DL-FOIL
Concept Learning in Description Logics

N. Fanizzi, C. d'Amato, and F. Esposito

LACAM – Dipartimento di Informatica – Università degli studi di Bari
Via Orabona, 4 – 70125 – Bari, Italy
{fanizzi|claudia.damato|esposito}@di.uniba.it

Abstract. In this paper we focus on learning concept descriptions expressed in Description Logics. After stating the learning problem in this context, a FOIL-like algorithm is presented that can be applied to general DL languages, discussing related theoretical aspects of learning with the inherent incompleteness underlying the semantics of this representation. Subsequently we present an experimental evaluation of the implementation of this algorithm performed on some real ontologies in order to empirically assess its performance.

1 Introduction

Description Logics (DLs) is the family of representation languages underlying the standard ontology languages designed for knowledge bases in the Semantic Web [3]. These logics constitute a specific fragment of First Order Logic (FOL) that differs from the standard clausal forms employed in Inductive Logic Programming and related multi-relational settings, namely they have a different syntax and especially very different semantics [4, 12]. These considerations justify the growing interest in investigating concept learning in such new formalisms.

Early work on learning in DLs essentially focused on demonstrating the PAC-learnability for various languages derived from Classic. In particular, Cohen and Hirsh investigate the CoreClassic DL proving that it is not PAC-learnable [6] as well as demonstrating the PAC-learnability of a peculiar class among its sub-languages such as C-CLASSIC [7], through the LcsLearn algorithm, together with an empirical evaluation of its implementation.

These approaches tend to cast supervised concept learning as performed through a structural generalizing operator working on equivalent graph representations of the concept descriptions. It is also worth mentioning unsupervised learning methodologies for DL concept descriptions, whose prototypical example is Kluster [16], a polynomial-time algorithm for the induction of BACK terminologies, which exploits the tractability of the standard inferences in this DL language [1].

More recently also learning in hybrid languages, mixing clausal and description logics, have been investigated. Kietz [15] studied the learnability of DL

programs. Other related approaches propose learning methods for hybrid languages, such as CARIN-$\mathcal{ALN}$ [21] and $\mathcal{AL}$-log [19], that allow simple DLs to be combined with DATALOG.

In this work, we focus on learning concepts in expressive DLs endowed with most of the constructors, seeking for a tradeoff between expressiveness, efficiency and completeness of the resulting learning system. Indeed, more expressiveness requires more computational resources for the most common inferences; hence, algorithms dealing with larger DL languages must face the complexity of reasoning. In our vision inductive inference should be employed in order to help the knowledge engineer construct new concept definitions that can eventually be further refined, either manually or by means of other semi-automatic tools. This would save the engineer from finding trivial regularities in the examples so that he/she could concentrate the efforts in refining the induced definition.

We implemented a specific new version of the FOIL algorithm [20], resulting in the DL-FOIL system, that is adapted to learning the DL representations supporting the OWL-DL language. The main components of this new systems are represented by a set of refinement operators borrowed from other similar systems [13, 18] proposed in the literature and by a different gain function which must take into account the open-world assumption, namely, many instances may be available which cannot be ascribed to the target concept nor to its negation. This requires a different setting, similar to learning with unknown class attributes [11], requiring a special treatment of the unlabeled individuals.

A preliminary experiment presented in this paper applies the DL-FOIL system to real ontologies which represent a real testbed w.r.t. the datasets employed for testing YINYANG [13] and DL-Learner [18] which limit their scope to $\mathcal{ALC}$ ontologies. This also demonstrates the usage of the method as a means for performing approximations of concepts across ontologies described with different languages [5, 1].

The outcomes in terms of precision and recall were satisfactory, despite of the employment of incomplete refinement operators. However these outcomes are not as meaningful as they might be in a standard (closed-world) setting. Namely, since many test instances might not be proved to be examples or counter-examples, we resort to different performance metrics, measuring the alignment of the classification decided by the concept descriptions induced by DL-FOIL with the classification derived deductively by a DL reasoner. This allows measuring the amount of unlabeled instances that may be ascribed to the newly induced concepts (or to their negations), which may constitute a real added value brought by the inductive method. Actually these abductive conclusions should be evaluated by the expert who helped during the construction of the ontology. However, this is not always possible.

The paper is organized as follows. After the next section introducing the representation, in Sect. 3 the refinement operators and the algorithm exploiting them are discussed and then, Sect. 4, the adaptation of the FOIL algorithm. In Sect. 5 the experiments proving the effectiveness of the approach are reported. Finally, possible developments are reported in Sect. 6.

2 Description Logics: Syntax and Semantics

In this section we shortly recall syntax and semantics of the DL representation. For brevity, we cannot report syntax and semantics of the various constructors, which can be easily be found in the reference manual [1]. In turn, the DL concept descriptions are straightforwardly mapped onto XML serializations of the standard ontology languages [9].

Roughly, the formalisms are concept-centric: they distinguish *concepts* from *relations* that are used to describe restrictions on concepts. In a DL language, primitive *concepts* $N_C = \{C, D, \ldots\}$ are interpreted as subsets of a domain of objects (resources) and primitive *roles* $N_R = \{R, S, \ldots\}$ are interpreted as binary relations on such a domain (properties). Individuals represent the objects through names from $N_I = \{a, b, \ldots\}$.

Complex concept descriptions are built using atomic concepts and primitive roles by means of specific constructors. The meaning of the descriptions 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 (respectively, a subset of the domain and a relation defined on such domain).

The *top* concept $\top$ is interpreted as the whole domain $\Delta^{\mathcal{I}}$, while the *bottom* concept $\bot$ corresponds to $\emptyset$. Complex descriptions can be built in $\mathcal{ALC}$ using the following constructors[1]. The language supports *full negation*: given any concept description C, denoted $\neg C$, it amounts to $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. The *conjunction* of concepts, denoted with $C_1 \sqcap C_2$, yields an extension $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ and, dually, concept *disjunction*, denoted with $C_1 \sqcup C_2$, yields $C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$. Finally, there are two restrictions on roles: the *existential restriction*, denoted with $\exists R.C$, and interpreted as the set $\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ and the *value restriction*, denoted with $\forall R.C$, whose extension is $\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$.

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 terminological axioms $C \sqsubseteq D$, yet we will consider only definitions $A \equiv D$, where $A \in N_C$ is a concept name (atomic) and D is a concept description given in terms of the language constructors, meaning $A^{\mathcal{I}} = D^{\mathcal{I}}$. The ABox $\mathcal{A}$ contains extensional assertions (ground facts) on concepts and roles, e.g. $C(a)$ and $R(a, b)$, meaning, respectively, that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. Note that the *unique names assumption* is not necessarily made[2].

Further constructors extend the expressiveness of the $\mathcal{ALC}$ language. We are interested in the languages that constitute the counterpart of OWL-DL, namely $\mathcal{SHOIQ}(D)$ that, roughly, extends $\mathcal{ALC}$ with transitive roles, role hierarchies,

[1] In fact, the $\mathcal{ALC}$ corresponds to the fragment of first-order logic obtained by restricting the syntax to formulae containing two variables. $\mathcal{ALC}$ has a modal logic counterpart, namely the multi-modal version of the logic K [1].

[2] Different individual names may be mapped onto the same domain object, in principle.

individual classes, inverse roles and qualified number restrictions. Besides, concrete domains[3] (**D**) can be dealt with.

The set-theoretic 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. A concept definition in the proposed language may be:
$\texttt{Father} \equiv \sqcap\texttt{Male} \sqcap \exists\texttt{hasChild}.\top$
which translates the sentence: *"a father is a male that has someone as his child"* ($\top$ denotes the most general concept).

Now, if we define two new concepts:
$\texttt{FatherWithoutSons} \equiv \texttt{Male} \sqcap \exists\texttt{hasChild}.\top \sqcap \forall\texttt{hasChild}.(\neg\texttt{Male})$
and
$\texttt{Parent} \equiv (\texttt{Male} \sqcup \texttt{Female}) \sqcap \exists\texttt{hasChild}.\top$
then it is easy to see that $\texttt{Father} \sqsupseteq \texttt{FatherWithoutSons}$ and $\texttt{Parent} \sqsupseteq \texttt{Father}$, yet $\texttt{Father} \not\sqsupseteq \texttt{Parent}$ and $\texttt{FatherWithoutSons} \not\sqsupseteq \texttt{Father}$.

A-box assertions are ground facts like:
$\texttt{Father}(\texttt{edward})$, $\texttt{Male}(\texttt{charles})$, $\texttt{hasChild.Male}(\texttt{edward},\texttt{charles})$,
$\geq 1.\texttt{hasChild}(\texttt{edward})$, $\exists\texttt{hasChild}.\top(\texttt{charles})$ and so on.

The most important inference service from the inductive point of view is *instance checking* [1], that amounts to ascertain class-membership assertions: $\mathcal{K} \models C(a)$, where $\mathcal{K}$ is the knowledge base a is an individual name and C is a concept definition given in terms of the concepts accounted for in $\mathcal{K}$.

An important difference with other FOL fragments is the *open-world assumption* (OWA) which makes it more difficult to answer class-membership queries. Thus it may happen that an object that cannot be proved to belong to a certain concept is not necessarily a counterexample for that concept. That would only be interpreted[4] as a case of insufficient (incomplete) knowledge for that assertion.

Example 2.2 (cont'd). Given the concepts
$\texttt{MotherWithoutDaughters} \equiv \texttt{Mother} \sqcap \forall\texttt{hasChild}.\neg\texttt{Female}$
and
$\texttt{Super-motherMother} \geq 3.\texttt{hasChild}$
and the ABox:
$\mathcal{A} = \{\ \texttt{Female}(\texttt{elisabeth}),\ \texttt{Female}(\texttt{diana}),$
$\texttt{Male}(\texttt{charles}),\ \texttt{Male}(\texttt{edward}),\ \texttt{Male}(\texttt{andrew}),$
$\texttt{MotherWithoutDaughters}(\texttt{diana}),$
$\texttt{hasChild}(\texttt{elisabeth},\texttt{charles}),\ \texttt{hasChild}(\texttt{elisabeth},\texttt{edward}),$
$\texttt{hasChild}(\texttt{elisabeth},\texttt{andrew}),\ \texttt{hasChild}(\texttt{diana},\texttt{william}),$
$\texttt{hasChild}(\texttt{charles},\texttt{william})\ \}$

[3] Concrete domains include data types such as numerical types, but also more elaborate domains, such as tuples of the relational calculus, spatial regions, or time intervals.

[4] A model could be constructed for both the membership and non-membership case [1].

One may infer
$\mathcal{K} \models$ `Super-mother(elisabeth)`
but not
$\mathcal{K} \models$ `MotherWithoutDaughters(elisabeth)`
because it may well be that a daughter is not known yet.

This is perfectly compatible with the typical scenario related to the Semantic Web, where new resources may continuously be made available across the Web, hence a complete knowledge cannot be assumed at any time.

The other inference service provided by DL reasoner is concept *retrieval*: given a certain concept, retrieve all the individuals that can be proved to belong to it.

3 Learning as Search in DLs

After recalling the basics of DLs, we are ready to formally define the learning problem in this setting.

Definition 3.1 (learning problem). *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base.*
Given

- *a (new) target concept name C*
- *a set of positive and negative examples $\mathsf{Ind}^+_C(\mathcal{A}) \cup \mathsf{Ind}^-_C(\mathcal{A}) \subseteq \mathsf{Ind}(\mathcal{A})$ where $\mathsf{Ind}(\mathcal{A})$ is the set of individuals occurring in $\mathcal{A}$, $\mathsf{Ind}^+_C(\mathcal{A}) = \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models C(a)\}$, $\mathsf{Ind}^-_C(\mathcal{A}) = \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models \neg C(a)\}$*

***Suppose**, in case a definition for C is already available (refinement problem), that it holds:*

$$\exists a \in \mathsf{Ind}^+_C(\mathcal{A}) \quad \mathcal{K} \not\models C(a) \qquad or \qquad \exists b \in \mathsf{Ind}^-_C(\mathcal{A})) \quad \mathcal{K} \not\models \neg C(b)$$

***Buld** a concept definition $C \equiv D$ such that*

$$\mathcal{K} \models C(a) \quad \forall a \in \mathsf{Ind}^+_C(\mathcal{A}) \qquad and \qquad \mathcal{K} \not\models C(b) \quad \forall b \in \mathsf{Ind}^-_C(\mathcal{A})$$

The definition given above can be interpreted as a generic supervised concept learning task. $\mathsf{Ind}^+_C(\mathcal{A})$ and $\mathsf{Ind}^-_C(\mathcal{A})$ represent respectively the sets of positive and negative examples, whereas $C \equiv D$ is the hypothesis to be induced. The intermediate clause was added for generalizing the definition covering also refinement problems, in which a definition for C is already available but it may be defective w.r.t. to some positive or negative examples.

As known from related works [10, 13, 17], the subsumption relationship (see Def. 2.1) induces a partial order on the space of all the possible concept descriptions. Hence the inductive problem stated above can be cast as a search of the right concept definition (hypothesis) in the induced search space.

In such a setting, one can define suitable operators in order to traverse the search space. As usual in inductive search, we will define two operators in order

to obtain, given a starting (incorrect) hypothesis in the search space, one (or some) of its generalizations/specializations.

Of course, given a set of concept definitions belonging to the space of the descriptions allowed by the reference language, which is partially ordered by subsumption, there is an infinite number of generalizations and specializations. Usually one tries to devise operators that can move efficiently throughout the space in pursuit of one of the target hypotheses.

Now we can define both the downward (specializing) operator ρ and the upward (generalizing) operator δ [13]:

Definition 3.2 (downward operator ρ). $\rho = (\rho_\sqcup, \rho_\sqcap)$, *where:*
$[\rho_\sqcup]$ *given a description in normal form* $D = D_1 \sqcup \cdots \sqcup D_n$*:*

- $D' \in \rho_\sqcup(D)$ *if* $D' = \bigsqcup_{1 \leq i,j \leq n} D_i$ *for some* $j \neq i, 1 \leq j \leq n$
- $D' \in \rho_\sqcup(D)$ *if* $D' = D'_i \sqcup \bigsqcup_{1 \leq i,j \leq n}^{j \neq i} D_k$ *for some* $D'_i \in \rho_\sqcap(D_i)$

$[\rho_\sqcap]$ *given a conjunctive description* $C = C_1 \sqcap \cdots \sqcap C_m$ *and a set of concept descriptions* $\mathcal{A}^- = \{E_k \mid 1 \leq k \leq p\}$*:*

- $C' \in \rho_\sqcap(C)$ *if* $C' = C \sqcap C_{j+1}$
  for some $C_{j+1} \not\sqsupseteq C$ *and* $C_{j+1} \not\sqsupseteq E_h$*, for* $h \in \{1, \ldots, p\}$
- $C' \in \rho_\sqcap(C)$ *if* $C' = (C \sqcup \neg C_j) \sqcap C'_j$ *for some* $j \in \{1, \ldots, m\}$*, 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* $D'_j \in \rho_\sqcup(D_j)$

$\rho_\sqcup$ simply drops one top-level disjunct or replaces it with a downward refinement obtained with $\rho_\sqcap$. $\rho_\sqcap$ adds new conjuncts or replaces one with a refinement obtained by specializing (through $\rho_\sqcup$) the concepts in the scope of a universal or existential restriction.

Definition 3.3 (upward operator δ). $\delta = (\delta_\sqcup, \delta_\sqcap)$, *where:*
$[\delta_\sqcup]$ *given a description in normal form* $D = D_1 \sqcup \cdots \sqcup D_n$ *and a set of concept descriptions* $\mathcal{A}^+ = \{E_h \mid 1 \leq h \leq p\}$*:*

- $D' \in \delta_\sqcup(D)$ *if* $D' = D \sqcup D_{n+1}$ *for some* D_{n+1} *such that* $D_{n+1} \not\sqsubseteq D_i$*,* $i \in \{1, \ldots, n\}$ *and* $D_{j+1} \sqsupseteq E_h$ *for some* $h \in \{1, \ldots, p\}$
- $D' \in \delta_\sqcup(D)$ *if* $D' = D'_i \sqcup \bigsqcup_{1 \leq i,k \leq n}^{k \neq i} D_i$ *for some* $D'_i \in \delta_\sqcap(D_i)$

$[\delta_\sqcap]$ *given a conjunctive description* $C = C_1 \sqcap \cdots \sqcap C_m$*:*

- $C' \in \delta_\sqcap(C)$ *if* $C' = \prod_{1 \leq i,j \leq m}^{j \neq i} C_i$
- $C' \in \delta_\sqcap(C)$ *if* $C' = \prod_{1 \leq i,j \leq m}^{j \neq i} C_i \quad \sqcap C'_j$ *, 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* $D'_j \in \delta_\sqcup(D_j)$

$\delta_{\sqcup}$ and $\delta_{\sqcap}$ simply perform dual operation w.r.t. $\rho_{\sqcup}$ and $\rho_{\sqcap}$, respectively. See [13] for examples.

Other operators [13] may exploit also the knowledge conveyed by the positive and negative examples in order to prune the possible results yielded by a single generalization/specialization step and to better direct the search for suitable solutions to the problem. Instead of using the examples in a mere *generate-and-test* strategy based on these operators, they can be exploited more directly, in order to influence the choices made during the refinement process.

These operators cannot be complete for most expressive DLs [17]. However, we are not looking for too precise operators that likely lead to overfit the data. E.g. the LCS function [7] is a generalizing operator that may be used to compute upper refinement of a concept w.r.t. to an uncovered positive instance (represented by its most specific concept, see [1, 13]), yet this results in a simple union of the two descriptions which deprives the result of any added generalization w.r.t. future examples.

The next step is embedding these simple operators in a suitable learning algorithm.

4 The Learning Algorithm

Various search strategies have been experimented as well as other evaluation measures. Those that we will present in the following are those which gave the best results.

The main aim of this work was conceiving a learning algorithm that could overcome two limitation of the current DL learning systems, namely avoiding the computation of the most specific concepts and the language dependence. Indeed, following the early work, YinYang [13] requires lifting the instances to the concept level through a suitable approximate operator and then start learning from such extremely specific concept descriptions. This setting has the disadvantages of approximation and language-dependence.

DL-Learner [18] partly mitigates these disadvantages for it does not need to compute such approximations since it is essentially based on a genetic programming procedure based on refinement operators whose fitness is computed on the grounds of the covered instances.

Also, in our new algorithm conceived in order to solve the learning problem, the (downward) refinement operators previously defined play a central role. A sketch of the main routine that makes up the algorithm is reported in Fig. 1.

Like in the original Foil algorithm [20], the generalization routine computes (partial) generalizations as long as they do not cover any negative example. If this occurs, the specialization routine is invoked for solving these sub-problems. This routine applies the idea of specializing using the (incomplete) refinement operator defined in the previous section. The specialization continues until no negative example is covered (or a very limited amount[5] of them). The partial

[5] The actual exit-condition for the inner loop being: $|Negatives| - |CoveredNegatives| < \varepsilon$, for some small constant ε.

function DL-FOIL(*Positives*, *Negatives*, *Unlabeled*): *Generalization*
input *Positives*, *Negatives*, *Unlabeled*: positive, negative and unlabeled individuals
output *Generalization*: concept definition

begin
$Generalization \leftarrow \bot$
$PositivesToCover \leftarrow Positives$
while $PositivesToCover \neq \emptyset$ **do**
  begin
  $PartialDef \leftarrow \top$
  $CoveredNegatives \leftarrow Negatives$
  while $CoveredNegatives \neq \emptyset$ **do**
    begin
    $PartialDef \leftarrow$ SPECIALIZE($PartialDef$, $PositivesToCover$, $CoveredNegatives$, $Unlabeled$)
    $CoveredNegatives \leftarrow \{n \in Negatives \mid \mathcal{K} \models \neg PartialDef(n)\}$
    end
  $CoveredPositives \leftarrow \{p \in PositivesToCover \mid \mathcal{K} \models PartialDef(p)\}$
  $Generalization \leftarrow Generalization \sqcup PartialDef$
  $PositivesToCover \leftarrow PositivesToCover \setminus CoveredPositives$
  end
return *Generalization*
end

Fig. 1. The main generalizing routine in DL-FOIL.

generalizations built on each outer loop are finally grouped together in a disjunction which is an allowed constructor for DLs more expressive than (or equal to) $\mathcal{ALC}$. Also the outer while-loop can be exited before covering all the positive examples for avoiding overfitting generalizations.

The specialization function SPECIALIZE (reported in Fig. 2) is called from within the inner loop of the generalization procedure in order to specialize an overly general partial generalization. The function searches for proper refinements that provide at least a minimal gain (see below) fixed with a threshold (*MINGAIN*).

In FOIL-I [14], the gain function has to take into account incomplete examples. Similarly to a semi-supervised learning setting, the gain function that is evaluated for choosing the best refinement is computed as follows:

$$p_1 \cdot \left[\log \frac{p_1 + u_1 w_1}{p_1 + n_1 + u_1} - \log \frac{p_0 + u_0 w_0}{p_0 + n_0 + u_0} \right]$$

where p_1, n_1 and u_1 represent, resp., the number of positive, negative and unlabeled examples covered by the specialization and p_0, n_0 and u_0 stand for the number of positive, negative and unlabeled examples covered by the former definition, and the weights w_0, w_1 are determined by the prior probability of the positive examples, resp., in the current and former concept definition. In order to avoid null numerators, a further correction of the probabilities is performed by resorting to the m-estimate procedure.

```
function SPECIALIZE(PartialDef, Positives, Negatives, Unlabeled): Refinement
input    PartialDef: concept definition
         Positives, Negatives, Unlabeled: positive, negative and unlabeled individuals
output   Refinement: concept definition
const    MAXNUM: maximum real number
         MINGAIN: minimal acceptable gain
         NUMSPECS: number of specializations to be generated

begin
bestGain ← −MAXNUM
while bestGain < MINGAIN do
    for i ← 1 to NUMSPECS do
        begin
        specialization ← GETRANDOMREFINEMENT(ρ, PartialDef)
        CoveredNegatives ← {n ∈ Negatives | K ⊨ ¬PartialDef(n)}
        CoveredPositives ← {p ∈ Positives | K ⊨ PartialDef(p)}
        thisGain ← GAIN(CoveredPositives, CoveredNegatives, Unlabeled, Positives, Negatives)
        if thisGain > bestGain then
            begin
            bestConcept ← refConcept
            bestGain ← thisGain
            end
        end
return Refinement
end
```

Fig. 2. The specializing routine in DL-FOIL.

Despite of its simplicity the complexity of the algorithm is largely determined by the calls to reasoning services, namely subsumption and instance-checking. If we consider the $\mathcal{ALC}$ logic the complexity of these inferences is P-space. However, the algorithm can be thought as building an (upper) $\mathcal{ALC}$-approximation of target concepts, given a knowledge base that can contain definitions expressed in more complex languages, which in turn require more complex reasoning algorithms (see details on $\mathcal{SHOIN}(\mathbf{D})$ [1]). The number of nodes visited during the search grows with the expressiveness of the language because the algorithm searches a sub-space of the actual search space induced by the adopted language.

5 Preliminary Experiments

5.1 Experimental Setting

In order to perform a preliminary experimentation on real ontologies DL-FOIL, it was applied to a number of concept retrieval problems solved by using inductive classification of the individuals w.r.t. a number of query concepts.

To this purpose, we selected a number of ontologies from different domains represented in OWL, namely: NEWTESTAMENTNAMES (NTN) from the Protégé

Table 1. Facts concerning the ontologies employed in the experiments.

Ontology	DL language	#concepts	#object prop.	#data prop.	#individuals
BioPAX	$\mathcal{ALCHF}(D)$	28	19	30	323
NTN	$\mathcal{SHIF}(D)$	47	27	8	676
Financial	$\mathcal{ALCIF}$	60	17	0	1000

library[6] accounting for characters and places mentioned in the book, the BioPax glycolysis ontology[7] (BioPax) describing the glycolysis pathway from the EcoCyc database, translated into BioPax format. It is intended to show what a pathway from an existing database might look like after being translated into BioPAX format. The Financial ontology[8], built for eBanking applications, deals with accounts, holders, loans and related events. Tab. 1 summarizes important details concerning these ontologies. The sizes of the ontologies are to be measured in terms of thousands of triples.

For each ontology, 30 target queries were randomly generated by composition of 2 through 8 primitive or defined concepts from each knowledge base by means of the concept constructors: intersection, union, universal or existential restrictions. Given the overall set of individuals mentioned in the ABox, this was split in training and test sets according to the ten-fold cross-validation procedure. A standard reasoner[9] was employed to decide their class-membership (and non-membership) w.r.t. the query concepts. The performance was evaluated comparing the definitions of the query concepts induced by the system to those that were randomly generated, determining the class-membership of the test examples.

Note that the constant *NUMSPECS* that determines the maximum number of specializations evaluated per turn was set to 15 (larger numbers yield better results but may lower the efficiency).

5.2 Results

Standard IR measures. Initially the standard IR measures precision, recall, F_1-measure were employed to evaluate the system performance. Specifically we considered a two-way classification where relevance coincides with class-membership and the rest of instances are considered irrelevant (true negatives if nothing could be concluded for their membership).

The outcomes are reported in Fig. 2. For each knowledge base, we report the average values obtained over the 30 queries as well as their standard deviation and minimum-maximum ranges of values.

[6] `http://protege.stanford.edu/plugins/owl/owl-library`

[7] `http://www.biopax.org/Downloads/Level1v1.4/biopax-example-ecocyc-glycolysis.owl`

[8] `http://www.cs.put.poznan.pl/alawrynowicz/financial.owl`

[9] Pellet v. 1.5.1 was employed for instance-checking. The reasoner is publicly available at: `http://pellet.owldl.com`.

Table 2. Experimental results in terms of standard IR measures: averages ± standard deviations and [min,max] intervals.

ontology	precision	recall	F_1-measure
BioPax	66.0 ± 24.1	76.5 ± 21.7	69.6 ± 21.0
	[28.3;99.4]	[36.5;100.0]	[31.9;99.7]
NTN	59.0 ± 36.8	64.9 ± 25.7	59.1 ± 30.0
	[18.8;100.0]	[27.9;100.0]	[22.5;100.0]
Financial	62.1 ± 40.7	64.8 ± 37.0	63.3 ± 39.1
	[19.1;99.1]	[24.3;99.0]	[66.7;21.3;99.0]

It is possible to note that precision and recall are generally good but not very high which is also reflected by the F-measure. This happens because these parameters are taken on the grounds of the positive instances which are likely to be in a limited amount w.r.t. the overall number of individuals, especially when random concepts are considered. This is also the cause for the variance to be quite high for all experiments. For the sake of repeatability, we did not employ artificially populated ontologies. We rather employed ontologies as they can be found in the Web. Of course more largely populated ontologies would help assess more stable results. Actually for Financial we selected a reduced number of instances. Selecting larger numbers of individuals, we could obtain better and more stable results.

The reason for precision being less than recall is probably due to the OWA. Indeed, in many cases it was observed that the inductive classification deemed some individuals as relevant for the query issued while the DL reasoner was not able to assess this relevance and this was computed as a mistake while it may likely turn out to be a correct inference when judged by a human agent.

Because of the problems issued by the OWA, different indices would be needed in this case that may make explicit both the rate of inductively classified individuals and the nature of the mistakes.

Alternative measures. Due to the OWA, cases were observed when, it could not be (deductively) ascertained whether a resource was relevant or not for a given query. Then a three-way classification is preferable. Hence, we introduced the following indices for a further evaluation [8]. Essentially they measure the correspondence between the classification provided by the reasoner for the instances w.r.t. the test concept and the definition induced by our system.

- *match rate*: number of cases of individuals that got exactly the same classification with both definitions;
- *omission error rate*: amount of individuals for which class-membership w.r.t. the given query could not determined using the induced definition, while they actually belong (do not belong) to the query concept;

Table 3. Results with alternative indices: averages ± standard deviations and [min,max] intervals.

ontology	match rate	commission error rate	omission error rate	induction rate
BioPax	**76.9** ± 15.7 [56.0;99.4]	**19.7** ± 15.9 [0.0;44.0]	**7.0** ± 20.0 [0.0;64.0]	**7.5** ± 23.7 [0.0;75.0]
NTN	**78.0** ± 19.2 [43.6;95.4]	**16.1** ± 4.0 [0.0;10.8]	**6.4** ± 8.1 [1.2;21.3]	**14.0** ± 10.1 [2.5;27.1]
Financial	**75.5** ± 20.8 [50.2;98.1]	**16.1** ± 12.8 [0.6;25.6]	**4.5** ± 5.1 [1.0;12.6]	**3.7** ± 7.9 [0.3;18.1]

- *commission error rate*: amount of individuals found not to belong to the query concept according to the induced definition, while they actually belong to it and vice-versa.
- *induction rate*: amount of individuals found to belong or not to belong to the query concept according to the induced definition, while either case is not logically derivable from the knowledge base with the original definition

Tab. 3 reports the outcomes in terms of these new indices. Preliminarily, we found that the search procedure was accurate enough: it made few critical mistakes especially when the considered concepts are known to have many examples (and counterexamples) in the ontology. However, it is important to note that, in each experiment, the commission error was limited but not absent, as in the experiments with other classification methods [8]. The cases of queries for which this measure was high are due to the limited amount of examples available (too narrow concepts). Even few mistakes provoked high error rates. This is also due to the absence of axioms stating explicitly the disjointness of some concepts.

Also the omission error rates are quite low. They are comparable with the amount of inductive conclusions that could be drawn with the induced definitions. Again these figures may vary as a consequence of the presence / absence of knowledge about the disjunction of (sibling) concepts in the subsumption hierarchies. In an ontology population perspective, the cases of induction are interesting because they suggest new assertions which cannot be logically derived by using a deductive reasoner yet they might be used to complete a knowledge base [2], e.g. after being validated by an ontology engineer. Better results were obtained on the same task with different inductive methods (instance-based learning [8]). Yet, with DL-Foil we have the added value of having an intensional definition of the target concepts.

The elapsed time (not reported here) was very limited: about 0.5 hour for a whole ten-fold cross validation experiment including the time consumed by the reasoner to make the judgments.

5.3 Learned Concepts

For each ontology, we report examples of the concept descriptions that were learned during the experiments and compare them to the query concept that generated the examples and counterexamples.

BioPax
induced:
`Or( And( physicalEntity protein) dataSource)`
original:
`Or( And( And( dataSource externalReferenceUtilityClass)`
`ForAll(ORGANISM ForAll(CONTROLLED phys icalInteraction))) protein)`

NTN
induced:
`Or( EvilSupernaturalBeing Not(God))`
original:
`Not(God)`

Financial
induced:
`Or( Not(Finished) NotPaidFinishedLoan Weekly)`
original:
`Or( LoanPayment Not(NoProblemsFinishedLoan))`

These concepts totally overlap in terms of their extensions w.r.t. the known individuals.

Of course for a correct qualitative interpretation of the value of these concepts some familiarity is assumed with the ontologies. As mentioned, they are all freely available at standard repositories.

6 Conclusions and Outlook

In this work, we investigated learning expressive DLs supporting ontology languages such as OWL. We implemented a Foil-like algorithm in the DL-Foil system, that is an adaptation to the issues related to the different representation. The main components of this new system are represented by a set of refinement operators and by a different gain function which takes into account the open-world assumption. Namely many instances may be available which cannot be ascribed to the target concept nor to its negation. This requires a different setting and a special treatment of the unlabeled individuals.

A preliminary experimentation has been presented in this paper, applying the DL-Foil system to learning from individuals in real ontologies which represent a harder testbed w.r.t. the datasets employed for testing YinYang [13] and DL-Learner [18] that limited their scope to $\mathcal{ALC}$ ontologies.

The outcomes in terms of precision and recall were satisfactory. However, since these outcomes are not as meaningful as they might be in a standard (closed-world) setting, we recurred to different performance metrics, measuring

the alignment of the classification decided by the concept descriptions induced by DL-Foil with the classification derived deductively by a DL reasoner. This allowed measuring the amount of unlabeled instances that may be ascribed to the newly induced concepts (or to their negations), which constituted a real added value brought by the inductive method. Actually these abductive conclusions should be evaluated by the expert who helped during the construction of the ontology which was not possible unless toy-ontologies were employed.

The experiments made on various ontologies showed that the method is quite effective, and its performance depends on the number (and distribution) of the available training instances. Besides, the procedure appears also robust to noise since commission errors were limited in the experiments carried out so far.

We plan to extend this work evaluating the benefits the algorithm can receive from the addition of other reasoning strategies such as abstraction and abduction. Another important issue is related to the employment of suitable distance or similarity measures that could influence chosen generalization strategy as well as example ordering in the training set. The measure may be applicable to other instance-based tasks which can be approached through machine learning techniques. Then the measure might be plugged in a hierarchical clustering algorithm where clusters would be formed grouping instances on the grounds of their similarity assessed through the measure.

References

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] F. Baader, B. Ganter, B. Sertkaya, and U. Sattler. Completing description logic knowledge bases using formal concept analysis. In M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 230–235, Hyderabad, India, 2007.

[3] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[4] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[5] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, editors, *Proceedings of the International Conference on Knowledge Representation*, pages 203–214. Morgan Kaufmann, 2002.

[6] W.W. Cohen and H. Hirsh. Learnability of description logics. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, Pittsburgh, PA, 1992. ACM Press.

[7] W.W. Cohen and H. Hirsh. Learning the CLASSIC description logic. In P. Torasso, J. Doyle, and E. Sandewall, editors, *Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning*, pages 121–133. Morgan Kaufmann, 1994.

[8] C. d'Amato, N. Fanizzi, and F. Esposito. Query answering and ontology population: An inductive approach. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *Proceedings of the 5th European Semantic Web Conference, ESWC2008*, volume 5021 of *LNCS*, pages 288–302. Springer, 2008.

[9] M. Dean and G. Schreiber. Web Ontology Language Reference. W3C recommendation, W3C, 2004. `http://www.w3.org/TR/owl-ref`.

[10] F. Esposito, N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro. Knowledge-intensive induction of terminologies from metadata. In F. van Harmelen, S. McIlraith, and D. Plexousakis, editors, *ISWC2004, Proceedings of the 3rd International Semantic Web Conference*, volume 3298 of *LNCS*, pages 441–455. Springer, 2004.

[11] S. A. Goldman, S. Kwek, and S. D. Scott. Learning from examples with unspecified attribute values. *Information and Computation*, 180(2):82–100, 2003.

[12] B.N. Grosof, I. Horrocks, R. Volz, and S.Decker. Description logic programs: combining logic programs with description logic. In *Proceedings of the 12th international conference on World Wide Web, WWW03*, pages 48–57, New York, NY, 2003. ACM.

[13] L. Iannone, I. Palmisano, and N. Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.

[14] N. Inuzuka, M. Kamo, N. Ishii, H. Seki, and H. Itoh. Tow-down induction of logic programs from incomplete samples. In S. Muggleton, editor, *Inductive Logic Programming Workshop*, volume 1314 of *LNAI*, pages 265–282. Springer, 1997.

[15] J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Proceedings of the 12th International Conference on Inductive Logic Programming*, volume 2583 of *LNAI*, pages 117–132, Sydney, 2002. Springer.

[16] J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2):193–218, 1994.

[17] J. Lehmann and P. Hitzler. Foundations of refinement operators for description logics. In H. Blockeel, J. Ramon, J. Shavlik, and P. Tadepalli, editors, *Proceedings of the 17th International Conference on Inductive Logic Programming, ILP2007*, volume 4894 of *LNCS*, pages 161–174. Springer, 2008.

[18] J. Lehmann and P. Hitzler. A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic. In H. Blockeel, J. Ramon, J. Shavlik, and P. Tadepalli, editors, *Proceedings of the 17th International Conference on Inductive Logic Programming, ILP2007*, volume 4894 of *LNCS*. Springer, 2008.

[19] F.A. Lisi. Principles of inductive reasoning on the Semantic Web: A framework for learning in $\mathcal{AL}$-Log. In F. Fages and S. Soliman, editors, *Proceedings of the 3rd International Workshop on Principles and Practice of Semantic Web Reasoning, PPSWR2005*, volume 3703 of *LNCS*, pages 118–132, Dagstuhl Castle, Germany, 2005. Springer.

[20] R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[21] C. Rouveirol and V. Ventos. Towards learning in CARIN-$\mathcal{ALN}$. In J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *LNAI*, pages 191–208. Springer, 2000.