Knowledge-Intensive Induction of Terminologies from Metadata

Floriana Esposito, Nicola Fanizzi, Luigi Iannone, Ignazio Palmisano, and Giovanni Semeraro

Dipartimento di Informatica, Università degli Studi di Bari
Via Orabona 4, 70125 Bari, Italy
{esposito,fanizzi,iannone,palmisano,semeraro}@di.uniba.it

Abstract. We focus on the induction and revision of terminologies from metadata. Following a Machine Learning approach, this setting can be cast as a search problem to be solved employing operators that traverse the search space expressed in a structural representation, aiming at correct concept definitions. The progressive refinement of such definitions in a terminology is driven by the available extensional knowledge (metadata). A knowledge-intensive inductive approach to this task is presented, that can deal with on the expressive Semantic Web representations based on Description Logics, which are endowed with well-founded reasoning capabilities. The core inferential mechanism, based on multilevel counterfactuals, can be used for either inducing new concept descriptions or refining existing (incorrect) ones. The soundness of the approach and its applicability are also proved and discussed[1].

1 Introduction

The Semantic Web (SW) relies on ontological knowledge in order to gain a semantic interoperability for organizing and processing resources on the ground of their meaning. Though SW actors adopt ontologies as a formal tool for semantic convergence, these are the outcome of a complex engineering process of acquisition and harmonization that is still limitedly machine-aided. Automated reasoning is typically exploited to assist knowledge engineers in the detection of failures in the ontologies. However, harder problems involving (non standard) inferences, such as the induction or revision of defective semantic knowledge is typically delegated to knowledge engineers. The actors ought to be able not only to detect inconsistent sections in the knowledge bases shared with their *peers*, but also to adopt proper counter-measures to revise them accordingly and maintain a sound source of semantic information. A particular aspect of this kind of

[1] This research was partially funded by the European Commission under the IST Integrated Project VIKEF - Virtual Information and Knowledge Environment Framework (Contract no. 507173 - more information at http://www.vikef.net) and by National Ministry of Instruction University and Research Project COFIN 2003 "Tecniche di intelligenza artificiale per il reperimento di informazione di qualita' sul Web".

S.A. McIlraith et al. (Eds.): ISWC 2004, LNCS 3298, pp. 441–455, 2004.

problems is investigated. A domain ontology, supposed to be described by an ontology data model, proves itself incorrect (incomplete or inconsistent) with respect to processed assertional knowledge (metadata). Such a flawed information undermines the very foundations of semantic interoperability. Ideally, the required knowledge revision should be operated in a (semi)automatic fashion at a machine level so that the knowledge engineer's duties reduce to the mere checking of the outcomes. The induction and revision of structural knowledge is not new issue for *Machine Learning*, though it has been investigated for poorly expressive languages. Description Logics (henceforth DLs) and derived markup languages are a sort of *de facto* standard for representing ontologies owing to their well-founded semantics and available reasoning services [1]. DLs represent a peculiar fragment of First Order Logic that is comparable to graph representations. The induction of such a structural knowledge is known as a hard task [2]. Recently, in *Logic Programming* contexts, attempts have been made to extend relational learning techniques toward more expressive languages such as *prenex conjunctive normal forms* [3] or hybrid representations [4,5]. In order to cope with the complexity issues, former methods use heuristic search and generally implement bottom-up algorithms, such as the *least common subsumer* (*lcs*) [6]. Yet this induces overly specific concept definitions, which may have poor predictive capabilities (the phenomenon of *overfitting*). Moreover, for many DLs, it is well known that lcs's do not yield a compact representation of the generalizations [7]. Hence, other works propose heuristic generalizations [8].

Further approaches have shown that also a top-down search of concept definitions is feasible [9], yet the proposed methods are less operational. They can be considered as a theoretical study of the search space properties since the exploitation of the assertions in the refinement process is neglected, in favor of a generate and test strategy that can turn out to be in practice very inefficient.

A more knowledge-intensive refinement method is needed for concept definitions. Besides, when new assertions become available and turn out to be inconsistent w.r.t. the previously defined ontology, it is computationally expensive to re-build the the concept definitions accordingly. Thus the revision process should be possibly performed incrementally on the ground of the metadata.

In our investigation the learning problem is cast as a search problem in the space determined by an expressive DL representation. Imposing an order over the space of concept descriptions, we give the definition of theoretical operators that can traverse the space seeking a solution to the problem. Intending to give operational instruments for inference and refinement of conceptual descriptions, we introduce an algorithm based on multilevel counterfactuals [10] for operating with an $\mathcal{ALC}$ [11] representation (yet extensions to other DLs should be possible).

The method is based on mutually recursive cycles of specialization and generalization with respect to the (newly) available assertions (examples) regarding the concepts to be revised. The intuition is that specialization can be obtained by learning exceptions, while generalizations stand as exceptions to exceptions, and so on. We show how this method can be adapted to the DLs representation,

Table 1. The constructors for $\mathcal{ALC}$ descriptions and their interpretation.

Name	Syntax	Semantics
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
concept conjunction	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
concept disjunction	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y\ (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y\ (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$

converging to correct concept definitions (with respect to the intended model of the knowledge).

The paper is organized as follows. After the next preliminary section introducing the representation, in Sect. 3 the search space and its properties are presented. The method for learning and refining DL concept definitions is illustrated in Sect. 4, where its applicability is also briefly analyzed. Finally, possible extensions of the method are discussed in Sect. 5.

2 Syntax and Semantics

Most of the SW languages (RDF Schema, DAML+OIL, OWL) have their theoretical foundations in Description Logics. These formalisms are fragments of First Order Logic with different degrees of expressive powers. They rely on well-founded descriptive style semantics and offer efficient reasoning services (deductive inferences) such as subsumption, consistency and instance check. It is straightforward that the efficiency of reasoning is strongly dependent on the language expressiveness [1]. In this section we recall syntax and semantics for the reference representation $\mathcal{ALC}$ [11] adopted in the paper, although the method is also extensible to other DLs. Moreover, examples are shown where the DL constructs are mapped onto the descriptions in SW languages.

In a DL language, primitive *concepts* $N_C = \{C, D, \ldots\}$ are interpreted as subsets of a certain domain of objects (resources) and primitive *roles* $N_R = \{R, S, \ldots\}$ are interpreted as binary relations on such a domain (properties). More complex concept descriptions are built using atomic concepts and primitive roles by means of the constructors in Table 1. Their meaning is defined by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the *domain* of the interpretation and the functor $\cdot^{\mathcal{I}}$ stands for the *interpretation function*, mapping the intension of concepts and roles to their extension.

A *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains two components: a T-box $\mathcal{T}$ and an A-box $\mathcal{A}$. $\mathcal{T}$ is a set of concept definitions $C \equiv D$, meaning $C^{\mathcal{I}} = D^{\mathcal{I}}$, where C is the concept name and D is a description given in terms of the language constructors. $\mathcal{A}$ contains extensional assertions on concepts and roles, e.g. $C(a)$ and $R(a, b)$, meaning, respectively, that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

The semantic notion of *subsumption* between concepts (or roles) can be given in terms of the interpretations:

Definition 2.1 (subsumption). *Given two concept descriptions C and D in $\mathcal{T}$, C* subsumes *D, denoted by $C \sqsupseteq D$, iff for every interpretation $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \supseteq D^{\mathcal{I}}$. Hence, $C \equiv D$ amounts to $C \sqsupseteq D$ and $D \sqsupseteq C$.*

Example 2.1. An instance of concept definition in the proposed language is:
$Father \equiv Human \sqcap Male \sqcap \exists hasChild.Human$
which translates the sentence: *"a father is a male human that has some humans as his children"*. A-box assertions look like:
$Father(Tom), Father(Bill), hasChild.Human(Bill, Joe)$ and so on.
Now, if we define two new concepts:
$FatherWithoutSons \equiv Human \sqcap Male \sqcap \exists hasChild.Human \sqcap \forall hasChild.(\neg Male)$
$Parent \equiv Human \sqcap (Male \sqcup Female) \sqcap \exists hasChild.Human$
then it is easy to see that $Father \sqsupseteq FatherWithoutSons$ and $Parent \sqsupseteq Father$, yet $Father \not\sqsupseteq Parent$ and $FatherWithoutSons \not\sqsupseteq Father$
The corresponding OWL/XML representation can be found in Figure 1.

Many semantically equivalent (yet syntactically different) descriptions can be given for the same concept. However they can be reduced to a canonical form by means of rewriting rules that preserve their equivalence, e.g. $\forall R.C_1 \sqcap \forall R.C_2 \equiv \forall R.(C_1 \sqcap C_2)$ (see [1] for normalization and simplification issues). Preliminarily, some notation is needed to name the different parts of an $\mathcal{ALC}$ description: $\mathsf{prim}(C)$ is the set of all the concepts at the top-level conjunction of C; if there exists a universal restriction $\forall R.C'$ on the top-level of C then $\mathsf{val}_R(C) = C'$ otherwise $\mathsf{val}_R(C) = \top$ (a single one because of the equivalence mentioned above). Finally, $\mathsf{ex}_R(C)$ is the set of the concept descriptions C' appearing in existential restrictions $\exists R.C'$ at the top-level conjunction of C.

Definition 2.2 (normal form). *A concept description D is in $\mathcal{ALC}$* normal form *iff $D \equiv \bot$ or $D \equiv \top$ or if $D = D_1 \sqcup \cdots \sqcup D_n$ with*

$$D_i = \prod_{A \in \mathsf{prim}(D_i)} A \quad \sqcap \prod_{R \in N_R} \left[\prod_{V \in \mathsf{val}_R(D_i)} \forall R.V \quad \sqcap \prod_{E \in \mathsf{ex}_R(D_i)} \exists R.E \right]$$

where, for all $i = 1, \ldots, n$, $D_i \not\equiv \bot$ and, for any R, every sub-description in $\mathsf{ex}_R(D_i)$ and $\mathsf{val}_R(D_i)$ is in normal form.

This form will be employed for defining an ordering over the concept descriptions and the refinement operators.

3 Induction as Search

The problem of induction in its simplest form can be now formally defined as a supervised learning task:

```
<!ENTITY ns1 "http://www.myontos/Friend#">
<rdf:RDF ... >
  <owl:Class rdf:about="&ns1;FatherWithoutSons">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty><rdf:Property rdf:about="&ns1;hasChild"/>
        </owl:onProperty>
        <owl:minCardinality>1</owl:minCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf><owl:Class rdf:about="&ns1;Male"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty><rdf:Property rdf:about="&ns1;hasFemaleChild"/>
        </owl:onProperty>
        <owl:maxCardinality>0</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="&ns1;Female">
    <rdfs:subClassOf><owl:Class rdf:about="&ns1;Human"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="&ns1;MaleOrFemale">
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="&ns1;Male"/>
      <owl:Class rdf:about="&ns1;Female"/>
    </owl:unionOf>
  </owl:Class>
  <owl:Class rdf:about="&ns1;Parent">
    <rdfs:subClassOf rdf:resource="&ns1;Human"/>
    <rdfs:subClassOf rdf:resource="&ns1;MaleOrFemale"/>
  </owl:Class>
  <owl:Class rdf:about="&ns1;Male">
    <rdfs:subClassOf rdf:resource="&ns1;Human"/>
  </owl:Class>
  <rdf:Property rdf:about="&ns1;hasFemaleChild">
    <rdfs:range rdf:resource="&ns1;Female"/>
    <rdfs:subPropertyOf rdf:resource="&ns1;hasChild"/>
  </rdf:Property>
</rdf:RDF>
```

Fig. 1. OWL/XML representation of Example 2.1

Definition 3.1 (learning problem). *In a search space* $(\mathcal{S}, \succeq)$
Given *a knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *and a set of positive and negative assertions* $\mathcal{A}_C = \mathcal{A}_C^+ \cup \mathcal{A}_C^-$ *regarding the membership (or non-membership) of some*

individuals to a concept C *such that:* $\mathcal{T} \not\models \mathcal{A} \cup \mathcal{A}_C$
Find *a new T-box* $\mathcal{T}' = (\mathcal{T} \setminus \{C \equiv D\}) \cup \{C \equiv D'\}$ *such that:* $\mathcal{T}' \models \mathcal{A} \cup \mathcal{A}_C$

Thus, if a concept C is not defined in the terminology $\mathcal{T}$ we have a case of an *induction problem* requiring to find definitions $C \equiv D$ entailing the (new) assertions in $\mathcal{A}_C$. Conversely, when an existing definition in $\mathcal{T}$ proves to be incorrect i.e. it is uncapable of entailing the positive assertions in $\mathcal{A}_C$ (*incomplete* definition) or it entails negative ones (*inconsistent* definition), this yields a *refinement problem* where a new correct definition $C \equiv D'$ is to be found on the ground of the previous one and the new examples.

The induction of the definitions can be modeled as a search in the space of candidate definitions. First, the search space of concept descriptions must be taken into account for the target language. This space is defined once a quasi-ordering on the descriptions is adopted (i.e. subsumption); then, definitions of refinement operators should be given for deciding the traversal of the search space. The (algebraic) properties of the *search space* obviously depend on the ordering imposed over its components (concept descriptions). This notion, in turn, induces a generalization model (a quasi-ordering) that gives a criterion for traversing the space of solutions by means of suitable operators.

In order to traverse the search space seeking for correct concept definitions solving the learning problem, we will employ the notion of particular operators:

Definition 3.2 (refinement operators). *In a quasi-ordered space* $(\mathcal{S}, \succeq)$*, a* downward *(respectively* upward*)* refinement operator ρ *(resp.* δ*) is a mapping from* $\mathcal{S}$ *to* $2^{\mathcal{S}}$*, such that* $D' \in \rho(D)$ *implies* $D \succeq D'$ *(resp.* $D' \in \delta(D)$ *implies* $D' \succeq D$*).* D' *is called a* specialization *(resp.* generalization*) of* D

Here the order should be based on the subsumption relationship. It is now possible to specify refinement operators for the reference search space. Preliminarily, the definition of a difference operator is needed for both conjunctive and disjunctive descriptions. In the former case, $C = C_1 \sqcap \cdots \sqcap C_n$, the difference is the generalized conjunct resulting from removing one conjunct: $C - C_i = \bigsqcap_{k \neq i} C_k$. In the latter case, $D = D_1 \sqcup \cdots \sqcup D_m$, the difference is the specialized disjunct $D - D_j = \bigsqcup_{k \neq j} D_k$. Taking into account the $\mathcal{ALLC}$ normal form, each level of a concept description interleaves disjunctive or conjunctive descriptions. Thus, the operators should accommodate either case.

Definition 3.3 (downward operator). *In the search space* $(\mathcal{ALLC}, \sqsupseteq)$*, the* downward refinement operator $\rho_\sqcup$ *for disjunctive concept descriptions (in* $\mathcal{ALLC}$ *normal form)* $D = D_1 \sqcup \cdots \sqcup D_n$ *is defined as follows:*

- $D' \in \rho_\sqcup(D)$ *if* $D' = D - D_i$ *for some* $1 \leq i \leq n$
- $D' \in \rho_\sqcup(D)$ *if* $D' = (D - D_i) \sqcup D'_i$ *for some* $D'_i \in \rho_\sqcap(D_i),\ 1 \leq i \leq n$

where the downward refinement operator $\rho_\sqcap$*, given a conjunctive concept description* $C = C_1 \sqcap \cdots \sqcap C_m$*, is defined as follows:*

- $C' \in \rho_\sqcap(C)$ *if* $C' = C \sqcap C_{j+1}$ *for some* $C_{j+1} \not\sqsupseteq C$

- $C' \in \rho_{\sqcap}(C)$ *if* $C' = (C - C_j) \sqcap C'_j$ *for some* $1 \leq j \leq m$, *where:*
  - $C'_j = \exists R.D'_j$, $C_j = \exists R.D_j$ *and* $D'_j \in \rho_{\sqcup}(D_j)$ *or*
  - $C'_j = \forall R.D'_j$, $C_j = \forall R.D_j$ *and* $D'_j \in \rho_{\sqcup}(D_j)$

It is straightforward to define the dual upward operator that seeks for more general hypotheses by dropping disjuncts or refining them.

Definition 3.4 (upward operator). *In the search space* $(\mathcal{ALC}, \sqsupseteq)$, *the* upward refinement operator $\delta_{\sqcup}$ *for disjunctive concept descriptions (in* $\mathcal{ALC}$ *normal form)* $D = D_1 \sqcup \cdots \sqcup D_n$ *is defined as follows:*

- $D' \in \delta_{\sqcup}(D)$ *if* $D' = D \sqcup D_{n+1}$ *for some* D_{n+1} *such that* $D_{n+1} \not\sqsupseteq D$
- $D' \in \delta_{\sqcup}(D)$ *if* $D' = (D - D_i) \sqcup D'_i$ *for some* $D'_i \in \delta_{\sqcap}(D_i)$, $1 \leq i \leq n$

where the upward refinement operator $\delta_{\sqcap}$, *given a conjunctive concept description* $C = C_1 \sqcap \cdots \sqcap C_m$, *is defined:*

- $C' \in \delta_{\sqcap}(C)$ *if* $C' = C - C_j$ *for some* $1 \leq j \leq m$
- $C' \in \delta_{\sqcap}(C)$ *if* $C' = (C - C_j) \sqcap C'_j$ *for some* $1 \leq j \leq m$, *where:*
  - $C'_j = \exists R.D'_j$, $C_j = \exists R.D_j$ *and* $D'_j \in \delta_{\sqcup}(D_j)$ *or*
  - $C'_j = \forall R.D'_j$, $C_j = \forall R.D_j$ *and* $D'_j \in \delta_{\sqcup}(D_j)$

Several properties of the refinement operators may be investigated such as *completeness*, *ideality*, *minimality* and *non-redundancy* [9]. Although theoretical results for these operators have been found (i.e. non ideality, completeness, local finiteness), these issues are beyond the intended scope of this paper. Defining both upward and downward operators allows for the specification of an inductive algorithm for T-box refinement that exploits the information provided by the examples and that is presented in the following section.

4 Induction of Concept Descriptions Based on Assertions

The main idea is to combine theoretic refinement operators with heuristics in order to converge toward a solution of the learning problem. The efficiency of this method derives from biasing inductive search with information coming from assertions. These, indeed, can indicate the search directions in order to avoid mere generate and test refinements and converge more rapidly to the solution. The methodology for the induction and refinement of T-boxes proposed in this work is based on the notion of counterfactuals built on the ground of residual learning problems [10]. We discuss its correctness, applicability and some issues related to its complexity.

4.1 The Method

It is assumed that the learning process can start exclusively from the examples and counterexamples in the A-box of the concept for which a new definition is required. This classification is assumed to be given by a trainer (the knowledge

engineer). However, the methodology would apply also for a similar yet different setting where the starting hypothesis for the target concept is already available in a given T-box, which may have turned out to be incorrect (overly general) for entailing some (new) assertions that have been classified as being negative for the target concept.

Each assertion is not processed as such: it is supposed that a representative at the concept language level (*single representation trick*) is preliminarily derived in the form of *most specific concept* (*msc*). The msc required by the algorithm is a maximally specific DL concept description that entails the given assertion. Since, in some DL it does not exist, we consider their approximations up to a certain depth [12]. Hence, in the algorithm the positive and negative examples will be very specific conjunctive descriptions obtained by means of the *realization* [13] of the assertions concerning the target concept. For many DLs the msc cannot be easily computed or simply it is not unique. For our purposes it suffices to have good (upper) approximations. The algorithm relies on two interleaving routines (see Figure 2) performing, respectively, generalization and specialization, that call each other for converging at a correct concept definition.

The generalization algorithm is a greedy covering one: it tries to explain the positive examples by constructing a disjunctive definition. At each outer iteration, a very specialized definition (the msc of an example) is selected as a starting seed for a new partial generalization; then, iteratively, the hypothesis is generalized by means of the upward operator δ (with a heuristic that privileges the refinements that cover the most of the positives) until all positive concept representatives are covered or some negative representatives are explained. In such a case, the current concept definition *ParGen* has to be specialized by some counterfactuals. The co-routine, which receives the covered examples as its input, finds a sub-description K that is capable of ruling out the negative examples previously covered. In the routine for building counterfactuals, given a previously computed hypothesis *ParGen*, which is supposed to be complete (covering the positive assertions) yet inconsistent with respect to some negative assertions, the aim is finding those counterfactuals to be conjuncted to the initial hypothesis for restoring a correct definition, that can rule out the negative instances. The algorithm is based on the construction of residual learning problems based on the sub-descriptions that caused the subsumption of the negative examples, represented by their msc's. In this case, for each model a residual is derived by considering that part of the incorrect definition *ParGen* that did not play a role in the subsumption. The residual will be successively employed as a positive instance of that part of description that should be ruled out of the definition (through negation). Analogously the msc's derived from positive assertions will play the opposite role of negative instances for the residual learning problem under construction. Finally, this problem is solved by calling the co-routine which generalizes these example descriptions and then conjoining its negation of the returned result.

generalization(*Positives*, *Negatives*, *Generalization*)
input *Positives*, *Negatives*: positive and negative instances at concept level;
output *Generalization*: generalized concept definition
begin
$ResPositives \leftarrow Positives$
$Generalization \leftarrow \bot$
while $ResPositives \neq \emptyset$ **do**
  $ParGen \leftarrow$ select_seed($ResPositives$)
  $CoveredPos \leftarrow \{Pos \in ResPositives \mid ParGen \sqsupseteq Pos\}$
  $CoveredNeg \leftarrow \{Neg \in Negatives \mid ParGen \sqsupseteq Neg\}$
  while $CoveredPos \neq ResPositives$ **and** $CoveredNeg = \emptyset$ **do**
    $ParGen \leftarrow$ select($\delta(ParGen), ResPositives$)
    $CoveredPos \leftarrow \{Pos \in ResPositives \mid ParGen \sqsupseteq Pos\}$
    $CoveredNeg \leftarrow \{Neg \in Negatives \mid ParGen \sqsupseteq Neg\}$
  if $CoveredNeg \neq \emptyset$ **then**
    $K \leftarrow$ **counterfactuals**($ParGen$, $CoveredPos$, $CoveredNeg$)
    $ParGen \leftarrow ParGen \sqcap \neg K$
  $Generalization \leftarrow Generalization \sqcup ParGen$
  $ResPositives \leftarrow ResPositives \setminus CoveredPos$
return *Generalization*
end

counterfactuals($ParGen$, $CoveredPos$, $CoveredNeg$, K)
input *ParGen*: inconsistent concept definition
  CoveredPos, *CoveredNeg*: covered positive and negative descriptions
output K: counterfactual
begin
$NewPositives \leftarrow \emptyset$
$NewNegatives \leftarrow \emptyset$
for each $N_i \in CoveredNeg$ **do**
  $NewP_i \leftarrow$ residual($N_i, ParGen$)
  $NewPositives \leftarrow NewPositives \cup \{NewP_i\}$
for each $P_j \in CoveredPos$ **do**
  $NewN_j \leftarrow$ residual($P_j, ParGen$)
  $NewNegatives \leftarrow NewNegatives \cup \{NewN_j\}$
$K \leftarrow$ **generalization**($NewPositives$, $NewNegatives$)
return K
end

Fig. 2. The co-routines used in the method.

Example 4.1. Suppose to have the RDF metadata fragment as in Figure 3, the resulting A-box[2] is:

$$\mathcal{A} = \{IP(j), \neg IP(f), \neg IP(m), \neg IP(h), R(j), F(j, j_1), I(j_1), F(j, j_2), I(j_2), R(f), R(m), F(m, m_1), I(m_1), F(m, m_2), F(h, h_1), F(h, h_2), I(h_2)\}$$

[2] where R stands for *rich*, I stands for *influential* and F stands for the *hasFriend* relationship (an example taken from [9]).

The msc's of the examples in the A-Box are the following[3]:

$msc(j) R \sqcap \exists F.I \sqcap \forall F.I \qquad mmsc(f) = R$

$msc(m) = R \sqcap \exists F.I \qquad mmsc(h) = \exists F.I \sqcap \forall F.I$

The algorithm is presented with the incorrect definition $G_1 = \top$ for the target concept IP. This definition turns out to cover the example description $msc(j)$ and all the counterexample descriptions. Thus the routine *counterfactuals* is in charge for building a residual learning problem by computing the new examples.

Then, $Neg_j^1 = msc(j)$ is the only counterexample and $Pos_f^1 = msc(f)$, $Pos_m^1 = msc(m)$ and $Pos_h^1 = msc(h)$ are the examples for the new learning problem. Suppose Pos_f^1 is chosen as a seed description G_2. It turns out to be overly general since it would cover Pos_f^1, Pos_m^1 and Neg_j^1 but not Pos_h^1. In a new inner call, *counterfactuals* computes the following residuals (beside $Neg_m^2 = \top$):

$Pos_j^2 = Neg_j^1 - G_2 = (R \sqcap \exists F.I \sqcap \forall F.I) \sqcup \neg R = \top \sqcap ((\exists F.I \sqcap \exists \forall F.I) \sqcup \neg R) = (\exists F.I \sqcap \forall F.I) \sqcup \neg R$

$Neg_m^2 = Pos_m^1 - G_2 = (R \sqcap \exists F.I) \sqcup \neg R = \top \sqcap (\neg R \sqcup \exists F.I) = \neg R \sqcup \exists F.I$

Now, a trivial generalization G_3 amounts to the example description Pos_j^2 that does not cover the negative example description Neg_m^2. Then, returning from the inner call, we got a new specialized generalization $G_2 := G_2 \sqcap \neg G_3 = R \sqcap \neg((\exists F.I \sqcap \forall F.I) \sqcup \neg R) = R \sqcap (\neg(\exists F.I \sqcap \forall F.I) \sqcap R) = R \sqcap \neg(\exists F.I \sqcap \forall F.I)$ which is consistent because it now rules out Neg_j^1. Yet it is not complete for it covers Pos_f^1 and Pos_m^1 but not Pos_h^1; then a new generalization problem is issued for producing a second disjunct.

The very Pos_h^1 is used as a seed for the new generalization G_2'. Since it covers the negative description Neg_j^1. Then the counterfactual found for this small problem is $K_2 := Neg_j^1 - Pos_h^1 = (R \sqcap \exists F.I \sqcap \forall F.I) \sqcup \neg(\exists F.I \sqcap \forall F.I) = (R \sqcup \neg(\exists F.I \sqcap \forall F.I)) \sqcap \top = R \sqcup \neg(\exists F.I \sqcap \forall F.I)$ and the corresponding generalization is $G_2' = Pos_h^1 \sqcap \neg K_2 = (\exists F.I \sqcap \forall F.I) \sqcap \neg(R \sqcup \neg(\exists F.I \sqcap \forall F.I) = \neg R \sqcap (\exists F.I \sqcap \forall F.I)$ that covers Pos_h^1 but not Neg_j^1.

Recollecting the two disjuncts $G_2 = (R \sqcap \neg(\exists F.I \sqcap \forall F.I)) \sqcup (\neg R \sqcap (\exists F.I \sqcap \forall F.I))$. Finally, the complete and consistent generalization for the original problem is $G_1 := G_1 \sqcap \neg G_2 = \top \sqcap \neg((R \sqcap \neg(\exists F.I \sqcap \forall F.I)) \sqcup (\neg R \sqcap (\exists F.I \sqcap \forall F.I))) = \neg(R \sqcap \neg(\exists F.I \sqcap \forall F.I)) \sqcap \neg(\neg R \sqcap (\exists F.I \sqcap \forall F.I)) = (\neg R \sqcup (\exists F.I \sqcap \forall F.I)) \sqcap (R \sqcup \neg(\exists F.I \sqcap \forall F.I)) = (R \sqcap (\exists F.I \sqcap \forall F.I)) \sqcup (\neg R \sqcap \neg(\exists F.I \sqcap \forall F.I))$. This generalization is complete and consistent. Its OWL/XML serialization is reported in Figure 4.

4.2 Discussion

For the sake of simplicity, the generalization routine described in the method is non-deterministic in the points where heuristic choices have to be made: among the possible upward refinements computed by the operator δ, those that maximize the number of covered positives should be preferred. The residual operator

[3] Value restriction can be included by making a temporary domain closure assumption, whose extent is limited to the time the msc's computation is being performed. After that, we adopt the Open Domain Assumption, as usual in inductive reasoning.

```
<!ENTITY ns1 "http://www.myontos/Friend#">
...
<rdf:RDF xmlns:ns0='http://www.myontos/Friend#' ... >
 <owl:Ontology rdf:about=''/>
 <owl:ObjectProperty rdf:about='&ns1;F'/>
  <rdf:Description rdf:about='&ns1;f'>
   <rdf:type><owl:Class rdf:about='&ns1;R'/></rdf:type>
  </rdf:Description>
  <owl:Thing rdf:about='&ns1;h'>
    <ns0:F rdf:resource='&ns1;h1'/><ns0:F rdf:resource='&ns1;h2'/>
  </owl:Thing>
  <rdf:Description rdf:about='&ns1;j'>
   <rdf:type><owl:Class rdf:about='&ns1;R'/></rdf:type>
    <ns0:F rdf:resource='&ns1;j1'/><ns0:F rdf:resource='&ns1;j2'/>
  </rdf:Description>
  <rdf:Description rdf:about='&ns1;m'>
   <rdf:type><owl:Class rdf:about='&ns1;R'/></rdf:type>
    <ns0:F rdf:resource='&ns1;m1'/><ns0:F rdf:resource='&ns1;m2'/>
  </rdf:Description>
  <owl:Thing rdf:about='&ns1;h1'/>
  <rdf:Description rdf:about='&ns1;h2'>
   <rdf:type><owl:Class rdf:about='&ns1;I'/></rdf:type>
  </rdf:Description>
  <rdf:Description rdf:about='&ns1;j1'>
   <rdf:type><owl:Class rdf:about='&ns1;I'/></rdf:type>
  </rdf:Description>
  <rdf:Description rdf:about='&ns1;j2'>
   <rdf:type><owl:Class rdf:about='&ns1;I'/></rdf:type>
  </rdf:Description>
  <rdf:Description rdf:about='&ns1;m1'>
   <rdf:type><owl:Class rdf:about='&ns1;I'/></rdf:type>
  </rdf:Description>
  <owl:Thing rdf:about='&ns1;m2'/>
</rdf:RDF>
```

Fig. 3. OWL/RDF representation of Example 4.1

is essentially a difference function [14]. In general a possible definition of the operator could be $C - D = \max\{E \mid E \sqcap D \equiv C\}$, where the maximum depends on the ordering induced by subsumption. Depending on the representation language, other forms of difference are possible, as discussed in [12]. In the case $\mathcal{ALC}$, the difference can be simply defined as $C - D = C \sqcup \neg D$.

Now let us turn to prove that the method actually converges to a solution of the learning problem:

Theorem 4.1 (correctness). *The algorithm eventually terminates computing a correct concept definition.*

Proof. *The two routines are discussed separately.*

```
<!ENTITY ns1 "http://www.myontos/Friend#">
...
<owl:Class rdf:about="&ns1;Learned">
 <rdfs:subClassOf>
  <owl:Class><owl:unionOf rdf:parseType="Collection">
   <owl:Class><owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="&ns1;R"/>
    <owl:Class><owl:intersectionOf rdf:parseType="Collection">
     <owl:Restriction><owl:onProperty rdf:resource="&ns1;F"/>
      <owl:someValuesFrom><owl:Class rdf:about="&ns1;I"/>
      </owl:someValuesFrom></owl:Restriction>
     <owl:Restriction><owl:onProperty rdf:resource="&ns1;F"/>
      <owl:allValuesFrom><owl:Class rdf:about="&ns1;I"/>
      </owl:allValuesFrom></owl:Restriction>
    </owl:intersectionOf></owl:Class>
   </owl:intersectionOf></owl:Class>
   <owl:Class><owl:intersectionOf rdf:parseType="Collection">
    <owl:Class><owl:complementOf><owl:Class rdf:about="&ns1;R"/>
    </owl:complementOf></owl:Class>
    <owl:Class><owl:complementOf>
     <owl:Class><owl:intersectionOf rdf:parseType="Collection">
      <owl:Restriction><owl:onProperty rdf:resource="&ns1;F"/>
       <owl:someValuesFrom><owl:Class rdf:about="&ns1;I"/>
       </owl:someValuesFrom></owl:Restriction>
      <owl:Restriction><owl:onProperty rdf:resource="&ns1;F"/>
       <owl:allValuesFrom><owl:Class rdf:about="&ns1;I"/>
       </owl:allValuesFrom></owl:Restriction>
     </owl:intersectionOf></owl:Class>
    </owl:complementOf></owl:Class>
   </owl:intersectionOf></owl:Class>
  </owl:unionOf></owl:Class>
 </rdfs:subClassOf>
</owl:Class>
```

Fig. 4. Induced Concept Description in the Example 4.1

(generalization) *The routine is controlled by the outer cycle that produces one disjunct at each iteration, accommodating at least one positive example which is successively removed. Thus eventually the termination of the routine is guaranteed, provided that the inner loop terminates. The inner loop generalizes monotonically the starting seed-definition (a conjunct) by applying* δ. *Eventually, the loop terminates either because the generalization is correct or because it covers some negative example. This is the case when the counterfactuals co-routine is invoked.*
Supposing that the latter is correct, the former can terminate by conjoining the negation of the counterfactuals produced to the incorrect definition, thus covering at each iteration new positives and no negative assertion.

(counterfactuals) *This routine contains two initial loops that are controlled by the sizes of the example sets in the input. Each iteration produces a residual concept definition to be used for the successive generalization. Provided that the other routine terminates with a correct generalization of the residual concept then also the counterfactual routine terminates. As regards the correctness, it is to be proven that, on return from the routine, the following relations hold:*
1. $\forall Pos \in CoveredPos$: $(ParGen \sqcap \neg K) \sqsupseteq Pos$
2. $\forall Neg \in Negatives$: $(ParGen \sqcap \neg K) \not\sqsupseteq Neg$. *If the call to* generalization *succeeds then for all* $NewN_j \in NewNegatives$: $K \not\sqsupseteq NewN_j$. *By definition of NewNegatives,* $P_j \in Positives$ *implies that the* residual$(P_j, ParGen) \in NewNegatives$. *Hence* $K \not\sqsupseteq$ residual$(P_j, ParGen)$. *Since* $\forall Pos \in Positives = CoveredPos$: $ParGen \sqsupseteq Pos$ *the first condition is proven. For the other condition, recall that* $(ParGen \sqcap \neg K) \not\sqsupseteq Neg$ *iff* $ParGen \not\sqsupseteq Neg$ *or* $ParGen \sqsupseteq Neg$ *and* $K \sqsupseteq$ residual$(Neg, ParGen)$. *Now, considering all* $Neg \in Negatives$, *if* $ParGen \not\sqsupseteq Neg$ *we have immediately the truth of the condition. Otherwise, if* $ParGen \sqsupseteq Neg$ *then the routine builds a residual element* residual$(Neg, ParGen)$ *for NewPositives. Thus, on return from the call to* generalization, *K is a generalization of every description in NewPositives, and hence* $K \sqsupseteq$ residual$(Neg, ParGen)$.

The algorithm may fail in two cases: a call to counterfactuals with null positive and negative descriptions indicates that it is impossible to discriminate between two identical descriptions. The second case is when generalization fails, which is unlikely to happen because trivially there exists the disjunction of all positive examples descriptions as basic generalization. Such cases should be eliminated beforehand by preprocessing the examples for obtaining the msc's. The counterfactuals algorithm is linear except for the dependency on the generalization algorithm. Then it suffices to discuss the complexity of such an algorithm. The generalization proposed here is a generic divide and conquer algorithm which performs a greedy search using the refinement operator δ. Introducing a better refinement operator based on examples, the heuristic information conveyed by the examples can be better exploited so to have a faster convergency. The cycles are linear on the number of instances. The subsumption tests, that are PSpace-complete in $\mathcal{ALC}$ [11], represent the real source of complexity. As mentioned in the introduction, an easy solution to the learning problem could be computing the minimal generalization: lcs(msc(Pos_1), ..., msc(Pos_m), msc($\neg Neg_1$), ..., msc($\neg Neg_n$)). Yet, suffers from lacking of predictive power, fitting precisely only the known examples (while it may turn out to be inconsistent wrt new ones). Moreover, for many expressive DLs, such as $\mathcal{ALE}$, msc's are difficult to compute (provided they exist). In $\mathcal{ALC}$, although lcs is simply the disjunction of the inputs, there is no algorithm for computing msc's. Our algorithm processes approximations of the msc's, for it is endowed with the specialization mechanism of the counterfactuals, whereas the lcs can only generalize starting from very specialized definitions. The algorithm could be the skeleton for similar ones, based on different DL representations. In such cases, it suffices to change the operator

for calculating residuals and proper methods to conjoin the counterfactual to the starting description should be devised.

5 Conclusions and Future Work

Structural knowledge is expected to play an important role in the Semantic Web. Yet, while techniques for deductive reasoning for such knowledge bases are now very well assessed, their construction is a hard task for knowledge engineers even for limited domains; they could be supported by (semi)automatic inductive tools that still require investigation. In this paper, we have investigated on the induction concept definitions in structural representations both theoretically casting the task as a search problem and operationally, presenting a method for inducing and/or refining concept definitions. The method illustrated in this paper (actually initially presented in [15]) has been implemented in a refinement system (YINGYANG, *Yet another INduction Yields A New Generalization*) that induces or refines $\mathcal{ALC}$ knowledge bases which can be maintained by means of OWL markup. This will allow to design a learning and refinement service for the Semantic Web. The proposed framework could be extended along three directions. First, an investigation on the properties of the refinement operators on DL languages is required. In order to increase the efficiency of learning, redundancy during the search for solutions is to be avoided. This can be done by defining minimal refinement operators [9]. Secondly, the method can be extended to other DLs by changing the residual operator and devising a proper representation for counterfactuals. Another promising direction is the study of *constructive induction* methods to overcome the mentioned problem of inadequate language bias to represent the concepts instantiated in the A-box, under the interactive supervision of the knowledge engineer as an oracle.

References

[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)

[2] Haussler, D.: Learning conjuntive concepts in structural domains. Machine Learning **4** (1989) 7–40

[3] Nienhuys-Cheng, S., Laer, W.V., Ramon, J., Raedt, L.D.: Generalizing refinement operators to learn prenex conjunctive normal forms. In: Proceedings of the International Conference on Inductive Logic Programming. Volume 1631 of LNAI., Springer (1999) 245–256

[4] Rouveirol, C., Ventos, V.: Towards learning in CARIN-$\mathcal{ALN}$. In Cussens, J., Frisch, A., eds.: Proceedings of the 10th International Conference on Inductive Logic Programming. Volume 1866 of LNAI., Springer (2000) 191–208

[5] Kietz, J.U.: Learnability of description logic programs. In Matwin, S., Sammut, C., eds.: Proceedings of the 12th International Conference on Inductive Logic Programming. Volume 2583 of LNAI., Sydney, Springer (2002) 117–132

[6] Cohen, W., Hirsh, H.: Learning the CLASSIC description logic. In Torasso, P., Doyle, J., Sandewall, E., eds.: Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning, Morgan Kaufmann (1994) 121–133
[7] Baader, F., Turhan, A.Y.: TBoxes do not yield a compact representation of least common subsumers. In: Working Notes of the International Description Logics Workshop. Volume 49 of CEUR Workshop Proceedings., Stanford, USA (2001)
[8] Kietz, J.U., Morik, K.: A polynomial approach to the constructive induction of structural knowledge. Machine Learning **14** (1994) 193–218
[9] Badea, L., Nienhuys-Cheng, S.H.: A refinement operator for description logics. In Cussens, J., Frisch, A., eds.: Proceedings of the 10th International Conference on Inductive Logic Programming. Volume 1866 of LNAI., Springer (2000) 40–59
[10] Vere, S.: Multilevel counterfactuals for generalizations of relational concepts and productions. Artificial Intelligence **14** (1980) 139–164
[11] Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence **48** (1991) 1–26
[12] Brandt, S., Küsters, R., Turhan, A.Y.: Approximation and difference in description logics. In Fensel, D., Giunchiglia, F., McGuinness, D., Williams, M.A., eds.: Proceedings of the International Conference on Knowledge Representation, Morgan Kaufmann (2002) 203–214
[13] Nebel, B.: Reasoning and Revision in Hybrid Representation Systems. Volume 422 of LNAI. Springer (1990)
[14] Teege, G.: A subtraction operation for description logics. In Torasso, P., Doyle, J., Sandewall, E., eds.: Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann (1994) 540–550
[15] Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Induction and revision of terminologies. In de Mataras, R.L., Saitta, L., eds.: Proceedings of the 16th European Conference on Artificial Intelligence, IOS Press (2004) 1007–1008