Reasoning with OWL-DL in Inductive Logic Programming

Francesca A. Lisi

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

Abstract. The use of background knowledge and the adoption of Horn clausal logic as a knowledge representation and reasoning framework are the distinguishing features of Inductive Logic Programming (ILP) with respect to other approaches to concept learning. We argue that ILP can not ignore the latest developments in Knowledge Engineering such as ontologies and formalisms based on Description Logics. In this paper we present an experience with OWL-DL reasoners in ILP within the application context of the Semantic Web.

1 Introduction

Inductive Logic Programming (ILP) has been historically concerned with concept learning from examples and *background knowledge* within the representation framework of Horn clausal logic and with the aim of prediction [17]. Though the use of background knowledge has been widely recognized as one of the strongest points of ILP when compared to other forms of inductive learning [19,21,10] and has been empirically studied in several application domains [11,26,25], the background knowledge in ILP systems is often not organized around a well-formed conceptual model. This practice seems to ignore latest developments in Knowledge Engineering such as ontologies [6] and formalisms based on Description Logics (DLs) [2] which are playing a relevant role in the definition of the Semantic Web [3]. Indeed the standard mark-up language OWL for the ontological layer of the Semantic Web has been based on the very expressive DL $\mathcal{SHIQ}$ [7]. In a recent position paper, Page and Srinivasan have pointed out that the use of special-purpose reasoners in ILP is among the pressing issues that have arisen from the most challenging ILP applications of today [18]. We think that this is the case for ILP applications in the Semantic Web area.

In this paper we report on an experience with OWL-DL reasoners in ILP. In particular, we choose $\mathcal{AL}$-QuIn [14,13] as the ILP system and Pellet as the OWL-DL reasoner [24]. The paper is structured as follows. Section 2 briefly describes $\mathcal{AL}$-QuIn. Section 3 illustrates the use of Pellet in $\mathcal{AL}$-QuIn. Section 4 draws conclusions and outlines directions of future work.

2 The ILP system $\mathcal{AL}$-QuIn

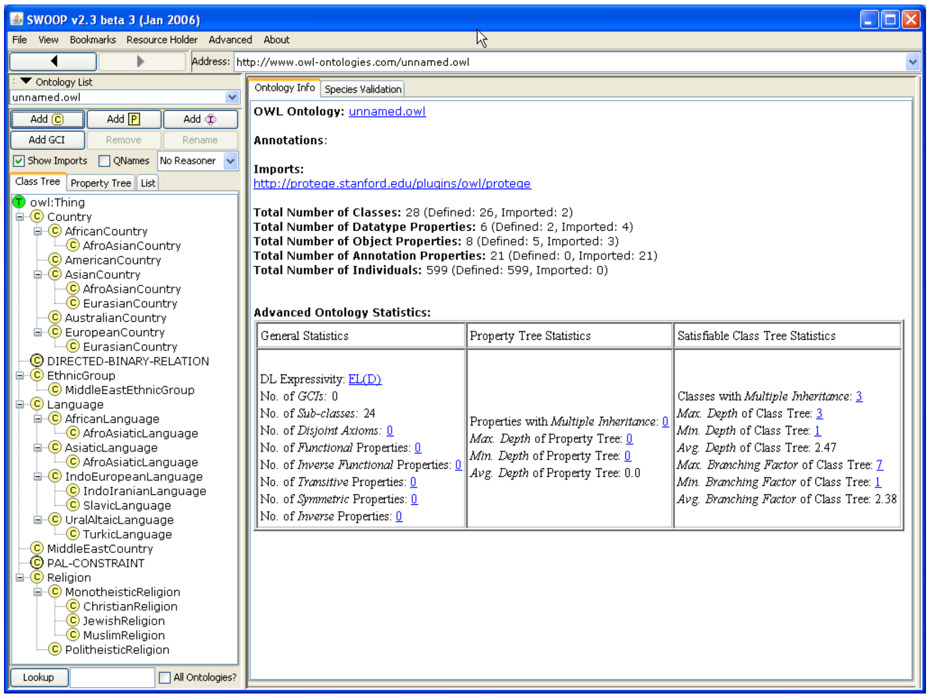
The ILP system $\mathcal{AL}$-QuIn ($\mathcal{AL}$-log Query Induction) [14,13] supports a data mining task known under the name of *frequent pattern discovery.* In data mining a *pattern* is considered as an intensional description (expressed in a given language $\mathcal{L}$) of a subset of a given data set $\mathbf{r}$. The *support* of a pattern is the relative frequency of the pattern within $\mathbf{r}$ and is computed with the evaluation function *supp*. The task of frequent pattern discovery aims at the extraction of all *frequent* patterns, i.e. all patterns whose support exceeds a user-defined threshold of *minimum support.* The blueprint of most algorithms for frequent pattern discovery is the *levelwise search* [16]. It is based on the following assumption: If a generality order $\succeq$ for the language $\mathcal{L}$ of patterns can be found such that $\succeq$ is monotonic w.r.t. *supp*, then the resulting space $(\mathcal{L}, \succeq)$ can be searched breadth-first starting from the most general pattern in $\mathcal{L}$ and by alternating *candidate generation* and *candidate evaluation* phases. In particular, candidate generation consists of a refinement step followed by a pruning step. The former derives candidates for the current search level from patterns found frequent in the previous search level. The latter allows some infrequent patterns to be detected and discarded prior to evaluation thanks to the monotonicity of $\succeq$. $\mathcal{AL}$-QuIn solves a variant of the frequent pattern discovery problem which takes concept hierarchies into account during the discovery process, thus yielding descriptions at multiple granularity levels up to a maximum level $maxG$. More formally, given

- a data set $\mathbf{r}$ including a taxonomy $\mathcal{T}$ where a reference concept C_{ref} and task-relevant concepts are designated,
- a multi-grained language $\{\mathcal{L}^l\}_{1 \leq l \leq maxG}$ of patterns
- a set $\{minsup^l\}_{1 \leq l \leq maxG}$ of user-defined minimum support thresholds

the problem of *frequent pattern discovery at l levels of description granularity*, $1 \leq l \leq maxG$, is to find the set $\mathcal{F}$ of all the patterns $P \in \mathcal{L}^l$ that describe the reference concept w.r.t. the task-relevant concepts and turn out to be frequent in $\mathbf{r}$. Note that P's with support s such that (i) $s \geq minsup^l$ and (ii) all ancestors of P w.r.t. $\mathcal{T}$ are frequent in $\mathbf{r}$. Note that a pattern Q is considered to be an ancestor of P if it is a coarser-grained version of P.

Example 1. As a showcase we consider the task of finding frequent patterns that describe Middle East countries (reference concept) w.r.t. the religions believed and the languages spoken (task-relevant concepts) at three levels of granularity ($maxG = 3$). Minimum support thresholds are set to the following values: $minsup^1 = 20\%$, $minsup^2 = 13\%$, and $minsup^3 = 10\%$. The data set and the language of patterns will be illustrated in Example 2 and Example 3, respectively.

In $\mathcal{AL}$-QuIn data and patterns are represented according to the hybrid knowledge representation and reasoning system $\mathcal{AL}$-log [5]. In particular, the data set $\mathbf{r}$ is represented as an $\mathcal{AL}$-log knowledge base $\mathcal{B}$, thus composed of a structural part and a relational part. The structural subsystem Σ is based



```
owl:Thing
  Country
    AfricanCountry
      AfroAsianCountry
    AmericanCountry
    AsianCountry
      AfroAsianCountry
      EurasianCountry
    AustralianCountry
    EuropeanCountry
      EurasianCountry
  DIRECTED-BINARY-RELATION
  EthnicGroup
    MiddleEastEthnicGroup
  Language
    AfricanLanguage
      AfroAsiaticLanguage
    AsiaticLanguage
      AfroAsiaticLanguage
    IndoEuropeanLanguage
      IndoIranianLanguage
      SlavicLanguage
    UralAltaicLanguage
      TurkicLanguage
  MiddleEastCountry
  PAL-CONSTRAINT
  Religion
    MonotheisticReligion
      ChristianReligion
      JewishReligion
      MuslimReligion
    PolitheisticReligion
```

OWL Ontology: unnamed.owl

Annotations:

Imports:
http://protege.stanford.edu/plugins/owl/protege

Total Number of Classes: 28 (Defined: 26, Imported: 2)
Total Number of Datatype Properties: 6 (Defined: 2, Imported: 4)
Total Number of Object Properties: 8 (Defined: 5, Imported: 3)
Total Number of Annotation Properties: 21 (Defined: 0, Imported: 21)
Total Number of Individuals: 599 (Defined: 599, Imported: 0)

Advanced Ontology Statistics:

General Statistics	Property Tree Statistics	Satisfiable Class Tree Statistics
DL Expressivity: EL(D) No. of *GCIs*: 0 No. of *Sub-classes*: 24 No. of *Disjoint Axioms*: 0 No. of *Functional* Properties: 0 No. of *Inverse Functional* Properties: 0 No. of *Transitive* Properties: 0 No. of *Symmetric* Properties: 0 No. of *Inverse* Properties: 0	Properties with *Multiple Inheritance*: 0 *Max. Depth* of Property Tree: 0 *Min. Depth* of Property Tree: 0 *Avg. Depth* of Property Tree: 0.0	Classes with *Multiple Inheritance*: 3 *Max. Depth* of Class Tree: 3 *Min. Depth* of Class Tree: 1 *Avg. Depth* of Class Tree: 2.47 *Max. Branching Factor* of Class Tree: 7 *Min. Branching Factor* of Class Tree: 1 *Avg. Branching Factor* of Class Tree: 2.38

Fig. 1. The ontology $\Sigma_{\texttt{CIA}}$ for $\mathcal{AL}$-QuIn.

on $\mathcal{ALC}$ [22] and allows for the specification of knowledge in terms of classes (*concepts*), binary relations between classes (*roles*), and instances (*individuals*). In particular, the TBox $\mathcal{T}$ contains is-a relations between concepts (*axioms*) whereas the ABox $\mathcal{A}$ contains instance-of relations between individuals (resp. pairs of individuals) and concepts (resp. roles) (*assertions*). The relational subsystem Π is based on an extended form of Datalog [4] that is obtained by using $\mathcal{ALC}$ concept assertions essentially as type constraints on variables.

Example 2. For the task of interest, we consider an $\mathcal{AL}$-log knowledge base $\mathcal{B}_{\texttt{CIA}}$ that integrates a $\mathcal{ALC}$ component $\Sigma_{\texttt{CIA}}$ containing taxonomies rooted into the concepts `Country`, `EthnicGroup`, `Language` and `Religion` and a Datalog component $\Pi_{\texttt{CIA}}$ containing facts[1] extracted from the on-line 1996 CIA World Fact Book[2]. Note that Middle East countries have been defined as Asian countries that host at least one Middle Eastern ethnic group:

$$\texttt{MiddleEastCountry} \equiv \texttt{AsianCountry} \sqcap \exists\texttt{Hosts}.\texttt{MiddleEastEthnicGroup}.$$

[1] http://www.dbis.informatik.uni-goettingen.de/Mondial/mondial-rel-facts.flp
[2] http://www.odci.gov/cia/publications/factbook/

In particular, Armenia (`'ARM'`) and Iran (`'IR'`) are classified as Middle East countries because the following membership assertions hold in $\Sigma_{\texttt{CIA}}$:

```
'ARM':AsianCountry.
'IR':AsianCountry.
'Arab':MiddleEastEthnicGroup.
'Armenian':MiddleEastEthnicGroup.
<'ARM','Armenian'>:Hosts.
<'IR','Arab'>:Hosts.
```

More details on $\Sigma_{\texttt{CIA}}$ can be found in Figure 1.[3] Also $\Pi_{\texttt{CIA}}$ includes constrained DATALOG clauses such as:

```
believes(Code, Name)←
        religion(Code, Name, Percent) & Code:Country, Name:Religion.
speaks(Code, Name)←
        language(Code, Name, Percent) & Code:Country, Name:Language.
```

that define views on the relations `religion` and `language`, respectively.

The language $\mathcal{L} = \{\mathcal{L}^l\}_{1 \leq l \leq maxG}$ of patterns allows for the generation of $\mathcal{AL}$-log unary conjunctive queries, called $\mathcal{O}$-queries. Given a reference concept C_{ref}, an *$\mathcal{O}$-query Q* to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ is a (linked and connected)[4] constrained DATALOG clause of the form

$$Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \& X : C_{ref}, \gamma_1, \ldots, \gamma_n$$

where X is the *distinguished variable* and the remaining variables occurring in the body of Q are the *existential variables*. Note that α_j, $1 \leq j \leq m$, is a DATALOG literal whereas γ_k, $1 \leq k \leq n$, is an assertion that constrains a variable already appearing in any of the α_j's to vary in the range of individuals of a concept defined in $\mathcal{B}$. The $\mathcal{O}$-query

$$Q_t = q(X) \leftarrow \& X : C_{ref}$$

is called *trivial* for $\mathcal{L}$ because it only contains the constraint for the *distinguished variable* X. Furthermore the language $\mathcal{L}$ is *multi-grained*, i.e. it contains expressions at multiple levels of description granularity. Indeed it is implicitly defined by a *declarative bias specification* which consists of a finite alphabet Δ of DATALOG predicate names and finite alphabets Γ^l (one for each level l of description granularity) of $\mathcal{ALC}$ concept names. Note that the α_i's are taken from $\mathcal{A}$ and γ_j's are taken from Γ^l. We impose $\mathcal{L}$ to be finite by specifying some bounds, mainly $maxD$ for the maximum depth of search and $maxG$ for the maximum level of granularity.

Example 3. To accomplish the task of Example 1 we define $\mathcal{L}_{\texttt{CIA}}$ as the set of $\mathcal{O}$-queries with C_{ref} = `MiddleEastCountry` that can be generated from the alphabet Δ= {`believes/2`, `speaks/2`} of DATALOG binary predicate names, and the alphabets

[3] We would like to remind the reader that $\mathcal{ALC}$ is a fragment of $\mathcal{SHIQ}$.
[4] For the definition of linkedness and connectedness see [17].

Γ^1= {`Language`, `Religion`}
Γ^2= {`IndoEuropeanLanguage`, ..., `MonotheisticReligion`, ...}
Γ^3= {`IndoIranianLanguage`, ..., `MuslimReligion`, ...}

of $\mathcal{ALC}$ concept names for $1 \leq l \leq 3$, up to $maxD = 5$. Examples of $\mathcal{O}$-queries in $\mathcal{L}_{\mathtt{CIA}}$ are:

Q_t= `q(X)` $\leftarrow$ & `X:MiddleEastCountry`
Q_1= `q(X)` $\leftarrow$ `speaks(X,Y)` & `X:MiddleEastCountry`, `Y:Language`
Q_2= `q(X)` $\leftarrow$ `speaks(X,Y)` & `X:MiddleEastCountry`, `Y:IndoEuropeanLanguage`
Q_3= `q(X)` $\leftarrow$ `believes(X,Y)`& `X:MiddleEastCountry`, `Y:MuslimReligion`

where Q_t is the trivial $\mathcal{O}$-query for $\mathcal{L}_{\mathtt{CIA}}$, $Q_1 \in \mathcal{L}^1_{\mathtt{CIA}}$, $Q_2 \in \mathcal{L}^2_{\mathtt{CIA}}$, and $Q_3 \in \mathcal{L}^3_{\mathtt{CIA}}$. Note that Q_1 is an ancestor of Q_2.

The *support* of an $\mathcal{O}$-query $Q \in \mathcal{L}^l$ w.r.t an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ is defined as

$$supp(Q, \mathcal{B}) = \mid answerset(Q, \mathcal{B}) \mid / \mid answerset(Q_t, \mathcal{B}) \mid$$

where $answerset(Q, \mathcal{B})$ is the set of correct answers to Q w.r.t. $\mathcal{B}$. An *answer* to Q is a ground substitution θ for the distinguished variable of Q. An answer θ to Q is a *correct (resp. computed) answer* w.r.t. $\mathcal{B}$ if there exists at least one correct (resp. computed) answer to $body(Q)\theta$ w.r.t. $\mathcal{B}$. Thus the computation of support relies on query answering in $\mathcal{AL}$-log.

Example 4. The pattern Q_2 turns out to be frequent because it has support $supp(Q_2, \mathcal{B}_{\mathtt{CIA}}) = (2/15)\% = 13.3\%$ ($\geq minsup^2$). It is to be read as '13.3 % of Middle East countries speak an Indoeuropean language'. The two correct answers to Q_2 w.r.t. $\mathcal{B}_{\mathtt{CIA}}$ are `'ARM'` and `'IR'`.

The system $\mathcal{AL}$-QuIn implements the aforementioned levelwise search method for frequent pattern discovery. In particular, candidate patterns of a certain level k (called *k-patterns*) are obtained by refinement of the frequent patterns discovered at level $k-1$. In $\mathcal{AL}$-QuIn patterns are ordered according to $\mathcal{B}$-subsumption (which has been proved to fulfill the abovementioned condition of monotonicity [15]). The search starts from the most general pattern in $\mathcal{L}$ and iterates through the generation-evaluation cycle for a number of times that is bounded with respect to both the granularity level l ($maxG$) and the depth level k ($maxD$).

Example 5. After $maxD = 5$ search stages, $\mathcal{AL}$-QuIn returns 53 frequent patterns out of 99 candidate patterns compliant with the parameter settings. One of these frequent patterns is Q_2.

3 Using Pellet in $\mathcal{AL}$-QuIn

3.1 The coverage test

In ILP the evaluation of inductive hypotheses (like candidate patterns in frequent pattern discovery) w.r.t. a set of observations (data units) is usually referred to as the *coverage test* because it checks which observations satisfy (are covered by) the hypothesis. Since evaluation is the most computationally expensive step when inducing hypotheses expressed in (fragments of) first-order logic, an appropriate choice of representation for observations can help speeding up this step. In $\mathcal{AL}$-QuIn the extensional part of Π is partitioned into portions $\mathcal{A}_i$ each of which refers to an individual a_i of C_{ref}. The link between $\mathcal{A}_i$ and a_i is represented with the Datalog literal $q(a_i)$. The pair $(q(a_i), \mathcal{A}_i)$ is called *observation*.

Example 6. By assuming `MiddleEastCountry` as reference concept, the observation $\mathcal{A}_{\mathtt{ARM}}$ contains Datalog facts such as

```
language('ARM','Armenian',96).
language('ARM','Russian',2).
```

concerning the individual `'ARM'` whereas the observation $\mathcal{A}_{\mathtt{IR}}$ consists of facts like

```
language('IR','Turkish',1).
language('IR','Kurdish',9).
language('IR','Baloch',1).
language('IR','Arabic',1).
language('IR','Luri',2).
language('IR','Persian',58).
language('IR','Turkic',26).
```

related to the individual `'IR'`.

In ILP the coverage test must take the background knowledge into account. The portion $\mathcal{K}$ of $\mathcal{B}$ which encompasses the whole Σ and the intensional part (IDB) of Π is considered as *background knowledge* for $\mathcal{AL}$-QuIn. Therefore proving that an $\mathcal{O}$-query Q covers an observation $(q(a_i), \mathcal{A}_i)$ w.r.t. $\mathcal{K}$ equals to proving that $\theta_i = \{X/a_i\}$ is a correct answer to Q w.r.t. $\mathcal{B}_i = \mathcal{K} \cup \mathcal{A}_i$.

Example 7. Checking whether Q_2 covers the observation $(\mathtt{q('ARM')}, \mathcal{A}_{\mathtt{ARM}})$ w.r.t. $\mathcal{K}_{\mathtt{CIA}}$ is equivalent to answering the query

$Q_2^{(0)} = \leftarrow \mathtt{q('ARM')}$

w.r.t. $\mathcal{K}_{\mathtt{CIA}} \cup \mathcal{A}_{\mathtt{ARM}} \cup Q_2$. The coverage test for $(\mathtt{q('IR')}, \mathcal{A}_{\mathtt{IR}})$ is analogous.

A common practice in ILP is to use a reformulation operator, called *saturation* [20], to speed-up the coverage test. It enables ILP systems to make background knowledge explicit within the observations instead of implicit and apart from the observations. In the following we will discuss the implementation of the coverage test in $\mathcal{AL}$-QuIn and clarify the role of Pellet in supporting the saturation of observations w.r.t. a OWL-DL background knowledge Σ.

3.2 Implementation issues

$\mathcal{AL}$-QuIn is implemented with Prolog as usual in ILP. Thus, the *actual* representation language in $\mathcal{AL}$-QuIn is a kind of Datalog^{OI} [23], i.e. the subset of Datalog$^{\neq}$ equipped with an equational theory that consists of the axioms of Clark's Equality Theory augmented with one rewriting rule that adds *inequality atoms* $s \neq t$ to any $P \in \mathcal{L}$ for each pair (s, t) of distinct terms occurring in P. Note that concept assertions are rendered as *membership atoms*, e.g. $a : C$ becomes $c\_C(a)$.

Example 8. The following query

```
q(X) ← c_MiddleEastCountry(X), believes(X,Y), c_MonotheisticReligion(Y),
        believes(X,Z), Y≠Z
```

is the Datalog^{OI} rewriting of:

```
q(X) ← believes(X,Y), believes(X,Z) &
        X:MiddleEastCountry, Y:MonotheisticReligion
```

where the absence of a $\mathcal{ALC}$ constraint for the variable `Z` explains the need for the inequality atom.

When implementing the coverage test in $\mathcal{AL}$-QuIn, the goal has been to reduce the reasoning mechanism of $\mathcal{AL}$-log (constrained SLD-resolution) to SLD-resolution on Datalog^{OI}. A crucial issue in this mapping is to deal with the satisfiability tests of $\mathcal{ALC}$ constraints w.r.t. Σ which are required by constrained SLD-resolution because they are performed by applying the tableau calculus for $\mathcal{ALC}$. The reasoning on the constraint part of $\mathcal{O}$-queries has been replaced by preliminary saturation steps of the observations w.r.t. the background knowledge Σ. By doing so, the observations are completed with concept assertions that can be derived from Σ.

Retrieving all the individuals of a concept C is known in DLs as the ***retrieval*** problem [2]. Here, the retrieval is called *levelwise* because it follows the layering of $\mathcal{T}$: individuals of concepts belonging to the l-th layer $\mathcal{T}^l$ of $\mathcal{T}$ are retrieved all together. Conversely the retrieval for the *reference concept* is made only once at the beginning of the whole discovery process because it makes explicit knowledge of interest to all the levels of granularity. This makes SLD-refutations of queries in $\mathcal{L}^l$ work only on extensional structural knowledge at the level l of description granularity.

A Java application, named OWL2Datalog, has been developed to support the saturation of observations w.r.t. a OWL-DL background knowledge Σ in $\mathcal{AL}$-QuIn. To achieve this goal, it supplies the following functionalities:

– levelwise retrieval w.r.t. Σ
– Datalog^{OI} rewriting of (asserted and derived) concept assertions of Σ

The former is implemented by a client for the DIG server Pellet. The latter relies on the former, meaning that the results of the levelwise retrieval are exported to Datalog^{OI}.

Example 9. The DATALOG^{OI} rewriting of the concept assertions derived for $\mathcal{T}^2$ produces facts like:

```
c_AfroAsiaticLanguage('Arabic').
...
c_IndoEuropeanLanguage('Armenian').
...
c_UralAltaicLanguage('Kazak').
...
c_MonotheisticReligion('ShiaMuslim').
...
c_PolytheisticReligion('Druze').
...
```

to be considered during coverage tests of $\mathcal{O}$-queries in $\mathcal{L}^2$.

The concept assertions, once translated to DATALOG^{OI}, are added to the facts derived from the IDB of Π at the loading of each observation. The coverage test therefore concerns DATALOG^{OI} rewritings of both $\mathcal{O}$-queries and saturated observations.

Example 10. The DATALOG^{OI} rewriting

```
q(X) ← c_MiddleEastCountry(X), speaks(X,Y), c_IndoEuropeanLanguage(Y)
```

of Q_2 covers the DATALOG^{OI} rewriting:

```
c_MiddleEastCountry('ARM').
speaks('ARM','Armenian').
...
c_IndoEuropeanLanguage('Armenian').
...
```

of the saturated observation $\hat{\mathcal{A}}_{\mathtt{ARM}}$.

Note that the translation from OWL-DL to DATALOG^{OI} is possible because we assume that *all* the concepts are named. This means that an equivalence axiom is required for each complex concept in the knowledge base. Equivalence axioms help keeping concept names (used within constrained DATALOG clauses) independent from concept definitions.

4 Conclusions and future work

In this paper we have shown how to use the OWL/DL reasoner Pellet to make an existing ILP system, $\mathcal{AL}$-QUIN, compliant with the latest developments in Knowledge Engineering, i.e. ontologies and DL-based ontology languages. We would like to emphasize that $\mathcal{AL}$-QUIN was originally conceived to deal with background knowledge in the form of taxonomic ontologies but the implementation of this feature was still lacking[5]. Therefore, we have also shown how to use

[5] $\mathcal{AL}$-QUIN could actually deal only with concept hierarchies in DATALOG^{OI}.

Pellet to make $\mathcal{AL}$-QuIn fulfill its design requirements. More precisely, the Java application OWL2Datalog relies on the reasoning services of Pellet to support the saturation of observations w.r.t. background knowledge in $\mathcal{AL}$-QuIn. In ILP saturation has been mentioned as a way of speeding-up the evaluation of candidate hypotheses. In our case it encompasses a transformation step that compiles DL-based background knowledge down to the usual Datalog-like formalisms of ILP systems. In this respect, the pre-processing method proposed by Kietz [9] to enable legacy ILP systems to work within the framework of the hybrid KR&R system CARIN [12] is related to ours but it lacks an implementation. Analogously, the method proposed in [8] for translating OWL-DL to disjunctive Datalog is far too general with respect to the specific needs of our application. Rather, the proposal of interfacing existing reasoners to combine ontologies and rules [1] is more similar to ours in the spirit.

For the future we plan to implement the functionalities of OWL2Datalog as a plugin for Protégé-2000. Also we intend to compare $\mathcal{AL}$-QuIn with other ILP systems able to deal with ontological background knowledge as soon as they are implemented and deployed.

References

1. U. Assmann, J. Henriksson, and J.Maluszynski. Combining safe rules and ontologies by interfacing of reasoners. In J.J. Alferes, J. Bailey, W. May, and U. Schwertel, editors, *Principles and Practice of Semantic Web Reasoning*, volume 4187 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2006.
2. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2003.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May, 2001.
4. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases.* Springer, 1990.
5. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.
6. A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering.* Springer, 2004.
7. I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
8. U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}$-description logic to disjunctive datalog programs. In D. Dubois, C.A. Welty, and M.-A. Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 152–162. AAAI Press, 2004.
9. J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Inductive Logic Programming*, volume 2583 of *Lecture Notes in Artificial Intelligence*, pages 117–132. Springer, 2003.
10. N. Lavrač and S. Džeroski. Background knowledge and declarative bias in inductive concept learning. In Klaus P. Jantke, editor, *Analogical and Inductive Inference*, volume 642 of *Lecture Notes in Computer Science*, pages 51–71. Springer, 1992.

11. N. Lavrač, S. Džeroski, V. Pirnat, and V. Krizman. The utility of background knowledge in learning medical diagnostic rules. *Applied Artificial Intelligence*, 7(3):273–293, 1993.
12. A.Y. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.
13. F.A. Lisi and F. Esposito. Efficient Evaluation of Candidate Hypotheses in $\mathcal{AL}$-log. In R. Camacho, R. King, and A. Srinivasan, editors, *Inductive Logic Programming*, volume 3194 of *Lecture Notes in Artificial Intelligence*, pages 216–233. Springer, 2004.
14. F.A. Lisi and D. Malerba. Ideal Refinement of Descriptions in $\mathcal{AL}$-log. In T. Horvath and A. Yamamoto, editors, *Inductive Logic Programming*, volume 2835 of *Lecture Notes in Artificial Intelligence*, pages 215–232. Springer, 2003.
15. F.A. Lisi and D. Malerba. Inducing Multi-Level Association Rules from Multiple Relations. *Machine Learning*, 55:175–210, 2004.
16. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
17. S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.
18. D. Page and A. Srinivasan. ILP: A short look back and a longer look forward. *Journal of Machine Learning Research*, 4:415–430, 2003.
19. M.J. Pazzani and D.F. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9:57–94, 1992.
20. C. Rouveirol. Flattening and saturation: Two representation changes for generalization. *Machine Learning*, 14(1):219–232, 1994.
21. C. Rouveirol and L. De Raedt. The use of background knowledge for generalization in ILP. In C. Rouveirol, editor, *Proceedings of the ECAI-92 Workshop on Logical Approaches to Machine Learning*, 1992.
22. M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
23. G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli. A logic framework for the incremental inductive synthesis of Datalog theories. In N.E. Fuchs, editor, *Proceedings of 7th International Workshop on Logic Program Synthesis and Transformation*, volume 1463 of *Lecture Notes in Computer Science*, pages 300–321. Springer, 1998.
24. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 2006.
25. A. Srinivasan, R.D. King, and M. Bain. An empirical study of the use of relevance information in inductive logic programming. *Journal of Machine Learning Research*, 4:369–383, 2003.
26. M. Turcotte, S. Muggleton, and M.J.E. Sternberg. The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Machine Learning*, 43(1/2):81–95, 2001.