ontology evolution process is supported by dedicated organizational structures and people that manually identify the necessary changes and update the ontology at regular intervals.

Unfortunately, both current ontology creation workbenches and ontology modeling approaches require the participation of ontology experts. These are modelers that are familiar with the rather cryptic syntax of modeling languages like OWL and RDF. The expert modelers specify ontology structures after interviewing domain experts or transform candidate lists from workbenches into legal ontology statements. The costs of developing ontologies and the competence needed to model them have effectively prevented smaller organizations or communities from developing their own ontologies.

This paper presents a new type of ontology creation methodology and a corresponding workbench that frees the user from having any in-depth knowledge of ontology modeling languages. The user is exploring ontological structures as simple tree structures, where all formal details are hidden, and he or she can interactively play around with different techniques and different ways of combining techniques until the user is satisfied with the end result. The tool is constructing formal OWL ontology models automatically from the results favored by thye user. The ontology workbench is now under development in a European semantic search project and will be evaluated against a set of movie documents on the web.

2 Approaches to Ontology Creation

With ontology creation we usually refer to the semi-automatic or automatic construction of ontology structures from domain text. The discipline is still rather immature with a plethora of approaches, algorithms, and constraints. There are many components that can produce complete OWL ontology models today, though few will argue at this stage that fully automatic approaches to ontology engineering are satisfactory. The quality of many algorithms is however at such a level that they can significantly speed up the construction and maintenance of ontologies.

In general, ontology creation consists of two main phases:

- **Linguistic preprocessing**. These are techniques for standardizing and normalizing the document sources into a unified format suitable for ontology extraction. There is no extraction in this phase, but the text is transformed and/or expanded with linguistic knowledge that address orthographical, morphological, or syntactic issues. The techniques here are domain-dependent and sometimes even data source dependent. They are well understood, and there are many freely available libraries or components for this phase.

- **Ontology Extraction**. These techniques use syntactic structures, statistical tests and/or external knowledge bases to extract ontology elements at all levels of the ontology creation layer cake. There are some well-understood standard algorithms like association rules and clustering, though the research field is immature and future research will most likely improve these techniques substantially.

The most successful part of ontology creation has so far been the extraction of prominent terms. There have been a number of recent proposals for extracting taxonomic and non-taxonomic relationships between terms, and many of them show promising results. Combinations of techniques produce substantially better results than individual techniques, which indicate that different techniques tend to capture different ontological aspects. Most approaches today are based on the assumption that prominent terms can be transferred into concepts at some stage of the process. Because there are few techniques that can deal with intensional or logical aspects of concepts, the approach has been to define concepts from their instances or linguistic realizations (terms).

Over the last ten years a number of ontology engineering methodologies have been proposed. They provide guidelines for collaborative ontology construction and maintenance, and they provide a structure for the organization of ontology development projects. Fernandez-Lopez et al. [27] tend to view ontology engineering as a specialized form of software engineering, making use of the same types of phases and the same type of modeling approaches.

Methodologies such as Methontology [30], UPON [29], Diligent [31], and Stanford's Ontology Development 101 [28], all split up the process into phases that guide you through the development work. Whereas Methontology can be very high level with project-related organizational issues, others provide very specific guidelines for the structures that should go into the ontology. As an example, Stanford's rather simple engineering method includes the following steps: a) determine the domain and scope of the ontology, b) consider reusing existing ontologies, c) enumerate important terms in the ontology, d) define classes and a class hierarchy, e) define the properties of the classes, f) define the facets of the slots, and g) create instances.

The methodologies are in principle independent of any tool support. Consequently, they base their approach on many traditional modeling approaches from database design and information systems analysis. The task of putting ontologies together is assumed to be supported by ontology editors that perform standard checking on syntactic and logical consistency, but the user needs to come up with all the concepts and all the structures that should go into the ontologies.

Currently, there is no comprehensive ontology engineering methodology that reflects the state-of-the-art on automatic ontology creation. Even though these tools may have a severe effect on the construction of new ontologies and the updating of existing ones, they are mostly ignored from a methodological point of view. They may acknowledge that the tools are of value to particular tasks or stages, but the tools should not affect the overall structure of the methodologies. On the other hand, there are now tools in the research community that may not only help the users in carrying out the individual steps in the Stanford methodology mentioned above, but also imply

a different approach to the whole modeling process. Instead of addressing each type of ontology element in isolation, as is advised in Stanford's Ontology Development 101 methodology, tools may extract all types of elements at the same time and encourage the users to explore and classify these elements while other parts of the ontology are identified. Other tools simply automate the entire steps of the modeling process or change the order of the steps to enable a more precise extraction of ontology elements.

3 Tool Support in Ontology Engineering

Our workbench borrows extraction techniques that are found in several existing ontology workbenches [4]. Text2Onto is a comprehensive ontology workbench that makes use of both Lucene and GATE [8] to produce ranked candidate lists that are rather similar to what we do with our workbench [3]. Text2Onto also has some additional features for ontology maintenance and incremental updating of existing ontologies. The JATKE workbench is a plug-in to the Protégé ontology editor and helps users develop ontologies for the Protégé editor [6]. OntoLT is another Protégé plug-in that transforms linguistically annotated entities into concepts and individuals in ontologies [2]. The OntoLearn workbench from Navigli and Velardi [35] concentrates on word sense disambiguation and makes use of the WordNet lexicon.

All the workbenches above suffer from the fact that they consider the ontology creation process a chain of static analysis components. The user does not have the flexibility to adapt the analysis to his or her preferences, the nature of the document collection, or aspects of the domain itself. Furthermore, it is assumed that the user is familiar with ontology editors and can afterwards refine the generated results manually. Our workbench provides more methodological flexibility to the user and allows him or her to finish the entire ontology without the use of complex ontology editors.

For advanced ontology structures like axiom schemata and rules, there is little support in current ontology creation workbenches. As long as the constructed ontology is intended for search, however, this may not be a serious problem. Rules and constraints are used neither in improving search relevance nor in helping users navigate in the search space. If future search applications should include true questions-answering facilities, though, it may be needed to derive rules that allow the application to reason its own knowledge.

4 Towards a Phased Methodology

Most ontologies today are developed manually in labor-intensive projects to satisfy a range of functional goals (data interchange, unification, representation, communication, etc.). Ontology development, however, is difficult even for experienced ontology experts. The developer uses personal knowledge as well as knowledge acquired from accessible sources to identify key terms that are coded into classes or subclasses, and to establish relationships among them. People of different

background usually need to collaborate to assess all the relevant terminology of a domain. It is difficult to assess an ontology's truthfulness, generality, or whether it contains all the information for a given domain. Often, the only relevant criterion is whether an ontology fulfills the functions for which it was created.

In semi-automatic ontology development, specially designed tools are used for some tasks, such as editing ontologies, gathering significant terms, or identifying appropriate relationships among them. Fully automated development of ontologies is difficult to attain because solving non-objective design criteria may require designer intervention. Examples include determining whether a concept is a class or a data type or determining the covering and disjointness constraints of a set of concepts. Ontology development is, thus, labor-intensive and expensive because of the broad range of skills and knowledge required [10]. Moreover, the scale and complexity of ontologies are growing quickly with the rapid development of the Semantic Web [24].

Warren [23] points out that the main obstacle for successful implementation of ontology-based knowledge management is the effort needed for ontology and metadata creation. He argues that developing tools for creating ontologies in a semi-automatic manner is a key research issue. In this research, we focus on developing a semi-automatic methodology for creating ontologies from various text documents.

Our proposed methodology uses a phased approach, as shown in Figure 1. It consists of the following three phases: a) collection phase, b) analysis phase, and c) creation phase.

The *collection phase* consists of two major tasks: 1) crawling, and 2) linguistic preprocessing. At the end of this phase, a database that contains the document and term statistics is created, which becomes the input to the analysis phase. The crawling task entails downloading HTML pages and transforming them into XML documents. A number of crawlers are used to accomplish this task. This task may require human intervention to set up the crawling criteria or decision rules to tune the crawlers. The human intervention aspect of this task is indicated using a stick figure, as shown in Figure 1. A unified document ontology is used to carryout focused crawling. This is part of the workbench configuration. Important document structures can be specified to improve the efficiency of crawling.

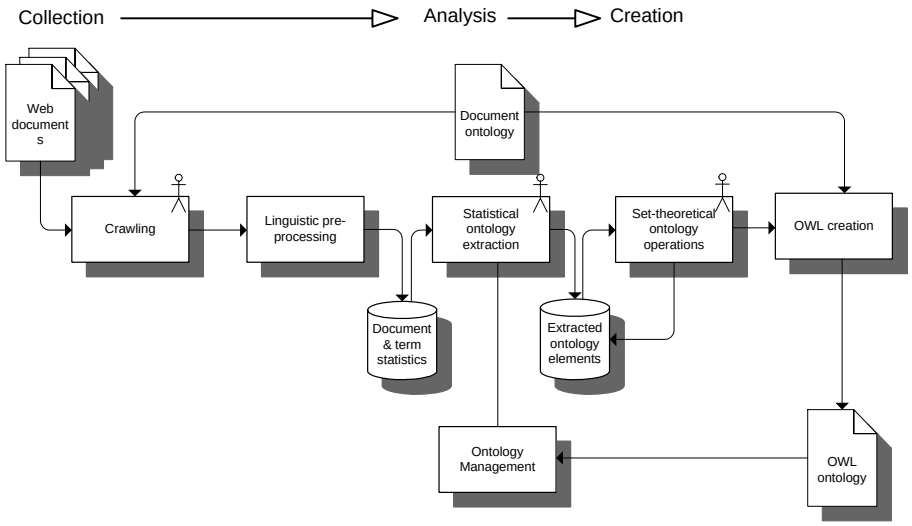


Figure 1. Ontology Creation Methodology

The linguistic pre-processing task uses the XML documents and transforms them using a number of operations such as tokenization and transliteration, named entity recognition, POS tagging, lemmatization or stemming, noun phrase recognition, stop word removal, etc. The processed documents are used for creating several indices.

The *analysis phase* consists of one major task, namely, statistical ontology extraction. This task may also require some human intervention. It involves identifying candidate concepts, instances and different types of relationships. A number of ontology extraction techniques are used to accomplish this task. Some of the extraction techniques are: instance/concept extraction with frequency based scores, instance extraction with structure, relationship extraction with similarity measures, relationship extraction using association rules, and relationship extraction using clustering. The output of this phase is another database that contains the extracted ontology elements. This serves as the input to the third phase, which deals with ontology creation.

The *creation phase* consists of the following two main tasks: a) set theoretical ontology operations, and b) OWL creation. The ontology operations that are part of our methodology are based on set theory. Currently we support the following operations: union, intersection, and difference. This task may also require user interaction. The OWL creation task involves generating the final ontology using OWL syntax. This task also uses the document ontology in creating the OWL ontology. Specifically, this operation involves converting instances to OWL individuals, concepts into classes and relationships into data and object properties.

The proposed ontology creation methodology is iterative and facilitates the evolution of ontologies over time. In other words, the OWL ontology that gets generated can be expanded with additional concepts and relationships, as new information becomes available. Typically, ontology creation requires two groups of people, namely, domain experts and knowledge experts to work together. Domain

experts have knowledge about the domain for which the ontology is being built, however, they may not know much about ontology languages, their syntax, etc. Similarly, ontology experts are well versed in the technical aspects of ontology creation, but lack the domain knowledge. Domain experts generally help in improving the quality of ontologies. Thus, if we can provide a toolset or a workbench that hides the complexity of ontology generation, domain experts can create high quality ontologies relatively easily. This compromise can greatly increase the number of ontologies that could be made available on the Web.

4 Workbench Components

Most ontology creation workbenches combine a linguistic preprocessing stage with various statistical tests on word frequencies. The linguistic techniques transform the actual tokens found in documents to standardized lexical forms that are potential concepts or instances of concepts. For example, the sentence "*In rural Texas, welder and hunter Llewelyn Moss discovers the remains of several drug runners*"[1], we may use a parts-of-speech tagger and a lemmatizer to replace the word inflections with their base forms. If we also remove stop words, i.e. words that are irrelevant to the statistical analysis later, we end up with the following set of phrases: {*rural texas, welder, hunter, llewelyn moss, discover, remain, drug dealer*}. Since we in ontology creation tend to use noun phrases only, we remove adjectives and verbs and keep the set {*texas, welder, hunter, llewelyn moss, remain, drug dealer*} for the subsequent statistical analysis of terms.

Standard ontology workbenches will now combine all sets of noun phrases extracted from all sentences of all documents available and run a number of pre-specified ontology creation algorithms. Each technique produces lists of ranked terms or term relationships that have to be carefully recreated as ontology structures.

Our workbench, on the other hand, has a more interactive approach to the statistical extraction of ontology structures. It consists of three main parts that are used independently of each other, and the user may use all of them simultaneously when new ontologies are created. Since each component also allows the user to manually influence the results, the created ontology will reflect both the quality of the extraction techniques and the contributions from the users. This supports an iterative collaborative ontology creation process, in which one or more users actively explore the underlying document text and construct ontologies without being bothered by the ontology representations themselves. In the following subsections, we discuss the workbench architecture and the components in some more detail.

Collection

At the *Collection* stage, there are a number of crawlers that download HTML pages and transform them into XML documents. A unified document ontology, which is part of the workbench configuration, specifies all important document structures for

[1] From the imdb.com description of the movie "No country for old men".

all data sources and defines a unified ontology format for all of them. This format will form a part of the ontology to be generated later.

The XML documents from the crawler are fed into a linguistic pipeline that carried out the following transformations:

Tokenization and *transliteration*: Word boundaries are identified, and standardized spelling is introduced.

Named entity recognition: Locations, persons, and organizations are identified and tagged for later indexing.

Parts-of-speech tagging: Parts-of-speech tags are associated with every word in the text.

Lemmatization or *stemming*: German and English dictionaries are used to lemmatize the text. If the word is not found in the dictionaries, stemming is used instead.

Noun phrase recognition: We recognize and tag noun phrases that consist of one of more adjectives followed by consecutive nouns.

Stop word removal: Words not relevant for ontology creation, like prepositions, conjunctions and determiners, are deleted.

The linguistically pre-processed document text is stored in Lucene indices according to the structures of the document ontology. For the movie domain, this means that there are specific indices for elements like movie titles, actors, and directors, but also noun phrase indices for plots and user reviews.

Analysis

At the *analysis* stage there is a battery of ontology creation techniques that use the indices to extract candidate concepts, candidate instances, and various types of relationships from the indexed text.

Each technique is specified with a *domain* and a *range* before it is run on the document index. While the domain determines which subset of the index should be used, the range is the type of ontology structures to be extracted. For example, if we extract the most prominent actors playing in drama movies, the domain is the drama movies and the range is actors.

Currently we support the following ontology extraction techniques:

Instance/concept extraction with frequency-based scores: The technique lists prominent terms and rank them according to several ranking schemes, among them the tf.idf scores. Terms are individual words, identified entities, or tagged noun phrases.

Instance extraction with structure-based techniques: Instances given by the structure of the documents are extracted directly using the corresponding XML tags.

Relationship extraction with similarity measures: Relationships between instances are calculated and ranked based on the cosine similarity between their vector representations.

Relationship extraction with association rules. Association rules are based on co-occurrence data and specify with sets of terms tend to imply the presence of another term. Terms involved in an association rule can be assumed to be semantically related.

Relationship extraction with clustering: Suffix tree clustering is used to group terms together in clusters based on common suffixes. The assumption is that there is a relationship between all terms in a particular cluster.

Each technique can be run numerous times with different domains and ranges. They can be used in any order and combined in any possible way to produce the final set of elements to be transformed into ontology structures. Each technique also has a configuration file that tailors the technique to the particular needs of the user.

The extracted concepts, instances and relationships – or a selection of them – are stored in a separate index for later ontology generation.

Creation

All result sets from the analysis phase may be manually modified and stored for later OWL generation. The generation itself is based on a combination of stored result sets, in which the following set operations are used to select the elements to include in the final ontology:

Union: a term or term relationship kept if it appears in at least one of the selected result sets.

Intersection: a term or term relationships is kept if it appears in all selected result sets.

Difference: a term or relationship is kept if it appears in the first result set and not in the other. None of entries in the second result set are kept.

The OWL creation component turns instances into OWL individuals, concepts into OWL classes, and relationships into OWL data and object properties. The exact nature of the mapping to OWL is given by the document ontology as well as the ontological definition of each extraction technique. Some structural simplifications are made to make sure that we avoid meta modeling issues [14] and respect the constraints of OWL DL.

The prototype architecture contains the necessary components that implement the various phases illustrated in Figure 1.

5 Using the Workbench-Supported Methodology

The following scenario illustrates the use of the workbench to construct ontology structures for the movie domain.

Initially, the workbench is building meta data structures for each movie based on the semi-structured text of the documents. This meta data include movie titles, release years, directors, actors, soundtrack, roles, etc.

The next step is to extract concepts from the unstructured parts and relate these to other concepts of the ontology. For example, we can extract all prominent single-word terms that describe the movie *About a man*. As seen from Figure 2, we specify *About a man* as the domain for this operation, and the result is a list of single-word noun phrases ranked by their tf.idf scores. Terms ranked high on this list appear frequently in the documents for About a man, and rather less frequently in other documents in the collection.

As seen from the Figure, the concepts *life*, *boy*, *story*, *friend* and *father* are on top of the list and should describe aspects of the movie's plot. Proper names, like *John* a little bit further down the list, may be recognized from our named entity lists and subsequently be encoded as instances in the ontology. All recognized concepts and instances will in OWL be related to About a boy by means of an OWL object property. If the user identifies two or more terms to refer to the same underlying concept or instance, he may group them together before the OWL statements are generated.

We use association rules to extract relationships between extracted concepts. The key idea is to assume that two terms are related if they tend to co-occur in the data set with a certain regularity and predictability. For the concept *good friend* we extract the following related terms from the movie collection: life, friend, love → good friend

This implies that the presence of *life*, *friend* and *love* tend to imply the presence of *good friend*. The algorithm used for the generation of association rules is the *a priori* algorithm introduced by Agrawal and Srikant [1]. Provided that these relationships are confirmed manually or by means of other techniques, they are mapped onto OWL structures with object properties linking the concepts together.

An important part of ontology creation is to identify important relationships between already approved domain concepts. The user may apply three different techniques for identifying movies related to Rain Man, and he may afterwards decide how the results should be combined.

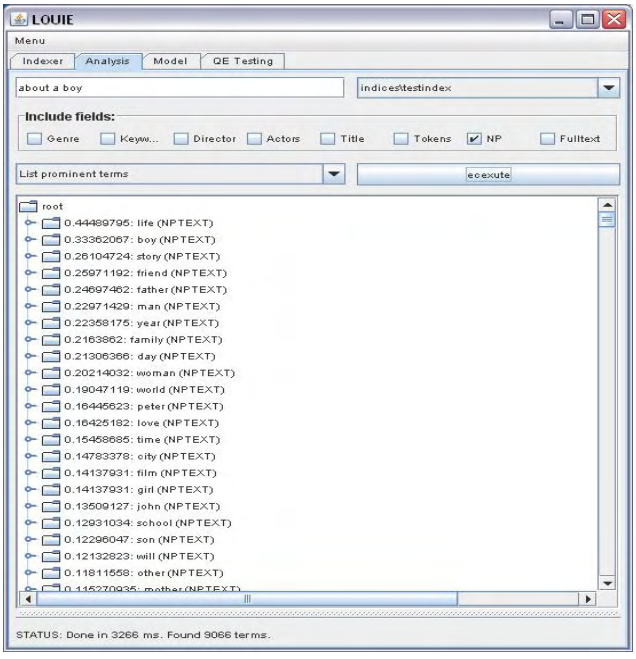


Figure 2. Terms extracted for the movie *About a boy.*

The *vector similarity* method calculates the cosine similarity between About a boy and any other movie in the collection:

$$cos(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}},$$

where $\vec{x}$ and $\vec{y}$ are term vectors of Rain Man and some other movie and x_i is the weight of the i^{th} term of vector $\vec{x}$. If the cosine similarity is above a certain threshold, we conclude that there is a relationship between the movies. This means that the movies have been described with similar terms and expressions. The set of all cosine similarities above this threshold for About a boy is the system's suggested list of related movies. More details about this method is found in [24].

The Association rule method identifies movies that tend to be discussed together in movie documents. Applied to About a boy, the method identifies ten movies that occur frequently with About a boy in the data set.

A third method, clustering, can also be used to generate potential relationships between movies. The approach implemented in this workbench, Suffix Tree Clustering, extracts suffixes from the text and uses these to group movies together. As opposed to traditional top-down or bottom-up clustering, STC is linear and well suited for large data sets. Details of the algorithm can be found in [25].

Figure 3 lists the movies that may be related to About a boy and have been suggested by the three methods. The user now has a chance to select the most suitable movies or use set operations to select the best results from the techniques that have been applied. For example, if the user decides to trust the association rules method the most, he may generate a new result set that contains the titles from the association rules method that have been confirmed by at least one of the other two methods:

$$R_{AR} \cap R_{STC} \cup R_{AR} \cap R_{VS}$$

In this case the selected movies are *Reign of fire, Catwoman, Hearts in Atlantis* and *The Interpreter*. In OWL these relationships between movies are represented as object properties of About a boy.

Technique	**Related to *About a boy***
Vector Similarity	– *Spider man 3* – *Switch* – *Reign of fire* – *My summer of love* – *Sky high* – *The stepdaughter* – *The good shepherd*
Association rules	– *21 grams* – *Artificial Intelligence* – *Barbershop* – *Catwoman*

	– *Get over it* – *hearts in Atlantis* – *The interpreter* – *The notebook* – *Reign of fire* – *Riding in cars with boys*
Clustering (STC)	– *The 40 year old virgin* – *The interpreter* – *Catwoman* – *Hearts in Atlantis* – *Spy kids* – *Daddy day care* – *Maid in Manhattan* – *The island* – *Miss Congeniality*

Figure 3. Three techniques extracting movies related to About a boy

5 Conclusions

The ontology engineering methodology discussed in this paper is intimately integrated with the workbench under development. Additional methodological flexibility is enabled because the workbench allows a more interactive exploration of domain documents. Temporary or preliminary results are kept and may be inspected by the user when he is gradually working out the elements to be included in the resulting ontology. The workbench guides the entire learning process from deciding on the documents to analyze all the way to complete OWL DL models. As long as complex restrictions or rules are not needed, the user may not need to use any additional tool for constructing and maintaining the ontologies.

The quality of ontology workbenches depend on the techniques available to the user. Our workbench has only a limited number of techniques implemented so far, and future versions need to add new ones for extracting hierarchical relationships and relevant concepts not directly mentioned in the text. Currently this information has to be added by the user manually while inspecting and grouping results from the workbench.

This workbench is developed in the context of a semantic search application. This means that the ontologies are geared towards search engines and their strategies for adding semantics to their syntax-based vector space retrieval models. The extraction of concepts and relationships is based on information retrieval techniques to make sure that they are useful as navigational help in the search applications. If the ontology was to be used in other types of applications, other extraction techniques may be more important than the ones implemented so far in our workbench.

Acknowledgments. This research has been partly funded by Deutsche Telekom, and we would like to thank T-Labs for their support and help.

References

1. Agrawal, R., T. Imielinski, and A.N. Swami, Mining Association Rules between Sets of Items in large Databases, in Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. 1993.
2. Buitelaar, P., D. Olejnik, M. Sintek. A Protégé Plug-In for Ontology Extraction from Text Based on Linguistic Analysis. In: Proceedings of the 1st European Semantic Web Symposium (ESWS), Heraklion, Greece, May 2004.
3. Cimiano, P. and J. Völker, Text2Onto, in Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'05), A. Montoyo, R. Muñoz, and E. Métais, Editors. 2005, Springer: Allicante. p. 227-238.
4. Cimiano, P., J. Völker, and R. Studer, Ontologies on Demand? A Description of the State-of-the-Art, Applications, Challenges and Trends for Ontology Learning from Text. Information, Wissenschaft und Praxis, 2006. 57(6-7): p. 315-320.
5. Cristiani, M. and R. Cuel, A Survey on Ontology Creation Methodologies. 2005: Idea Group Publishing.
6. Endres, B. JATKE: A Platform for the Integration of Ontology Learning Approaches. 2005
7. Fernandez, M., A. Goméz-Peréz, and N. Juristo, Methontology: from ontological art towards ontological engineering, in Proceedings of the AAAI'97 Spring Symposium Series on Ontological Engineering. 1997: Stanford. p. 33-40.
8. Gaizauskas, R., et al., GATE User Guide. URL: http://gate.ac.uk/sale/tao/index.html#x1-40001.2. 1996.
9. Gulla, J.A., H.O. Borch, and J.E. Ingvaldsen, Ontology Learning for Search Applications, in Proceedings of the 6th International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2007). 2007, Springer: Vilamoura.
10. Haase, P. and J. Völker, Ontology Learning and Reasoning - Dealing with Uncertainty and Inconsistency, in Proceedings of the International Semantic Web Conference. Workshop 3: Uncertainty Reasoning for the Semantic Web (ISWC-URSW'05), P.C.G. da Costa, et al., Editors. 2005: Galway. p. 45-55.
11. Maedche, A. and S. Staab, Semi-automatic Engineering of Ontologies from Text. In Proceedings of the 12th Internal Conference on Software and Knowledge Engineering, Chicago, 2000.
12. Navigli, R. and P. Velardi, Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites. Computational Linguistics, 2004. 30(2): p. 151-179.
13. Sabou, M., et al., Learning Domain Ontologies for Semantic Web Service Descriptions. Accepted for publication in Journal of Web Semantics, 2008.
14. Motik, B. On the Properties of Metamodeling in OWL. Journal of Logic and Computation 2007 17(4):617-637;
15. Grant, R.M. 1996. "Prospering in dynamically-competitive environments: Organizational capability as knowledge integration". Organizational Science, 7(4):375-387.
16. Nidumolu, S., Mani.R. Subramani and Alan Aldrich, (2001). Situated Learning and theSituated Knowledge Web: Exploring the Ground beneath Knowledge Management, Journal of Management Information Systems, 18(1), pp. 115-151.
17. Rus, M. Lindvall, (2002) "Knowledge Management in Software Engineering", IEEE Software, 19(3), 26-38.
18. Ju, T.L. 2006. Representing Organizational Memory for Computer-Aided Utilization. Journal of Information Science. 32(5), pp. 420 – 433.
19. Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz, R. 2003. Ontologies for Enterprise Knowledge Management. IEEE Intelligent Systems. 18(2), pp. 26 – 33.
20. Fensel, D. 2002. Ontology-Based Knowledge Management. IEEE Computer. 35(11), pp. 56 – 59.

21. Chang, J., Choi, B., Lee, H. 2004. An Organizational Memory for Facilitating Knowledge: An Application to E-Business Architecture. Expert Systems with Applications. 26(2), pp. 203 – 215.
22. Berners-Lee, T., Hendler, J., and Lassila, O. 2001. The Semantic Web, Scientific American, May 2001, pp. 1-19.
23. Warren, P. 2006. Knowledge Management and the Semantic Web: From Scenario to Technology. IEEE Intelligent Systems, January/February 2006, pp. 53 – 59.
24. Gulla, J. A. and T. Brasethvik. 2008. A Hybrid Approach to Ontology Relationship Learning. Accepted at 13th International Conference on Applications of Natural Language to Information Systems (NLDB 2008), London.
25. Zamir, O., Etzioni, O., Madani, O., & Karp, R. 1997. Fast and intuitive clustering of Web documents. In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, pp. 287—290.
26. Agrawal, R. and Srikant, R. 1994. Fast Algorithms for Mining Association Rules, in Proceedings of the 20th International Conference on Very Large Data Bases (VLDS'94). 1994.
27. Fernandez-Lopez, M. and A. Gomez-Perez. 2002. Overview and analysis of Methodologies for Building Ontologies. The Knowledge Engineering Review, Vol. 17, No. 2, pp 129-156, 2002.
28. Noy, N. F., and D. L. McGuinness. 2001. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.
29. De Nicola, A., Navigli, R., Missikoff, M. 2005. Building an eProcurement Ontology with UPON methodology, Proc. of 15th e-Challenges Conference, Lijubiana, Slovenia, October 19-21st, 2005.
30. Gomez-Perez, A., M. Fernandez-Lopez, and A. de Vicente. 1996. Towards a method to Conceptualize Domain Ontologies. In Proceedings of ECAI96 Workshop on Ontological Engineering, pp. 41-51.
31. Pinto, H. S., S. Staab and C. Tempich. 2004. Diligent: Towards a fine-grained methodology for Distributed, Loosely-controlled and Evolving Engineering of Ontologies. In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), August, 2004, Valencia.
32. Barth, S. “KM Horror Stories,” Knowledge Management, Volume 3, Number 10, Oct 2000, pp. 36-40.
33. Davenport, T.H. (1997) “Known Evils: The Common Pitfalls of Knowledge Management,” CIO Magazine, June 15.
34. Lenat, D. B. (1995). "CYC: a large-scale investment in knowledge infrastructure." Communications of the ACM 38(11): 33-38.
35. Navigli, R. and Velardi, P. 2004. Learning domain ontologies from document warehouses and dedi-cated web sites. Computational Linguistics, 50, 2004.

A Theoretical Model to Handle Ontology Debugging & Change Through Argumentation

Martín O. Moguillansky, Nicolás D. Rotstein, and Marcelo A. Falappa

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Department of Computer Science and Engineering (DCIC)
Artificial Intelligence Research and Development Laboratory (LIDIA)
Universidad Nacional del Sur (UNS), Bahía Blanca, ARGENTINA
{mom,ndr,maf}@cs.uns.edu.ar

Abstract. A dynamic argumentation framework based on $\mathcal{ALC}$ description logics is presented by extending notions from argumentation. Since argumentation frameworks reason over graphs that relate arguments through attack, a methodology is proposed to bridge ontology-specific concepts to argumentation notions. In this way, inconsistency in the former will be represented as an attack in the latter. This approach benefits from (argumentation) acceptability semantics to restore consistency to ontologies. Finally, a model of ontology change is presented along with a rational characterization.

1 Introduction

In this article we adapt the abstract version of the dynamic argumentation framework (DAF) originally presented in [18] and [17] to deal with a specific logic for arguments: $\mathcal{ALC}$ description logics (DLs). Some classical argumentation elements [4] are extended to be properly applied on such a specialized DAF. For instance, attack and support in argumentation are here related to identify different levels of inconsistency in ontologies [6]. Therefore, an acceptability semantics [3] applied to the DAF, determines a methodology for ontology debugging.

Afterwards, a model of change for ontologies is presented. Such model prioritizes the new knowledge to be incorporated, in a way that the evolved ontology will fully accept the new information, while ending up consistent at all levels. That is, (ontology) consistency as well as (terminology) coherency will be assured. Rationality of change in ontologies is stated in an abstract manner by the proposal of a set of postulates adapted from [6] and [1]. Afterwards, the model of change proposed in this article is studied under the light of the presented postulates, and an axiomatization is finally determined.

It is important to note that the argumentation machinery here proposed is semantically determined –by effect of the semantic entailment– and prepared to deal with $\mathcal{ALC}$ fragments of knowledge. This means that tableaux technics may be easily reused towards further implementations. Consequently, the actual model could recognize the sources of inconsistency directly, with no need to any translation to a DAF. In this sense, this methodology could be implemented on top of the DL reasoner. A different practical approach will be analyzed in Sect. 5.

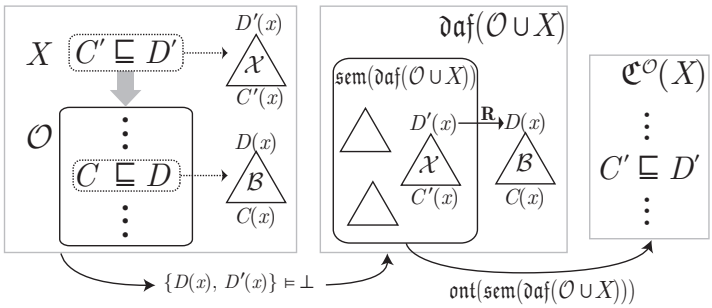


Fig. 1. Schema of the theoretical model of ontology change.

In Fig. 1 a schema of the presented theoretical model of change is shown (arguments are depicted as triangles). Its full understanding will be made clear throughout this paper but, on the other hand, the understanding of this paper will be made clearer by referring to the figure from time to time. In general, we define a DAF interpreting ontology concept descriptions as arguments. The reader should be aware that the actual translation from concepts to arguments has some subtleties that are not depicted in the figure. An $\mathcal{ALC}$ ontology $\mathcal{O}$ will be required to accept in a consistent and coherent manner a second $\mathcal{ALC}$ ontology X by means of the change operation $\mathfrak{C}^{\mathcal{O}}(X)$. A DAF determined by a function "$\mathfrak{daf}$" will return the DAF associated to the union of both ontologies. Afterwards, the argumentation machinery manages to recognize argument defeaters, that is arguments that are contradictory in some way. Consequently, the set of attack relations $\mathbf{R}$ is identified, leading to a graph of arguments. An acceptability semantics "$\mathfrak{sem}$" determines a consistent set of accepted arguments, thus yielding a disconnected graph. Finally, the translation back "$\mathfrak{ont}$" is done obtaining the new evolved consistent-coherent ontology $\mathfrak{C}^{\mathcal{O}}(X)$.

2 Description Logics Brief Overview

The following constitutes a very brief overview of the description logics (DLs) used in this paper, for more detailed information refer to [2]. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty domain $\Delta^{\mathcal{I}}$, and an interpretation function $\cdot^{\mathcal{I}}$ that maps every concept to a subset of $\Delta^{\mathcal{I}}$, every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every individual to an element of $\Delta^{\mathcal{I}}$.

The description language $\mathcal{ALC}$ is formed by concept definitions according to the syntax $C, D ::= A|\bot|\top|\neg C|C \sqcap D|C \sqcup D|\forall R.C|\exists R.C$ where A is an atomic concept, R is an atomic role; and the interpretation function $\cdot^{\mathcal{I}}$ is extended to the universal concept as $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$; the bottom concept as $\bot^{\mathcal{I}} = \emptyset$; the full negation or complement as $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$; the intersection as $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$; the union as $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$; the universal quantification as $(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \forall b.(a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$; and the full existential quantification as $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b.(a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$.

An ontology is a pair $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ represents the TBox, containing the terminologies (or axioms) of the application domain, and $\mathcal{A}$, the ABox, which contains assertions about named individuals in terms of these terminologies. Regarding the TBox $\mathcal{T}$, axioms are sketched as $C \sqsubseteq D$ and $C \equiv D$, therefore, an interpretation $\mathcal{I}$ satisfies them whenever $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ and $C^{\mathcal{I}} = D^{\mathcal{I}}$ respectively. An interpretation $\mathcal{I}$ is a model for the TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies all the axioms in $\mathcal{T}$. Thus, the TBox $\mathcal{T}$ is said to be satisfiable if it admits a model. Besides, in the ABox $\mathcal{A}$, $\mathcal{I}$ satisfies $C(a)$ if $a \in C^{\mathcal{I}}$, and $R(a, b)$ if $(a, b) \in R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is said to be a model of the ABox $\mathcal{A}$ if every assertion of $\mathcal{A}$ is satisfied by $\mathcal{I}$. Hence, the ABox $\mathcal{A}$ is said to be satisfiable if it admits a model. Finally, regarding the entire ontology, an interpretation $\mathcal{I}$ is said to be a model of $\mathcal{O}$ if every statement in $\mathcal{O}$ is satisfied by $\mathcal{I}$, and $\mathcal{O}$ is said to be satisfiable if it admits a model. An ontology contains implicit knowledge that is made explicit through inferences. The notion of semantic entailment is given by $\mathcal{O} \models \alpha$, meaning that every model of the ontology $\mathcal{O}$ is also a model of the statement α. Formally, **(semantic entailment)** $\mathcal{O} \models \alpha$ *iff* $\mathcal{M}(\mathcal{O}) \subseteq \mathcal{M}(\{\alpha\})$. Just for simplicity, we shall abuse notation writing $\mathcal{O} = \mathcal{T} \cup \mathcal{A}$ (*eg.*, $\mathcal{O} = \{C \sqsubseteq D, A(a)\}$) to identify an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ (*eg.*, $\mathcal{O} = \langle \{C \sqsubseteq D\}, \{A(a)\} \rangle$). Throughout this paper, inferences will be obtained following the notion of semantic entailment.

3 $\mathcal{ALC}$-Based Dynamic Argumentation Framework

In the rest of this article we will use symbols $A, A_1, A_2, \ldots$ and $B, B_1, B_2, \ldots$ to denote atomic DL concepts, $C, C_1, C_2, \ldots$ and $D, D_1, D_2, \ldots$ to denote general DL concepts, $R, R_1, R_2, \ldots$ to denote atomic DL roles, x, y to denote free variables, and a, b to denote individual names. The following grammars are necessary to specify the language used to represent $\mathcal{ALC}$-based ontologies into a dynamic argumentation framework (DAF) for description logics. It is important to mention that the DAF is an extension of the abstract argumentation framework proposed by Dung [4]. In this work, the original DAF formalism [17,18] requires to be extended in order to handle knowledge represented by $\mathcal{ALC}$ DL statements.

$$\phi ::= \top | A | \neg A | \forall R.\mathcal{L}_{disj} | \exists R.\mathcal{L}_{conj}$$

$$\mathcal{L}_{conj} ::= \phi | \mathcal{L}_{conj} \sqcap \mathcal{L}_{conj} \qquad \mathcal{L}_{disj} ::= \phi | \mathcal{L}_{disj} \sqcup \mathcal{L}_{disj}$$

$$\mathcal{L}_{\mathcal{T}} ::= \mathcal{L}_{conj} \sqsubseteq \mathcal{L}_{disj} \qquad \mathcal{L}_{\mathcal{A}} ::= A(\mathcal{L}_{var}) | \neg A(\mathcal{L}_{var}) | R(\mathcal{L}_{var}, \mathcal{L}_{var})$$

$$\mathcal{L}_{\mathtt{pr}} ::= \phi(\mathcal{L}_{var}) \qquad \mathcal{L}_{\mathtt{cl}} ::= \mathcal{L}_{disj}(\mathcal{L}_{var}) | R(\mathcal{L}_{var}, \mathcal{L}_{var})$$

$$\mathcal{L}_{var} ::= a | b | x | y \qquad \mathbb{A}\mathtt{rgs} ::= 2^{\mathcal{L}_{\mathtt{pr}}} \times \mathcal{L}_{\mathtt{cl}}$$

An argument is interpreted as an atomic (indivisible) piece of knowledge. To the argumentation machinery, an argument is a primitive element of reasoning supporting a claim from its set of premises. Usually, argumentation frameworks consider ground arguments, that is, a claim is directly inferred if the set of premises are conformed. In our framework, we will consider two different kinds of arguments: ground and schematic. In this sense, a set of premises might consider free variables, meaning that the claim, and therefore the inference, will depend on them. When an argument has its premises supported, its variables may be instantiated as a result of that. These notions are carefully detailed throughout this section. Next we formalize the notions of argument and DAF for DLs.

Definition 1 (Argument). *An argument* $\mathcal{B} \in \mathbb{A}\mathtt{rgs}$ *is a pair* $\langle P, c\rangle$*, where* $P \in 2^{\mathcal{L}_{\mathfrak{pr}}}$ *is the finite set of premises from* $\mathcal{L}_{\mathfrak{pr}}$*, and* $c \in \mathcal{L}_{\mathfrak{cl}}$*, the claim. An argument* $\mathcal{B}$ *guarantees* $P \cup \{c\} \not\models \bot$ **(consistency)**. *Both premises and claims are represented as finite formulae from their respective language.*

Definition 2 (Dynamic Argumentation Framework for DLs). *Let* $T \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$ *be a Dynamic Argumentation Framework for DLs* (DAF)*, specified by a pair* $\langle \mathbf{U}, \mathbf{A}\rangle$*, where* $\mathbf{U} \subseteq \mathbb{A}\mathtt{rgs}$ *is the universal set of arguments, and* $\mathbf{A} \subseteq \mathbf{U}$ *is the framework's active set containing the unique set of arguments considered by the argumentation reasoning process.*

As usual in argumentation, pairs of conflictive arguments may appear. Such pairs will be contained in an attack relation set $\mathbf{R}$, dynamically recognized from the current DAF specification. This notion will be made clear later. Besides, inactive arguments –ignored by the reasoning process– might be identified by means of a set $\mathbf{I} = \mathbf{U} \setminus \mathbf{A}$. Some arguments may count with no premises to be satisfied. Such arguments, referred as evidence, will be individually considered a self-conclusive piece of knowledge, and will be enclosed by a set $\mathbf{E} \subseteq \mathbf{A}$. However, there could be inactive arguments with an empty set of premises which will be recognized as non-evidential facts, enclosed in a set $\mathbf{F} \subseteq \mathbf{I}$.

Definition 3 (Evidence & Non-Evidential Fact). *Given a* DAF $\langle \mathbf{U}, \mathbf{A}\rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$*. An argument* $\mathcal{B} \in \mathbf{U}$ *such that* $\mathcal{B} = \langle\{\}, c\rangle$ *and* $c \in \mathcal{L}_{\mathcal{A}}$*, is referred either as:* ***Evidence*** *iff* $\mathcal{B} \in \mathbf{A}$*, or* ***Non-Evidential Fact*** *iff* $\mathcal{B} \notin \mathbf{A}$*.*

Given an argument $\mathcal{B} \in \mathbb{A}\mathtt{rgs}$, its claim and set of premises are identified by the functions $\mathfrak{cl} : \mathbb{A}\mathtt{rgs} \rightarrowtail \mathcal{L}_{\mathfrak{cl}}$, and $\mathfrak{pr} : \mathbb{A}\mathtt{rgs} \rightarrowtail 2^{\mathcal{L}_{\mathfrak{pr}}}$, respectively. For instance, given $\mathcal{B} = \langle\{p_1, p_2\}, c\rangle$, its premises are $\mathfrak{pr}(\mathcal{B}) = \{p_1, p_2\}$, and its claim, $\mathfrak{cl}(\mathcal{B}) = c$.

In order to obtain a DAF from an $\mathcal{ALC}$ ontology $\mathcal{O}$, it is needed to translate each axiom in $\mathcal{O}$ to *negation normal form*, so that negation appears only in front of atomic concepts. Afterwards, each axiom should turn to *disjunctive normal form* for the left-hand-side (*lhs*) part of the description, and to *conjunctive normal form* for its right-hand-side (*rhs*), conforming axioms $lhs \sqsubseteq rhs$ or $lhs \equiv rhs$, where $lhs ::= \bot | \mathcal{L}_{conj} \sqcup \ldots \sqcup \mathcal{L}_{conj}$ and $rhs ::= \bot | \mathcal{L}_{disj} \sqcap \ldots \sqcap \mathcal{L}_{disj}$, referred as *pre-argumental normal form* (pANF). An ontology in pANF could trigger multiple arguments from each axiom, as states the following intuition: each *lhs* disjunction (in $\mathcal{L}_{conj}$) is interpreted as a set of premises $\mathcal{L}_{\mathfrak{pr}}$–one for each conjunction– and each *rhs* conjunction (in $\mathcal{L}_{disj}$), as a claim in $\mathcal{L}_{\mathfrak{cl}}$ (*c.f.* Ex. 1). Concept equivalences as $C_1 \equiv C_2$, are assumed as pairs $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. Inclusions $\bot \sqsubseteq C$ and $C \sqsubseteq \bot$, are assumed as $\neg C \sqsubseteq \top$ and $\top \sqsubseteq \neg C$, respectively, given that arguments cannot accept $\bot$ in any of their components (*c.f. consistency* in Def. 1). Finally, assertions as $A(a)$, trigger evidential arguments as $\langle\{\}, A(a)\rangle$. A formal specification of a systematic translation was left out due to space reasons.

Example 1. Let $(A_1 \sqcap A_2) \sqcup (\forall R_1.A_3 \sqcap \exists R_2.\forall R_3.\neg A_4) \sqsubseteq (A_1 \sqcup A_2) \sqcap A_5$ be an axiom conforming the pANF. Four arguments appear in the related DAF: $\langle\{A_1(x), A_2(x)\}, (A_1 \sqcup A_2)(x)\rangle$, $\langle\{(\forall R_1.A_3)(x), (\exists R_2.\forall R_3.\neg A_4)(x)\}, (A_1 \sqcup A_2)(x)\rangle$, $\langle\{A_1(x), A_2(x)\},\ A_5(x)\rangle$, and $\langle\{(\forall R_1.A_3)(x), (\exists R_2.\forall R_3.\neg A_4)(x)\}, A_5(x)\rangle$.

Given an $\mathcal{ALC}$ ontology $\mathcal{O}$, a function $\mathfrak{daf} : \mathcal{ALC} \rightarrowtail \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$, is the mapping $\mathfrak{daf}(\mathcal{O}) = \langle \mathbf{U}, \mathbf{A} \rangle$, which follows the translation methodology described before. That is, $\mathcal{O}$ is turned into pANF, and consequently the DAF is obtained, where each argument identified is considered active, *i.e.*, $\mathbf{A} = \mathbf{U}$.

We will define as $\mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$ to the logic for ontologies $\mathcal{O} \subseteq \mathcal{L}_{\mathcal{T}} \times \mathcal{L}_{\mathcal{A}}$, using $\mathcal{L}_{\mathcal{T}}$ for axioms and $\mathcal{L}_{\mathcal{A}}$ for assertions. It is clear that any $\mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$ ontology conforms the $\mathcal{ALC}$ DL, and it is always in pANF. Moreover, we will assume a function $\mathfrak{af} : \mathcal{ALC} \rightarrowtail \mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$, the *argumental-DL function* that translates any $\mathcal{ALC}$ ontology $\mathcal{O}$ into an equivalent $\mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$ ontology $\mathfrak{af}(\mathcal{O})$. A desirable property of an $\mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$ ontology is that each statement in it generates a single argument in its related DAF, except for obvious unsatisfiable inclusions as $A \sqsubseteq \neg A$, which are filtered by *consistency* in Def. 1 –triggering no related argument in the DAF.

Proposition 1. [1] *Let $\mathcal{O}$ and $\mathcal{O}'$ be two ontologies. If $\mathcal{O}$ conforms the logic $\mathcal{ALC}$, and $\mathcal{O}'$ conforms $\mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$ then*

a) $\mathcal{O}'$ conforms the logic $\mathcal{ALC}$ and is in pANF*,*
b) If $\mathfrak{af}(\mathcal{O}) = \mathcal{O}'$ then $\mathcal{O}$ is equivalent to $\mathcal{O}'$, and
c) If $\mathcal{O}$ conforms the logic $\mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$ then $|\mathcal{O}| \geq |\mathbf{U}|$ where $\mathfrak{daf}(\mathcal{O}) = \langle \mathbf{U}, \mathbf{A} \rangle$.

3.1 The Argumentation Machinery

An argument needs to find its premises supported as a functional part of the reasoning process to reach its claim. In this framework, due to the logic used to represent arguments derived from that of the ontology languages, a single argument is sometimes not enough to support a premise. This is the reason why we introduce the notion of coalitions: to identify a minimal set of arguments verifying some specific properties. For instance, a coalition $\widehat{\mathcal{C}} \subseteq \mathbb{A}\mathtt{rgs}$ may provide support for an argument $\mathcal{B} \in \mathbb{A}\mathtt{rgs}$ through some of its premises. For that matter, we present the functions $\widehat{\mathfrak{clset}}(\widehat{\mathcal{C}}) = \{\mathfrak{cl}(\mathcal{B}) | \mathcal{B} \in \widehat{\mathcal{C}}\}$, and $\widehat{\mathfrak{prset}}(\widehat{\mathcal{C}}) = \bigcup_{\mathcal{B} \in \widehat{\mathcal{C}}} \mathfrak{pr}(\mathcal{B})$.

Definition 4 (Supporter). *Given a* DAF $\langle \mathbf{U}, \mathbf{A} \rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$*, and an argument $\mathcal{B} \in \mathbf{U}$ such that $p \in \mathfrak{pr}(\mathcal{B})$. A set of arguments $\widehat{\mathcal{C}} \subseteq \mathbf{U}$ is a supporting-coalition, or just a* ***supporter****, of $\mathcal{B}$ through p iff it guarantees:*

(support) $\widehat{\mathfrak{clset}}(\widehat{\mathcal{C}}) \models p$,
(consistency) $\widehat{\mathfrak{prset}}(\widehat{\mathcal{C}}) \cup \widehat{\mathfrak{clset}}(\widehat{\mathcal{C}}) \cup \mathfrak{pr}(\mathcal{B}) \cup \{\mathfrak{cl}(\mathcal{B})\} \not\models \bot$*, and*
(minimality) *no $\widehat{\mathcal{C}}' \subset \widehat{\mathcal{C}}$ is a supporter of $\mathcal{B}$ through p.*

Definition 5 (Free Premise). *Given a* DAF $\langle \mathbf{U}, \mathbf{A} \rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$ *and an argument $\mathcal{B} \in \mathbf{U}$, a premise $p \in \mathfrak{pr}(\mathcal{B})$ of $\mathcal{B}$ is* ***free*** *wrt. $\mathbf{U}$ iff there is no supporting-coalition $\widehat{\mathcal{C}} \subseteq \mathbf{U}$ of $\mathcal{B}$ through p.*

Example 2. Suppose we have a set $\mathbf{U} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$, where arguments $\mathcal{B}_1 = \langle \{(\exists R.A_1)(x),\ A_2(x)\},\ B(x) \rangle$, $\mathcal{B}_2 = \langle \{\}, R(a,b) \rangle$, and $\mathcal{B}_3 = \langle \{\}, A_1(b) \rangle$. The set $\widehat{\mathcal{C}} = \{\mathcal{B}_2, \mathcal{B}_3\}$ is a supporting-coalition of $\mathcal{B}_1$ given that $\{R(a,b), A_1(b)\} \models (\exists R.A_1)(x)$. Note that premise $A_2(x)$ is free wrt. $\mathbf{U}$.

[1] In this work, proofs will be omitted due to space reasons.

Example 3. Assume $\mathbf{U} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4\}$, where $\mathcal{B}_1 = \langle\{A_1(x)\}, B_1(x)\rangle$, $\mathcal{B}_2 = \langle\{A_1(x)\}, B_2(x)\rangle$, $\mathcal{B}_3 = \langle\{A_2(x)\}, (A_1 \sqcup B_1)(x)\rangle$, and $\mathcal{B}_4 = \langle\{A_3(x)\}, \neg B_1(x)\rangle$. The set $\widehat{\mathcal{C}} = \{\mathcal{B}_3, \mathcal{B}_4\}$ is a supporting-coalition of $\mathcal{B}_2$. Note that $\widehat{\mathcal{C}}$ cannot be a supporting-coalition of $\mathcal{B}_1$ since it violates (supporter) consistency.

Definition 6 (Supporting-Chain). *Given a* DAF $\langle \mathbf{U}, \mathbf{A}\rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$, *and a sequence* $\lambda = \mathcal{B} \xleftarrow{p} \widehat{\mathcal{C}}_1 \xleftarrow{p_1} \widehat{\mathcal{C}}_2 \xleftarrow{p_2} \cdots$, *where* $(\bigcup_{i\geq 1} \widehat{\mathcal{C}}_i) \cup \{\mathcal{B}\} \subseteq \mathbf{U}$, $p \in \mathfrak{pr}(\mathcal{B})$, $\widehat{\mathcal{C}}_1$ *is a supporting-coalition of* $\mathcal{B}$ *through* p, *and for every* $i > 1$, $p_{i-1} \in \widehat{\mathfrak{prset}}(\widehat{\mathcal{C}}_{i-1})$, *and* $\widehat{\mathcal{C}}_i$ *is a supporting-coalition of* $\widehat{\mathcal{C}}_{i-1}$ *through* p_{i-1}. *Thus,* λ *is referred as a (possible infinite)* ***supporting-chain for*** p *of* $\mathcal{B}$ *wrt.* $\mathbf{U}$.

Whenever λ *has a last identifiable element* $\widehat{\mathcal{C}}_n$, *it follows that every premise in* $\widehat{\mathfrak{prset}}(\widehat{\mathcal{C}}_n)$ *is free wrt.* $\mathbf{U}$, *or* $\widehat{\mathfrak{prset}}(\widehat{\mathcal{C}}_n) = \emptyset$. *In such a case,* λ *is said to be a* ***finite supporting-chain for*** p *of length* n *wrt.* $\mathbf{U}$.

Definitions 4 and 5 are reviewed in Ex. 2 and 3. The iterated aggregation of arguments via the support relation (*c.f.* Def. 4) may conform both, chains of supporting-coalitions for a premise in some argument (*c.f.* Def. 6), as well as sets of interrelated arguments (*c.f.* Def. 7). We will refer to such sets as structures and will be a core part of the argumentation machinery for the proposed DAF.

Definition 7 (Structure). *Given a* DAF $\langle \mathbf{U}, \mathbf{A}\rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$, $\mathbb{S} \subseteq \mathbf{U}$ *is a* ***structure for*** c *iff it guarantees:*

(top argument) *there exists a unique* $\mathcal{B}^{top} \in \mathbb{S}$ *such that* $\mathfrak{cl}(\mathcal{B}^{top}) = c$,
(connectivity) *for every* $\mathcal{B} \in \mathbb{S} \setminus \{\mathcal{B}^{top}\}$, *there exists a unique subset* $\widehat{\mathcal{C}} \subseteq \mathbb{S}$ *such that* $\mathcal{B} \in \widehat{\mathcal{C}}$ *where* $\widehat{\mathcal{C}}$ *is a supporting-coalition of an argument in* $\mathbb{S}$,
(self-consistency) $\widehat{\mathfrak{prset}}(\mathbb{S}) \cup \widehat{\mathfrak{clset}}(\mathbb{S}) \not\models \bot$, *and*
(acyclicity) *every supporting-chain for every* p *of every* $\mathcal{B} \in \mathbb{S}$ *wrt.* $\mathbb{S}$ *is finite.*

The claim and premises of $\mathbb{S}$ *are determined by the functions* $\mathfrak{cl}(\mathbb{S}) = c$ *and* $\mathfrak{pr}(\mathbb{S}) = \{p \in \widehat{\mathfrak{prset}}(\mathbb{S}) \mid p \text{ is a free premise wrt. } \mathbb{S}\}$, *respectively.*

Note that functions "$\mathfrak{pr}$" and "$\mathfrak{cl}$" are overloaded and can be applied both to arguments and structures. This is not going to be problematic since either usage will be rather explicit. In addition to that, we will identify the top argument of a structure $\mathbb{S}$ using the function $\mathfrak{top} : 2^{\mathbb{A}\mathtt{rgs}} \rightarrowtail \mathbb{A}\mathtt{rgs}$. Note that $\mathfrak{cl}(\mathfrak{top}(\mathbb{S})) = \mathfrak{cl}(\mathbb{S})$. Next, it is shown how our theory manages to handle $\mathcal{ALC}$ cyclic terminologies.

Example 4. Given an $\mathcal{ALC}$ ontology $\mathcal{O} = \{A \equiv B\}$, after applying $\mathfrak{daf}(\mathcal{O})$ arguments $\mathcal{B}_1 = \langle\{A(x)\}, B(x)\rangle$, and $\mathcal{B}_2 = \langle\{B(x)\}, A(x)\rangle$, appear. However, a set $\{\mathcal{B}_1, \mathcal{B}_2\}$ cannot be part of any structure since the infinite supporting-chain $\lambda = \mathcal{B}_1 \xleftarrow{A(x)} \{\mathcal{B}_2\} \xleftarrow{B(x)} \{\mathcal{B}_1\} \xleftarrow{A(x)} \cdots$ for $A(x)$ would violate (structure) acyclicity.

A structure $\mathbb{S}$ trivially formed by a single argument is referred as *primitive iff* $|\mathbb{S}| = 1$. Thus, if $\mathbb{S} = \{\mathcal{B}\}$ then $\mathfrak{pr}(\mathcal{B}) = \mathfrak{pr}(\mathbb{S})$ and $\mathfrak{cl}(\mathcal{B}) = \mathfrak{cl}(\mathbb{S})$. However, not every single argument has an associated primitive structure. For instance, from an axiom $A \sqsubseteq A$, no structure could contain its related argument given that it would violate (structure) acyclicity. Depending on the condition of the set of premises in a structure we may identify two different kinds of structures.

Definition 8 (Schematic & Argumental Structure). *Given a* DAF $\langle \mathbf{U}, \mathbf{A}\rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$*, a structure* $\mathbb{S} \subseteq \mathbf{U}$ *is referred either as:* ***Argumental*** *iff* $\mathfrak{pr}(\mathbb{S}) = \emptyset$*, or* ***Schematic*** *iff* $\mathfrak{pr}(\mathbb{S}) \neq \emptyset$*.*

When no distinction is needed, we refer to primitive, schematic, or argumental structures, simply as structures. A **sub-structure relation** "$\trianglelefteq$" relating structures $\mathbb{S} \subseteq \mathbb{A}\mathtt{rgs}$ is defined as stated by the following intuition: given a structure $\mathbb{S}$ for a claim c, if it contains a subset $\mathbb{S}'$ verifying the conditions in Def. 7 for a claim c', then $\mathbb{S}'$ is a structure for c' and $\mathbb{S}' \trianglelefteq \mathbb{S}$.

From a schematic structure and a supporting-coalition for it, a new structure is formed. If this new structure has no free premises, it means that a *variable substitution* was made over the schematic structure leading to an argumental structure. In general, a structure that adds some evidential argument about an individual name, say a, as part of the support for a schematic structure, provokes a variable substitution in the latter. In that case, the argumental structure ends up asserting some property –through its claim– about the individual a. Finally, it is clear that if a structure states a property about some element of the world by means of a free variable x then it is schematic.

Two argumental structures $\mathbb{S}_1$ and $\mathbb{S}_2$ are in conflict whenever they cannot be assumed together. This notion may be made extensive to sets of argumental structures, namely coalition of argumental structures. Coalition of structures is analogous to that of arguments; its formalization is not given due to lack of space. Therefore, the functions "$\widehat{\mathfrak{clset}}$" and "$\widehat{\mathfrak{prset}}$" are overloaded and can be applied both to coalitions $\widehat{\mathcal{C}}$ of arguments and to coalitions $\widehat{\mathbb{C}}$ of structures. Formally, $\widehat{\mathfrak{clset}}(\widehat{\mathbb{C}}) = \{\mathfrak{cl}(\mathbb{S}) | \mathbb{S} \in \widehat{\mathbb{C}}\}$, and $\widehat{\mathfrak{prset}}(\widehat{\mathbb{C}}) = \bigcup_{\mathbb{S}\in\widehat{\mathbb{C}}} \mathfrak{pr}(\mathbb{S})$. Next, we specify the notion of conflict between coalitions of structures as a generalization, since one of them has to be necessarily a singleton. This is required to preserve conflict minimality.

Definition 9 (Conflict). *Let* $\langle \mathbf{U}, \mathbf{A}\rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$ *be a* DAF*, and* $\widehat{\mathbb{C}}_1$ *and* $\widehat{\mathbb{C}}_2$*, two* ***minimal****, and* ***consistent*** *coalitions of structures in* $\mathbf{U}$ *verifying:*

a) $|\widehat{\mathbb{C}}_1| = 1$*, or* $|\widehat{\mathbb{C}}_2| = 1$*, and*
b) $\widehat{\mathfrak{prset}}(\widehat{\mathbb{C}}_1) \models \widehat{\mathfrak{prset}}(\widehat{\mathbb{C}}_2)$ ***(dependency)****, or* $\widehat{\mathfrak{clset}}(\widehat{\mathbb{C}}_1) \models \widehat{\mathfrak{prset}}(\widehat{\mathbb{C}}_2)$ ***(support)****.*

Coalitions $\widehat{\mathbb{C}}_1$ *and* $\widehat{\mathbb{C}}_2$ *are in* ***conflict*** *iff every structure* $\mathbb{S} \subseteq (\widehat{\mathbb{C}}_1 \cup \widehat{\mathbb{C}}_2)$*, is the* ***smallest*** $\mathbb{S}$ *needed to guarantee either:*

(claim-conflict) $\widehat{\mathfrak{clset}}(\widehat{\mathbb{C}}_1) \cup \widehat{\mathfrak{clset}}(\widehat{\mathbb{C}}_2) \models \bot$*, or*
(premise-conflict) $\widehat{\mathfrak{clset}}(\widehat{\mathbb{C}}_1) \cup \widehat{\mathfrak{prset}}(\widehat{\mathbb{C}}_2)) \models \bot$.

Example 5. Assume we have $A_1 \sqcap A_2 \sqsubseteq B$, $A_3 \sqsubseteq A_1$, and $A_3 \sqsubseteq \neg A_2$. Hence, from arguments $\mathcal{B}_1 = \langle\{A_1(x), A_2(x)\}, B(x)\rangle$, $\mathcal{B}_2 = \langle\{A_3(x)\}, A_1(x)\rangle$, and $\mathcal{B}_3 = \langle\{A_3(x)\}, \neg A_2(x)\rangle$, two structures $\mathbb{S}_1 = \{\mathcal{B}_1, \mathcal{B}_2\}$ and $\mathbb{S}_2 = \{\mathcal{B}_3\}$ appear. The trivial coalitions $\widehat{\mathbb{C}}_1 = \{\mathbb{S}_1\}$ and $\widehat{\mathbb{C}}_2 = \{\mathbb{S}_2\}$ model a **premise-conflict**. Note that $\widehat{\mathfrak{prset}}(\widehat{\mathbb{C}}_1) = \{A_3(x), A_2(x)\}$, $\widehat{\mathfrak{prset}}(\widehat{\mathbb{C}}_2) = \{A_3(x)\}$, and $\widehat{\mathfrak{clset}}(\widehat{\mathbb{C}}_2) = \{\neg A_2(x)\}$.

Assume now that we have the same axioms determining arguments $\mathcal{B}_1$ and $\mathcal{B}_2$, and $A_3 \sqsubseteq \neg B$ triggering $\mathcal{B}'_3 = \langle\{A_3(x)\}, \neg B(x)\rangle$. It is easy to verify that a **claim-conflict** will be modeled from $\widehat{\mathbb{C}}_1$ and $\{\{\mathcal{B}'_3\}\}$.

Note that both conflicts in Ex. 5 come from *dependency* (*c.f.* Def. 9*b*). Examples of *claim-conflict* from *support* appear from an ontology $\{\top \sqsubseteq A, \neg A(a)\}$, or either in Ex. 3. It is clear that no *premise-conflict from support* is possible since both support and premise-conflict conditions cannot be mutually verified.

Deciding which coalition of structures succeeds between a conflicting pair, requires a comparison criterion. Such a criterion should be defined upon (*a*) entrenchment of knowledge and (*b*) novelty. For the former case, the ontology engineer may decide to give different levels of importance to individual pieces of knowledge, and for the latter, to prefer new knowledge to older one.

In that sense, we will assume there exists a partial order of arguments called *argument comparison criterion* "$\succcurlyeq$", such that $\mathcal{B}_1 \succcurlyeq \mathcal{B}_2$ states that $\mathcal{B}_1$ has more priority than $\mathcal{B}_2$. Afterwards, two conflictive coalitions of structures $\widehat{\mathbb{C}}_1$ and $\widehat{\mathbb{C}}_2$ are assumed to be ordered by a function "$\mathfrak{pref}$" relying on "$\succcurlyeq$", where $\mathfrak{pref}(\widehat{\mathbb{C}}_1, \widehat{\mathbb{C}}_2) = (\widehat{\mathbb{C}}_1, \widehat{\mathbb{C}}_2)$ implies the attack relation $\widehat{\mathbb{C}}_1 \mathbf{R} \widehat{\mathbb{C}}_2$, *i.e.*, $\widehat{\mathbb{C}}_1$ is a defeater of (or it defeats) $\widehat{\mathbb{C}}_2$. Note that when no pair of arguments is related by "$\succcurlyeq$", both $\widehat{\mathbb{C}}_1 \mathbf{R} \widehat{\mathbb{C}}_2$ and $\widehat{\mathbb{C}}_2 \mathbf{R} \widehat{\mathbb{C}}_1$ appear from any conflicting pair $\widehat{\mathbb{C}}_1$ and $\widehat{\mathbb{C}}_2$.

Definition 10 (Attack Relation Set). *Given a* DAF $\langle \mathbf{U}, \mathbf{A} \rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$*, the set* $\mathbf{R}$ *of attack relations is defined as* $\mathbf{R} = \{(\widehat{\mathbb{C}}_1, \widehat{\mathbb{C}}_2) \mid \widehat{\mathbb{C}}_1 \subseteq 2^{\mathbf{U}}$ *and* $\widehat{\mathbb{C}}_2 \subseteq 2^{\mathbf{U}}$ *are two conflictive coalitions of structures and* $\mathfrak{pref}(\widehat{\mathbb{C}}_1, \widehat{\mathbb{C}}_2) = (\widehat{\mathbb{C}}_1, \widehat{\mathbb{C}}_2)\}$.

Regarding the active condition of the components of the framework, a structure is active *iff* all its arguments are active. This notion is also extended to coalitions of structures by considering a coalition $\widehat{\mathbb{C}}$ active *iff* all its structures are active. Finally, an attack relation $\widehat{\mathbb{C}}_1 \mathbf{R} \widehat{\mathbb{C}}_2$ is active *iff* both $\widehat{\mathbb{C}}_1$ and $\widehat{\mathbb{C}}_2$ are active. That is, if $(\widehat{\mathbb{C}}_1, \widehat{\mathbb{C}}_2) \in \mathbf{R}^{\mathbf{A}} \subseteq \mathbf{R}$ then $\widehat{\mathbb{C}}_1 \subseteq 2^{\mathbf{A}}$ and $\widehat{\mathbb{C}}_2 \subseteq 2^{\mathbf{A}}$, where $\mathbf{R}^{\mathbf{A}}$ is the set standing for every active attack relation in $\mathbf{R}$.

3.2 Acceptability Analysis

In an ontology, inconsistency implies that there are contradictory concept definitions, or assertions that will lead to conflicting arguments within the equivalent DAF. Thus, once the translation is performed, each inconsistency in the original ontology will be reflected as an attack in the resulting DAF. Since the objective of converting an ontology to an argumentation framework is to remove inconsistency from the former, there is a need for a mechanism that allows us to obtain those arguments that prevail over the rest. That is, those arguments that can be consistently assumed together, following some policy. For instance, structures with no defeaters should always prevail, since there is nothing strong enough to be posed against them. The tool we need to resolve inconsistency of concept definitions via an argumentation framework is the notion of *acceptability of arguments*, which is defined on top of an *argumentation semantics* [3]. There are several well-known argumentation semantics, such as the grounded, the stable, and the preferred semantics [4]. These semantics ensure the obtention of a consistent set of arguments, namely an extension. That is, the set of accepted arguments calculated following any of these semantics is such that no pair of

conflictive arguments appears in that same extension. Finally, when we translate an ontology to a DAF, all what is left to do to resolve inconsistencies is to calculate the set of accepted arguments following some semantics, which is going to be translated back to a consistent ontology. It is important to notice that the chosen semantics will greatly affect the resulting ontology. Moreover, problems like multiple extensions from semantics like both the *stable* and the *preferred* may appear, requiring to make a choice among them. On the other hand, the outcome of the *grounded semantics* is always a single extension, which could be empty. Finally, since dealing with multiple extensions is a problem that falls outside the scope of this article, we will choose the grounded semantics, which can be implemented with a simple algorithm. Consequently, we define a mapping $\mathfrak{sem} : 2^{\mathbb{A}\mathtt{rgs}} \times 2^{\mathbb{A}\mathtt{rgs}} \rightarrowtail 2^{\mathbb{A}\mathtt{rgs}} \times 2^{\mathbb{A}\mathtt{rgs}}$, that intuitively behaves as follows.

For every pair of active attack $(\widehat{\mathbb{C}}_1, \widehat{\mathbb{C}}_2) \in \mathbf{R^A}$, if there is no active coalition of structures defeating $\widehat{\mathbb{C}}_1$ **(undefeat)**, then we deactivate some argument from some structure in $\widehat{\mathbb{C}}_2$ **(deactivation)**. As a side-effect, any attack $(\widehat{\mathbb{C}}_2, \widehat{\mathbb{C}}_3)$ will disappear. This process is recursively applied on $\mathbf{R^A}$ until every attack relation is deactivated. As stated before, the outcome of a grounded semantics could be an empty extension. Such an issue arises when there is a loop in the structures attack graph. To overcome this, if *undefeat* is not verified for any $(\widehat{\mathbb{C}}_1, \widehat{\mathbb{C}}_2) \in \mathbf{R^A}$, then *deactivation* is applied to some active attack. Thus, the loop is broken, and the process determined by applying "$\mathfrak{sem}$" can be reconsidered.

Proposition 2. *Given a* DAF $T \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$*, if* $\mathfrak{sem}(T) = \langle \mathbf{U}, \mathbf{A} \rangle$ *then* $\mathbf{R^A} = \emptyset$.

4 Ontology Debugging & Change Through the DAF

In order to provide an ontology change model, we will formalize some properties regarding the relation between the DAF here proposed and its related ontology. Such properties are relevant also as a repairing methodology for terminologies, and in general in the area of ontology debugging. In this sense, we will first characterize the different classes of inconsistencies in an ontology.

Given an ontology $\mathcal{O}$, a concept C is *unsatisfiable iff* for each interpretation $\mathcal{I} \in \mathcal{M}(\mathcal{O})$, $C^{\mathcal{I}} = \emptyset$. As stated in [6], an ontology $\mathcal{O}$ is *incoherent iff* there exists an unsatisfiable concept in $\mathcal{O}$. An incoherence may be considered a kind of inconsistency in the TBox. However, the incoherence does not replace the classical meaning of inconsistency, given that an incoherent ontology may admit models. Hence, an ontology $\mathcal{O}$ is *inconsistent iff* it admits no model.

Let "$\mathfrak{ont}$" be a mapping from a DAF $\langle \mathbf{U}, \mathbf{A} \rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$ to an $\mathcal{ALC}$ ontology $\mathfrak{ont}(\langle \mathbf{U}, \mathbf{A} \rangle)$, following backwards the intuitions given to obtain a DAF by "$\mathfrak{daf}$". Consistency-coherency of the $\mathfrak{ont}$-outcome is related to the attacks in the DAF by Prop. 3. Such relation along with that in Prop. 2 motivates Lemma 1.

Proposition 3. *Given a* DAF $\langle \mathbf{U}, \mathbf{A} \rangle \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$, $\mathbf{R^A} = \emptyset$ *iff* $\mathfrak{ont}(\langle \mathbf{U}, \mathbf{A} \rangle)$ *is a consistent-coherent* $\mathcal{ALC}$ *ontology.*

Lemma 1. *Given a* DAF $T \subseteq \mathbb{A}\mathtt{rgs} \times \mathbb{A}\mathtt{rgs}$, $\mathfrak{ont}(\mathfrak{sem}(T))$ *specifies a consistent-coherent* $\mathcal{ALC}$ *ontology.*

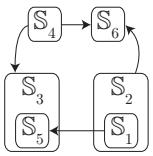
The main contribution of the DAF regarding ontology debugging is sated by Theorem 1, afterwards, Corollary 1 relates that result through "$\mathfrak{af}$" (*c.f.* Sect. 3).

Theorem 1. *Given an $\mathcal{ALC}$ ontology $\mathcal{O}$, if $\mathcal{O}$ is inconsistent and\or incoherent then $\mathfrak{ont}(\mathfrak{sem}(\mathfrak{daf}(\mathcal{O})))$ is a consistent-coherent $\mathcal{ALC}$ ontology.*

Corollary 1. *Given an inconsistent-incoherent $\mathcal{ALC}$ ontology $\mathcal{O}$, there exists a consistent-coherent ontology $\mathcal{O}'$, such that $\mathfrak{af}(\mathcal{O}') \subseteq \mathfrak{af}(\mathcal{O})$.*

Example 6. Let $\mathcal{O} = \{A_1 \sqsubseteq B_1 \sqcap B_2,\ A_2 \sqsubseteq A_1 \sqcap \neg B_2,\ A_1(a),\ B_1(a),\ \neg B_2(a),\ A_2(a)\}$ be an $\mathcal{ALC}$ ontology, we want to debug $\mathcal{O}$ to obtain a related consistent-coherent ontology $\mathcal{O}^R$. Applying $\mathfrak{daf}(\mathcal{O})$, a DAF $\langle \mathbf{U}, \mathbf{A}\rangle$, where $\mathbf{U} = \mathbf{A}$ appears:

Statement	Args.
$A_1 \sqsubseteq B_1 \sqcap B_2$	$\{\mathcal{B}_1, \mathcal{B}_2\}$
$A_2 \sqsubseteq A_1 \sqcap \neg B_2$	$\{\mathcal{B}_3, \mathcal{B}_4\}$
$A_1(a)$	$\{\mathcal{B}_5\}$
$B_1(a)$	$\{\mathcal{B}_6\}$
$\neg B_2(a)$	$\{\mathcal{B}_7\}$
$A_2(a)$	$\{\mathcal{B}_8\}$



$$\left\{\begin{array}{l}\mathcal{B}_1 = \langle\{A_1(x)\}, B_1(x)\rangle\\ \mathcal{B}_2 = \langle\{A_1(x)\}, B_2(x)\rangle\\ \mathcal{B}_3 = \langle\{A_2(x)\}, A_1(x)\rangle\\ \mathcal{B}_4 = \langle\{A_2(x)\}, \neg B_2(x)\rangle\\ \mathcal{B}_5 = \langle\{\}, A_1(a)\rangle\\ \mathcal{B}_6 = \langle\{\}, B_1(a)\rangle\\ \mathcal{B}_7 = \langle\{\}, \neg B_2(a)\rangle\\ \mathcal{B}_8 = \langle\{\}, A_2(a)\rangle\end{array}\right\}$$

Consider the structures $\mathbb{S}_1 = \{\mathcal{B}_3, \mathcal{B}_2\}$, $\mathbb{S}_2 = \{\mathcal{B}_8\} \cup \mathbb{S}_1$, $\mathbb{S}_3 = \{\mathcal{B}_8, \mathcal{B}_4\}$, and $\mathbb{S}_4 = \{\mathcal{B}_5, \mathcal{B}_2\}$; and the primitive structures $\mathbb{S}_5 = \{\mathcal{B}_4\}$ and $\mathbb{S}_6 = \{\mathcal{B}_7\}$.Assuming $\mathcal{B}_2 \succcurlyeq \mathcal{B}_4$ and $\mathcal{B}_2 \succcurlyeq \mathcal{B}_7$, the attack relation set is $\mathbf{R} = \{(\{\mathbb{S}_1\}, \{\mathbb{S}_5\}), (\{\mathbb{S}_2\}, \{\mathbb{S}_6\}), (\{\mathbb{S}_4\}, \{\mathbb{S}_6\}), (\{\mathbb{S}_4\}, \{\mathbb{S}_3\})\}$ (see the graph depicted above). Note that $(\{\mathbb{S}_2\}, \{\mathbb{S}_3\})$ is not in $\mathbf{R}$ given that $\mathbb{S}_1 \trianglelefteq \mathbb{S}_2$, $\mathbb{S}_5 \trianglelefteq \mathbb{S}_3$, and $(\{\mathbb{S}_1\}, \{\mathbb{S}_5\}) \in \mathbf{R}$ (*c.f.* Def. 9).

The acceptability analysis determines $\mathbb{S}_3$, $\mathbb{S}_5$, and $\mathbb{S}_6$ to be deactivated, and since $\mathbb{S}_5 \trianglelefteq \mathbb{S}_3$, deactivating $\mathcal{B}_4$ and $\mathcal{B}_7$ is enough. Afterwards, $\mathfrak{sem}(\mathfrak{daf}(\mathcal{O}))$ determines the new set of active arguments $\{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_5, \mathcal{B}_6, \mathcal{B}_8\}$. Finally, following the table above, the operation $\mathfrak{ont}(\mathfrak{sem}(\mathfrak{daf}(\mathcal{O})))$ constructs the repaired ontology $\mathcal{O}^R = \{A_1 \sqsubseteq B_1 \sqcap B_2,\ A_2 \sqsubseteq A_1,\ A_1(a),\ B_1(a),\ A_2(a)\}$.

Note that, assuming $\mathcal{B}_7 \succcurlyeq \mathcal{B}_2 \succcurlyeq \mathcal{B}_4$, conflicts involving $\mathbb{S}_6$ are inverted leading to $(\{\mathbb{S}_6\}, \{\mathbb{S}_2\})$ and $(\{\mathbb{S}_6\}, \{\mathbb{S}_4\})$. In such a case, only $\mathcal{B}_2$ would be deactivated.

In the sequel, we propose an ontology change operator "$\mathfrak{C}$" for $\mathcal{ALC}$ ontologies wrt. a consistent-coherent ontology X. Afterwards, its rationality is analyzed.

Definition 11 (Ontology Change Operation). *Let $\mathcal{O}$ and X be two $\mathcal{ALC}$ ontologies, where X is consistent-coherent. The* ***ontology change operator*** *"$\mathfrak{C}$" is defined as $\mathfrak{C}^{\mathcal{O}}(X) = \mathfrak{ont}(\mathfrak{sem}(\mathfrak{daf}(\mathcal{O} \cup X)))$.*

A prioritized approach of change is assumed given that each piece of knowledge from X is considered the ultimate perception of the world, consequently, X will be fully accepted in the evolved ontology $\mathfrak{C}^{\mathcal{O}}(X)$. In this sense, $\mathcal{X} \succcurlyeq \mathcal{B}$ is assumed for any argument $\mathcal{X}$ determined from X, and any $\mathcal{B}$ from $\mathcal{O}$. Afterwards, if $\mathcal{X}$ appears in some conflicting coalition, the function "$\mathfrak{sem}$" will deactivate some other $\mathcal{B}$ involved in the attack. Since X is required to be consistent-coherent, the previous statement is verified given that no pair of arguments from X will be in direct conflict. Finally, every argument generated from X will be kept active.

Example 7. Let $\mathcal{O} = \{R(a,b),\ R(b,c),\ R(c,d),\ A(a),\ \neg A(c),\ \neg A(d)\}$ be an ontology where R and A stand for "supervised-by" and "researcher", respectively. Now suppose that a new regulation poses that no academic researcher might be supervised by a non-researcher. Therefore, we would need to provoke the ontology to evolve by an operation $\mathfrak{C}^{\mathcal{O}}(X)$, where $X = \{A \sqsubseteq \forall R.A\}$.

Applying $\mathfrak{daf}(\mathcal{O} \cup X)$, the $\mathtt{DAF}\ \langle \mathbf{U}, \mathbf{A} \rangle$ is determined as $\mathbf{U} = \mathbf{A} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_6, \mathcal{X}_1\}$, where $\mathcal{B}_1 = \langle\{\}, R(a,b)\rangle$, $\mathcal{B}_2 = \langle\{\}, R(b,c)\rangle$, $\mathcal{B}_3 = \langle\{\}, R(c,d)\rangle$, $\mathcal{B}_4 = \langle\{\}, A(a)\rangle$, $\mathcal{B}_5 = \langle\{\}, \neg A(c)\rangle$, $\mathcal{B}_6 = \langle\{\}, \neg A(d)\rangle$, $\mathcal{X}_1 = \langle\{A(x)\}, (\forall R.A)(x)\rangle$.

Since the new information is prioritized it follows $\mathcal{X}_1 \succcurlyeq \mathcal{B}_i$, $i \in \{1, \ldots, 6\}$. The argumental structure $\mathbb{S}_1 = \{\mathcal{B}_4, \mathcal{X}_1\}$ appears. Later on, the set $\widehat{\mathcal{C}}_1 = \{\mathcal{X}_1, \mathcal{B}_1\}$ is a supporting-coalition of $\mathcal{X}_1$ through $A(b)$, $\widehat{\mathcal{C}}_2 = \{\mathcal{X}_1, \mathcal{B}_2\}$ is a supporting-coalition of $\mathcal{X}_1$ through $A(c)$, and $\widehat{\mathcal{C}}_3 = \{\mathcal{X}_1, \mathcal{B}_3\}$ is a supporting-coalition of $\mathcal{X}_1$ through $A(d)$. Hence, the schematic structures $\mathbb{S}_2 = \{\mathcal{X}_1, \mathcal{B}_1\}$, $\mathbb{S}_3 = \{\mathcal{X}_1, \mathcal{B}_2\}$, and $\mathbb{S}_4 = \{\mathcal{X}_1, \mathcal{B}_3\}$, appear with claims $\mathfrak{cl}(\mathbb{S}_2) = (\forall R.A)(b)$, $\mathfrak{cl}(\mathbb{S}_3) = (\forall R.A)(c)$, and $\mathfrak{cl}(\mathbb{S}_4) = (\forall R.A)(d)$; and premises $\mathfrak{pr}(\mathbb{S}_2) = A(a)$, $\mathfrak{pr}(\mathbb{S}_3) = A(b)$, and $\mathfrak{pr}(\mathbb{S}_4) = A(c)$. Thus, appear the related argumental structures $\mathbb{S}_5 = \{\mathcal{B}_4, \mathcal{X}_1, \mathcal{B}_1\}$, $\mathbb{S}_6 = \{\mathcal{B}_4, \mathcal{X}_1, \mathcal{B}_1, \mathcal{B}_2\}$, and $\mathbb{S}_7 = \{\mathcal{B}_4, \mathcal{X}_1, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$, where $\mathbb{S}_2 \trianglelefteq \mathbb{S}_5$, $\mathbb{S}_3 \trianglelefteq \mathbb{S}_6$, and $\mathbb{S}_4 \trianglelefteq \mathbb{S}_7$, as well as $\mathbb{S}_5 \trianglelefteq \mathbb{S}_6$, and $\mathbb{S}_6 \trianglelefteq \mathbb{S}_7$. Note also that, $\mathbb{S}_1$ is sub-structure of $\mathbb{S}_5$, $\mathbb{S}_6$, and $\mathbb{S}_7$.

Consider now the coalitions of structures $\widehat{\mathbb{C}}_1 = \{\{\mathcal{B}_2\}, \{\mathcal{B}_5\}\}$, and $\widehat{\mathbb{C}}_2 = \{\{\mathcal{B}_3\}, \{\mathcal{B}_6\}\}$. The following attack relations appear: $\{\mathbb{S}_5\}\mathbf{R}\widehat{\mathbb{C}}_1$ and $\{\mathbb{S}_6\}\mathbf{R}\widehat{\mathbb{C}}_2$ (refer to Fig. 2). Later on, considering also the coalitions of structures $\widehat{\mathbb{C}}_3 = \{\mathbb{S}_5, \{\mathcal{B}_2\}\}$, and $\widehat{\mathbb{C}}_4 = \{\mathbb{S}_6, \{\mathcal{B}_3\}\}$, attacks $\widehat{\mathbb{C}}_3\mathbf{R}\{\{\mathcal{B}_5\}\}$ and $\widehat{\mathbb{C}}_4\mathbf{R}\{\{\mathcal{B}_6\}\}$ appear.

The acceptability analysis determines coalitions $\widehat{\mathbb{C}}_1$, $\widehat{\mathbb{C}}_2$, $\{\{\mathcal{B}_5\}\}$, and $\{\{\mathcal{B}_6\}\}$ to deactivate. Later on, the deactivation of $\mathcal{B}_5$ and $\mathcal{B}_6$ deactivates every attack. Afterwards, $\mathfrak{sem}(\mathfrak{daf}(\mathcal{O} \cup X))$ determines the set of active arguments as $\{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{X}_1\}$. Finally, the operation $\mathfrak{ont}(\mathfrak{sem}(\mathfrak{daf}(\mathcal{O} \cup X)))$ constructs the evolved ontology $\mathcal{O}^R = \{A \sqsubseteq \forall R.A,\ R(a,b),\ R(b,c),\ R(c,d),\ A(a)\}$.

Assuming also $\mathcal{B}_5 \succcurlyeq \mathcal{B}_2$ and $\mathcal{B}_6 \succcurlyeq \mathcal{B}_6$, $\{\{\mathcal{B}_5\}\}\mathbf{R}\widehat{\mathbb{C}}_3$ and $\{\{\mathcal{B}_6\}\}\mathbf{R}\widehat{\mathbb{C}}_4$ appear along with the attacks from Fig. 2. Hence, only $\mathcal{B}_2$ and $\mathcal{B}_3$ would be deactivated.

In the last few years, different ontology change operations have been proposed usually along with some characterization for its rationality. Some examples may be found in [15,16,14,12]. Particularly, in [6], a set of rationality postulates is analyzed abstracting away from the definition of any change operation. In this work, we analyze the model of change proposed wrt. the following general set of basic postulates for an ontology change operation $\mathcal{O} * X$.

$(\mathcal{O} * 1)$ $X \subseteq \mathcal{O} * X$.
$(\mathcal{O} * 2)$ If $\mathcal{O} \cup X$ is consistent-coherent then $\mathcal{O} * X = \mathcal{O} \cup X$.
$(\mathcal{O} * 3)$ If X is consistent-coherent then $\mathcal{O} * X$ is consistent-coherent.
$(\mathcal{O} * 4)$ If $X \cong Y$ then $\mathcal{O} * X \cong \mathcal{O} * Y$ (assuming "$\cong$" as logically equivalent to).
$(\mathcal{O} * 5)$ $\mathcal{O} * X \subseteq \mathcal{O} \cup X$.

Properties $(\mathcal{O}*1)$ to $(\mathcal{O}*4)$ are adapted from the revision postulates proposed in [6] which in turn follow the original basic AGM postulates for revision exposed in [1]. Besides, $(\mathcal{O} * 5)$ is adapted from [9].

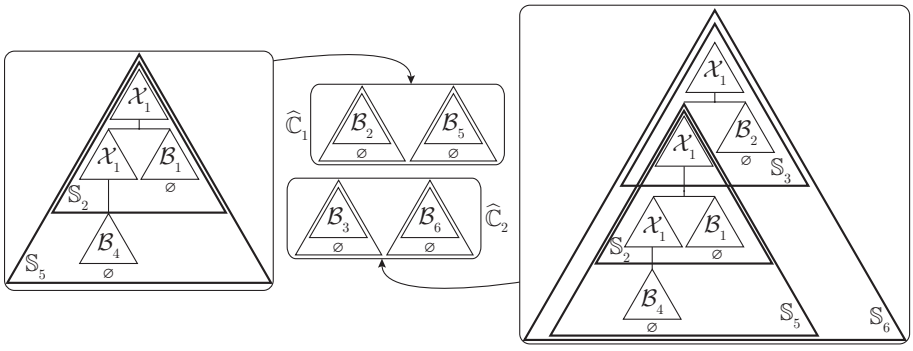


Fig. 2. Some attacks from Ex. 7. Multiple occurrences of an argument within a structure refer to its different instances determined by every possible variable substitution.

Lemma 2. *Given two ontologies $\mathcal{O}$ and X, and assuming $\mathfrak{C}^{\mathcal{O}}(X)$ as $\mathcal{O} * X$:*

a) If $\mathcal{O}$ and X conform the $\mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$ DL then "$\mathfrak{C}$" satisfies $(\mathcal{O} * 1)$ to $(\mathcal{O} * 5)$.
b) If $\mathcal{O}$ and X conform the $\mathcal{ALC}$ DL then "$\mathfrak{C}$" satisfies $(\mathcal{O} * 1)$ to $(\mathcal{O} * 4)$.

As stated by Lemma 2*b*), "$\mathfrak{C}$" fails to satisfy $(\mathcal{O} * 5)$ for $\mathcal{ALC}$ ontologies. For instance, assume two $\mathcal{ALC}$ ontologies $\mathcal{O} = \{A \sqsubseteq B_1 \sqcap B_2, A(a)\}$, and $X = \{\neg B_1(a)\}$. If assertional loss is required to be avoided, then we prefer (via "$\succcurlyeq$") evidential arguments over any other argument. Finally, the evolved ontology would end up as $\mathfrak{C}^{\mathcal{O}}(X) = \{A \sqsubseteq B_2, A(a), \neg B_1(a)\}$.

Proposition 4. *Given the $\mathcal{ALC}$ ontologies $\mathcal{O}$, and X, "$\mathfrak{af}$" is distributable wrt. the ontology change operator "$\mathfrak{C}$". That is, $\mathfrak{af}(\mathfrak{C}^{\mathcal{O}}(X)) = \mathfrak{C}^{\mathfrak{af}(\mathcal{O})}(\mathfrak{af}(X))$.*

Since every $\mathcal{ALC}$ ontology $\mathcal{O}$ can be translated into an equivalent $\mathcal{ALC}^{\mathbb{A}\mathtt{rgs}}$ $\mathfrak{af}(\mathcal{O})$ (*c.f.* Prop. 1), Theorem 2 captures the change operator "$\mathfrak{C}$" in a rational manner by means of $(\mathcal{O} * 1)$ to $(\mathcal{O} * 5)$, Theorem 1, Lemma 2, and Prop. 4.

Theorem 2. *Given the $\mathcal{ALC}$ ontologies $\mathcal{O}$, and X, an ontology change operator "$\mathfrak{C}$" satisfies $(\mathcal{O} * 1)$ $(\mathcal{O} * 5)$ through "$\mathfrak{af}$" iff $\mathfrak{af}(\mathfrak{C}^{\mathcal{O}}(X)) = \mathfrak{af}(\mathcal{O}) * \mathfrak{af}(X)$.*
*Finally, "$\mathfrak{C}$" is said to be **rational through argumental-DL form**.*

5 Related and Future Work

Debugging of terminologies is usually focused on the recognition of sources of concept-unsatisfiability. In this sense, the union of conflictive coalitions of structures presented in this work, may be related to constructions like Kernel Sets [8] or, *minimal inconsistent preserving sub-terminologies* (MIPS) [20], which have been previously used in ontology debugging [19] and change [14]. MIPS may be also related to works in ontology integration [10], and debugging like [11], where *maximally concept-satisfiable subsets* (MCSS) were proposed for that matter.

As minimal sets inferring $\perp$, kernel sets are also similar to the union of conflictive coalitions of structures. Moreover, in works like [16,14,12], incision functions are used to cut the appropriate piece of knowledge from every kernel such that they do not appear in the evolved ontology. In that sense, the function "$\mathfrak{sem}$" deactivates the appropriate argument from each attack in order to deactivate every possible argument conflict from the DAF, just like incision functions do.

As stated before, further implementations of the model here presented could be done (1) as a module to be incorporated to the DL reasoner, or (2) as a DL-argumentation reasoner. For the second option, a DL reasoner based on argumentation could be an interesting alternative to those like RACER, FaCT, and FaCT++. An approach relying on defeasible logic to enrich ontologies with argumentation is presented in [21], but our proposal differs in that we abstract away from the logic to specify arguments, and also in the argumentation framework and machinery proposed to handle ontology dynamics. A dynamic argumentative approach could decide "on the fly" what to keep or discard from different sources without applying any changes to them. Moreover, an ontology may keep inconsistencies leaving its resolution up to the argumentation process, that is, the ontology reasoner would manage to dynamically handle inconsistency.

It also seems interesting to further investigate this proposal from a theoretical point of view. This would allow, to relate it to other approaches of ontology change like [15,16,14,12], and even to investigate the existence of equivalences regarding other models of change beyond the scope of the semantic web, like the classical AGM model [1]. To this matter, works like [7,5] should be considered. In this sense, since the approach here presented was constructed as an extension of widespread accepted argumentation methodologies [4], it could benefit from previous results in that area. Moreover, a preliminary abstract DAF was presented in [17], and formalized in detail in [18]. Besides, a model of change in argumentation systems was recently proposed in [17], known as Argument Theory Change, and reified to defeasible logic programming in [13], meaning that properties and methodologies from that model of change could also be adapted to the model here proposed, and even implemented.

As mentioned before, the grounded semantics [4] could return empty extensions. For instance, refer to Ex. 6 assuming an empty comparison criterion "$\succcurlyeq$". Thus, the usage of different semantics [3] could be studied to overcome this issue.

6 Conclusion

A novel theoretical approach to handle ontology debugging through argumentation was presented. Such approach manages ontology dynamics by proposing an ontology change operator along with a characterization for its rationality.

The methodology here proposed was deemed as theoretical given the complexity to translate ontologies into a DAF. In this sense, we claimed this proposal to be a starting point to two different practical approaches: a specialized ontology reasoner based on argumentation, and the theoretical model at issue. The former case exposes an interesting proposal to incorporate to the semantic web

the most characteristic feature of argumentation reasoners: to keep inconsistency while managing to reason on top of it.

References

1. C. Alchourrón, P. Gärdenfors, and D. Makinson. *On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. The Journal of Symbolic Logic*, 50:510–530, 1985.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *Description Logic Handbook: Theory, Implementation and Application*. Cambridge University Press, Cambridge, 2003.
3. P. Baroni and M. Giacomin. On Principle-Based Evaluation of Extension-Based Argumentation Semantics. *Artificial Intelligence*, 171(10-15):675–700, 2007.
4. P. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning and Logic Programming and n-person Games. *Artificial Intelligence*, 77:321–357, 1995.
5. G. Flouris. On Belief Change and Ontology Evolution. *Doctoral Dissertation, Department of Computer Science, University of Crete*, February 2006.
6. G. Flouris, Z. Huang, J. Z. Pan, D. Plexousakis, and H. Wache. Inconsistencies, Negations and Changes in Ontologies. In *AAAI*, pages 1295–1300, 2006.
7. G. Flouris, D. Plexousakis, and G. Antoniou. On Applying the AGM Theory to DLs and OWL. In *ISWC*, pages 216–231, 2005.
8. S. O. Hansson. Kernel Contraction. *Journal of Symbolic Logic*, 59:845–859, 1994.
9. S. O. Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating. Springer*. 1999.
10. T. Meyer, K. Lee, and R. Booth. Knowledge Integration for Description Logics. In *AAAI*, pages 645–650, 2005.
11. T. Meyer, K. Lee, R. Booth, and J. Z. Pan. Finding Maximally Satisfiable Terminologies for the Description Logic ALC. In *AAAI*, 2006.
12. M. Moguillansky, M. Falappa, and G. Simari. Model-Based Contractions for Description Logics. In *NMR*, pages 34–42, 2008.
13. M. Moguillansky, N. Rotstein, M. Falappa, A. García, and G. Simari. Argument Theory Change Applied to Defeasible Logic Programming. In *AAAI*, pages 132–137, 2008.
14. G. Qi, P. Haase, Z. Huang, and J. Z. Pan. A Kernel Revision Operator for Terminologies. In *DL*, 2008.
15. G. Qi, W. Liu, and D. A. Bell. Knowledge Base Revision in Description Logics. In *JELIA*, pages 386–398, 2006.
16. M. M. Ribeiro and R. Wassermann. Base Revision in Description Logics - preliminary results. In *IWOD*, 2007.
17. N. Rotstein, M. Moguillansky, M. Falappa, A. García, and G. Simari. Argument Theory Change: Revision Upon Warrant. In *COMMA*, pages 336–347, 2008.
18. N. Rotstein, M. Moguillansky, M. Falappa, A. García, and G. Simari. An Abstract Argumentation Framework for Handling Dynamics. In *NMR*, pages 131–139, 2008.
19. S. Schlobach. Debugging and Semantic Clarification by Pinpointing. In *ESWC*, pages 226–240, 2005.
20. S. Schlobach and R. Cornet. Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In *IJCAI*, pages 355–362, 2003.
21. M. Williams and A. Hunter. Harnessing Ontologies for Argument-Based Decision-Making in Breast Cancer. *ICTAI*, 2:254–261, 2007.

Using Background Knowledge for Ontology Evolution

Fouad Zablith, Marta Sabou, Mathieu d'Aquin, and Enrico Motta

Knowledge Media Institute (KMi), The Open University
Walton Hall, Milton Keynes, MK7 6AA, United Kingdom
{f.zablith, r.m.sabou, m.daquin, e.motta}@open.ac.uk

Abstract. One of the current bottlenecks for automating ontology evolution is resolving the right links between newly arising information and the existing knowledge in the ontology. Most of existing approaches mainly rely on the user when it comes to capturing and representing new knowledge. Our ontology evolution framework intends to reduce or even eliminate user input through the use of background knowledge. In this paper, we show how various sources of background knowledge could be exploited for relation discovery. We perform a relation discovery experiment focusing on the use of WordNet and Semantic Web ontologies as sources of background knowledge. We back our experiment with a thorough analysis that highlights various issues on how to improve and validate relation discovery in the future, which will directly improve the task of automatically performing ontology changes during evolution.

1 Introduction

Ontologies are fundamental building blocks of the Semantic Web and are often used as the knowledge backbones of advanced information systems. As such, they need to be kept up to date in order to reflect the changes that affect the life-cycle of such systems (e.g. changes in the underlying data sets, need for new functionalities, etc). This task, described as the "timely adaptation of an ontology to the arisen changes and the consistent management of these changes", is called *ontology evolution* [11].

While it seems necessary to apply such a process consistently for most ontology-based systems, it is often a time-consuming and knowledge intensive task, as it requires a knowledge engineer to identify the need for change, perform appropriate changes on the base ontology and manage its various versions. Several research efforts have addressed various phases of this complex process. A first category of approaches [12, 16, 19, 20] are concerned with formalisms for representing changes and for facilitating the versioning process. Another category of work [2, 5, 14, 15, 17] aim to identify potential novel information that should be added to the ontology. They do this primarily by exploiting the changes occurring in the various data sources underlying an information system (e.g. databases, text corpora, etc), or by interpreting trends in the behavior of the users of the system [2, 5]. A few systems from this category also investigate methods that

propose appropriate changes to an ontology given a piece of novel information. These methods typically rely on agent negotiation/multi-agent systems [14, 15, 17] to propose concrete changes which then are verified by the ontology curator.

Our hypothesis is that this process of prescribing ways in which a piece of new information can be added to a base ontology could be made more cost effective by relying on various sources of background knowledge to reduce human intervention. These sources can have varying levels of formality, ranging from formal knowledge structures, such as ontologies available on the Semantic Web, to thesauri (e.g. WordNet), or even unstructured textual data from the Web. We believe that, by combining such complementary sources of knowledge, a large part of the ontology evolution process can be automated. Therefore, we propose an ontology evolution framework, called Evolva[1], which once a set of terms has been identified as potentially relevant concepts to add to the ontology, makes use of external sources of background knowledge to establish relations between these terms and the knowledge already present in the ontology.

In this paper, we present a first implementation of the *relation discovery for ontology changes* process of Evolva, exploiting WordNet and Semantic Web ontologies as sources of background knowledge. We describe initial experiments realized with this process, showing the feasibility of the approach and pointing out possible ways to better exploit external sources of knowledge, as well as the base ontology, in the evolution process.

We start by describing a motivating scenario in Section 2 and providing a brief overview of Evolva in Section 3. In Section 4 we detail our ideas about the use of background knowledge sources. We describe the implementation details of our prototype (Section 5) followed by a discussion of our experimental results obtained in the context of the example scenario (Section 6). We finalize the paper with some notes on related work and our conclusions (Sections 7 and 8).

2 Example Scenario: The KMi Semantic Web Portal

The Knowledge Media Institute's (KMi) Semantic Web portal[2] is a typical example of an ontology based information system. The portal provides access to various data sources (e.g. staff and publication databases, news stories) by relying on an ontology that represents the academic domain, namely the AKT ontology[3]. The ontology has been originally built manually and is automatically populated by relying on a set of manually established mapping rules [13].

However, apart from the population process, the evolution of this ontology was performed entirely manually. Indeed, as in this scenario ontology population is bound by strict and limited mapping expressions, when a new type of term (i.e. not covered by the mapping rules) is extracted, the intervention of the ontology administrator is required to modify the mappings. Moreover, the ontology schema can only be updated by the administrator. Finally, with no mechanism

[1] An overview of Evolva can be found in [22].
[2] http://semanticweb.kmi.open.ac.uk
[3] http://kmi.open.ac.uk/projects/akt/ref-onto/index.html

to support recording and managing changes, it is difficult to maintain a proper versioning of the ontology. Therefore, as this manual evolution of the ontology could not follow the changes in the underlying data (which happen on a daily basis), the ontology was finally left outdated.

3 Evolva: An Ontology Evolution Framework

Evolva is a complete ontology evolution framework that relies on various sources of background knowledge to support its process. We hereby provide a brief overview of its five components design (Figure 1), which is only partially implemented now. More details are available in [22].

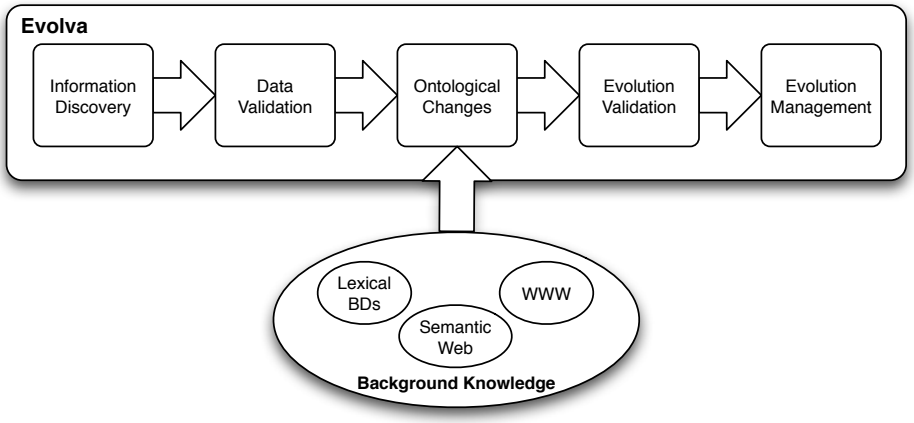


Fig. 1. The main components of Evolva.

Information Discovery. Our approach starts with discovering potentially new information from the data sources associated with the information system. Contrasting the ontology with the content of these sources is a way of detecting new knowledge that should be reflected by the base ontology. Data sources exist in various formats from unstructured data such as text documents or tags, to structured data such as databases and ontologies. This component handles each data source differently: (1) Text documents are processed using information extraction, ontology learning or entity recognition techniques. (2) Other external ontologies are subject to translation for language compatibility with the base ontology, and (3) database content is translated into ontology languages.

Data Validation. Discovered information are validated in this component. We rely on a set of heuristic rules such as the length of the extracted terms. This is especially needed for information discovered from text documents, as information extraction techniques are likely to introduce noise. For example, most of the two-letter terms extracted from KMi's news corpora are meaningless and should be

discarded. In the case of structured data, this validation is less needed as the type of information is explicitly defined.

Ontological Changes. This component is in charge of establishing relations between the extracted terms and the concepts in the base ontology. These relations are identified by exploring a variety of background knowledge sources, as we will describe in the next section. Appropriate changes will be performed directly to the base ontology and recorded by using the formal representations proposed by [12] and [19], which we will explore at a later stage of our research.

Evolution Validation. Performing ontology changes automatically may introduce inconsistencies and incoherences in the base ontology. Also, due to having multiple data sources, data duplication is likely to arise. Conflicting knowledge is highly possible to occur and should be handled by automated reasoning. As evolution is an ongoing process, many statements are time dependent and should be treated accordingly by applying temporal reasoning techniques.

Evolution Management. Managing the evolution will be about giving the ontology curator a degree of control over the evolution, as well as propagating changes to the dependent components of the ontology such as other ontologies or applications. User control will deal with tracking ontology changes, spotting and solving unresolved problems.

While this section presents an overview of our ontology evolution framework, our focus in this paper is on the relation discovery process that occurs in the *ontological changes* component.

4 The Role of Background Knowledge in Evolva

A core task in most ontology evolution scenarios is the integration of new knowledge into the base ontology. We focus on those scenarios in which such new knowledge is extracted as a set of emerging terms from textual corpora, databases, or domain ontologies. Traditionally this process of integrating a new set of emerging terms is performed by the ontology curator. For a given term, he/she would rely on his/her own knowledge of the domain to identify, in the base ontology, elements related to the term, as well as the actual relations they share. As such, it is a time consuming process, which requires the ontology curator to know well the ontology, as well as being an expert in the domain it covers.

Evolva makes use of various background knowledge sources to identify relations between new terms and ontology elements. The hypothesis is that a large part of the process of updating an ontology with new terms can be automated by using these sources as an alternative to the curator's domain knowledge. We have identified several potential sources of background knowledge. For example, thesauri such as WordNet have been long used as a reference resource for establishing relations between two given concepts, based on the relation that exists between their synsets. Because WordNet's dictionary can be downloaded and accessed locally by the system and because a variety of relation discovery techniques have been proposed and optimized, exploring this resource is quite fast. Online ontologies constitute another source of background knowledge which has

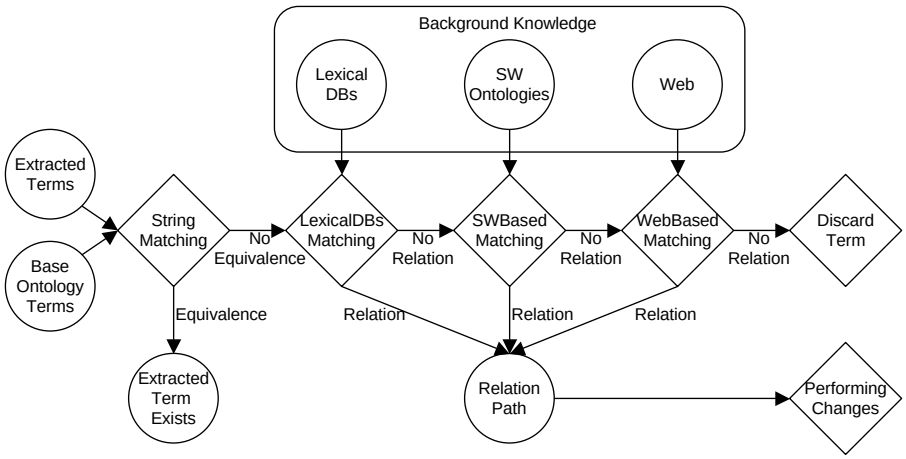


Fig. 2. Finding relations between new terms and the base ontology in Evolva.

been recently explored to support various tasks such as ontology matching [18] or enrichment [4]. While the initial results in employing these ontologies are encouraging, these techniques are still novel and in need of further optimizations (in particular regarding time-performance). Finally, the Web itself has been recognized as a vast source of information that can be exploited for relation discovery through the use of so-called lexico-syntactic patterns [6]. Because they rely on unstructured, textual sources, these techniques are more likely to introduce noise than the previously mentioned techniques which rely on already formalized knowledge. Additionally, these techniques are time consuming given that they operate at Web scale.

Taking into account these considerations, we devised a relation discovery process that combines various background knowledge sources with the goal of optimizing time-performance and precision. As shown in Figure 2, the relation discovery starts from quick methods that are likely to return good results, and continues with slower methods which are likely to introduce a higher percentage of noise: (1) The process begins with string matching for detecting already existing terms in the ontology. This will identify equivalence relations between the new terms and the ontology elements. (2) Extracted elements that do not exist in the base ontology are passed to a module that performs relation discovery by exploring WordNet's synset hierarchy. (3) Terms that could not be incorporated by using WordNet are passed to the next module which explores Semantic Web ontologies. (4) If no relation is found, we resort to the slower and more noisy methods which explore the Web itself through search engines' APIs and lexico-syntactic patterns [6]. In case no relation is found at the final level, the extracted term is discarded or, optionally, forwarded for manual check.

5 Implementation of Evolva's Relation Discovery

We have partially implemented the algorithm presented in Figure 2 by making use of methods for exploring two main background knowledge sources: WordNet and online ontologies. We have not yet implemented methods for exploiting the Web as a source of knowledge. The first part of the implementation performs the string matching between the extracted terms and the ontology elements. We rely on the Jaro distance metric similarity [8] which takes into account the number and positions of the common characters between a term and an ontology concept label. This string similarity technique performs well on short strings, and offers a way to find a match between strings that are slightly different only because of typos or the use of different naming conventions.

The WordNet based relation discovery uses the Wu and Palmer similarity [21] for identifying the best similarity measure between the two terms. This measure is computed according to the following formula:

$$Sim(C1, C2) = \frac{2*N3}{N1+N2+2*N3}$$

where C1 and C2 are the concepts to check for similarity, N1 is the number of nodes on the path from C1 to the least common superconcept (C3) of C1 and C2, N2 is the number of nodes between C2 and C3, and N3 is the number of nodes on the path from C3 to the root [21]. For those terms that are most closely related to each other, we derive a relation by exploring WordNet's hierarchy using a functionality built into its Java library[4]. This will result in a relation between a term, as well as an inference path which lead to its discovery.

The terms that could not be related to the base ontology are forwarded to the next module which makes use of online ontologies. For this component, we rely on the Scarlet relation discovery engine[5]. It is worth to note that we handle ontologies at the level of statements, i.e. ontologies are not processed as one block of statements. Thus we focus on knowledge reuse without taking care of the validation of the sources as a whole with respect to the base ontology. Scarlet [18] automatically selects and explores online ontologies *to discover relations between two given concepts*. For example, when relating two concepts labeled *Researcher* and *AcademicStaff*, Scarlet 1) identifies (at run-time) online ontologies that can provide information about how these two concepts inter-relate and then 2) combines this information to infer their relation. [18] describes two increasingly sophisticated strategies to identify and to exploit online ontologies for relation discovery. Hereby, we rely on the first strategy that derives a relation between two concepts if this relation is defined within a single online ontology, e.g., stating that *Researcher* $\sqsubseteq$ *AcademicStaff*. Besides subsumption relations, Scarlet is also able to identify disjoint and named relations. All relations are obtained

[4] http://jwordnet.sourceforge.net/
[5] http://scarlet.open.ac.uk/

by using derivation rules which explore not only direct relations but also relations deduced by applying subsumption reasoning within a given ontology. For example, when matching two concepts labeled *Drinking Water* and *tap_water*, appropriate anchor terms are discovered in the TAP ontology and the following subsumption chain in the external ontology is used to deduce a subsumption relation: $DrinkingWater \sqsubseteq FlatDrinkingWater \sqsubseteq TapWater$. Note, that as in the case of WordNet, the derived relations are accompanied by a path of inferences that lead to them.

6 Relation Discovery Experiment

We performed an experimental evaluation of the current implementation of the relation discovery module on the data sets provided by the KMi scenario. Our goal was to answer three main questions. First, we wanted to get an insight into the efficiency, in particular in terms of precision, of the relation discovery relying on our two main background knowledge sources: WordNet and online ontologies. Second, we wished to understand the main reasons behind the incorrect relations, leading to ways of identifying these automatically. Tackling these issues would further increase the precision of the identified relations and bring us closer to a full automation of this task. Finally, as a preparation for implementing Evolva's algorithm for performing ontology changes, we also wanted to identify a few typical cases of relations to integrate into the base ontology.

6.1 Experimental Data

We relied on 20 documents from KMi's news repository as a source of potentially new information. We used Text2Onto's extraction algorithm [7] and discovered 520 terms from these text documents.

The base ontology which we wish to evolve (i.e. KMi's ontology) currently contains 256 concepts, although it has not been updated for well over one year, since April 2007. By using the Jaro matcher we identified that 21 of the extracted terms have exact correspondences within the base ontology and that 7 are closely related to some concepts (i.e. their similarity coefficient is above the threshold of 0.92).

6.2 Evaluation of the WordNet Based Relation Discovery

Out of the 492 remaining new terms, 162 have been related to concepts of the ontology thanks to the WordNet based relation discovery module. Some of these relations were duplicates as they related the same pair of term and concept through the relation of different synsets. For evaluation purposes, we eliminated duplicate relations and obtained 413 distinct relations (see examples in Table 1).

We evaluated a sample of randomly selected 205 relations (i.e. half of the total) in three parallel evaluations performed by three of the authors of this

Extracted Term	Ontology Concept	Relation	Relation Path
Contact	Person	$\sqsubseteq$	contact $\sqsubseteq$ representative $\sqsubseteq$ negotiator $\sqsubseteq$communicator$\sqsubseteq$ person
Business	Partnership	$\sqsubseteq$	business $\sqsubseteq$ partnership
Child	Person	$\sqsubseteq$	child $\sqsubseteq$ person

Table 1. Examples of relations derived by using WordNet.

paper. This manual evaluation[6], which is not part of our evolution framework, helped to identify those relations which we considered correct or false, as well as those for which we could not decide on a correctness value ("Don't know"). Our results are shown in Table 2. We computed a precision value for each evaluator, however, because there was a considerable variation between these, we decided to also compute a precision value on the sample on which they all agreed. Even though, because of the rather high disagreement level between evaluators (more than 50%), we cannot draw a generally valid conclusion from these values. Nevertheless, they already give us an indication that, even in the worst case scenario, more than half of the obtained relations would be correct. Moreover, this experiment helped us to identify typical incorrect relations that could be filtered out automatically. These will be discussed in Section 6.4.

	Evaluator 1	Evaluator 2	Evaluator 3	Agreed by all
Correct	106	137	132	76
False	96	53	73	26
Don't know	2	15	0	0
Precision	53 %	73 %	65 %	75 %

Table 2. Evaluation results for the relations derived from WordNet.

6.3 Evaluation Results for Scarlet

The Scarlet based relation discovery processed the 327 terms for which no relation has been found in WordNet. It identified 786 relations of different types (subsumption, disjointness, named relations) for 68 of these terms (see some examples in Table 3). Some of these relations were duplicates, as the same relation can often be derived from several online ontologies. Duplicate elimination lead to 478 distinct relations.

For the evaluation, we randomly selected 240 of the distinct relations (i.e. 50% of them). They were then evaluated in the same setting as the WordNet-based

[6] To our knowledge, there are no benchmarks of similar experimental data against which our results could be compared.

relations. Our results are shown in Table 4, where, as in the case of the WordNet-based relations, precision values were computed both individually and for the jointly agreed relations. These values were in the same ranges as for WordNet. One particular issue we faced here was the evaluation of the named relations. These proved difficult because the names of the relations did not always make their meanings clear. Different evaluators provided different interpretations for these and thus increased the disagreement levels. Therefore, again, we cannot provide a definitive conclusion of the performance of this particular algorithm. Nevertheless, each evaluator identified more correct than incorrect relations.

6.4 Error Analysis

One of the main goals of our experiment was to identify typical errors and to envisage ways to avoid them. We hereby describe some of our observations.

As already mentioned, in addition to the actual relation discovered between a new term and an ontology concept, our method also provides the path that lead to this relation, either in the WordNet synsets hierarchy or in the external ontology in the case of Scarlet. Related to that, a straightforward observation was that there seem to be a correlation between the length of this path and the correctness of the relation, i.e. relations derived form longer paths are more likely to be incorrect. To verify this intuition, we inspected groups of relations with different path lengths and for each computed the percentage of correct, false, unranked relations, as well as the relations on which an agreement was not reached. These results are shown in Table 5. As expected, we observe that the percentage of correct relations decreases for relations with longer paths (although, a similar observation cannot be derived for the incorrect relations). We also note that the percentages of relations which were not ranked and of those on which no agreement was reached are higher for relations established through a longer path. This indicates that relations generated from longer paths are more difficult to interpret, and so, may be less suitable for automatic integration.

Another observation was that several relations were derived for the *Thing* concept (e.g. $Lecturer \sqsubseteq Thing$). While these relations cannot be considered incorrect, they are of little relevance for the domain ontology, as they would not

No.	Extracted Term	Ontology Concept	Relation	Relation Path
1	Funding	Grant	$\sqsubseteq$	funding $\sqsubseteq$ grant
2	Region	Event	occurredIn	region $\sqsubseteq$ place $\leftarrow$occurredIn- event
3	Hour	Duration	$\sqsubseteq$	hour $\sqsubseteq$ duration
4	Broker	Person	isOccupationOf	broker -isOccupationOf$\rightarrow$ person
5	Lecturer	Book	editor	lecturer $\sqsubseteq$ academicStaff $\sqsubseteq$ employee $\sqsubseteq$ person$\leftarrow$editor-book
6	Innovation	Event	$\sqsubseteq$	innovation $\sqsubseteq$ activity $\sqsubseteq$ event

Table 3. Examples of relations discovered using Scarlet.

	Evaluator 1	**Evaluator 2**	**Evaluator 3**	**Agreed by all**
Correct	118	126	81	62
False	96	56	57	17
Don't know	11	47	102	8
Precision	56 %	70 %	59 %	79 %

Table 4. Evaluation results for the relations derived with Scarlet.

contribute in making it evolve in a useful way. Therefore, they should simply be discarded. Similarly, relations that contained in their path abstract concepts such as *Event*, *Individual* or *Resource* tended to be incorrect.

Relation Path Length	**True**	**False**	**Don't Know**	**No agreement**
1	33 %	10 %	0 %	58 %
2	26 %	8 %	4 %	64 %
3	30 %	5 %	5 %	63 %
4	23 %	10 %	2 %	67 %
5	20 %	3 %	9 %	69 %

Table 5. Correlation between the length of the path and the correctness of a relation.

Finally, during the evaluation we also identified a set of relations to concepts that are not relevant for the domain (e.g. death, doubt). While they sometimes lead to correct relations (e.g. $Death \sqsubseteq Event$), these were rather irrelevant for the domain and thus should be avoided. We concluded that it would be beneficial to include a filtering step that eliminates, prior to the relation discovery step, those terms which are less relevant for the base ontology.

6.5 Observations on Integrating Relations into the Base Ontology

A particularity of the use of Scarlet is that different relations are derived from different online ontologies, reflecting various perspectives and subscribing to different design decisions.

One side effect of exploring multiple knowledge sources is that the derived knowledge is sometimes redundant. Duplicates often appear when two or more ontologies state the same relation between two concepts. These are easy to eliminate for subsumption and disjoint relations, but become non-trivial for named relations.

Another side effect is that we can derive contradictory relations between the same pair of concepts originating from different ontologies. For example, between *Process* and *Event* we found three different relations: "disjoint", "subClassOf"

and "superClassOf". Such a case is a clear indication that at least one of the relations should be discarded, as they cannot be all integrated into the ontology.

As we mentioned previously, both our methods provide a relation as well as an inference path that lead to its derivation. This makes the integration with the base ontology easier as more information is available.

An interesting situation arises when a part of the path supporting the relation contradicts the base ontology. For example, the second relation in the path relating *Innovation* to *Event*, Row 6 of Table 3, contradicts the base ontology where *Event* and *Activity* are siblings. This is a nice illustration of how the base ontology can be used as a context for checking the validity of a relation. Indeed, we could envision a mechanism that increases the confidence value for those paths which have a high correlation with the ontology (i.e. when they "agree" at least on some parts).

In the process of matching a path to an ontology, we can encounter situations where some elements of the path only have a partial syntactic match with the labels of some ontology concepts. Referring to Row 5 of Table 3, some of the terms in the relation path connecting *Lecturer* to *Book* partially map to labels in the subsumption hierarchy of the base ontology:

$$LecturerInAcademia \sqsubseteq AcademicStaffMember \sqsubseteq HigherEducationalOrganizationEmployee \sqsubseteq EducationalEmployee \sqsubseteq Employee \sqsubseteq AffiliatedPerson \sqsubseteq Person$$

While our Jaro based matcher could not identify a match between *Lecturer* and *LecturerInAcademia*, this association can be done by taking into account the discovered path and the base ontology, therefore avoiding the addition of already existing concepts, and giving further indications on the way to integrate the discovered relations.

A final interesting observation relates to the appropriate abstraction level where a named relation should be added. We listed in Row 5 of Table 3 a relation path where *Lecturer* inherits a named relation to *Book* from its superclass, *Person*. Because *Person* also exists in the base ontology, we think that it is more appropriate to add the relation to this concept rather than to the more specific concept.

7 Related Work

As mentioned in the introduction, the work presented in this paper is mostly related to those ontology evolution approaches which aim to automate the process of incorporating potentially novel information into a base ontology. In particular DINO [14, 15] makes use of ontology alignment and agent-negotiation techniques to achieve such integration. These are then validated by a human user. Similarly, Dynamo [17] uses an adaptive multi-agent system architecture to evolve ontologies. The system considers the extracted entities as individual agents related to other entities (agents) through a certain relationship. Unfortunately, Dynamo

only generates concept hierarchies and its output depends on the order of the input data fed in the system. Unlike these systems, Evolva explores various background knowledge sources to reduce the amount of human intervention required for ontology evolution.

Similarly to Evolva, background knowledge has been used to successfully support various other tasks. Ontology matching techniques have been built to exploit such sources: WordNet [9], medical domain ontologies [3] or online ontologies [18]. Online ontologies have also been used for supporting the enrichment of folksonomy tagspaces [4] and ontology learning [1].

8 Conclusion and Future Work

Ontology evolution is a tedious and time consuming task, especially at the level of introducing new knowledge to the ontology. Most of current ontology evolution approaches rely on the ontology curator's expertise to come up with the right integration decisions. We discussed in this paper how background knowledge can support Evolva, our ontology evolution framework, for automating the process of relation discovery. Our technique is based on gradually harvesting background knowledge sources in order to exploit the most specific and easily accessible sources first and later investigate more generic but noisier sources. In our experiments, we explored WordNet and Semantic Web ontologies (through the Scarlet relation discovery engine).

Our relation discovery experiments using WordNet and Scarlet helped identifying various possible ways in which the overall quality of the process can be improved. First, it became evident that relations established with abstract concepts or concepts that are poorly related to the base ontology have a low relevance. We could avoid deriving such relations in the first place by simply maintaining a list of common abstract concepts (e.g. *Thing*, *Resource*) that should be avoided. Another approach would be to check the relevance of the terms with respect to the ontology, by measuring their co-occurence in Web documents with concepts from the base ontology.

In addition, the observation of the result of the relation discovery process can lead to the design of a number of heuristic-based methods to improve the general quality of the output. For example, we observed a correlation between the length of the path from which a relation was derived and its quality. Note, however, that more in-depth analysis is needed to verify such hypothesis.

Finally, and most interestingly, the base ontology itself can be used for validating the correctness of a relation. Indeed, an overlap between the statements in the path and the base ontology is an indication that the relation is likely to be correct, and, inversely, if contradictions exist between the path and the ontology, the relation should be discarded. We have also shown cases where during the integration of a path in the ontology, some concepts can be considered equivalent even if their labels only partially match at a syntactic level.

While the work presented in this paper has shown the feasibility of exploiting external background knowledge sources to automate, at least partially, the ontology evolution process, there are still a number of aspects that need to be considered to make this approach fully operational. First, the way different sources are combined (here, WordNet and Scarlet) should be better studied, so that the method could better benefit from the complementarity of local, well established knowledge bases and of dynamic, distributed and heterogeneous Semantic Web ontologies. While we use a linear combination of background knowledge sources in our experiment, it would be worth to test a parallel combination to assess its effect on precision. Moreover, we plan to enhance the matching and term anchoring process by introducing word sense disambiguation techniques [10] that should increase precision. Also, one element not considered in this paper concerns the computational performance of our approach to ontology evolution. Additional work is currently ongoing to optimize particularly complex components like Scarlet.

Acknowledgements

This work was funded by the NeOn project sponsored under EC grant number IST-FF6-027595.

References

1. H. Alani. Position Paper: Ontology Construction from Online Ontologies. In *Proc. of WWW*, 2006.
2. H. Alani, S. Harris, and B. O'Neil. Winnowing ontologies based on application use. *Proceedings of 3rd European Semantic Web Conference (ESWC-06), Montenegro*, 2006.
3. Z. Aleksovski, M. Klein, W. ten Katen, and F. van Harmelen. Matching Unstructured Vocabularies using a Background Ontology. In S. Staab and V. Svatek, editors, *Proc. of EKAW*, LNAI. Springer-Verlag, 2006.
4. S. Angeletou, M. Sabou, L. Specia, and E. Motta. Bridging the Gap Between Folksonomies and the Semantic Web: An Experience Report. In *Proc. of the ESWC Workshop on Bridging the Gap between Semantic Web and Web 2.0*, 2007.
5. S. Bloehdorn, P. Haase, Y. Sure, and J. Voelker. *Ontology Evolution*, pages 51–70. John Wiley & Sons, June 2006.
6. P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. *Proceedings of the 13th international conference on World Wide Web*, pages 462–471, 2004.
7. P. Cimiano and J. Völker. Text2Onto - a framework for ontology learning and data-driven change discovery. *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB-05), Alicante*, pages 15–17, 2005.
8. W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.

9. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Discovering Missing Background Knowledge in Ontology Matching. In *Proc. of ECAI*, 2006.
10. J. Gracia, V. Lopez, M. d'Aquin, M. Sabou, E. Motta, and E. Mena. Solving semantic ambiguity to improve semantic web based ontology matching. *Workshop on Ontology Matching, The 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference*, 2007.
11. P. Haase and L. Stojanovic. Consistent evolution of owl ontologies. *Proceedings of the Second European Semantic Web Conference, Heraklion, Greece, 2005*, pages 182–197, 2005.
12. M. Klein. *Change Management for Distributed Ontologies*. PhD thesis, Vrije Universiteit in Amsterdam, 2004.
13. Y. Lei, M. Sabou, V. Lopez, J. Zhu, V. Uren, and E. Motta. An infrastructure for acquiring high quality semantic metadata. *Proceedings of the 3rd European Semantic Web Conference*, 2006.
14. V. Novacek, L. Laera, and S. Handschuh. Semi-automatic integration of learned ontologies into a collaborative framework. *International Workshop on Ontology Dynamics (IWOD-07)*, 2007.
15. V. Novacek, L. Laera, S. Handschuh, J. Zemanek, M. Volkel, R. Bendaoud, M. R. Hacene, Y. Toussaint, B. Delecroix, and A. Napoli. D2. 3.8 v2 report and prototype of dynamics in the ontology lifecycle. 2008.
16. N. F. Noy, A. Chugh, W. Liu, and M. A. Musen. A framework for ontology evolution in collaborative environments. *Proceedings of the 5th Int. Semantic Web Conference (ISWC-06). Athens, Georgia, USA*, pages 544–558, 2006.
17. K. Ottens and P. Glize. A multi-agent system for building dynamic ontologies. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 2007.
18. M. Sabou, M. d'Aquin, and E. Motta. Exploring the Semantic Web as Background Knowledge for Ontology Matching. *Journal on Data Semantics*, XI, 2008.
19. L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, FZI - Research Center for Information Technologies at the University of Karslruhe, 2004.
20. D. Vrandecic, H. S. Pinto, Y. Sure, and C. Tempich. The diligent knowledge processes. *Journal of Knowledge Management*, 9:85–96, 2005.
21. Z. Wu and M. Palmer. Verb semantics and lexical selection. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, 1994.
22. F. Zablith. Dynamic ontology evolution. *To appear in: Proceedings of the ISWC Doctoral Consortium*, 2008.

Supporting the Evolution of $\mathcal{SHIQ}$ Ontologies with Inductive Logic Programming

A Preliminary Study

Francesca A. Lisi and Floriana Esposito

Dipartimento di Informatica, Università degli Studi di Bari
Via E. Orabona 4, 70125 Bari, Italy
{lisi, esposito}@di.uniba.it

Abstract. The definition of new concepts or roles for which extensional knowledge become available can turn out to be necessary to make a DL ontology evolve. In this paper we reformulate this task as a machine learning problem and study a solution based on techniques borrowed from that form of logic-based machine learning known under the name of Inductive Logic Programming (ILP). More precisely, we propose to adapt previous ILP results to the knowledge representation framework of $\mathcal{DL}$+log in order to learn rules to be used for changing $\mathcal{SHIQ}$ ontologies.

1 Introduction

Encoding of human knowledge in ontologies using logical formalisms, e.g. Description Logics (DLs) [1], is one of the crucial tasks to be performed towards the realization of the vision of the Semantic Web. Actually building ontologies is simply the first step because ontologies, just like any structure holding knowledge, need to be maintained as well. Ontology Evolution is the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology-based application, as well as the consistent management/propagation of these changes to dependent elements [28]. As opposite to Ontology Modification, Ontology Evolution is accommodated when an ontology management system facilitates the modification of an ontology by preserving its consistency. To define change operations for ontologies is not easy, because one has to take into account all the possible effects a change can have on the components of an ontology. According to [21] one can distinguish between three kinds of changes:

- Conceptual changes, i.e. changes in the conceptualisation itself, for example changing concepts relations;
- Specification changes, i.e. changes in the specifications of the conceptualisations, for example to add new properties to a concept;
- Representation changes, i.e. changes in the representation of the conceptualisation, for example by using another ontology representation language.

Noy and Klein [21] define also a set of change operations for ontologies considering the effects on the compatibility between two versions of an ontology. E.g., the creation of a class/slot is a lossless change operation because no data is lost.

In this paper we consider the conceptual changes of a DL ontology due to extensional knowledge (e.g., facts of the instance level of the ontology) previously unknown but classified which may become available. In particular, we consider the task of defining new concepts or roles which provide the intensional counterpart of this extensional knowledge. One such task, if adequately reformulated, can be partially automated by applying machine learning algorithms. E.g., the following new facts concerning known individuals

```
LONER(Joe)
LONER(Mary)
LONER(Paul)
```

may raise the need for having a definition of the concept `LONER` in the ontology. One such definition can be learned from these facts together with prior knowledge about Joe, Mary and Paul, i.e. facts concerning them and already available in the ontology. An important requirement is that the definition must be expressed as a DL formula or similar.

The use of prior or background knowledge (BK) during the learning process is a distinguishing feature of Inductive Logic Programming (ILP) [20]. ILP has been historically concerned with the induction of rules from examples for classification purposes. Unfortunately, ILP adopts the Knowledge Representation (KR) framework of Logic Programming, i.e. Clausal Logics (CLs) [15], which differ from DLs in several respects. Yet, KR hybrid formalisms exist that combine DLs and CLs. Among the many recent KR proposals, $\mathcal{DL}$+log [24] is a very powerful framework that allows for the tight integration of DLs and disjunctive DATALOG with negation (DATALOG$^{\neg\vee}$) [7]. A point in favour of $\mathcal{DL}$+log is its decidability for many DLs, notably for $\mathcal{SHIQ}$ [12]. Since the design of OWL has been based on the $\mathcal{SH}$ family of very expressive DLs [11], $\mathcal{SHIQ}$+log is a good candidate for investigation in the (Semantic) Web context. In this paper, we consider a decidable instantiation of $\mathcal{DL}$+log obtained by choosing $\mathcal{SHIQ}$ for the DL part and DATALOG$^{\neg}$ for the CL part, denoted as $\mathcal{SHIQ}$+log$^{\neg}$, and adapt ILP techniques to $\mathcal{SHIQ}$+log$^{\neg}$ in order to learn rules to be used for changing $\mathcal{SHIQ}$ ontologies.

The paper is organized as follows. Section 2 briefly describes the techniques commonly used in ILP. Section 3 introduces the KR framework of $\mathcal{DL}$+log. Section 4 states the learning problem of interest, defines the core ingredients of an ILP solution to it and sketches an application scenario in Ontology Evolution. Section 5 concludes the paper with final remarks.

2 Learning with Inductive Logic Programming

Inductive Logic Programming (ILP) was born at the intersection of Inductive Learning, more precisely Concept Learning [17], and Logic Programming [15].

From Logic Programming it has borrowed the KR framework, i.e. Horn Clausal Logic. From Concept Learning it has inherited the inferential mechanisms for induction, the most prominent of which is *generalization* characterized as search through a partially ordered space of hypotheses [17]. According to this vision, in ILP a hypothesis is a clausal theory (i.e., a set of rules) and the induction of a single clause (rule) requires (i) structuring, (ii) searching and (iii) bounding the space of hypotheses.

First we focus on (i) by clarifying the notion of *ordering* for clauses. An ordering allows for determining which one, between two clauses, is more general than the other. Actually quasi-orders are considered, therefore uncomparable pairs of clauses are admitted. One such ordering is *θ-subsumption* [22]: Given two clauses C and D, we say that C θ-subsumes D if there exists a substitution θ, such that $C\theta \subseteq D$. Given the usefulness of BK, orders have been proposed that reckon with it, e.g. Buntine's *generalized subsumption* [2]. Generalized subsumption only applies to definite clauses and the BK should be a definite program.

Once structured, the space of hypotheses can be searched (ii) by means of refinement operators. The definition of refinement operators presupposes the investigation of the properties of the various quasi-orders. In Shapiro's sense [27], a refinement operator is a function which computes a set of specializations of a clause. Specialization is suited for the direction of search in his approach. His kind of refinement operator has been therefore called a *downward* refinement operator in ILP. Dually, operators can be also defined to compute generalizations of clauses. These can be applied in a bottom-up search, so they have been named *upward* refinement operators.

The definition of refinement operators is usually coupled with the specification of a declarative bias for bounding the space of clauses (iii). *Bias* concerns anything which constrains the search for theories [29]. Following [19] we will distinguish three kinds of bias: (a) *Language bias* that specifies constraints on the clauses in the search space; (b) *Search bias* that has to do with the way an ILP system searches its space of permitted clauses; (c) *Validation bias* that concerns the stopping criterion of the ILP system.

Induction with ILP generalizes from individual instances/observations in the presence of background knowledge, finding *valid hypotheses*. Validity depends on the underlying *setting*. At present, there exist several formalizations of induction in clausal logic that can be classified according to the following two orthogonal dimensions: the *scope of induction* (discrimination vs characterization) and the *representation of observations* (ground definite clauses vs ground unit clauses) [4]. *Discriminant induction* aims at inducing hypotheses with discriminant power as required in tasks such as classification. In classification, observations encompass both positive and negative examples. *Characteristic induction* is more suitable for finding regularities in a data set. This corresponds to learning from positive examples only. For a thorough discussion of differences between discriminant and characteristic induction see [16]. The second dimension affects the notion of *coverage*, i.e. the condition under which a hypothesis explains an observation. In *learning from entailment* (also called *learning from implications*), hypotheses are

clausal theories, observations are ground definite clauses, and a hypothesis covers an observation if the hypothesis logically entails the observation. In *learning from interpretations*, hypotheses are clausal theories, observations are Herbrand interpretations (ground unit clauses) and a hypothesis covers an observation if the observation is a model for the hypothesis. A deeper investigation of learning from entailment and learning from interpretations can be found in [3].

3 Representing in $\mathcal{DL}$+log

The KR framework of $\mathcal{DL}$+log [24] allows for the tight integration of DLs [1] and DATALOG$^{\neg\vee}$ [7]. More precisely, it allows a DL knowledge base (DL-KB) to be extended with *weakly-safe* DATALOG$^{\neg\vee}$ rules. Note that the condition of weak safeness allows to overcome the main representational limits of the approaches based on the DL-safeness condition, e.g. the possibility of expressing conjunctive queries and unions of conjunctive queries, by keeping the integration scheme still decidable. $\mathcal{DL}$+log follows the tradition of KR research on *hybrid systems*, i.e. those systems which are constituted by two or more subsystems dealing with distinct portions of a single KB by performing specific reasoning procedures [8]. The motivation for investigating and developing such systems is to improve on two basic features of KR formalisms, namely *representational adequacy* and *deductive power*, by preserving the other crucial feature, i.e. *decidability*. In particular, combining DLs and CLs can easily yield to undecidability if the interface between them is not reduced. The resulting KR systems will be referred to as DL-CL hybrid systems in the rest of the paper. Examples of those systems are $\mathcal{AL}$-log [6] and CARIN [13].

3.1 Syntax

We start from three mutually disjoint predicate alphabets:

- an alphabet of concept names P_C;
- an alphabet of role names P_R;
- an alphabet of DATALOG predicates P_D.

We call a predicate p a *DL-predicate* if either $p \in P_C$ or $p \in P_R$. Then, we denote by $\mathcal{C}$ a countably infinite alphabet of constant names. An *atom* is an expression of the form $p(X)$, where p is a predicate of arity n and X is a n-tuple of variables and constants. If no variable symbol occurs in X, then $p(X)$ is called a *ground atom* (or *fact*). If $p \in P_C \cup P_R$, the atom is called a *DL-atom*, while if $p \in P_D$, it is called a DATALOG *atom*.

Given a description logic $\mathcal{DL}$, a $\mathcal{DL}$- KB with weakly-safe DATALOG$^{\neg\vee}$ rules ($\mathcal{DL}$+log-KB for short) $\mathcal{B}$ is a pair (Σ, Π), where:

- Σ is a $\mathcal{DL}$-KB, i.e., a pair $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is the TBox and $\mathcal{A}$ is the ABox;
- Π is a set of DATALOG$^{\neg\vee}$ rules, where each rule R has the form

$$p_1(\boldsymbol{X_1}) \vee \ldots \vee p_n(\boldsymbol{X_n}) \leftarrow$$
$$r_1(\boldsymbol{Y_1}), \ldots, r_m(\boldsymbol{Y_m}), s_1(\boldsymbol{Z_1}), \ldots, s_k(\boldsymbol{Z_k}), \neg u_1(\boldsymbol{W_1}), \ldots, \neg u_h(\boldsymbol{W_h})$$

$n \geq 0$, $m \geq 0$, $k \geq 0$, $h \geq 0$, each $p_i(\boldsymbol{X_i})$, $r_j(\boldsymbol{Y_j})$, $s_l(\boldsymbol{Z_l})$, $u_k(\boldsymbol{W_k})$ is an atom and:

- each p_i is either a DL-predicate or a DATALOG predicate;
- each r_j, u_k is a DATALOG predicate;
- each s_l is a DL-predicate;
- (DATALOG safeness) every variable occurring in R must appear in at least one of the atoms $r_1(\boldsymbol{Y_1}), \ldots, r_m(\boldsymbol{Y_m}), s_1(\boldsymbol{Z_1}), \ldots, s_k(\boldsymbol{Z_k})$;
- (weak safeness) every head variable of R must appear in at least one of the atoms $r_1(\boldsymbol{Y_1}), \ldots, r_m(\boldsymbol{Y_m})$.

We remark that the above notion of weak safeness allows for the presence of variables that only occur in DL-atoms in the body of R. On the other hand, the notion of DL-safeness of variables adopted in previous approaches [18,23] can be expressed as follows: every variable of R must appear in at least one of the atoms $r_1(\boldsymbol{Y_1}), \ldots, r_m(\boldsymbol{Y_m})$. Therefore, DL-safeness forces every variable of R to occur also in the DATALOG atoms in the body of R, while weak safeness allows for the presence of variables that only occur in DL-atoms in the body of R. Without loss of generality, we can assume that in a $\mathcal{DL}$+log-KB (Σ, Π) all constants occurring in Σ also occur in Π.

3.2 Semantics

For $\mathcal{DL}$+log two semantics have been defined: a first-order logic (FOL) semantics and a nonmonotonic (NM) semantics. In particular, the latter extends the stable model semantics of DATALOG$^{\neg\vee}$ [9].

According to the NM semantics, DL-predicates are still interpreted under the classical open-world assumption (OWA), while DATALOG predicates are interpreted under a closed-world assumption (CWA). Notice that, both under the FOL semantics and the NM semantics, entailment can be reduced to satisfiability, since it is possible to express constraints in the DATALOG program. More precisely, under both semantics, it is immediate to verify that (Σ, Π) entails $p(c)$ iff $(\Sigma, \Pi \cup \{\leftarrow p(c)\})$ is unsatisfiable. In a similar way, it can be seen that *conjunctive query answering* can be reduced to satisfiability in $\mathcal{DL}$+log. Consequently, Rosati [24] concentrates on the satisfiability problem in $\mathcal{DL}$+log-KBs.

It has been shown that, when the rules are positive disjunctive, i.e., there are no negated atoms in the bodies of rules, the above two semantics are equivalent with respect to the satisfiability problem. In particular, FOL-satisfiability can always be reduced (in linear time) to NM-satisfiability. Hence, the satisfiability problem under the NM semantics is in the focus of interest.

3.3 Reasoning

The problem statement of satisfiability for finite $\mathcal{DL}$+log-KBs requires some preliminary definitions. We start by introducing Boolean conjunctive queries

(CQs) and Boolean unions of conjunctive queries (UCQs), and the containment problem for such queries. A *Boolean UCQ* over a predicate alphabet P is a first-order sentence of the form $\exists \boldsymbol{X}.conj_1(\boldsymbol{X}) \vee \ldots \vee conj_n(\boldsymbol{X})$, where $\boldsymbol{X}$ is a tuple of variable symbols and each $conj_i(\boldsymbol{X})$ is a set of atoms whose predicates are in P and whose arguments are either constants or variables from $\boldsymbol{X}$. A *Boolean CQ* corresponds to a Boolean UCQ in the case when $n = 1$. Given a $\mathcal{DL}$-TBox $\mathcal{T}$, a Boolean CQ Q_1 and a Boolean UCQ Q_2 over the alphabet $P_C \cup P_R$, Q_1 is contained in Q_2 with respect to $\mathcal{T}$, denoted by $\mathcal{T} \models Q_1 \subseteq Q_2$, iff, for every model $\mathcal{I}$ of $\mathcal{T}$, if Q_1 is satisfied in $\mathcal{I}$ then Q_2 is satisfied in $\mathcal{I}$. In the following, we call the problem of deciding $\mathcal{T} \models Q_1 \subseteq Q_2$ the *Boolean CQ/UCQ containment problem.*[1]

Besides the Boolean CQ/UCQ containment problem, it is important to clarify how the grounding operation used in stable model semantics is adapted to the $\mathcal{DL}$+log case. Given a $\mathcal{DL}$+log KB $\mathcal{B} = (\Sigma, \Pi)$, the DL-grounding of Π, denoted as $gr_p(\Pi)$, is a set of Boolean CQs. Note that $gr_p(\Pi)$ constitutes a partial grounding of the conjunctions of DL-atoms that occur in Π with respect to the constants in $\mathcal{C}_\Pi$, since the variables that only occur in DL-atoms in the body of rules are not replaced by constants in $gr_p(\Pi)$.

The algorithm NMSAT-$\mathcal{DL}$+log for deciding NM-satisfiability of $\mathcal{DL}$+log-KBs has a very simple structure, since it decides satisfiability by looking for a guess (G_P, G_N) of the Boolean CQs in $gr_p(\Pi)$ that is consistent with the $\mathcal{DL}$-KB Σ (Boolean CQ/UCQ containment problem) and such that the DATALOG$^{\neg\vee}$ program $\Pi(G_P, G_N)$ has a stable model. Notice that $\Pi(G_P, G_N)$ is a ground DATALOG$^{\neg\vee}$ program over P_D, i.e. no DL-predicate occurs in such a program. The decidability of reasoning in $\mathcal{DL}$+log depends on the decidability of the Boolean CQ/UCQ containment problem in $\mathcal{DL}$.

Theorem 1 *[24] For every description logic $\mathcal{DL}$, satisfiability of $\mathcal{DL}$+log-KBs (both under FOL semantics and under NM semantics) is decidable iff Boolean CQ/UCQ containment is decidable in $\mathcal{DL}$.*

The decidability of ground query answering follows from Theorem 1.

Corollary 1. *Given a $\mathcal{DL}$+log KB (Σ, Π) and a ground atom α, $(\Sigma, \Pi) \models \alpha$ iff $(\Sigma, \Pi \cup \{\leftarrow \alpha\})$ is unsatisfiable.*

Thus ground queries can be answered by means of the abovementioned algorithm NMSAT-$\mathcal{DL}$+log for deciding NM-satisfiability of $\mathcal{DL}$+log-KBs.

From Theorem 1 and from previous results on query answering and query containment in DLs, we are able to state decidability of reasoning in several instantiations of $\mathcal{DL}$+log. In particular, for the DL $\mathcal{SHIQ}$ it is known that Boolean CQ/UCQ containment is decidable [10]. Since $\mathcal{SHIQ}$ is the most expressive DL for which this property has been proved, we consider $\mathcal{SHIQ}$+log$^{\neg}$ (i.e. $\mathcal{SHIQ}$ extended with weakly-safe DATALOG$^{\neg}$ rules) as the KR framework in our preliminary study of an ILP framework for ontology evolution.

[1] This problem was called *existential entailment* in [13].

4 Learning Concepts and Roles in $\mathcal{SHIQ}$+log$^\neg$ with ILP

4.1 The problem statement

We consider the problem of inducing rule-based definitions of concepts/roles that do not occur in an existing $\mathcal{SHIQ}$ ontology. At this stage of work the scope of induction does not matter. Therefore the term 'observation' is to be preferred to the term 'example'.

Definition 1. *Let $\mathcal{B}$ a $\mathcal{SHIQ}$+log$^\neg$ KB composed of a $\mathcal{SHIQ}$ ontology and a* DATALOG$^\neg$ *program.*
***Given**:*

- *a new target $\mathcal{SHIQ}$ predicate name p*
- *a set O of observations for p*
- *a language $\mathcal{L}$ of hypotheses*

***Build** a hypothesis $\mathcal{H} \in \mathcal{L}$ for p such that $\mathcal{B} \cup \mathcal{H}$ is correct w.r.t. O.*

We assume that the intensional part $\mathcal{K}$ (i.e., the TBox $\mathcal{T}$ plus the set Π_R of rules) of $\mathcal{B}$ plays the role of BK and the extensional part $\mathcal{F}$ (i.e., the ABox $\mathcal{A}$ plus the set Π_F of facts) contributes to the definition of observations. We choose to work within the setting of *learning from interpretations* [5] which requires an observation to be represented as a set of ground unit clauses.

Example 1. Suppose we have a $\mathcal{SHIQ}$+log$^\neg$ KB (adapted from [24]) consisting of the following intensional knowledge $\mathcal{K}$:

$[A1]$ `RICH`$\sqcap$`UNMARRIED` $\sqsubseteq \exists$ `WANTS-TO-MARRY`$^-$.$\top$
$[R1]$ `RICH(X)` $\leftarrow$ `famous(X)`, $\neg$ `scientist(X)`
$[R2]$ `happy(X)` $\leftarrow$ `famous(X)`, `WANTS-TO-MARRY(Y,X)`

and the following extensional knowledge $\mathcal{F}$:

```
UNMARRIED(Mary)
UNMARRIED(Joe)
famous(Mary)
famous(Paul)
famous(Joe)
scientist(Joe)
```

that can be split into $\mathcal{F}_{\texttt{Joe}} = \{\texttt{UNMARRIED(Joe)}, \texttt{famous(Joe)}, \texttt{scientist(Joe)}\}$, $\mathcal{F}_{\texttt{Mary}} = \{\texttt{UNMARRIED(Mary)}, \texttt{famous(Mary)}\}$, and $\mathcal{F}_{\texttt{Paul}} = \{\texttt{famous(Paul)}\}$. Note that $[R2]$ is weakly-safe but not DL-safe because the variable `Y` does not occur in any DATALOG literal of $[R2]$.

The language $\mathcal{L}$ of hypotheses must allow for the generation of $\mathcal{SHIQ}$+log$^\neg$ rules starting from three disjoint alphabets $P_C(\mathcal{L}) \subseteq P_C(\mathcal{B})$, $P_R(\mathcal{L}) \subseteq P_R(\mathcal{B})$,

and $P_D(\mathcal{L}) \subseteq P_D(\mathcal{B})$. More precisely, we consider linked[2] and range-restricted[3] weakly-safe DATALOG$^\neg$ clauses of the form

$$p(\boldsymbol{X}) \leftarrow r_1(\boldsymbol{Y_1}), \ldots, r_m(\boldsymbol{Y_m}), s_1(\boldsymbol{Z_1}), \ldots, s_k(\boldsymbol{Z_k}), \neg u_1(\boldsymbol{W_1}), \ldots, \neg u_h(\boldsymbol{W_h})$$

where $m \geq 0$, $k \geq 0$, $h \geq 0$, each $p(\boldsymbol{X})$, $r_j(\boldsymbol{Y_j})$, $s_l(\boldsymbol{Z_l})$, $u_k(\boldsymbol{W_k})$ is an atom and:

- p is a $\mathcal{SHIQ}$-predicate;
- each r_j, u_k is a DATALOG-predicate;
- each s_l is a $\mathcal{SHIQ}$-predicate.

Note that the literal $p(\boldsymbol{X})$ in the head represents the target predicate, i.e. the predicate to be defined by learned $\mathcal{SHIQ}$+log$^\neg$ rules.

Example 2. Suppose that the target predicate is the $\mathcal{SHIQ}$-concept `LONER`. If $\mathcal{L}^{\texttt{LONER}}$ is defined over $P_D(\mathcal{L}^{\texttt{LONER}}) \cup P_C(\mathcal{L}^{\texttt{LONER}}) = \{\texttt{famous/1}, \texttt{scientist/1}\} \cup \{\texttt{UNMARRIED/1}\}$, then the following $\mathcal{SHIQ}$+log$^\neg$ rules

$H_1^{\texttt{LONER}}$ `LONER(X)` $\leftarrow$ `scientist(X)`
$H_2^{\texttt{LONER}}$ `LONER(X)` $\leftarrow$ `scientist(X)`, `UNMARRIED(X)`
$H_3^{\texttt{LONER}}$ `LONER(X)` $\leftarrow$ `UNMARRIED(X)`
$H_4^{\texttt{LONER}}$ `LONER(X)` $\leftarrow \neg$`famous(X)`

belong to $\mathcal{L}^{\texttt{LONER}}$ and represent hypotheses of definition for `LONER`.

Example 3. Suppose now that the $\mathcal{SHIQ}$-role `LIKES` is the target predicate and the set $P_D(\mathcal{L}^{\texttt{LIKES}}) \cup P_C(\mathcal{L}^{\texttt{LIKES}}) \cup P_R(\mathcal{L}^{\texttt{LIKES}}) = \{\texttt{happy/1}\} \cup \{\texttt{RICH/1}\} \cup \{\texttt{WANTS-TO-MARRY/2}\}$ provides the building blocks for the language $\mathcal{L}^{\texttt{LIKES}}$. The following $\mathcal{SHIQ}$+log$^\neg$ rules

$H_1^{\texttt{LIKES}}$ `LIKES(X,Y)` $\leftarrow$ `WANTS-TO-MARRY(X,Y)`
$H_2^{\texttt{LIKES}}$ `LIKES(X,Y)` $\leftarrow$ `WANTS-TO-MARRY(X,Y)`, `happy(X)`
$H_3^{\texttt{LIKES}}$ `LIKES(X,Y)` $\leftarrow$ `WANTS-TO-MARRY(X,Y)`, `RICH(Y)`
$H_4^{\texttt{LIKES}}$ `LIKES(X,Y)` $\leftarrow$ `happy(X)`, `RICH(Y)`

belonging to $\mathcal{L}^{\texttt{LIKES}}$ can be considered hypotheses of definition for `LIKES`.

Note that a hypothesis $\mathcal{H}$ may consist of more than one $\mathcal{SHIQ}$+log$^\neg$ rule. Also $\mathcal{H}$ is valid as a solution to the learning problem in hand if it changes the input ontology by keeping it consistent. This requirement is guaranteed by the correcteness condition in Definition 1.

[2] Let H be a clause. A term t in some literal $l_i \in H$ is *linked* with linking-chain of length 0, if t occurs in $head(H)$, and is linked with linking-chain of length $d+1$, if some other term in l_i is linked with linking-chain of length d. The link-depth of a term t in some $l_i \in H$ is the length of the shortest linking-chain of t. A literal $l_i \in H$ is linked if at least one of its terms is linked.

[3] A clause H is *range-restricted* or *connected* if each variable occurring in $head(H)$ also occur in $body(H)$.

4.2 The ingredients for an ILP solution

In order to solve with ILP techniques the learning problem in hand, the language $\mathcal{L}$ of hypotheses needs to be equipped with:

- a *generality* order $\succeq$, and
- a *coverage* relation *covers*

so that $(\mathcal{L}, \succeq)$ is a search space and *covers* defines the mappings from $(\mathcal{L}, \succeq)$ to the set O of observations.

A generality order for $\mathcal{SHIQ}$+log$^\neg$ rules The definition of a generality order for hypotheses in $\mathcal{L}$ can disregard neither the peculiarities of $\mathcal{SHIQ}$+log$^\neg$ nor the methodological apparatus of ILP. One issue arises from the presence of NAF literals (i.e., negated DATALOG literals) both in the background knowledge and in the language of hypotheses. As pointed out in [26], rules in normal logic programs are syntactically regarded as Horn clauses by viewing the NAF-literal $\neg p(X)$ as an atom $not\_p(X)$ with the new predicate not_p. Then any result obtained on Horn logic programs is directly carried over to normal logic programs. Assuming one such treatment of NAF literals, we propose to adapt generalized subsumption [2] to the case of $\mathcal{SHIQ}$+log$^\neg$ rules. The resulting generality relation will be called $\mathcal{K}$-*subsumption*, briefly $\succeq_\mathcal{K}$, from now on. We provide a characterization of $\succeq_\mathcal{K}$ that relies on the reasoning tasks known for $\mathcal{DL}$+log and from which a test procedure can be derived.

Definition 2. *Let $H_1, H_2 \in \mathcal{L}$ be two hypotheses standardized apart, $\mathcal{K}$ a background knowledge, and σ a Skolem substitution*[4] *for H_2 with respect to $\{H_1\} \cup \mathcal{K}$. We say that $H_1 \succeq_\mathcal{K} H_2$ iff there exists a ground substitution θ for H_1 such that (i) $head(H_1)\theta = head(H_2)\sigma$ and (ii) $\mathcal{K} \cup body(H_2)\sigma \models body(H_1)\theta$.*

Note that condition (ii) is a variant of the Boolean CQ/UCQ containment problem because $body(H_2)\sigma$ and $body(H_1)\theta$ are both Boolean CQs. The difference between (ii) and the original formulation of the problem is that $\mathcal{K}$ encompasses not only a TBox but also a set of rules. Nonetheless this variant can be reduced to the satisfiability problem for finite $\mathcal{SHIQ}$+log$^\neg$ KBs. Indeed the skolemization of $body(H_2)$ allows to reduce the Boolean CQ/UCQ containment problem to a CQ answering problem. Due to the aforementioned link between CQ answering and satisfiability, checking (ii) can be reformulated as proving that the KB $(\mathcal{T}, \Pi_R \cup body(H_2)\sigma \cup \{\leftarrow body(H_1)\theta\})$ is unsatisfiable. Once reformulated this way, (ii) can be solved by applying the algorithm NMSAT-$\mathcal{DL}$+log.

Example 4. Let us consider the hypotheses

[4] Let $\mathcal{B}$ be a clausal theory and H be a clause. Let $X_1, \ldots, X_n$ be all the variables appearing in H, and $a_1, \ldots, a_n$ be distinct constants (individuals) not appearing in $\mathcal{B}$ or H. Then the substitution $\{X_1/a_1, \ldots, X_n/a_n\}$ is called a *Skolem substitution* for H w.r.t. $\mathcal{B}$.

$H_1^{\texttt{LONER}}$ `LONER(A)` $\leftarrow$ `scientist(A)`
$H_2^{\texttt{LONER}}$ `LONER(X)` $\leftarrow$ `scientist(X),UNMARRIED(X)`

reported in Example 2 up to variable renaming. We want to check whether $H_1^{\texttt{LONER}} \succeq_{\mathcal{K}} H_2^{\texttt{LONER}}$ holds. Let $\sigma = \{\texttt{X/a}\}$ a Skolem substitution for $H_2^{\texttt{LONER}}$ with respect to $\mathcal{K} \cup H_1^{\texttt{LONER}}$ and $\theta = \{\texttt{A/a}\}$ a ground substitution for $H_1^{\texttt{LONER}}$. The condition (i) is immediately verified. The condition

$$\text{(ii) } \mathcal{K} \cup \{\texttt{scientist(a)}, \texttt{UNMARRIED(a)}\} \models \{\texttt{scientist(a)}\}$$

is a ground query answering problem in $\mathcal{SHIQ}$+log. It can be easily proved that all NM-models for $\mathcal{K} \cup \{\texttt{scientist(a)}, \texttt{UNMARRIED(a)}\}$ satisfy `scientist(a)`. Thus, $H_1^{\texttt{LONER}} \succeq_{\mathcal{K}} H_2^{\texttt{LONER}}$. The viceversa does not hold. Also, $H_3^{\texttt{LONER}} \succ_{\mathcal{K}} H_2^{\texttt{LONER}}$ (i.e. strictly more general than) and $H_4^{\texttt{LONER}}$ is incomparable with all the first three hypotheses.

Example 5. With reference to Example 3, it can be proved that $H_1^{\texttt{LIKES}} \succ_{\mathcal{K}} H_2^{\texttt{LIKES}}$ and $H_1^{\texttt{LIKES}} \succ_{\mathcal{K}} H_3^{\texttt{LIKES}}$. Conversely, the hypotheses $H_2^{\texttt{LIKES}}$, $H_3^{\texttt{LIKES}}$, and $H_4^{\texttt{LIKES}}$ are incomparable under $\mathcal{K}$-subsumption.

It is straightforward to see that the decidability of $\mathcal{K}$-subsumption follows from the decidability of $\mathcal{SHIQ}$+log$^{\neg}$. It can be proved that $\succeq_{\mathcal{K}}$ is a quasi-order (i.e. it is a reflexive and transitive relation) for $\mathcal{SHIQ}$+log$^{\neg}$ rules, therefore the space of hypotheses can be searched by refinement operators.

A coverage relation for $\mathcal{SHIQ}$+log$^{\neg}$ rules The definition of a coverage relation depends on the representation choice for observations. An observation $o_i \in O$ is represented as a couple $(p(\boldsymbol{a_i}), \mathcal{F}_i)$ where p is the target $\mathcal{SHIQ}$ predicate, $\boldsymbol{a_i}$ is a tuple of individuals occurring in the ABox $\mathcal{A}$ and $\mathcal{F}_i$ is a set containing ground facts concerning individuals in $\boldsymbol{a_i}$. Note that when p is a $\mathcal{SHIQ}$ role name, the tuple $\boldsymbol{a_i}$ is a pair $< a_i^1, a_i^2 >$ of individuals and the set $\mathcal{F}_i$ is given by the union of $\mathcal{F}_i^1$ and $\mathcal{F}_i^2$. We assume $\mathcal{K} \cap O = \emptyset$.

Definition 3. *Let $H \in \mathcal{L}$ be a hypothesis, $\mathcal{K}$ a background knowledge and $o_i \in O$ an observation. We say that H* covers *o_i* under interpretations w.r.t. *$\mathcal{K}$ iff $\mathcal{K} \cup \mathcal{F}_i \cup H \models p(\boldsymbol{a_i})$.*

Therefore the coverage test can be reduced to query answering in $\mathcal{SHIQ}$+log$^{\neg}$ KBs which in its turn can be reformulated as a satisfiability problem of the KB.

Example 6. With reference to Example 2, the hypothesis $H_1^{\texttt{LONER}}$ covers the observation $o_{\texttt{Joe}} = (\texttt{LONER(Joe)}, \mathcal{F}_{\texttt{Joe}})$ because all NM-models for $\mathcal{B} = \mathcal{K} \cup \mathcal{F}_{\texttt{Joe}} \cup H_1^{\texttt{LONER}}$ do satisfy `scientist(Joe)`. Note that it does not cover the observations $o_{\texttt{Paul}} = (\texttt{LONER(Paul)}, \mathcal{F}_{\texttt{Paul}})$ and $o_{\texttt{Mary}} = (\texttt{LONER(Mary)}, \mathcal{F}_{\texttt{Mary}})$. The hypothesis $H_2^{\texttt{LONER}}$ convers only $o_{\texttt{Joe}}$ for analogous reasons. It can be proved that $H_3^{\texttt{LONER}}$ covers $o_{\texttt{Mary}}$ and $o_{\texttt{Joe}}$ while $H_4^{\texttt{LONER}}$ none of the three observations.

Example 7. None of the hypotheses $H_1^{\texttt{LIKES}}$, $H_2^{\texttt{LIKES}}$, and $H_3^{\texttt{LIKES}}$ reported in Example 3 cover observations concerning couples of known individuals. Conversely, $H_4^{\texttt{LIKES}}$ covers the observation $o_{\texttt{<Mary,Paul>}} = (\texttt{LIKES(Mary,Paul)}, \mathcal{F}_{\texttt{Mary}} \cup \mathcal{F}_{\texttt{Paul}})$ because all NM-models for $\mathcal{B} = \mathcal{K} \cup \mathcal{F}_{\texttt{Mary}} \cup \mathcal{F}_{\texttt{Paul}} \cup H_4^{\texttt{LIKES}}$ satisfy:

- `happy(Mary)`, due to the axiom A1 and to the rule R2. Indeed, since from A1 $\exists$`WANTS-TO-MARRY`$^{-}$.$\top$`(Mary)` holds in every model of $\mathcal{B}$, it follows that in every model there exists a constant `x` such that `WANTS-TO-MARRY(x,Mary)` holds in the model, consequently from rule R1 it follows that `happy(Mary)` also holds in the model;
- `RICH(Paul)`, since the default rule R1 is always applicable for `Paul`.

Note that $H_4^{\texttt{LIKES}}$ covers also $o_{\texttt{<Mary,Mary>}} = (\texttt{LIKES(Mary,Mary)}, \mathcal{F}_{\texttt{Mary}})$.

4.3 A proof-of-concept application scenario in Ontology Evolution

The ingredients identified in the previous section are the starting point for the definition of ILP algorithms, that once implemented, can support the evolution of ontologies. Before clarifying how, we remind the reader that - according to [28] - the ontology evolution process is composed of the following six phases:

1. *Change capturing*: This phase encapsulates the process of deciding to apply a change on an ontology. This might be forced by explicit requirements (the ontology engineer decides to make a change) or by results of automatic change discovery methods. The first kind of changes are called top-down changes, whereas the second one are called bottom-up changes. Bottom-up changes can be proposed by three different approaches to change discovery: structure-driven, data-driven, and usage-driven change discovery.
2. *Change representation*: In order to resolve changes, they should be identified and represented clearly and in a suitable format. They can be represented as elementary or complex changes.
3. *Semantic of changes*: How a change can affect the ontologys consistency must be understood in advance, whereas the meaning of consistency depends on the underlying ontology model.
4. *Change propagation*: To preserve consistency, affected artefacts should be handled appropriately as well. In a distributed environment, affected artefacts are not bound to local components of the changed ontology, but contain also distributed ontologies that reuse or extend the changed ontology, or even applications that are based on the changed ontology.
5. *Change implementation*: Before applying a change, all implications of it have to be presented to the user, who then can accept or discard it. If the user agrees with the changes, all activities to apply the change have to performed.
6. *Change validation*: It should be possible for a user to validate performed changes and to reverse the effects of them when necessary.

We argue that the phases 1.-3. are crucial from our point of view. Indeed change capturing (1.) provides the target predicate and the observations for one or more learning problems of the form as in Definition 1. Once captured this way, each change is represented (2.) as the hypothesis inductively generated according to Definition 1. A particular attention must then be paid to the semantics of those changes (3.) that contain NAF literals because they can affect the ontology consistency. Indeed the change operations considered in this paper, i.e. the

creation of a concept and the creation of a role, both boil down to the addition of new rules to the input $\mathcal{SHIQ}$+log$^{\neg}$ KB as illustrated in the following example.

Example 8. Let us suppose that for the concept `LONER` we have $o_{\texttt{Joe}}$ as a positive example and $o_{\texttt{Mary}}$ and $o_{\texttt{Paul}}$ as negative examples. From this set of observations, an ILP algorithm implementing the ingredients identified in Section 4.2 and adopting a top-down strategy can induce $H_1^{\texttt{LONER}}$ as the hypothesis of rule-based definition for `LONER` because it covers all positive examples and none of the negative examples w.r.t. the BK of Example 1. Conversely, if $o_{\texttt{Joe}}$ and $o_{\texttt{Mary}}$ are both positive examples and $o_{\texttt{Paul}}$ is the only negative example for `LONER`, the hypothesis $H_3^{\texttt{LONER}}$ will be returned.

Let us now suppose that $o_{\texttt{<Mary,Paul>}}$ and $o_{\texttt{<Mary,Mary>}}$ are positive examples for the role `LIKES` and any other observation is considered as negative example. In this case, the hypothesis $H_4^{\texttt{LIKES}}$ is the inductively correct definition of `LIKES`.

5 Final Remarks

In this paper, we have proposed an ILP framework built upon a decidable instantiation , $\mathcal{SHIQ}$+log$^{\neg}$, of the most powerful KR framework currently available for the integration of DLs and CLs. Indeed, well-known ILP techniques for induction have been reformulated in terms of the deductive reasoning mechanims of $\mathcal{DL}$+log. Notably, we have defined a decidable generality ordering, $\mathcal{K}$-subsumption, for $\mathcal{SHIQ}$+log$^{\neg}$ rules on the basis of the decidable algorithm NMSAT-$\mathcal{SHIQ}$+log. We would like to point out that the ILP framework proposed is suitable for supporting the evolution of a $\mathcal{SHIQ}$ ontology for two main reasons. First, it induces rules with a $\mathcal{SHIQ}$ predicate in the head. Second, it can deal with incomplete knowledge, thus coping with a more plausible scenario of ontology evolution. Though the work presented in this paper can be considered as a feasibility study, it provides the principles for learning in $\mathcal{SHIQ}$+log$^{\neg}$. We would like to emphasize that they will be still valid for any other upcoming decidable instantiation of $\mathcal{DL}$+log, provided that DATALOG$^{\neg}$ is still considered for the CL part.

The ILP framework presented in this paper differs from the related proposals [25] and [14] in several respects, notably the following ones. First, it relies on a more expressive DL (i.e., $\mathcal{SHIQ}$). Second, it allows for inducing definitions for new DL concepts (i.e., rules with a $\mathcal{SHIQ}$ literal in the head). Third, it relies on a more expressive yet decidable CL (i.e., DATALOG$^{\neg}$). Forth, it adopts a tighter form of integration between the DL part and the CL part of rules (i.e., the weakly-safe one). Similarities also emerge such as the definition of a *semantic* generality relation for hypotheses in order to accommodate ontologies in ILP. Note that generalized subsumption is chosen for adaptation because in all three ILP frameworks definite clauses, though enriched with DL literals and NAF literals, are still used.

As next step towards any practice, we plan to define ILP algorithms starting from the ingredients identified in this paper. Also we would like to investigate

the impact of having DATALOG$^{\neg\vee}$ both in the language of hypotheses and in the language for the background theory. The inclusion of the nonmonotonic features of $\mathcal{SHIQ}$+log *full* will strengthen the ability of our ILP framework to deal with incomplete knowledge by performing an inductive form of commonsense reasoning. One such ability can turn out to be useful in application domains, such as the Semantic Web, and complementary to reasoning with uncertainty and under inconsistency. Finally, we would like to study in more depth the phase of change validation in our approach to Ontology Evolution.

Acknowledgements We are grateful to Riccardo Rosati for his precious advice on $\mathcal{DL}$+log and Diego Calvanese for his valid support on the Boolean CQ/UCQ containment problem.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.
3. L. De Raedt. Logical Settings for Concept-Learning. *Artificial Intelligence*, 95(1):187–201, 1997.
4. L. De Raedt and L. Dehaspe. Clausal Discovery. *Machine Learning*, 26(2–3):99–146, 1997.
5. L. De Raedt and S. Džeroski. First order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.
6. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
7. T. Eiter, G. Gottlob, and H. Mannila. Disjunctive DATALOG. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.
8. A.M. Frisch and A.G. Cohn. Thoughts and afterthoughts on the 1988 workshop on principles of hybrid reasoning. *AI Magazine*, 11(5):84–87, 1991.
9. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.
10. B. Glimm, I. Horrocks, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. *Journal of Artificial Intelligence Research*, 31:151–198, 2008.
11. I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
12. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.
13. A.Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.
14. F.A. Lisi. Building Rules on Top of Ontologies for the Semantic Web with Inductive Logic Programming. *Theory and Practice of Logic Programming*, 8(03):271–300, 2008.
15. J.W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd edition, 1987.

16. R.S. Michalski. A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an artificial intelligence approach*, volume I. Morgan Kaufmann, San Mateo, CA, 1983.
17. T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
18. B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. *Journal on Web Semantics*, 3(1):41–60, 2005.
19. C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. Declarative bias in ILP. In L. De Raedt, editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS Press, 1996.
20. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.
21. N. Fridman Noy and M.C.A. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004.
22. G.D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.
23. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1), 2005.
24. R. Rosati. $\mathcal{DL}$+log: Tight integration of description logics and disjunctive datalog. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *Proc. of Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 68–78. AAAI Press, 2006.
25. C. Rouveirol and V. Ventos. Towards Learning in CARIN-$\mathcal{ALN}$. In J. Cussens and A. Frisch, editors, *Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 191–208. Springer, 2000.
26. C. Sakama. Nonmonotonic inductive logic programming. In T. Eiter, W. Faber, and M. Truszczynski, editors, *Logic Programming and Nonmonotonic Reasoning*, volume 2173 of *Lecture Notes in Computer Science*, pages 62–80. Springer, 2001.
27. E.Y. Shapiro. Inductive inference of theories from facts. Technical Report 624, Dept. of Computer Science, Yale University, 1981.
28. L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. User-driven ontology evolution management. In A. Gómez-Pérez and V. Richard Benjamins, editors, *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, volume 2473 of *Lecture Notes in Computer Science*, pages 285–300. Springer, 2002.
29. P.E. Utgoff and T.M. Mitchell. Acquisition of appropriate bias for inductive concept learning. In *Proceedings of the 2nd National Conference on Artificial Intelligence*, pages 414–418, Los Altos, CA, 1982. Morgan Kaufmann.

The 7th International Semantic Web Conference

October 26 – 30, 2008

Congress Center, Karlsruhe, Germany