

# Aspects of Automatic Ontology Extension: Adapting and Regeneralizing Dynamic Updates

Ekaterina Ovchinnikova<sup>1</sup>

Kai-Uwe Kühnberger<sup>2</sup>

<sup>1</sup> Seminar für Sprachwissenschaft  
University of Tübingen  
72074 Tübingen, Wilhelmstr. 19, Germany  
Email: e.ovchinnikova@gmail.com

<sup>2</sup>Institute of Cognitive Science  
University of Osnabrück  
49076 Osnabrück, Albrechtstr. 12, Germany  
Email: kkuehnbe@uos.de

## Abstract

Ontologies are widely used in text technology and artificial intelligence. The need to develop large ontologies for real-life applications provokes researchers to automatize ontology extension procedures. Automatic updates without the control of a human expert can generate potential conflicts between original and new knowledge. As a consequence the resulting ontology can contain inconsistencies. On the other hand, even if the information extracted from the external sources automatically is consistent with the original ontology it can be generalized unsystematically and conceptually wrong. This in turn can lead to mistakes in applications of the extended ontology. We propose an algorithm that models the process of the adaptation of an ontology to new information and regeneralizes the resulting ontology in a more intuitive way inserting additional knowledge where this is possible.

## 1 Introduction

There is an increasing interest in applying and using ontological knowledge in artificial intelligence. Examples for applications of ontologies in AI are expert systems, dialogue systems, robotics, reasoning systems, web services, and text technological tools. In general, knowledge-based systems are prototypical examples for using and applying ontological knowledge. The interested reader is referred to [www.cs.utexas.edu/users/mfkb/related.html](http://www.cs.utexas.edu/users/mfkb/related.html), where a long list of different knowledge-based systems and ontology projects can be found.

An important motivation for research in ontology design is the fact that inference processes can be made more efficient. For example, an ontology with a subsumption relation based on a many-sorted logic allows to restrict inferences to those rules that are in accordance to the sortal constraints. A classical (and famous) application in the field of theorem proving is the steamroller problem (Walter 1985) where the number of clauses that are necessary to solve the problem can be significantly reduced by introducing hierarchical constraints on these sorts. Besides such technical aspects, there is a further very general reason for the endeavor to develop models for ontological

systems: ontologies are still one of the few possibilities to explore the hard problem, whether machines can assign meanings to symbols, i.e. whether machines can develop an important aspect of human-level intelligence.

The most important new development motivating many researchers on focusing on ontologies is the omnipresence of the world wide web together with its numerous applications and its economic importance. It is often claimed that ontological (i.e. semantic) knowledge about domains of interest is one of the most important steps in order to develop new and intelligent web applications (Berners-Lee, Hendler & Lassila 2001). Examples for such services are intelligent search tools for large archives of multi-modal information, multi-modal resources for artificial agents and personal assistants (that are permanently connected with the internet), or intelligent document management tools for libraries and companies. But also e-commerce applications, the development of portals, or geospatial applications could benefit from ontological knowledge.

Since the manual development of large ontologies has been proven to be a very tedious, time-consuming and expensive task, automatic procedures for semantic annotations of relevant resources (texts and web content) and the possibility to automatically adapt and extend such ontologies would be desirable. Therefore one can find many current investigations that are devoted towards a development of automatic ontology learning methods (Gómez-Pérez & Manzano-Macho 2003).

During the last decades several formalisms have been proposed to represent ontological knowledge. In recent years the world wide web and its connection to various economically important applications has been provided the environment for dynamic developments in representation language standards. Probably the most important one of existing markup languages for ontology design is the Web Ontology Language *OWL* (OWL 2004) in its three different versions *OWL Lite*, *OWL DL*, and *OWL Full* (W3C 2004). The mentioned OWL versions are hierarchically ordered, such that *OWL Full* includes *OWL DL*, and *OWL DL* includes *OWL Lite*. Consequently they differ in their expressive strengths with respect to possible concept formations.

All versions of OWL are based on the logical formalism called Description Logic DL (Baader et al. 2003). Description logics were originally designed for the representation of terminological knowledge and reasoning processes. They can be characterized as subsystems of first-order predicate logic using at most two variables. Two points should be mentioned:

- In comparison to full first-order logic, description logics are – due to their restrictions concerning quantification – rather weak logics with respect to their expressive strength. Nevertheless they are considered as appropriate representation formalisms for ontological knowledge.
- DL can be used to characterize the different OWL versions. For example, *OWL DL* can be logically characterized as a syntactic variant of the description logic *SHOIN(D)* (Motik, Sattler & Studer 2004). As a consequence of the clear logical foundation of the OWL versions using description logics, important formal properties of the different OWL versions can be specified, for example, their decidability properties: whereas *OWL Full* is undecidable (due to the lack of restrictions to transitive properties), *OWL DL* and *OWL Lite* are decidable.

Although most of the tools extracting or extending ontologies automatically output the knowledge in the OWL-format, they usually use only a small subset of the underlying description logic. Core ontologies generated in practice consist of a set of concepts, the subsumption relation defined on concepts (inducing a taxonomy) and general relations (such as part-of) defined on concepts. At present complex ontologies making use of the whole expressive power and advances of the various versions of description logics can be achieved only manually or semi-automatically. Disadvantages of manual or semi-automatic applications of knowledge representation formalisms are costs (expensive and time-consuming).

However, several approaches appeared recently tending not only to learn taxonomic and general relations but also to state which concepts in the knowledge base are equivalent or disjoint (Haase 2005). In the present paper, we concentrate on these approaches. We will consider only terminological knowledge (called TBox in DL) leaving the information about assertions in the knowledge base (called ABox in DL) for the further investigation.<sup>1</sup>

Approaches of automatic ontology learning and automatic ontology extension – in particular if they are based on rather expressive logics – are often faced with the so-called generalization problem.<sup>2</sup> The appropriate level of granularity of an underlying ontology is usually hard to achieve. Two major problems can be distinguished:

- Inappropriate generalizations of concepts can lead, in the worst case, to inconsistencies if ontologies are automatically extended. Assume a concept  $C$  in the ontology was overgeneralized, then a new axiom that must be added to the ontology – due to new available information – can represent an exception towards  $C$  and can conflict with its definition. In this case  $C$  is too coarse. Resolving inconsistencies in logic based systems is well-known to be a hard problem.
- The undergeneralization of concepts in an ontology does not provoke inconsistencies, but it can lead to a loss of information by an ontology application (Ceusters et al. 2003). In this case, the underlying concept must be generalized because in its original form it is too fine-grained.

In this paper, we provide an overview of the mentioned generalization problems in ontologies automatically learned from external sources. Sections 3 and 4

<sup>1</sup>The formal definition of terminological knowledge coded in a TBox is stated in Section 2.

<sup>2</sup>For a motivation compare (Haase et al. 2005).

discuss the overgeneralization problem, whereas Section 5 deals with the undergeneralization problem. We give algorithmic solutions for both problems, in particular we specify how to resolve occurring contradictions and get additional knowledge where this is possible, and how regeneralizations of an ontology can be achieved if some underlying concepts is too fine-grained.

The paper has the following structure: In Section 2, we roughly summarize some important definitions of the syntax and semantics of description logics and we introduce the notion of least common subsumer. Section 3 starts with a rough summary of classical existing approaches to model inconsistent information and presents some intuitive ideas how occurring inconsistencies in ontology extension processes can be resolved. In Section 4, we present the algorithm *AdaptOnto* that allows the extension of ontologies with inconsistent information by an adaptation process. Section 5 addresses the generalization problem in consistent ontologies and proposes an algorithmic solution of this problem by the algorithm *Regen* as well as the prototype implementation. Last but not least, Section 6 adds some remarks concerning semantic issues and Section 7 concludes the paper.

## 2 Description Logic

### 2.1 Basic Definitions

In this section, we define the DL-logic underlying the ontological knowledge representation considered in this paper.<sup>3</sup> For a detailed presentation and an overview of various description logics, including their syntax and semantics, the reader is referred to (Baader et al. 2003).

Given a set of concept names  $N_C$  and a set of role names  $N_R$  a *TBox* (terminological box) is a finite set of axioms of the form  $A_1 \equiv A_2$  (equalities) or  $A \sqsubseteq C$  (inclusions) where  $A$  stands for a concept name and  $C$  (called *concept description*) is defined as follows ( $R$  denotes a role name):

$$C \rightarrow A \mid \neg A \mid \forall R.A$$

The symbol  $\doteq$  denotes the syntactical equality of concept descriptions. The concepts occurring on the left side of an axiom are called *axiomatized* ( $ax$ ). In an axiom with a concept  $A$  on the left side the concept on its right side is called *definition* of  $A$ .

Concept descriptions are interpreted in a classical model-theoretic sense (for details compare (Baader et al. 2003)). An *interpretation*  $\mathcal{I}$  is a pair  $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$  where  $\Delta^{\mathcal{I}}$  is a non-empty domain of individuals and the interpretation function  $\cdot^{\mathcal{I}}$  maps concept names to subsets of  $\Delta^{\mathcal{I}}$  and role names to subsets of  $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ . Concept descriptions are interpreted as follows:

$$\begin{aligned} (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\ (\forall R.A)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y. \langle x, y \rangle \in R^{\mathcal{I}} \rightarrow y \in A^{\mathcal{I}}\} \end{aligned}$$

An interpretation  $\mathcal{I}$  is a *model* of a TBox  $\mathcal{T}$  if for every inclusion  $A \sqsubseteq C$  in  $\mathcal{T}$  it holds  $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$  and for every equality  $A_1 \equiv A_2$  we have  $A_1^{\mathcal{I}} = A_2^{\mathcal{I}}$ . A concept description  $D$  *subsumes*  $C$  in  $\mathcal{T}$  (formally represented by  $\mathcal{T} \models C \sqsubseteq D$ ) if for every model  $\mathcal{I}$  of  $\mathcal{T}$ :  $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ . A concept  $C$  is called *satisfiable towards*  $\mathcal{T}$  if there is a model  $\mathcal{I}$  of  $\mathcal{T}$  such that  $C^{\mathcal{I}}$  is nonempty. Otherwise  $C$  is called *unsatisfiable* and it holds  $\mathcal{T} \models C \sqsubseteq \perp$ .

<sup>3</sup>In the following definitions we closely follow (Haase et al. 2005) who present an approach using one of the most powerful DL-version in ontology learning procedure.

Algorithms for checking satisfiability of concept descriptions have been implemented in several reasoning systems.<sup>4</sup> For example, the FaCT system implements subsumption for a very expressive DL  $\mathcal{SHIQ}$  (Horrocks 1998). Most of these systems are based on Tableau calculi. The idea is to use facts about the world (coded in the ABox) in order to construct a model for these facts (relative to the given TBox). The construction is usually performed by so-called *expansion rules* decomposing underlying concepts until no further application of a rule is possible or a contradiction is reached. It should be noted that these reasoners usually use the well-known correspondence between the subsumption  $C \sqsubseteq D$  and the unsatisfiability of  $C \sqcap \neg D$ .

## 2.2 Least Common Subsumers

Recent research in description logics is strongly concerned with *non-standard inferences* (Baader & Küsters 2006) tending to support bottom-up constructions of knowledge bases. A knowledge engineer introduces typical examples of a new concept and the system tries to find commonalities between them and generalizes it in a definition.

An important task for generalizing a new concept is the computation of the *least common subsumer* for a set of concepts (first mentioned in (Cohen et al. 1993)). Intuitively, the least common subsumer (lcs) for two concept descriptions  $C_1$  and  $C_2$  is a concept description that collects all common features of  $C_1$  and  $C_2$  and is most specific towards subsumption.

**Definition 1** *A concept description  $E$  of DL  $\mathcal{L}$  is a least common subsumer (lcs) of the concept descriptions  $C_1, \dots, C_n$  in  $\mathcal{L}$  ( $\text{lcs}_{\mathcal{L}}(C_1, \dots, C_n)$  for short) if and only if the following two conditions are satisfied:*

1.  $\forall i \in \{1, \dots, n\} : C_i \sqsubseteq E$  and
2.  $\forall E' \in \mathcal{L} : \text{if } \forall i \in \{1, \dots, n\} : C_i \sqsubseteq E' \text{ then } E \sqsubseteq E'$

There are algorithms for computing lcs for different DL logics (Baader & Küsters 2006). All of these algorithms work with logics allowing at least the existence of a top element. Because of the specificity of the logic considered in this paper (no conjunction, no top and bottom elements, multiple definitions for one concept) we define the set of the least common subsumers *lcss* for the set of concept descriptions  $C_1, \dots, C_n$  towards a TBox  $\mathcal{T}$  slightly differently from the usual way.

First of all, let us recursively define the function *subsumers* computing the set of all possible subsumers (formulated in the DL under consideration) for a concept  $C$  towards a TBox  $\mathcal{T}$ :

$$\begin{aligned} \text{subsumers}_{\mathcal{T}}(C) = \{ & D \mid D \doteq C \vee C \sqsubseteq D \in \mathcal{T} \vee \\ & \exists D' : C \sqsubseteq D' \in \mathcal{T} \wedge D \in \text{subsumers}_{\mathcal{T}}(D') \} \cup \\ & \{ \neg A \mid \exists A' : C \doteq \neg A' \wedge A \in \text{subsumers}_{\mathcal{T}}(A') \} \cup \\ & \{ \forall R.A \mid \exists A' : C \doteq \forall R.A' \wedge A' \in \text{subsumers}_{\mathcal{T}}(A') \} \end{aligned}$$

The definition can be summarized as follows: For every concept description  $C$  the set  $\text{subsumers}_{\mathcal{T}}(C)$  contains the following elements:

- (a)  $C$  itself;
- (b) Concept descriptions occurring on the right side of the axioms in  $\mathcal{T}$  that axiomatize  $C$ ;
- (c) Concept descriptions that subsume the concept descriptions from (b) (the subsumption relation is transitive).

If  $C$  is a negated atomic concept ( $C \doteq \neg A'$ ), then the set  $\text{subsumers}_{\mathcal{T}}(C)$  collects the negated subconcepts of  $A'$ . Since general axioms (with concept descriptions on the left side) are disallowed in our logic, atomic concepts can subsume only atomic concepts and no negations or value restrictions. Every value restriction  $\forall R.A'$  is subsumed only by the relational restrictions  $\forall R.A$  where  $A$  subsumes  $A'$ .

According to this constructive definition the set *subsumers* is exhaustive in our DL as stated explicitly in Fact 1:

**Fact 1** *For every TBox  $\mathcal{T}$ , for all concept descriptions  $C, C'$ :*

$$C' \in \text{subsumers}_{\mathcal{T}}(C) \Leftrightarrow \mathcal{T} \models C \sqsubseteq C'$$

Now we define the function *lcss* computing the set of least common subsumers for a set of concept descriptions  $C_1, \dots, C_n$  towards a TBox  $\mathcal{T}$  according to Definition 1.

**Definition 2** *A set  $L$  is the set of least common subsumers of the concept descriptions  $C_1, \dots, C_n$  towards the TBox  $\mathcal{T}$  ( $\text{lcss}_{\mathcal{T}}(C_1, \dots, C_n)$ ) if and only if it is specified as follows:*

1.  $CS = \bigcap_{i \in \{1, \dots, n\}} \text{subsumers}_{\mathcal{T}}(C_i)$  and
2.  $L = \{ C \in CS \mid \forall C' \in CS : \mathcal{T} \models C' \sqsubseteq C \rightarrow C' \doteq C \}$

In the following sections, we show how the notion of the least common subsumer can be used for regeneralizing a TBox.

## 3 Ontology Extension: Inconsistencies

### 3.1 Classical Approaches for Inconsistent Information

Inconsistencies occurring in reasoning processes do have a long history in artificial intelligence. Due to the fact that many approaches in AI are based on one or the other form of classical logic and inconsistencies, for example, triggered by new information added to a knowledge base, cannot be easily treated in classical logic, researchers proposed many approaches to solve this problem. The difficulties in modeling inconsistencies in classical logic are strongly connected to the monotonicity property of logic. This property can be described as follows: for all sets  $\Delta$  of first-order formulas and all first-order formulas  $\phi$  and  $\psi$  it holds:

$$\text{if } \Delta \vdash \phi \text{ then } \Delta \cup \psi \vdash \phi$$

In an application, based on classical logic, an update of the set of premises  $\Delta$  with  $\psi$  such that  $\psi \leftrightarrow \neg\phi$  the consequence  $\phi$  is still provable, contrary to the intuition that we have evidence for  $\neg\phi$ . In order to avoid this type of conclusion, so-called non-monotonic reasoning techniques were developed and extensively discussed in the literature (Bibel et al. 1993). We mention three prominent approaches that were proposed to model non-monotonicity.

- Default logic (Reiter 1980): A default theory  $\langle W, \Delta \rangle$  distinguishes two types of rules.  $W$  represents a world description, i.e. strict background knowledge, whereas  $\Delta$  denotes a set of defaults, representing revisable information. Intuitively (and very simplified) this means: if we have no evidence that a formula  $\neg\theta$  is true, then assume that  $\theta$  holds.

<sup>4</sup>Some of the DL reasoners are listed at <http://www.cs.man.ac.uk/~sattler/reasoners.html>.

- Answer set programming (Baral 2003): A natural idea of modeling inconsistencies is to introduce a ranking order on rules that can be applied in reasoning systems. Intuitively more specific rules are higher ranked than very general rules, i.e. general rules can be overwritten (revised) by specific information.
- Circumscription (Lifschitz 1994): Based on the idea of logical minimization, the circumscription of a predicate  $P$  relative to a world description  $W$  means that there is no other predicate  $P'$  such that  $W$  still holds and the extension of  $P'$  is strictly smaller than the extension of  $P$ . In other words,  $P$  is minimal with respect to  $W$ .

The listed approaches represent only a few examples of theories that were proposed to model inconsistencies and non-monotonicity. Nevertheless no generally accepted solution for non-monotonic reasoning seems to be available.

### 3.2 Inconsistencies and Ontologies

We want to consider inconsistencies in ontologies more closely. An ontology based on description logic can contain contradictions only if its underlying logic allows negation. Ontologies share this property with every logical system (like, for example, first-order logic). For the approaches concerned with core ontologies no contradictions in the ontological knowledge base is possible. But for approaches using more powerful logics, the problem of inconsistency becomes very important (Haase et al. 2005). In order to make the notion of inconsistency of a TBox precise we give the following definition.

**Definition 3** A TBox  $\mathcal{T}$  is inconsistent if there exist a concept  $C \in ax(\mathcal{T})$  that is unsatisfiable.

A number of approaches have been proposed treating inconsistencies by extending the underlying description logic with additional syntactical means. Some examples are extensions by default sets (Heymans & Vermeir 2002), by planning systems (Baader et al. 2005), by belief-revision processes (Flouris et al. 2005), or by epistemic operators (Katz & Parsi 2005). Unfortunately, these approaches are beyond ordinary description logics, i.e. they cannot be coded in standard versions of description logic. Therefore the application of classical DL reasoners is impossible due to the fact that standard DL inferences cannot be performed.

There are several theoretical approaches treating occurring inconsistencies for quite expressive DL-logics, but – contrary to the cases above – do not go beyond description logic. The following list summarizes some of these approaches:

- (Ghilardi et al. 2006) suggests a characterization of non-conservative extensions of an ontology: If a concept description is satisfiable prior to an extension, but becomes unsatisfiable after the extension, then a witness concept description demonstrating this fact will be suggested to the ontology engineer. Occurring inconsistencies caused by an ontology extension can be considered as a special case of a non-conservative extension. This approach does not provide a general solution of the inconsistency problem, but can help the human expert to discover occurring inconsistencies in certain cases.
- In (Fanizzi et al. 2005), the authors propose an ontology refinement procedure based on positive

and negative assertions for concepts. If a concept  $C$  becomes unsatisfiable after an ontology extension, then the axiom defining  $C$  is replaced by a new axiom constructed on the basis of the positive assertions for this concept. Thus, the information previously defined in the TBox for the concept  $C$  gets lost.

- (Ovchinnikova & Kühnberger 2006a) introduce a procedure automatically changing the original ontology if it conflicts with new information. The changes in the conflicting axioms are performed in order to achieve a resulting ontology that is consistent. Additionally these changes can be interpreted as an adaptation process, amalgamating previous knowledge to new data.

The listed approaches (and many others dealing with non-monotonic and non-conservative extensions) are concerned with quite expressive DL-logics and tend to support a semi-automatic development of ontologies. The situation with automatic ontology learning seems to be different:

- First, there is no ontology engineer who supervises the procedure.
- Second, the changes in the ontology are supposed to be relevant and possibly minimal.
- Third, the axioms extracted automatically from the external sources can be inconsistent.
- Finally, the underlying logic tends to be rather weak with respect to its expressive power.

In the next subsection, we consider some types of examples for which an automatic adaptation process can be implemented.

### 3.3 Resolving Occurring Inconsistencies in Automatic Ontology Learning

An interesting approach towards an automatic ontology learning procedure was proposed in (Haase et al. 2005). If the ontology under consideration is provably inconsistent, then one or more axioms must be deleted from this ontology. The axioms to be deleted are chosen according to a confidence rating. This rating is computed on the basis of the term distribution in texts used for learning.

Intuitively plausible is the consequence that in some cases the removal of a whole axiom can lead to a loss of relevant information. We consider an example to make this point clear.

TBox:  $\{\text{Bird} \sqsubseteq \text{Flying}, \text{Flying} \sqsubseteq \text{Moving}, \text{Canary} \sqsubseteq \text{Bird}, \text{Penguin} \sqsubseteq \text{Bird}\}$   
 New axiom:  $\text{Penguin} \sqsubseteq \neg \text{Flying}$

By removing the information *birds fly* we will obtain the proper generalization, but lose the knowledge that all birds considered before the ontology extension can fly (such as *Canary* - the subconcept of *Bird*) and that all birds can move. We propose the following solution of how to adapt the ontology to the new information *penguin cannot fly*.

Adapted TBox:  
 $\{\text{Bird} \sqsubseteq \text{Moving}, \text{Flying} \sqsubseteq \text{Moving}, \text{FlyingBird} \sqsubseteq \text{Bird}, \text{FlyingBird} \sqsubseteq \text{Flying}, \text{Canary} \sqsubseteq \text{FlyingBird}, \text{Penguin} \sqsubseteq \text{Bird}, \text{Penguin} \sqsubseteq \neg \text{Flying}\}$

The proposed solution is simple: We want to keep in the definition of the concept *Bird* subsuming *Penguin* a maximum of information that does not

conflict with the definition of **Penguin**. The conflicting information is moved to the definition of the new concept **FlyingBird**, which is declared to subsume all former subconcepts of **Bird** (such as **Canary** for example).

Presupposing that the new axioms extracted from external sources in order to be added to the ontology contain true information we conclude that the inconsistency is provoked by overgeneralized concepts. In the example above, the statement *all birds fly* is too general for the described update, namely the introduction of a counterexample.

The example below represents a case where two overgeneralized definitions of the same concept conflict with each other:

TBox:  $\{\text{Tomato} \sqsubseteq \forall \text{hasColor.Red},$   
 $\text{Red} \sqsubseteq \text{Color}, \text{Red} \sqsubseteq \neg \text{Yellow},$   
 $\text{Yellow} \sqsubseteq \text{Color}, \text{Yellow} \sqsubseteq \neg \text{Red}\}$   
 New axiom:  $\text{Tomato} \sqsubseteq \forall \text{hasColor.Yellow}$   
 Adapted TBox:  
 $\{\text{Tomato} \sqsubseteq \forall \text{hasColor.Color}, \text{Red} \sqsubseteq \text{Color},$   
 $\text{Red} \sqsubseteq \neg \text{Yellow}, \text{Yellow} \sqsubseteq \text{Color}, \text{Yellow} \sqsubseteq \neg \text{Red}\}$

In the example above, both definitions of **Tomato** ( $\forall \text{hasColor.Red}$  and  $\forall \text{hasColor.Yellow}$ ) are too specific. **Red** and **Yellow** being disjoint concepts produce a conflict. It seems to be an intuitive solution to replace these concepts by their least common subsumer **Color**. Furthermore it is plausible to claim that all tomatoes have color without specifying this color precisely.

Unfortunately, not all types of inconsistency can be resolved automatically. For axioms of the form  $A \sqsubseteq D$  and  $A \sqsubseteq \neg D$  no other alternative can be found to guarantee consistency, except to removing one of the problematic axioms. Without appealing to external knowledge (such as the confidence rating of the axioms) one cannot decide which axiom need to be removed.

We want to generalize the examples discussed so far. If a concept  $X$  is defined in the TBox  $T$  by the axioms  $X \sqsubseteq A$  and  $X \sqsubseteq B$  such that  $A$  conflicts with  $B$  in  $T$ , then the following options can be distinguished:

1.  $A$  and  $B$  are disjoint concept descriptions having common subsumers (the tomato example given above):  
 The solution in this case is to replace the axioms  $X \sqsubseteq A, X \sqsubseteq B$  with the corresponding definitions taken from the set  $lc_{ss_T}(A, B)$ .
2.  $A \in ax(T)$  and some definition  $D$  of  $A$  conflicts with  $B$  (the penguin example given above):  
 This case can be considered as the overgeneralization of  $A$ , because the concept  $X$  being subconcept of  $A$  represents an exception towards the definition  $D$ . The definition  $D$  must be revised as follows: a)  $D$  is replaced with its most specific superconcepts that do not conflict with  $B$ ; if there is no such concept then  $A \sqsubseteq D$  is just deleted; b) a new concept  $A'$  is added to the TBox as a subconcept of  $A$  and  $D$ ;  $A$  is replaced with  $A'$  in the definition of all its subconcepts except in the definition of  $X$ .
3.  $A, B \in ax(T)$ , a definition  $D_A$  of  $A$  conflicts with  $B$ , and a definition  $D_B$  of  $B$  conflicts with  $A$ :  
 In this case there is no automatic logical solution. Any of the definitions ( $D_A$  or  $D_B$ ) can be changed in the way described in the previous option (2) in order to achieve a consistent ontology. A confidence rating as suggested in (Haase et al. 2005) can be used for a selection of the axioms to be changed. For example, a function

$r_{conf} : Concepts \rightarrow \mathbb{R}$  can be used to assign a confidence rating to every concept. In this paper, we do not discuss this rating in more detail.

4. Otherwise:

One of the conflicting axioms ( $X \sqsubseteq A$  or  $X \sqsubseteq B$ ) must be removed from the TBox. A confidence rating as suggested in (Haase et al. 2005) can be used for the selection of the relevant axiom to be removed.

In order to formalize the procedure described above we need to introduce the notion of a *conflict* in the following definition.

**Definition 4** For every TBox  $T$ , two concept descriptions  $C_1$  and  $C_2$  conflict with each other in  $T$  if and only if for every model  $\mathcal{M}$  of  $T$ :

$$C_1^{\mathcal{M}} \cap C_2^{\mathcal{M}} = \emptyset$$

If the underlying logic allows conjunction (what is not true for our logic), then the definition above is equivalent to the following statement: two concept descriptions  $C_1$  and  $C_2$  conflict with each other in TBox  $T$  if and only if

$$T \models C_1 \cap C_2 \sqsubseteq \perp$$

The ontology adaptation algorithm described in the next section is a modification of the ideas that have been developed for the  $\mathcal{AL}\mathcal{E}$ -DL in (Ovchinnikova & Kühnberger 2006a), in order to model the less expressive logic defined in (Haase et al. 2005). This relatively weak logic seems to be appropriate for an implementation of an automatic ontology extension procedure.

## 4 Ontology Adaptation Algorithm

In this section, we describe the algorithm adapting an ontology to a new axiom. Before applying the adaptation algorithm *AdaptOnto* to a TBox, all equalities must be replaced by inclusions:

$$A_1 \equiv A_2 \longrightarrow A_1 \sqsubseteq A_2, A_2 \sqsubseteq A_1$$

The proposed algorithm *AdaptOnto* adapts a TBox  $T$  to a new axiom  $X \sqsubseteq Y$ . The algorithm revises the definitions of the subconcepts of  $X$  because the introduction of a new definition of a concept  $X$  can have an influence on the semantics of its subconcepts.

If a concept  $A$  is a subconcept of  $X$ , then every two definitions  $D_1$  and  $D_2$  of  $A$  are checked whether conflicts occur. If  $D_1$  conflicts with  $D_2$  and the set of common subsumers for  $D_1$  and  $D_2$  is non-empty, then  $D_1$  and  $D_2$  in the definitions of  $A$  will be replaced with their least common subsumers.

A definition of  $A$  ( $D_1$  or  $D_2$ ) is overgeneralized (and denoted by  $D_o$ ) if it is axiomatized in  $T$  and some of its definitions conflicts with the alternative definition of  $A^5$ . The definition of  $D_o$  will be changed. The alternative concept taken from the set  $\{D_1, D_2\}$  is denoted by  $D_c$  and is called contradicting definition.

The set  $C_c$  collects superconcepts of  $D_o$  that conflict with  $D_c$ . On the other hand, the set  $C_n$  denotes non-contradicting concepts. As explained in Sec. 3, the conflicting definitions of  $D_o$  (from the set  $C_c$ ) are removed from  $T$  and assigned to the new concept  $A_N$  if they are minimal towards subsumption.  $A_N$  is declared to be a subconcept of  $D_o$ . Non-contradicting

<sup>5</sup>In the case of two overgeneralized concepts the axioms to be changed are chosen according to the confidence rating.

**Input:** a TBox  $\mathcal{T}$ , an axiom  $X \sqsubseteq Y$   
**Output:** an adapted TBox  $\mathcal{T}'$

```

 $\mathcal{T}' := \mathcal{T} \cup \{X \sqsubseteq Y\}$ 
FOR  $A \in \{A' \in ax(\mathcal{T}') \mid \mathcal{T}' \models A' \sqsubseteq X\}$ 
  FOR  $\{D_1, D_2\} : A \sqsubseteq D_1 \in \mathcal{T}' \wedge A \sqsubseteq D_2 \in \mathcal{T}' \wedge D_1 \neq D_2$ 
    IF  $D_1$  conflicts with  $D_2$  in  $\mathcal{T}'$  THEN
      IF  $lcss_{\mathcal{T}'}(D_1, D_2) \neq \emptyset$  THEN
         $\mathcal{T}' := \mathcal{T}' \setminus \{A \sqsubseteq D_1, A \sqsubseteq D_2\} \cup \{A \sqsubseteq C' \mid C' \in lcss_{\mathcal{T}'}(D_1, D_2)\}$ 
      ELSE
        IF  $\exists i, j \in \{1, 2\} : \exists D'_i : D_i \sqsubseteq D'_i \in \mathcal{T}' \wedge D'_i$  conflicts with  $D_j$  in  $\mathcal{T}'$  THEN
          IF  $\exists D'_j : D_{j \neq i} \sqsubseteq D'_j \in \mathcal{T}' \wedge D'_j$  conflicts with  $D_i$  in  $\mathcal{T}'$  THEN
             $D_o := D_{k \in \{1, 2\}}$  such that  $r_{conf}(D_k) < r_{conf}(D_{m \in \{1, 2\}, m \neq k})$ 
             $D_c \in \{D_1, D_2\} \setminus \{D_o\}$ 
            ELSE  $D_o := D_i, D_c := D_j$ 
             $C_c := \{C \in subsumers_{\mathcal{T}'}(D_o) \mid C$  conflicts with  $D_c$  in  $\mathcal{T}'\}$ 
             $C_n := \{C \in subsumers_{\mathcal{T}'}(D_o) \mid C$  does not conflict with  $D_c$  in  $\mathcal{T}'\}$ 
             $\mathcal{T}' := \mathcal{T}' \setminus (\{D_o \sqsubseteq C \mid C \in C_c\} \cup \{Z \sqsubseteq D_o \mid Z \neq A\}) \cup$ 
               $\{D_o \sqsubseteq C \mid C \in C_n \wedge \forall C' \in C_n : \mathcal{T}' \models C' \sqsubseteq C \rightarrow C' \dot{=} C\} \cup$ 
               $\{A_N \sqsubseteq C \mid C \in C_c \wedge \forall C' \in C_c : \mathcal{T}' \models C' \sqsubseteq C \rightarrow C' \dot{=} C\} \cup$ 
               $\{A_N \sqsubseteq D_o\} \cup \{W \sqsubseteq A_N \mid W \neq A \wedge W \sqsubseteq D_o \in \mathcal{T}'\}$ 
          ELSE  $D_r := D_{k \in \{1, 2\}}$  such that  $r_{conf}(D_k) < r_{conf}(D_{m \in \{1, 2\}, m \neq k})$ 
             $\mathcal{T}' := \mathcal{T}' \setminus \{A \sqsubseteq D_r\}$ 
        END FOR
      END FOR
    END FOR
  END FOR

```

Figure 1: The Algorithm *AdaptOnto* for adapting a TBox  $\mathcal{T}$  to a new axiom given by a concept description. The output is a new TBox  $\mathcal{T}'$  resolving overgeneralized definitions.

concepts from  $C_n$  that are minimal towards subsumption are added to the TBox as definitions of  $D_o$ . Previous subconcepts of  $D_o$  are declared to be subsumed by the new concept  $A_N$  that captures the original semantics of  $D_o$ .

If the overgeneralization can not be defined then the definition of  $A$  ( $D_1$  or  $D_2$ ) that has the least confidence rating is removed from the TBox.

## 5 Generalization Problem in Consistent Ontologies

### 5.1 The Problem

Even a consistent ontology can contain generalization errors. Automatic ontology learning procedures often rely on random facts that are extracted from external sources and are not observed by a human expert. Therefore the proper generalization of an automatically extracted ontology is rather accidental than intended.

We saw in the previous sections – particularly in Section 3 – that definitions of overgeneralized concepts can be detected and appropriately revised by the appearance of exceptions. On the other hand, undergeneralized concepts can sometimes be revised without additional information. Let us consider a simple example. Suppose that our ontology contains the facts:

- *Dogs are animals that breathe and drink water.*
- *Cats are animals that breathe and drink milk.*
- *Horses are animals that breathe and eat hay.*
- ... and so on for other animals.

Probably we would like to conclude from such a collection of facts that all animals can breathe and reformulate the ontology in the following way:

- *All animals breathe.*
- *Dogs are animals that drink water.*
- *Cats are animals that drink milk.*
- *Horses are animals that eat hay.*
- ... and so on.

Undergeneralization does not lead to inconsistencies of an ontology. But it is also not only a matter of design. Suppose that by a further extension of an ontology or by an instantiation it will be derived that a mole is an animal, but nothing else is known about it. We would like to infer that a mole can also breathe like other animals do. A proper generalization will help us to do so.

In practical applications, the undergeneralization problem is well-known and usually treated either semi-automatically or by analyzing external linguistic data. For example, in (Ceusters et al. 2003) it is shown how cross-lingual information can be used to detect undergeneralization in large ontologies.

In (Ovchinnikova & Kühnberger 2006b), the authors introduce an induction procedure designed for the regeneration of the axioms in an  $\mathcal{ALCN}$  description logic. In the following sections we adapt this procedure to the simple DL logic under consideration in order to make it applicable in automatic ontology learning.

### 5.2 Induction

The idea of an induction procedure proposed in (Ovchinnikova & Kühnberger 2006b) is simple. If all subconcepts of a concept  $C$  are also subconcepts of some other concept  $G$ , then  $C$  is likely to be a subconcept of  $G$ . Such generalizations are very similar to what is called upward inheritance in feature logic or programming: if all subtypes of a type  $C$  share the same feature, this feature should be inherited by  $C$ .

For practical applications it is useful to introduce certain heuristics and generalize a concept only if it has more than  $t$  many subconcepts with the same feature, where  $t$  is considered as an empirical parameter. In other words, statistical information can be used in order to decide whether a concept should be generalized or not.

After the execution of inductions it can happen that mistakes occur provided that more information is available. For example, if only bird species occur in a certain context and we apply induction it could

happen that we end up with an ontology where all animals can fly. All inductions can be checked and adapted during the next steps of the ontology extension by applying the procedure described in the previous sections.

In Subsection 5.3 and Subsection 5.4, we will consider the algorithmic details of induction: first, we discuss the algorithm *Regen* in Subsection 5.3. Second, we add some remarks concerning the prototype implementation of this algorithm in Subsection 5.4. The following definition specifies the induction procedure:

**Definition 5** For every TBox  $\mathcal{T}$ , for every concept name  $A$  the induction function  $Ind : TBox \times A \rightarrow TBox$  is defined as follows:

$$Leaves(A) := \{concept\ name\ L \mid \mathcal{T} \models L \sqsubseteq A \wedge \forall\ concept\ name\ B : L \doteq B \vee \mathcal{T} \not\models B \sqsubseteq A\}$$

$$LCS = lcss_{\mathcal{T}}(A_1, \dots, A_n) \text{ where } \{A_1, \dots, A_n\} = Leaves(A)$$

$$Ind(\mathcal{T}, A) = \mathcal{T} \cup \{A \sqsubseteq C \mid C \in LCS \wedge \forall C' \in LCS : \mathcal{T} \models C' \sqsubseteq C \rightarrow C' \doteq C\}$$

In Definition 5, the induction function  $Ind$  is defined for a TBox  $\mathcal{T}$  and a concept name  $A$ . The set  $Leaves$  collects all the subconcepts of  $A$  that have no further subconcepts. The set  $LCS$  represents the least common subsumers of the leaves of  $A$ . The induction function returns the TBox  $\mathcal{T}'$  extending  $\mathcal{T}$  with axioms that are minimal towards subsumption concepts taken from  $LCS$ .

In order to make the induction procedure more transparent, we give a simple example<sup>6</sup> of the application of the induction function to the TBox below and the concept **Person**:

TBox:  
 $\{Woman \sqsubseteq Person \sqcap \forall hasSpouse.Man, Man \sqsubseteq Person \sqcap \forall hasSpouse.Woman\}$

Regeneralized TBox:  
 $\{Person \sqsubseteq \forall hasSpouse.Person, Woman \sqsubseteq Person \sqcap \forall hasSpouse.Man, Man \sqsubseteq Person \sqcap \forall hasSpouse.Woman\}$

In the example above, new information is added to the definition of the concept **Person**. From the definitions of its leaves **Man** and **Woman** it can be induced that every spouse of a person is also a person.

In the next section, we introduce the algorithm regeneralizing a TBox formalized in the DL logic under consideration.

### 5.3 Ontology Regeneralization

This section presents the regeneralization algorithm *Regen* inducing new definitions for concepts in the TBox  $\mathcal{T}$  on basis of the definitions of their subconcepts.

The algorithm proposed in Figure 2 tries to regeneralize every concept  $A$  in  $\mathcal{T}$ . The set of leaves  $Leaves(A)$  is computed for  $A$ . If the cardinality of this set exceeds the empirical parameter  $t$  (for  $t \in \mathbb{N}$ ), then the set of the least common subsumers  $LCS$  is computed relative to the concepts in  $Leaves(A)$ . If the set  $LCS$  is non-empty, then the concept  $A$  will be regeneralized as follows:

- a) Concept descriptions from the set  $LCS$  will be declared to subsume  $A$ ;

<sup>6</sup>In the following examples we use the symbol  $\sqcap$  for abbreviation:  $\{C \sqsubseteq D_1 \sqcap D_2\}$  stands for  $\{C \sqsubseteq D_1, C \sqsubseteq D_2\}$ .

**Input:** a TBox  $\mathcal{T}$ , a parameter  $t$   
**Output:** a regeneralized TBox  $\mathcal{T}'$

```

 $\mathcal{T}' := \mathcal{T}$ 
FOR concept name  $A$ 
   $Leaves(A) := \{concept\ name\ L \mid \mathcal{T} \models L \sqsubseteq A \wedge \forall\ concept\ name\ B : L \doteq B \vee \mathcal{T} \not\models B \sqsubseteq A\}$ 
  IF  $|Leaves(A)| \geq t$  THEN
     $LCS = lcss_{\mathcal{T}}(A_1, \dots, A_n)$  where  $\{A_1, \dots, A_n\} = Leaves(A)$ 
    FOR  $C \in LCS$ 
      IF  $\mathcal{T}' \not\models A \sqsubseteq C$  AND  $\forall C' \in LCS : \mathcal{T}' \models C' \sqsubseteq C \rightarrow C' \doteq C$ 
        THEN  $\mathcal{T}' := \mathcal{T}' \cup \{A \sqsubseteq C\}$ 
    FOR  $A' : \mathcal{T}' \models A' \sqsubseteq A$ 
       $\mathcal{T}' := \mathcal{T}' \setminus \{A' \sqsubseteq D \mid \mathcal{T}' \models C \sqsubseteq D\}$ 
    END FOR
  END FOR
END FOR

```

Figure 2: Algorithm *Regen* for the regeneralization of a TBox  $\mathcal{T}$ . *Regen* resolves undergeneralized concept definitions.

- b) The definitions of the subconcepts of  $A$  that subsume concepts from  $LCS$  will be removed (since they become redundant).

The algorithm *Regen* computes the construction specified in Definition 5 by computing the generalization of a given TBox. The following example illustrates the application of the *Regen* algorithm:

TBox:  
 $\{Cat \sqsubseteq Animal \sqcap Breathing \sqcap \forall drink.Milk, Dog \sqsubseteq Animal \sqcap Breathing \sqcap \forall drink.Water, Man \sqsubseteq Human \sqcap Breathing, Milk \sqsubseteq Liquid, Water \sqsubseteq Liquid\}$

Regeneralized TBox:  
 $\{Animal \sqsubseteq Breathing \sqcap \forall drink.Liquid, Cat \sqsubseteq Animal \sqcap \forall drink.Milk, Dog \sqsubseteq Animal \sqcap \forall drink.Water, Man \sqsubseteq Human \sqcap Breathing, Milk \sqsubseteq Liquid, Water \sqsubseteq Liquid\}$

In the example above, new definition is generated for the concept **Animal**. The redundant information about breathing is removed from the definitions of **Cat** and **Dog**.

### 5.4 Prototype Implementation

The prototype implementation of the regeneralization procedure has been tested successfully on an example ontology automatically extracted with the tools developed in the framework of the ASADO project ([www.cogsci.uni-osnabrueck.de/~ASADO](http://www.cogsci.uni-osnabrueck.de/~ASADO)). The basis of the ASADO project were scanned documents (a majority of them taken from the aviation industry). In the project, standard tools mainly taken from computational linguistics research were applied to make these documents electronically available. Examples of such tools were an OCR-software, a tagger, or a state-of-the-art statistical parser. Based on the resulting electronically enriched documents an ontology for these documents was automatically extracted. For cross-evaluation purposes other document corpora were also used.

The ASADO ontology contains only the taxonomy, general relations and as logical connective conjunction. Here is an example of a concept definition in the RDF format:

```
<rdfs:Class rdfs:about="o:telephone-account">
  <rdfs:subClassOf rdfs:resource="o:account"/>
</rdfs:Class>
```

According to this definition the concept *telephone account* is a subconcept of the concept *account* having the property to be "telephone". In description logics this information can be formalized as follows:

$$\text{telephone-account} \sqsubseteq \text{attribute\_telephone} \sqcap \text{account}$$

The considered ASADO ontology contains approximately 3000 concepts. Among them we have found 20 undergeneralized concepts. It is clearly a matter of discussion if all regeneralizations proposed by the system are relevant for the thematic area under consideration. But the discovered undergeneralization cases can give the ontology engineer important hints of how to refine an ontology extracted automatically. Furthermore the undergeneralization cases can serve as an additional evaluation criteria of the ontology learning procedure.

Let us consider a quite simple and obvious example of an undergeneralized concept. The ASADO ontology contains several concepts using the concept *hong* in the definitions: *epson-hong-kong*, *epson-hong-kong-limited*, *epson-hong-kong-ltd*, *hong-kong-phone*, *hong-kong-user*. It is obvious that *hong* never occurs in the ontology without the concept *kong*. The system suggests to unite the concepts *hong* and *kong* in one concept.

In the near future, we plan to test the proposed algorithm on ontologies formalized in more expressive description logics. Nevertheless, the described prototype implementation already supports the claim that the presented induction procedure is relevant for ontology engineering.

## 6 Semantic Issues

Although we do not focus in detail on semantic issues in this paper, some remarks concerning the semantics of a regeneralized TBox are added in this subsection. We postpone a thorough discussion of this important issue to another paper.

Let us consider the regeneralization of just one concept *A* axiomatized in a given TBox. By slightly simplifying things, we get the following situation: If *A* is undergeneralized, then its definition will be extended by the *Regen* algorithm. In the case of an overgeneralization some concept descriptions will be removed from the definition of *A* by *AdaptOnto*. It is easy to show that in both cases only the semantics of *A* itself changes, whereas the semantics of its subconcepts remains unchanged. Obviously, the two processes change the semantics of *A* in two different directions. Whereas the induction procedure narrows the semantics of *A* by adding further constraints, the adaptation procedure *AdaptOnto* extends it by removing removing constraints on *A*. The following Fact is a direct consequence of the considerations so far.

**Fact 2** *For every TBox  $\mathcal{T}$ , for every interpretation  $\mathcal{I}$ , and for every axiom  $Ax$  the following two claims hold:*

- (i) *If  $\mathcal{I}$  models  $\mathcal{T}$ , then it holds  $\mathcal{I}$  models the adapted TBox  $\text{AdaptOnto}(\mathcal{T}, Ax)$ .*
- (ii) *If  $\mathcal{I}$  models the regeneralized TBox  $\text{Regen}(\mathcal{T})$ , then it holds  $\mathcal{I}$  models  $\mathcal{T}$ .*

Assuming that the function *subsumers*(*C*) introduced above computes all possible concept descriptions subsuming *C* in the DL under consideration (Fact 1), we claim that the induction procedure computes "the best" regeneralization for *A* relative to a chosen heuristics. In other words nothing more can be induced about *A* in the induction process relative to the chosen heuristic.<sup>7</sup>

**Fact 3** *For every TBox  $\mathcal{T}$ , for every concept name  $A$ , and for every sets concept description  $C$  the following equivalence holds:*

$$(\mathcal{T} \models A' \sqsubseteq A \rightarrow \mathcal{T} \models A' \sqsubseteq C) \Leftrightarrow \text{Ind}(\mathcal{T}, A) \models A \sqsubseteq C$$

In the adaptation procedure, the choice of the overgeneralized concept is based on a heuristics. Therefore it is impossible to prove strictly logically that the *AdaptOnto* algorithm guarantees the minimality of changes<sup>8</sup> of an inconsistent ontology. But once the overgeneralized concept has been chosen, then it is quite obvious that *AdaptOnto* removes a minimum of information from its definition. This claim follows directly from the definition of the *subsumers* function (Fact 1). For an overgeneralized concept *A* the sets  $C_c$  and  $C_n$  of conflicting and non-contradicting concept descriptions subsuming *A* are computed on the basis of the *subsumers* function. Therefore these sets are exhaustive in the description logic under consideration.

## 7 Conclusion and Future Work

In this paper, we presented an approach for dynamically resolving conflicts appearing by automatic ontology learning. We have adopted ideas firstly presented in (Ovchinnikova & Kühnberger 2006a) for the subset of description logics corresponding to the logic practically used in systems for ontology learning (Haase et al. 2005). The main contributions of this paper are the specification of an algorithm for ontology adaptation for the mentioned weak logic practically used in systems and the specification of an algorithm generalizing ontologies (based on the same logic). Whereas ontology adaptation is strongly connected to non-monotonicity and the problem of handling inconsistencies, generalizations are strongly related to inductive reasoning.

The two algorithms *AdaptOnto* and *Regen* can be embedded into an overall architecture amalgamating ontologies automatically if new information about the original TBox is available. Figure 3 represents diagrammatically how these two algorithms can be embedded into an ontology framework. The following list summarizes some important steps in this integration process.

- Starting with a given ontology *O* new information (represented by axioms) updates the logical description of *O*. The result is a new (updated) ontology  $O^+$ , i.e. the underlying TBox is updated by these new axioms.
- In a second step, a standard reasoning system checks the consistency of  $O^+$ .

<sup>7</sup>Recall that the heuristics is based on the chosen natural number *t* of leaves of *A* with the same feature.

<sup>8</sup>Due to a long history of non-monotonic reasoning in AI where minimality conditions play an important role, the formal definition of the notion of minimality in this context requires an additional investigation. In this section, we use this term informally just to give an idea of how to evaluate the proposed procedures from the semantical point of view.

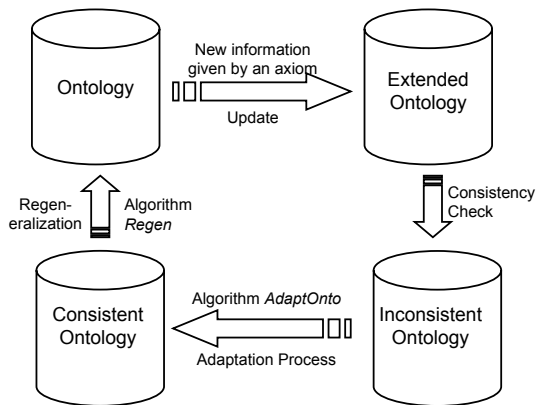


Figure 3: The diagrammatic representation of integrating the algorithms described in this paper into an ontology system. The resulting circle is permanently updating ontological knowledge with new information in a consistent way and keeps the ontology as compact as possible.

- If an inconsistency occurs, the presented algorithm *AdaptOnto* generates a new consistent ontology  $O_{con}^+$ . If no inconsistency occurs no adaptation is necessary.
- Finally the algorithm *Regen* rewrites the ontology  $O_{con}^+$  into an ontology representing knowledge in a more compact way by rewriting under-generalized concepts.
- The circle starts again by an update given of new axioms.

In the near future, we plan to develop a prototype implementation of the proposed architecture by combining the presented algorithms and test them on existing ontologies. It is of particular interest to see to what extent statistical information about the distribution and co-occurrence of concepts in texts can help to improve the adaptation procedure for making it more adequate to human intuition. Similarly, generalizations defined on ontologies are also dependent on statistically relevant information as can be seen in Figure 2 where generalization is only possible if a concept has more than  $t$  subsumers with the same feature.

Last but not least, an important theoretical issue concerns the complexity of the proposed algorithms. In the future, we plan to show characterization results specifying the complexities classes of the algorithms *AdaptOnto* and *Regen*, in order to prove theoretically the practical relevance and the tractability of the proposed architecture.

### Acknowledgment

This research was partially supported by the grant MO 386/3-4, a subproject of the collaborative research unit FOR 437 sponsored by the German Research Foundation (DFG).

### References

Baader, F., Lutz, C., Miličić, M., Sattler, U. & Wolter, F. (2005), Integrating Description Logics and Action Formalisms: First Results. In *Proc. of the 20th National Conference on Artificial Intelligence (AAAI'05)*, AAAI Press (2005).

Baader, F., Calvanese, D., McGuinness, D., Nardi, D. & Patel-Schneider, P. (eds.) (2003), *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F. & Küsters, R. (2006), Non-Standard Inferences in Description Logics: The Story So Far. *International Mathematical Series*, volume 4, *Mathematical Problems from Applied Logic. New Logics for the XXIst Century*.

Baader, F. & Sattler, U. (2001), An overview of tableau algorithms for description logics, *Studia Logica*, 69:5–40.

Baral, C. (2006), *Knowledge Representation, Reasoning and Declarative Problem Solving with Answer Sets*. Cambridge University Press.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001), The Semantic Web – A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, May 17, 2001, available on the world wide web (10th of March, 2006): [http://www.sciam.com/print\\_version.cfm?articleID=0004814410D21C7084A9809EC588EF21](http://www.sciam.com/print_version.cfm?articleID=0004814410D21C7084A9809EC588EF21).

Bibel, W., Hölldobler, S. & Schaub, T. (1993), *Wissensrepräsentation und Inferenz. Eine grundlegende Einführung*, Vieweg, Braunschweig.

Ceusters, W., Desimpel, I., Smith, B. & Schulz, S. (2003), Using Cross-Lingual Information to Cope with Underspecification in Formal Ontologies, In *Studies in Health Technology and Informatics*, 391–396.

Cohen, W., Borgida, A. & Hirsh, H. (1993), Computing Least Common Subsumers in Description Logics, In *Proc. of the 10th Nat.Conf. on Artificial Intelligence (AAAI-92)*. AAAI Press, 754–761.

Fanizzi, N., Ferilli, S., Iannone, L., Palmisano, I. & Semeraro, G. (2005), Downward Refinement in the ALN Description Logic. In: Masumi Ishikawa, Shuji Hashimoto, Marcin Paprzycki, Emilia Barakova, Kaori Yoshida, Mario Köppen, David W. Corne and Ajith Abraham (Eds.), *Hybrid Intelligent Systems (HIS'04)*, 68–73

Flouris, G., Plexousakis, D. & Antoniou, G. (2005), Updating Description Logics using the AGM Theory. In *Proc. of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning*.

Ghilardi, S., Lutz, C. & Wolter, F. (2006), Did I damage my ontology: A Case for Conservative Extensions of Description Logics. In *Proc. of Principles of Knowledge Representation and Reasoning 2006 (KR06)* (to appear).

Gómez-Pérez, A. & Manzano-Macho, D. (2003), A survey of ontology learning methods and techniques, <http://ontoweb.aifb.uni-karlsruhe.de/Members/ruben/Deliverable>

Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H. & Sure, Y. (2005), A Framework for Handling Inconsistency in Changing Ontologies In *Proc. of the Fourth International Semantic Web Conference (ISWC2005)*, v. 3729, pp. 353-367. Springer (2005).

Heymans, S. & Vermeir, D. (2002), A Defeasible Ontology Language, In Robert Meersman and Zahir Tari et al., editors, *Confederated International Conferences: CoopIS, DOA and ODBASE 2002*.

- Horrocks, I. (1998), Using an expressive description logic: FaCT or fiction? In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, 636–645. Morgan Kaufmann, San Francisco, California.
- Katz, Y. & Parsia, B. (2005), OWL: Experiences and Directions, Galway Ireland, online available at: <http://www.mindswap.org/2005/OWLWorkshop/sub7.pdf>.
- Lifschitz, V. (1994), Circumscription. In: *Handbook of Logic in Artificial Intelligence and Logic Programming*, Volume 3, 297–352, Oxford University Press.
- Motik, B., Sattler, U. & Studer, R. (2004), Query Answering for OWL-DL with Rules. In *Proc. of ISWC 2004*, LNCS 3298, 549–563, Springer (2004).
- Ovchinnikova, E. & Kühnberger, K. (2006a), Adaptive  $\mathcal{ALC}$ -TBox for Extending Terminological Knowledge, to appear in: A. Sattar and B. H. Kang (eds.): AI 2006, Proceedings of the 19th ACS Australian Joint Conference on Artificial Intelligence, LNAI 4304 (Lecture Notes in Artificial Intelligence), Springer, pp. 1111–1115.
- Ovchinnikova, E. & Kühnberger, K. (2006b), The Undergeneralization Problem in Ontology Design. In preparation.
- Reiter, R. (1980), A logic for default reasoning. *Artificial Intelligence* 13:81–132.
- OWL Web Ontology Language (2004), Overview. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/owl-features/>.
- Walter, C. (1985), A Mechanical Solution of Schubert's Steamroller by Many-Sorted Resolution. *Artificial Intelligence* 26:217–224.