Acquisition of OWL DL Axioms from Lexical Resources

Johanna Völker, Pascal Hitzler and Philipp Cimiano

Institute AIFB, University of Karlsruhe, Germany

Abstract. State-of-the-art research on automated learning of ontologies from text currently focuses on inexpressive ontologies. The acquisition of complex axioms involving logical connectives, role restrictions, and other expressive features of the Web Ontology Language OWL remains largely unexplored. In this paper, we present a method and implementation for enriching inexpressive OWL ontologies with expressive axioms which is based on a deep syntactic analysis of natural language definitions. We argue that it can serve as a core for a semi-automatic ontology engineering process supported by a methodology that integrates methods for both ontology learning and evaluation. The feasibility of our approach is demonstrated by generating complex class descriptions from Wikipedia definitions and from a fishery glossary provided by the Food and Agriculture Organization of the United Nations.

1 Introduction

Knowledge modeling for semantic applications, particularly the creation of ontologies, is a difficult and time-consuming task. It usually requires to combine the knowledge of domain experts with the skills and experience of ontology engineers into a single effort with high demand on scarce expert resources. We believe that this bottleneck currently constitutes a severe obstacle for the transfer of semantic technologies into practice. In order to address this bottleneck, it is reasonable to draw on available data, applying automated analyses to create ontological knowledge from given resources or to assist ontology engineers and domain experts by semi-automatic means. Accordingly, a significant number of ontology learning tools and frameworks has been developed aiming at the automatic or semi-automatic construction of ontologies from structured, unstructured or semi-structured documents. However, both quality and expressivity of the ontologies which can be generated by the current state of the art in lexical ontology learning have failed to meet the expectations of people who argue in favor of powerful, knowledge-intensive applications based on ontological reasoning. Purely logical approaches on the other hand presuppose a large number of manually created ABox statements, and lack the scalability required by many application scenarios.

In this paper, we focus on text corpora as the source for automated ontology creation. Texts are available in abundance from the internet, and the knowledge expressed e.g. through defining sentences given by experts can be expected to be a good base for the creation of ontology axioms describing the same knowledge. Our approach is essentially based on a syntactic transformation of natural language definitions into description logic axioms. It hinges critically on the availability of sentences which have definitory character, like "*Enzymes are proteins that catalyse chemical reactions.*" Such

sentences could be obtained e.g. from glossaries or software documentation related to the underlying ontology-based application. Here, we exemplify our approach by using definitions taken from Wikipedia[1] and a fishery glossary provided by the *Food and Agriculture Organization of the United Nations* (FAO), while also sketching a number of alternatives which could suggest themselves in different application scenarios. One of these alternatives, and a particularly interesting case in point is the exploitation of comments in less expressive ontologies given by means of annotation properties.

Consider, for example, an RDFS ontology (lying within OWL DL) essentially consisting of a class hierarchy with some simple use of properties. This ontology, e.g. modeling the scientific domain of Bioinformatics, could well describe a class *Enzyme* being annotated with a comment like the sentence given before, which documents the class for the ontology engineer – note that the sentence cannot be modeled properly in RDFS. In order to enhance the RDFS ontology, our automated tool LExO can be used to analyze this sentence and encode it as the OWL DL axiom

$$\text{Enzyme} \equiv (\text{Protein} \sqcap \exists \text{catalyse}.(\text{Chemical} \sqcap \text{Reaction})).$$

This OWL DL axiom might then be conveyed to the ontology engineer as a suggestion to enhance the ontology. We present here the initial work which can serve as an acquisition core for realizing such a semi-automated process. While we think that our approach has the necessary potential, several non-trivial obstacles remain to be addressed in forthcoming work. We will discuss these obstacles in detail and lay out a work plan for realizing our vision.

The paper is structured as follows. We first give some preliminaries in Section 2. In Section 3, we then describe our approach in detail, giving an ample supply of examples, and also describe our prototype implementation. This leads to a discussion, in Sections 4 and 5, of the obstacles which need to be overcome to realize the vision based on our initial work. We discuss related work and conclude in Section 6.

2 Preliminaries

The Web Ontology Language OWL [1] is being recommended as a web standard by the World Wide Web Consortium for modeling ontological knowledge which requires more expressivity than RDFS. Since then, OWL has found a multitude of uses, not only on the web, but for knowledge representation in general. In essence OWL, or more precisely its most important variant OWL DL, is a so-called *description logic*, or DL for short [2]. DLs combine a rigorous semantics based on first-order predicate logic with an intuitive way of structuring and encoding expressive conceptual knowledge.

In the following we give a formal introduction to the description logic underlying OWL DL. We will keep this introduction very brief, as an intuitive understanding of OWL DL suffices for understanding our approach. For the same reason, some finer details of the definitions will be omitted; the interested reader can find them in [2, 1]. We will introduce a syntax which is known as *DL-syntax*, as it is most convenient (and the easiest to read) for the purpose of this paper.

OWL DL is essentially the description logic known as $\mathcal{SHOIN}(\mathbf{D})$, which we introduce below. Knowledge bases in OWL DL are called *ontologies*, and they express

[1] `http://en.wikipedia.org`

relationships between the basic entities in OWL DL, which are *concepts* (or *classes*, like *Enzyme* or *Bank*), *roles* (or *properties*, denoting relationships between things, like *provides* or *represents*), and *individuals* (or *instances*, like *Rudi* or *Lactase*). There are actually two types of roles, namely abstract roles, which relate individuals to individuals, like *fatherOf*, and concrete roles, like *hasAge*, which assign an element of a concrete datatype $\mathcal{D}$ – in this case a number – to an individual.

For describing a $\mathcal{SHOIN}(\mathbf{D})$ (i.e. an OWL DL) ontology, we thus require three sets: a set of concept names (called *atomic* or *named concepts*), a set of role names, and a set of individual names. $\mathcal{SHOIN}(\mathbf{D})$ now allows to combine concepts to (*complex*) *concepts* as defined by the following grammar, where A is an atomic concept, r is an abstract role, s is a concrete role, d is a concrete domain predicate, a_i are individuals, c_i are elements of a datatype, and n is a non-negative integer:

$$C \rightarrow A \mid \bot \mid \top \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists r.C \mid \forall r.C \mid \geq n\, r \mid \leq n\, r \mid =n\, r \mid$$
$$\mid \{a_1, \ldots, a_n\} \mid \exists s.D \mid \forall s.D \mid \geq n\, s \mid \leq n\, s \mid =n\, s$$
$$D \rightarrow d \mid \{c_1, \ldots, c_n\}$$

Formally, the semantics of concepts is given by means of *interpretations*, which are mappings from the concept, role and individual names into sets, called *domains of interpretation* which are to be considered as their extensions. This is done essentially as in first-order predicate logic. We omit the details, which can be found in [1], and rather provide some examples to convey the intuition. *Female* $\sqcap$ *Human* stands for all human females, i.e. for all women. $\exists$ *hasChild*.*Male* denotes all things which have a male child. ≤ 3 *hasChild* is a so-called *cardinality restriction* and denotes all things which have at most 3 children. $\{a, b, c\}$ stands for the class containing exactly the instances a, b, c.

$\mathcal{SHOIN}(\mathbf{D})$ furthermore allows to specify relations between roles. In particular, two roles r_1 and r_2 can be in subrole relation ($r_1 \sqsubseteq r_2$), can be equivalent ($r_1 \equiv r_2$) or can be inverse to each other ($r_1 \equiv r_2^-$). Roles can also be specified as transitive.

Information about individuals is given in terms of the *ABox* of an ontology, which consists of statements of the form $C(a)$ and $r(a, b)$, where a, b are individuals, C is a (complex) concept and r is a role. Information about concrete roles like $s(a, d)$ can also be given, here d is an element of a concrete datatype, and the semantics is analogous.

Complex concepts can now be related in the following way: If C, D are complex concepts, then $C \sqsubseteq D$ and $C \equiv D$ are *inclusion axioms*, where the first denotes that C is a subconcept of D, and the second says that C and D are equivalent. The collection of all axioms of an ontology together with information about roles is called the *TBox* of the ontology. An ontology thus consists of a TBox and an ABox. By *OWL axioms* or *statements* we denote everything which can be contained in an ABox or a TBox. An *OWL element* is either an OWL axiom or a concept, role, or individual.

By means of the formal semantics which can be assigned to an ontology, it is possible to draw non-trivial logical inferences from an ontology. The ontology thus carries *implicit knowledge* by means of its semantics.

3 The LExO Approach and Examples

In this section we present a conceptual approach for transforming natural language definitions (e.g. annotation properties or associated dictionary entries) into sets of OWL

```
( E1 () (fin) C
   ( 3 is (be) VBE i
      ( 2 number (~) N s
         ( 1 A (~) Det det )
      )
      ( 6 entity (~) N pred
         ( E3 () (number) N subj 2 )
         ( 4 an (~) Det det )
         ( 5 abstract (~) A mod )
         ( E0 () (fin) C rel
            ( 7 that (~) THAT whn 6 )
            ( 8 represents (represent) V i
               ( E4 () (that) THAT subj 6 )
               ( 10 count (~) N obj
                  ( 9 a (~) Det det )
                  ( 11 or (~) U punc )
                  ( 12 measurement (~) N conj )
) ) ) ) ) )
```

Fig. 1. Dependency Tree (Minipar)

```
rule: relative clause {
    arg_0:  //N
    arg_1:  arg_0 /C[@role='rel']
    arg_2:  arg_1 /V
    result: [equivalent 0 [and 0-1 2]]
}
rule: verb and object {
    arg_0:  //V
    arg_1:  arg_0 /N[@role='obj']
    result: [equivalent 0 [some 0-1 1]]
    result: [subObjectPropertyOf 0 0-1]
}
```

Fig. 3. Transformation Rules

axioms. The technical feasibility of this approach is demonstrated by a prototypical implementation called LExO (*Learning EXpressive Ontologies*). However, the goal of our work is not to put forward our prototype, but rather to investigate the potential, limitations and challenges posed by attempting to acquire complex ontological knowledge from lexical evidence. We will thus abstract from some implementation details in favor of a critical discussion based on key examples, see Sections 3.1 and 4, and the development of a general methodology for putting our ideas to practical use, see Section 5.

The implementation of LExO basically relies on KAON2[2], an ontology management infrastructure for OWL DL, and the Minipar dependency parser [3]. Given a natural language definition of a class, LExO starts by analyzing the syntactic structure of the input sentence. The resulting dependency tree is then transformed into a set of OWL axioms by means of manually engineered transformation rules. In the following, we provide a step-by-step example to illustrate the complete transformation process. For more (and more complicated) examples please refer to Section 3.1.

Here, we assume that we would like to refine the description of the class *Number* which is part of the Proton ontology (see Section 3.1). The following definition of *Number* was taken from its corresponding Wikipedia article: *A **number** is an abstract entity that represents a count or measurement.*[3]

Initially, LExO applies the Minipar dependency parser wrapped by our own Java-based API in order to produce a structured output as shown in Figure 1. Every node in the dependency tree contains information about the token such as its lemma (base form), its syntactic category (e.g. *N* (noun)) and role (e.g. *subj*), as well as its surface position. Indentation in this notation visualizes direct dependency, i.e. each child node is syntactically dominated by its parent.

This dependency structure is now being transformed into an XML-based format (see Figure 2) in order to facilitate the subsequent transformation process, and to make LExO more independent of the particular parsing component.

[2] `http://kaon2.semanticweb.org`

[3] `http://en.wikipedia.org/wiki/Number`. In our experiments the word sense disambiguation required for identifying the correct article was done manually, although one could well imagine an automatic or semi-automatic solution depending on the requirements of the regarding application.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <C id="E1" pos="0">
    <VBE id="3" pos="3" role="i" phrase="is" base="be">
      <N id="2" pos="2" role="s" phrase="number">
        <Det id="1" pos="1" role="det" phrase="A"/>
      </N>
      <N id="6" pos="7" role="pred" phrase="entity">
        <N id="E3" pos="4" role="subj" base="number" antecedent="2"/>
        <Det id="4" pos="5" role="det" phrase="an"/>
        <A id="5" pos="6" role="mod" phrase="abstract"/>
        <C id="E0" pos="8" role="rel">
          <THAT id="7" pos="9" role="whn" phrase="that" antecedent="6"/>
          <V id="8" pos="10" role="i" phrase="represents" base="represent">
            <THAT id="E4" pos="11" role="subj" base="that" antecedent="6"/>
            <N id="10" pos="13" role="obj" phrase="count">
              <Det id="9" pos="12" role="det" phrase="a"/>
              <U id="11" pos="14" role="punc" phrase="or"/>
              <N id="12" pos="15" role="conj" phrase="measurement"/>
</N> </V> </C> </N> </VBE> </C> </root>
```

Fig. 2. XML Representation of Dependency Tree

```
[equivalent lexo:a_number lexo:an_abstract_entity_that_represents_a_count_or_measurement]
[equivalent lexo:an_abstract_entity_that_represents_a_count_or_measurement
    [and lexo:an_abstract_entity lexo:represents_a_count_or_measurement]]
[equivalent lexo:represents_a_count_or_measurement [some lexo:represents lexo:a_count_or_measurement]]
[equivalent lexo:a_count_or_measurement [or lexo:a_count lexo:measurement]]
[equivalent lexo:abstract_entity [and lexo:entity lexo:abstract]]
```

Fig. 4. Resulting Axioms

The set of rules which are then applied to the XML-based parse tree make use of XPath expressions for transforming the dependency structure into one or more OWL DL axioms. Figure 3 shows a few examples of such transformation rules in original syntax. Each of them consists of several arguments (e.g. *arg_1:*...), the values of which are defined by an optional prefix, i.e. a reference to a previously matched argument (*arg_0*), plus an XPath expression such as */C[@role='rel']* being evaluated relative to that prefix. The last lines of each transformation rule define one or more templates for OWL axioms, with variables to be replaced by the values of the arguments. Complex expressions such as *0-1* allow for "subtracting" individual subtrees from the overall tree structure. A more complete listing of the transformation rules we applied can be found further below.

A minimal set of rules for building a complete axiomatization of the *Number* example could be, e.g. *Copula*, *Relative Clause*, *Transitive Verb Phrase*, *Disjunction* and *Subsective Adjective* (see Table 1). The resulting list of axioms (see Figure 4) in KAON2 internal syntax is directly fed into KAON2 which interprets the textual representation of these axioms, and finally builds an unfolded[4] class description as shown in Figure 5.

In DL syntax, this final, unfolded axiomatization reads:

$$\text{A\_number} \equiv ((\text{Entity} \sqcap \text{Abstract}) \sqcap \exists \text{represents}.(\text{A\_count} \sqcup \text{Measurement}))$$

Obviously, all parts of this class description have to be normalized and possibly mapped to already existing content of the ontology before the results can be used to generate suggestions for ontology changes (cf. Section 5). As shown by the large body of re-

[4] By *unfolding*, a term borrowed from logic programming, we mean transformations like that of $\{A \equiv \exists R.B, C \equiv A \sqcap D\}$ to $\{C \equiv \exists R.B \sqcap D\}$. The specific for of output which we receive allows us to remove many of the newly generated class names by unfolding, in order to obtain a more concise output.

```
[equivalent lexo:a_number [and [and lexo:entity lexo:abstract] [some lexo:represents [or lexo:a_count lexo:
    measurement ]]]]
```

Fig. 5. Class Description (unfolded)

Rule	Natural Language Syntax	OWL Axioms
Disjunction	NP_0 `or` NP_1	$X \equiv (\text{NP}_0 \sqcup \text{NP}_1)$
Conjunction	NP_0 `and` NP_1	$X \equiv (\text{NP}_0 \sqcap \text{NP}_1)$
Determiner	Det_0 NP_0	$X \equiv \text{NP}_0$
Intersective Adjective	Adj_0 NP_0	$X \equiv (\text{Adj}_0 \sqcap \text{NP}_0)$
Subsective Adjective	Adj_0 NP_0	$X \sqsubseteq \text{NP}_0$
Privative Adjective	Adj_0 NP_0	$X \sqsubseteq \neg\text{NP}_0$
Copula	NP_0 VBE NP_1	$\text{NP}_0 \equiv \text{NP}_1$
Relative Clause	NP_0 C(*rel*) VP_0	$X \equiv (\text{NP}_0 \sqcap \text{VP}_0)$
Number Restriction	V_0 Num NP(obj)_0	$X \equiv =\text{Num}\ \text{V}_0.\text{NP}_0$
Negation (not)	`not` V_0 NP_0	$X \sqsubseteq \neg\exists \text{V}_0.\text{NP}_0$
Negation (without)	NP_0 `without` $\text{NP}(pcomp\text{-}n)_1$	$X \equiv (\text{NP}_0 \sqcap \neg\text{with}.\text{NP}_1)$
Participle	NP_0 $\text{VP}(vrel)_0$	$X \equiv (\text{NP}_0 \sqcap \text{VP}_0)$
Transitive Verb Phrase	V_0 $\text{NP}(obj)_0$	$X \equiv \exists \text{V}_0.\text{NP}_0$
Verb with Prep. Compl.	V_0 Prep_0 $\text{NP}(pcomp\text{-}n)_0$	$X \equiv \exists \text{V}_0\_\text{Prep}_0.\text{NP}_0$
Noun with Prep. Compl.	NP_0 Prep_0 $\text{NP}(pcomp\text{-}n)_1$	$X \equiv (\text{NP}_0 \sqcap \exists \text{Prep}_0.\text{NP}_1)$
Prepositional Phrase	Prep_0 NP_0	$X \equiv \exists \text{Prep}_0.\text{NP}_0$
...	...	...

Table 1. Transformation Rules

search done in the domain of ontology mapping, this task is not trivial at all. Semantic ambiguities of labels (e.g. homonymy or polysemy), as well as the fact that a single entity or axiom in the ontology can have arbitrarily many lexicalizations – differing even in their syntactic category – make it necessary to consider a multitude of possible mappings. Moreover, idiomatic expressions, i.e. expressions the meaning of which cannot be directly derived from the meaning of their individual components, need to be treated properly. Therefore, in addition to integrating a state-of-the-art mapping framework, a significant degree of user involvement will be unavoidable in the end (see Section 5).

Table 1 gives an overview of the most frequently used transformation rules. Each row in the table contains the rule name (e.g. *Verb with Prepositional Complement*) and an expression describing the natural language syntax matched by that rule – like, for example, $V_0\ Prep_0\ NP(\textit{pcomp-n})_0$, where V_0 represents a verb, $Prep_0$ a preposition and $NP(\textit{pcomp-n})$ denotes a noun phrase acting as a prepositional complement. Please note that these expressions are very much simplified due to lack of space. The last column shows the OWL axioms generated in each case, where X denotes the atomic class name represented by the surface string of the complete expression matched by the regarding transformation rule.

It is important to emphasize that this set of rules is by no means exhaustive, nor does it define the only possible way to perform the transformation. In fact, there are many different modeling possibilities, and the choice of appropriate rules very much depends on the particular application or individual preferences of the user (see example *Tetraploid* in Section 3.1).

3.1 Examples

We exemplify our approach by giving a number of axiomatizations automatically generated by LExO. The example sentences are not artificial, but were selected from real sources. The first is a fishery glossary provided by the Food and Agriculture Organization (FAO) of the United Nations within the NeOn project[5] – these are examples 1 through 8 below. The remaining examples concern classes of the Proton ontology [4], which has been developed in the SEKT project.[6] While Proton is an OWL ontology, it is rather inexpressive, so we undertook to create more complex OWL axiomatizations for Proton classes. For this purpose, we checked whether for a given Proton class name there was a corresponding Wikipedia article, and took the first sentence of this article as definitorial sentence for the class name. This approach worked reasonably well, as we will see. We first list the example sentences together with their axiomatizations.

1. ***Data:*** *Facts that result from measurements or observations.*
   $\text{Data} \equiv (\text{Fact} \sqcap \exists \text{result\_from}.(\text{Measurement} \sqcup \text{Observation}))$
2. ***InternalRateOfReturn:*** *A financial or economic indicator of the net benefits expected from a project or enterprise, expressed as a percentage.*
   $\text{InternalRateOfReturn} \equiv ((\text{Financial} \sqcup \text{Economic}) \sqcap \text{indicator} \sqcap \exists \text{of}.(\text{Net} \sqcap \text{Benefit} \sqcap \exists \text{expected\_from}.(\text{Project} \sqcup \text{Enterprise})) \sqcap \exists \text{expressed\_as}.\text{Percentage})$
3. ***Vector:*** *An organism which carries or transmits a pathogen.*
   $\text{Vector} \equiv (\text{Organism} \sqcap (\text{carry} \sqcup \exists \text{transmit}.\text{Pathogen}))$
4. ***Juvenile:*** *A young fish or animal that has not reached sexual maturity.*
   $\text{Juvenile} \equiv (\text{Young} \sqcap (\text{Fish} \sqcup \text{Animal}) \sqcap \neg \exists \text{reached}.(\text{Sexual} \sqcap \text{Maturity}))$
5. ***Tetraploid:*** *Cell or organism having four sets of chromosomes.*
   $\text{Tetraploid} \equiv ((\text{Cell} \sqcup \text{Organism}) \sqcup =4 \text{ having}.(\text{Set} \sqcap \exists \text{of}.\text{Chromosomes}))$
6. ***Pair Trawling:*** *Bottom or mid-water trawling by two vessels towing the same net.*
   $\text{PairTrawling} \equiv ((\text{Bottom} \sqcup \text{MidWater}) \sqcap \text{Trawling} \sqcap =2 \text{ by}.(\text{Vessel} \sqcap \exists \text{tow}.(\text{Same} \sqcap \text{Net})))$
7. ***Sustained Use:*** *Continuing use without severe or permanent deterioration in the resources.*
   $\text{SustainedUse} \equiv (\text{Continuing} \sqcap \text{Use} \sqcap \neg \exists \text{with}.((\text{Severe} \sqcup \text{Permanent}) \sqcap \text{Deterioration} \sqcap \exists \text{in}.\text{Resources}))$
8. ***Biosphere****: The portion of Earth and its atmosphere that can support life.*
   $\text{Biosphere} \equiv (\text{Portion} \sqcap \exists \text{of}.((\text{Earth} \sqcup (\text{Its} \sqcap \text{Atmosphere})) \sqcup \exists \text{can\_support}.\text{Life}))$
9. ***Vehicles*** *are non-living means of transportation.*
   $\text{Vehicle} \equiv (\neg \text{Living} \sqcap \text{Means} \sqcap \exists \text{of}.\text{Transportation})$
10. *A* ***minister*** *or a secretary is a politician who holds significant public office in a national or regional government.*
    $(\text{Minister} \sqcup \text{Secretary}) \equiv (\text{Politician} \sqcap \exists \text{holds}.((\text{Office} \sqcap \text{Significant} \sqcap \text{Public}) \sqcap \exists \text{in}.(\text{Government} \sqcap (\text{National} \sqcup \text{Regional}))))$
11. *A* ***currency*** *is a unit of exchange, facilitating the transfer of goods and services.*
    $\text{Currency} \equiv (\text{Unit} \sqcap \exists \text{of}.\text{Exchange} \sqcap \exists \text{facilitate}.(\text{Transfer} \sqcap \exists \text{of}.(\text{Good} \sqcap \text{Service})))$
12. *An* ***island*** *or isle is any piece of land that is completely surrounded by water.*
    $(\text{Island} \sqcup \text{Isle}) \equiv (\text{Piece} \sqcap \exists \text{of}.Land \sqcap \exists completely\_surrounded\_by.Water)$
13. ***Days of the week*** *are: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.*
    $\text{DayOfWeek} \equiv \{\text{Monday}, \text{Tuesday}, \text{Wednesday}, \text{Thursday}, \text{Friday}, \text{Saturday}, \text{Sunday}\}$

[5] `http://www.neon-project.org`
[6] `http://sekt-project.com`

Some critical remarks and observations on the examples:

1. This is a simple example, which works out very well.
2. This example shows the complex axiomatizations which can be obtained using our approach. Here (and in other examples) we note that adjectives are so far interpreted as being intersective – we discuss this in Section 4. Another recurring problem is the generic nature of the role *of*. Nevertheless, the output is a reasonable approximation of the intended meaning and would serve well as suggestion for an ontology engineer within an interactive process as we will draft in Section 5.
3. This is a Minipar parse error. The desired solution would be Vector $\equiv$ (Organism $\sqcap$ ($\exists$carry.Pathogen $\sqcup$ transmit.Pathogen)).
4. Take particular attention to the handling of negation and of the present perfect tense.
5. The natural language sentence is actually ambiguous whether the number should be read as *exactly four* or *at least four*, and the role name *having* is certainly not satisfactory. Even more difficult is how *set of chromosomes* is resolved. A correct treatment is rather intricate, even if modeling is done manually. The class name *Chromosomes* should probably rather be a nominal containing the class name as individual – which cannot be modeled in OWL DL, but only in OWL Full. Note also that the cardinality restriction is used as a so-called *qualified* one, which is not allowed in OWL DL but is supported by most DL reasoners.
6. *Same* is difficult to resolve. In fact, OWL DL is not expressive enough for properly modeling the sentence!
7. Apart from the very generic role *in* and the problem with adjectives already mentioned, this is a complex example which works very well.
8. The possessive pronoun *its* would have to be resolved.
9. Here, *means of transportation* should probably not be further broken down.
10. *Or* is ambiguous. Here it indicates that *minister* and *secretary* mean the same.
11. The word *and* actually indicates a disjunction in this example.
12. The handling of the adverb *completely* is insufficient.
13. This example would be easy to implement, but we have not done it yet because KAON2 currently cannot handle nominals.

4 Critical Discussion

The syntactic transformation proposed in Section 3 creates a set of OWL axioms which can be used to extend the axiomatization of any given class in an ontology. Our naive implementation of this approach is as simple as efficient, but obviously requires a significant amount of manual or automatic post-processing. This is to a major extent due to a number of problems which relate to limitations of the linguistic analysis and the transformation process, as well as fundamental differences between lexical and ontological semantics. In the following we will discuss some of these problems in more detail, and present possible solutions.

Semantic Aspects. Although the transformation takes into account some aspects of lexical semantics, it is certainly not capable of capturing much of the intension of the terms involved in the natural language expression that serves as an input for the transformation process. Much of the meaning of the resulting axioms is still brought in by the

semantics of the underlying natural language terms. This does not necessarily constitute a significant problem as long as the semantics of the description logic expressions is sufficiently "in line" with the lexical semantics of the terms involved in their formation. Actually, the semantics of ontological elements – not of the constructs of the ontology language, but of the classes, properties and instances defined by means of these constructs – will always be grounded to some extent in natural language semantics.

As it is impossible to express all possible aspects of a concept's meaning by virtue of description logic axioms, natural language labels and comments undoubtedly play a key role in ontological knowledge representation. In fact, an ontology without natural language labels attached to classes or properties is almost useless, because without this kind of grounding it is very difficult, if not impossible, for humans to map an ontology to their own conceptualization, i.e. the ontology lacks human-interpretability.

However, a grounding of ontologies in natural language is highly problematic due to different semantics and the dynamic nature of natural language. It is important to mention that many problems linked to either of these aspects are not necessarily specific to ontology learning approaches such as the one we present in this paper. Since the way people conceive and describe the world is very much influenced by the way they speak and vice-versa (also known as the *Sapir-Whorf hypothesis*), ontology engineering is often subject to our intuitive understanding of natural language semantics.

Lexical Semantics. The semantics of lexical relations fundamentally differs from the model-theoretic semantics of ontologies. While lexical relations such as hyponymy, antinomy or synonymy are defined over lexemes, ontological relations are used for relating classes.[7] And it is not obvious in all cases how to map words – especially very abstract notions – to classes, as their extension often remains unclear.

For practical reasons it might be sensible to assume a correspondence between lexical relations and some types of axioms. Traditional ontology learning approaches often rely on information about hyponymy for creating subsumption hierarchies [5], or meronymy for identifying part-of relationships [6]. However, one has to be aware of the fact that a one-to-one mapping between lexical and model-theoretic semantics may affect the formal correctness of ontologies – even more, if ontology learning or engineering exclusively relies on clues by means of lexico-syntactic patterns for inferring lexical relationships. Due to the informal character of natural language it is no trouble to say, for instance, "*A person is an amount of matter*". But from the perspective of formal semantics this might be problematic as pointed out by [7].

Dynamics of Natural Language. Further problems with respect to the use of natural language in ontology engineering relate to the way in which semantics are defined. While ontologies have a clear model-theoretic semantics, the semantics of lexical relations is defined by so-called *diagnostic frames*, i.e. by typical sentences describing the context in which a pair of words may or may not occur given a certain lexical relation among them. This way of defining lexical relations does not guarantee for stable

[7] For example, each of these classes could be associated with one or more natural language expressions describing the intended meaning (intension) of the class. And still, since hyponymy is not "transitive" over (near-)synonymy it is not necessarily the case that all mutually synonymous words associated with a subclass are hyponyms of all synonymous words associated with its superclass.

semantics, since natural languages, other than ontology representation languages, are dynamic. That means, each (*open-class*) word slightly changes its meaning every time it is used in a new linguistic context. These *semantic shifts*, if big enough, can affect the lexical relationships between any pair of words. And considering that natural language expressions are regularly used for the grounding of ontologies they can potentially lead to semantic "inconsistencies", i.e. conflicting intensional descriptions. This kind of inconsistencies can be avoided by more precise, formal axiomatizations of ontological elements. However, it is an open issue how many axioms are required to "pin down" the meaning of a given class or property.

Technical Problems. A significant objection one might have with respect to the technical implementation of our approach certainly refers to the rather sophisticated linguistic analysis which is required prior to the actual transformation process. And indeed, the Minipar[8] dependency parser we use sometimes fails to deliver a parse, particularly in the case of ill-formed or structurally complex sentences, or wrongly resolves **syntactic ambiguities** such as prepositional phrase attachments. However, dependency parsers are known to be much more robust than parsers using phrase structure grammar, for example. And as most of our transformation rules can be mapped to surface structure heuristics (see Table 1) in a relatively straightforward way, a chunker or shallow parser could complement Minipar in case of failure. Efficiency is not so much an issue as the parser is extremely fast, processing a few hundred sentences per second.

However, there are more severe problems apart from the quality or efficiency of the syntactic analysis. Many of them concern **semantic ambiguity** related to quantifier scope (e.g. "*any*", see example *Island*) or homonymy (e.g. "*net*"). Both types of problems are not appropriately handled at the moment. And our implementation also lacks an **anaphora resolution** step which would help to identify antecedents of pronouns (e.g. "*its atmosphere*") or nominal anaphora, for instance. Although some types of coreference including relative pronouns can be handled by Minipar itself, the language we defined for describing the transformation rules is not expressive enough to deal with phenomena such as long distance dependencies or deictic expressions. Therefore, user intervention is still essential during the post-processing phase to replace pronouns and to map co-referring nominals to the same class. Similarly, depending on the desired degree of modeling granularity, user input might be required to support the semantic analysis of **compound nominals** (e.g. "*Pair Trawling*").

Moreover, the different semantics of **adjectives** are not taken into account by the translation rules. Ideally, one would have to distinguish between at least three types of adjectives – subsective ($Young\_Fish \sqsubseteq Fish$), intersective ($Sexual\_Maturity \sqsubseteq (Sexual \sqcap Maturity)$) and privative ($Fake\_Fish \sqsubseteq \neg Fish$). But since an automatic classification of adjectives into these classes as proposed by [8], for example, is a very challenging task, we currently assume intersective semantics for all adjectives. Even more difficult is the semantics of **adverbs** (e.g. "*completely surrounded*") and some types of auxiliary verbs which express a spatial, temporal or behavioral modality (e.g. "*can support life*"). And of course, temporal relationships expressed by past or future **verb tense** are also very difficult to handle without temporal reasoning.

[8] `http://www.cs.ualberta.ca/~lindek/minipar.htm`

Another problem which is not yet sufficiently handled by our transformation rules are so-called **empty heads**, i.e. nominals which do not contribute to the actual meaning of a genus phrase (cf. the "any" in the *Island* example). In particular, the rules relying on Hearst-style patterns [5] for the identification of hyponymy relationships may be mislead by expressions such as *one*, *any*, *kind*, *type*. This phenomenon has already been described [9, 10] and could be handled by appropriate exception rules. An alternative solution to this and similar problems could be to increase the expressiveness of the **rule language** used in the transformation process. The language as it is defined by now does not permit the usage of regular expressions, for instance, which might be valuable means to generalize particular transformation rules. XSLT and *tgrep* could help to overcome these limitations.

Finally, our approach is restricted to texts with **definitory character** such as glossary entries or encyclopedic descriptions which have a universal reading and a more or less canonical form, i.e. including a genus category and additional information to distinguish the term from other members of the same category [11]. In order to extend the applicability of LExO to a greater variety of textual resources, one would need a component for the automatic identification of natural language definitions.

Further Remarks. Although we see a great potential in our approach (cf. Section 5), the discussion shows that there are still many open issues – technical, but also very fundamental questions. The most important ones according to our perception relate to the relationship of lexical and ontological semantics. Given a purely syntactical transformation such as ours, it will be crucial to investigate at which stage of the process and in which manner particularities of both semantics have to be considered. And finally, we will have to answer the question where the principal limitations of our approach with respect to the expressivity of the learned ontologies really are. It is reasonable to assume that at least some aspects of ontological semantics cannot (or not so easily) be captured by purely lexical ontology learning methods. However, we believe that a combination of lexical and logical approaches could help to overcome these limitations.

5 Realising the Vision

Despite the fact that our approach currently has a number of limitations as pointed out in Section 4, we believe that it has the potential to become a valuable component of a semi-automatic ontology engineering environment. Many of the technical drawbacks of the approach can be alleviated by integrating more sophisticated methods for natural language processing, ontology mapping or evaluation, and as a matter of course, by adding a human factor to the ontology acquisition process.

In this section we sketch our vision of a semi-automatic ontology engineering process involving a set of complementary methods for ontology engineering and evaluation along with an elaborate methodology. We describe the potential role of our approach within this scenario and identify the missing components.

Semi-automatic Ontology Engineering The overall scenario we envision for engineering expressive OWL ontologies is a semi-automatic cyclic process of ontology learning, evaluation and refinement, see Figure 6. The process starts with a relatively inexpressive ontology, possibly a bare taxonomy given in RDFS, which is supposed to

be enriched and refined to meet the requirements, e.g. of a reasoning-based application. In each iteration of the process, the user selects the class to be refined, and optionally specifies appropriate resources for the ontology generation phase (Step 1) such as

- manual user input,
- comments contained in the ontology,
- definitions extracted from ontology engineering discussions by email or Wiki,
- documentation of the underlying application and use cases,
- available glossaries and encyclopedias (e.g. Wikipedia), or
- textual descriptions of the domain which could be obtained by initiating a Google™search for definitions (e.g. "*define: DNS*").

A tool such as LExO can analyze the given resources to identify and extract definitory sentences, i.e. natural language descriptions of the class previously selected by the user. These definitions are parsed and transformed into OWL DL axioms (Step 2) that can be presented to the user, if she wants to intervene at this point. Otherwise, the system directly proceeds to the mapping phase which aims at relating the newly generated entities and axioms to elements in the initial ontology (Step 3). The outcome of this phase are a number of mapping axioms which can be added to the class axiomatization after being confirmed by the user.

Then, methods for ontology evaluation check for logical inconsistencies or potential modeling errors (Step 4). Based on the learned axiomatization and additional mappings the system now suggests ontology changes or extensions to the user (Step 5). The user now revises the ontology by modifying or removing some of the axioms (Steps 6 and 7), before the whole process starts over again. Further entities, e.g. those introduced by previous iterations, can be refined until the user or application needs are satisfied.

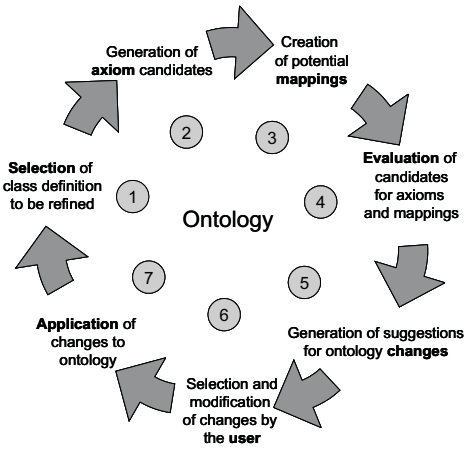

Fig. 6. Ontology Refinement Process

Ontology Evaluation. It is certainly necessary to add further functionalities to the interactive process. We point out two aspects which we judge to be of particular importance, namely how to aid the ontology engineer to ensure high *quality* of the ontology, and to ensure *completeness* of the modeling process in terms of the application domain.

Quality insurance will have to be based on previous work on the field of ontology evaluation. Since the automatic generation of expressive ontologies can potentially lead to a substantial increase in complexity, a simple manual revision of the ontology generated by a system such as the one described here might be infeasible. Therefore, we believe that automatic techniques for ontology evaluation will play a crucial role in the ontology learning and engineering cycle. These techniques could check, for instance, the ontology's validity with respect to the OntoClean methodology [12], or assure the logical consistency of the ontology. In particular, debugging techniques like pinpointing

[13] will be indispensable as soon as cardinality restrictions or any kinds of negation (e.g. class complement, disjointness) are introduced into learned ontologies.

Since this aspect has not emerged in lexical ontology learning up to now, the problem of integrating ontology evaluation and debugging into the learning process has not received much attention yet. As pointed out in [14] we see a great potential in exploiting metadata such as confidence and relevance values generated during the ontology learning process for resolving inconsistencies in learned ontologies. But still, the perfect synthesis of ontology learning and evaluation is a challenging problem.

In order to ensure completeness of the modeling process in terms of the application domain, a structured approach for an exhaustive exploration of complex relationships between classes is required. This can be realized e.g. by employing methods like *relational exploration* [15] which is an adaptation of attribute exploration from Formal Concept Analysis [16] to description logics. And finally, it might also be worthwhile to consider an integration of LExO with other learning approaches which could compensate for some of its limitations, e.g. with respect to the learnability of particular relations between roles [17], or disjointness axioms [18].

The issues discussed for creating an interactive ontology engineering tool are under investigation by the authors. In the medium term, we expect to develop a corresponding system as part of a powerful ontology engineering environment like OntoStudio, Protégé, or the forthcoming NeOn Toolkit[9]. It will also be worth investigating the use of LExO for automated question answering. Integrating LExO into any of these application scenarios will allow for a much more target-oriented evaluation of our approach.

6 Related Work and Conclusions

We have presented an approach which can support the semi-automatic engineering of ontologies by automatically processing dictionary definitions or ontology comments and translating these to axioms exploiting the expressive power of description logic languages, in particular $\mathcal{SHOIN}(\mathbf{D})$, i.e. OWL DL. An alternative to automatically processing definitory descriptions is to offer a natural language interface allowing users to interact with an ontology editor using natural language. Recently, several approaches relying on controlled language have been presented [19–21]. The drawback of such approaches is that users have to actually learn a restricted language which might sometimes even seem unnatural [22]. Though our approach also has limitations, it aims at processing language as used in dictionary definitions without notable restrictions.

There is also a large body of work on dictionary parsing reaching back to the 80s and 90s [23, Chapter 6]. Recent work has focused on extracting knowledge from online glossaries [24] and Wikipedia [25–27]. However, most of the work on processing machine-readable dictionaries up to now has mainly focused on extracting lexical relations, in particular hyponymy or meronymy relations. The aspect which distinguishes our approach from others is the fact that it aims at exploiting an expressive description logic language. In this line, our approach is related to the work of Gardent and Jacquey [28], who translate hypernyms, troponyms and antonyms as found in WordNet into DL axioms to be used within a question answering application.

[9] `http://www.neon-project.org`

Other work which indeed has aimed at inducing DL class descriptions from data are the ones of Lisi et al. [29] as well as Fanizzi et al. [30]. However, these approaches rely on extensional data, i.e. they assume the availability of an ABox from which a TBox is obtained by generalization. It remains an open issue how, and if these approaches can be applied to learning knowledge bases from texts.

Summarizing, the potential of our treatment lies in its flexibility and simpleness, as well as its suitability for an interactive process as spelled out in Section 5. We have reported on the decisive initial steps in realizing this vision, and accompanied it with a critical discussion of the obstacles which need to be overcome. We believe that these efforts will eventually result in an interactive system which will aid ontology engineers in the construction of expressive OWL DL ontologies.

Acknowledgments This research has been partially supported by the European Commission under contracts IST-2003-506826 SEKT, IST-2006-027595 NeOn and IST-FP6-026978 X-Media, and by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project (grant 01 IMD01 B).

References

1. W3C: Web Ontology Language (OWL) (2004) `http://www.w3.org/2004/OWL/`.
2. Baader, F., et al., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Lin, D.: Dependency-based evaluation of MINIPAR. In: Proceedings of the Workshop on the Evaluation of Parsing Systems. (1998)
4. Terziev, I., Kiryakov, A., Manov, D.: Base Upper-Level Ontology (BULO) guidance. SEKT deliverable 1.8.1, Ontotext Lab, Sirma AI EAD (Ltd.) (2004)
5. Hearst, M.: Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th International Conference on Computational Linguistics. (1992) 539–545
6. Poesio, M., Ishikawa, T., im Walde, S.S., Vieira, R.: Acquiring lexical knowledge for anaphora resolution. In: Proceedings of the 3rd Conference on Language Resources and Evaluation. (2002)
7. Guarino, N., Welty, C.A.: A formal ontology of properties. In: Knowledge Acquisition, Modeling and Management. (2000) 97–112
8. Amoia, M., Gardent, C.: Adjective based inference. In: Proceedings of the EACL Workshop on Knowledge and Reasoning for Answering Questions (KRAQ'06). (2006)
9. Guthrie, L., Slator, B., Wilks, Y., Bruce, R.: Is there content in empty heads? In: Proceedings of the 13th conference on Computational linguistics, Morristown, NJ, USA, Association for Computational Linguistics (1990) 138–143
10. Chodorow, M., Byrd, R., Heidorn, G.: Extracting semantic hierarchies from a large on-line dictionary. In: Proceedings of the 23rd annual meeting on Association for Computational Linguistics. (1985) 299–304
11. Klavans, J., Popper, S., Passonneau, B.: Tackling the internet glossary glut: Automatic extraction and evaluation of genus phrases. In: Proceedings of the SIGIR'03 Workshop on Semantic Web. (2003)
12. Völker, J., Vrandecic, D., Sure, Y.: Automatic evaluation of ontologies (AEON). In: Proc. of the 4th International Semantic Web Conference (ISWC2005). Volume 3729 of LNCS., Springer (2005) 716–731
13. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: Proc. of the 2nd European Semantic Web Conference (ESWC'05). Volume 3532 of LNCS., Springer (2005) 226–240

14. Haase, P., Völker, J.: Ontology learning and reasoning – dealing with uncertainty and inconsistency. In: Proc. of the ISWC Workshop on Uncertainty Reasoning for the Semantic Web (URSW). (2005) 45–55
15. Rudolph, S.: Exploring relational structures via FLE. In Wolff, K.E., Pfeiffer, H.D., Delugach, H.S., eds.: Conceptual Structures at Work: 12th International Conference on Conceptual Structures. Volume 3127 of LNCS., Springer (2004) 196–212
16. Ganter, B., Wille, R.: Formal Concept Analysis – Mathematical Foundations. Springer, Berlin (1999)
17. Lin, D., Pantel, P.: DIRT – discovery of inference rules from text. In: Knowledge Discovery and Data Mining. (2001) 323–328
18. Völker, J., Vrandecic, D., Sure, Y., Hotho, A.: Learning disjointness. In: Proceedings of the 4th European Semantic Web Conference (ESWC'07). (2007)
19. Tablan, V., Polajnar, T., Cunningham, H., Bontcheva, K.: User-friendly ontology authoring using a controlled language. In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'06). (2006)
20. Bernstein, A., Kaufmann, E., Göhring, A., Kiefer, C.: Querying ontologies: A controlled english interface for end-users. In: Proceedings of the 4th International Semantic Web Conference (ISWC'05). (2005) 112–126
21. Pease, A., Murray, W.: An English to Logic Translator for ontology-based knowledge representation languages. In: Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering. (2003) 777–783
22. Fuchs, N., Kaljurand, K., Schneider, G.: Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. In: Proceedings of FLAIRS'06. (2006)
23. Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer Verlag (2006)
24. Hovy, E., Philpot, A., Klavans, J., Germann, U., Davis, P., Popper, S.: Extending metadata definitions by automatically extracting and organizing glossary definitions. In: Proceedings of the 2003 annual national conference on Digital government research, Digital Government Research Center (2003) 1–6
25. Ruiz-Casado, M., Alfonseca, E., Castells, P.: Automatic extraction of semantic relationships for WordNet by means of pattern learning from Wikipedia. In: Proceedings of the International Conference on Natural Language for Information Systems (NDLB'05). Number 3513 in LNCS, Springer Verlag (2005) 67–79
26. Suh, S., Halpin, H., Klein, E.: Extracting common sense knowledge from Wikipedia. In: Proceedings of the Workshop on Web Content Mining with Human Language Technologies at ISWC'06. (2006)
27. Weber, N., Buitelaar, P.: Web-based ontology learning with ISOLDE. In: Proc. of the ISWC Workshop on Web Content Mining with Human Language Technologies. (2006)
28. Gardent, C., Jacquey, E.: Lexical reasoning. In: Proceedings of the International Conference on Natural Language Processing (ICON'03). (2003)
29. Lisi, F., Esposito, F.: ILP meets knowledge engineering: A case study. In: Proc. of the International Conference on Inductive Logic Programming (ILP), Springer (2005) 209–226
30. Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Concept formation in expressive description logics. In: Proc. of the 15th European Conference on Machine Learning (ECML'04), Springer Verlag (2004)