ONTOSEARCH2: Searching Ontologies Semantically

Edward Thomas, Jeff Z. Pan and Derek Sleeman

Department of Computing Science, University of Aberdeen, Aberdeen AB24 3UE, UK

Abstract. While the Web makes an increasing number of ontologies widely available for applications, how to discover ontologies becomes a more challenging issue. Existing approaches are mainly based on keywords and metadata information of ontologies, rather than semantic entailments of ontologies. In this paper, we present an ontology search engine, ONTOSEARCH2, which provides three approaches to searching ontologies semantically. These approaches take advantage of ONTOSEARCH2's efficient query answering capability.

1 Introduction

Searching for relevant ontologies is one of the key tasks to enable ontology reuse. Now the W3C ontology language OWL has become the defacto standard for ontologies and semantic data, there are progressively more ontology libraries online, such as the DAML Ontology Library and Protégé Ontologies Library. While the Web makes an increasing number of ontologies widely available for applications, how to discover ontologies becomes a more challenging issue.

Currently it is difficult to find ontologies suitable for a particular purpose. Semantic Web search engines like Swoogle [1] and OntoKhoj [2] allow ontologies to be searched using keywords, but further refinement of the search criteria based on semantic entailments is not possible. For example, if one wants to search ontologies in which Professor is a sub-class of Client, using the keywords based approach is not satisfactory, as any ontologies that contain the classes Professor and Client would be returned, whether or not the subsumption relationship holds between them. In this paper, we present an ontology search engine, called ONTOSEARCH2, which provides three approaches to searching ontologies semantically. A keyword-based search tool, a search tool based on query answering and a search tool based on fuzzy query answering. A detailed example of the three different kinds of searches is presented in Section 4.

The core of ONTOSEARCH2 is a DL-Lite [3] query engine. Semantic approximation recasts the idea of knowledge compilation to provide *sound approximation* for query answering over OWL DL ontologies. For queries without non-distinguished variables[1], this approach guarantees soundness and completeness. ONTOSEARCH2 also contains a fuzzy DL-Lite [4] query engine, which is

[1] Variables which exist in the body of a query, but not its head

able to answer threshold queries and some general fuzzy queries over fuzzy DL-Lite ontologies. Preliminary evaluations indicate the two querying components in ONTOSEARCH2 are very efficient. They provide a good infrastructure for the search functionality in ONTOSEARCH2.

The rest of the paper is organised as follows. Section 2 briefly introduces OWL, DL-Lite and fuzzy DL-Lite. Section 3 presents the ONTOSEARCH2 system, explaining the idea of semantic approximation and fuzzy query answering, as well as the search functionality. Section 4 presents an example to illustrate these three search functionalities of ONTOSEARCH2. We briefly review related work, before concluding the paper.

2 Background

2.1 Ontologies and OWL

An ontology [5] typically consists of a set of classes, properties, and constraints about these classes and properties. An ontology language provides constructors to construct class and property descriptions based on named classes and properties, as well as some forms of axioms about classes, properties and individuals. For example, RDFS [6] provides some axioms (such as domain and range axioms), but no class or property constructors. OWL DL [7] provides class constructors (e.g. conjunction $C \sqcup D$ and number restriction $\geq nR$), property constructors (e.g. inverse properties R) and more kinds of axioms (such as individual equality axioms $a \equiv b$) than RDFS. Usually, we call the set of class and property axioms TBox, while the set of individual axioms are called ABox. OWL DL is an expressive Description Logic [8], in which the complexity of logical entailment is NEXPTIME.

2.2 DL-Lite

The trade-off between expressive power and efficiency has been one of the main themes in Description Logic research. Recently, Calvanese et al [3] proposed DL-Lite which can express most features in UML class diagrams but still has a low reasoning overhead (worst case polynomial time, compared to worst case exponential time in the case of most Description Logics). DL-Lite supports the following axioms:

1. class inclusion axioms: $B \sqsubseteq C$ where B is a basic class $B := \mathsf{A} \mid \exists R \mid \exists R^-$ and C is a general class $C := B \mid \neg B \mid C1 \sqcap C2$ (where A denotes an named class and R denotes a named property);
2. functional property axioms: $Func(R), Func(R^-)$, where R is a named property;
3. individual axioms: $B(\mathtt{a}), R(\mathtt{a}, \mathtt{b})$ where $\mathtt{a}$ and $\mathtt{b}$ are named individuals.

After some careful query rewriting, query answering over DL-Lite ontologies can be carried out by an SQL engine, so as to take advantage of existing query optimisation strategies and algorithms provided by modern database management systems.

2.3 Fuzzy DL-Lite

Straccia [4] proposed fuzzy DL-Lite (which we call f-DL-Lite), which extends DL-Lite core with fuzzy assertions of the forms $B(\mathsf{a}) \geq n, R(\mathsf{a},\mathsf{b}) \geq n$, where B is basic class, R is a property, $\mathtt{a}$ and $\mathtt{b}$ are individuals and n is a real number in the range $[0,1]$.

The semantics of f-DL-Lite ontologies are defined in terms of *fuzzy interpretations* [9]. A fuzzy interpretation is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where the domain $\Delta^{\mathcal{I}}$ is a non-empty set of objects and $\cdot^{\mathcal{I}}$ is a fuzzy interpretation function, which maps:

- an individual a to an element of $\mathsf{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$,
- a named class A to a membership function $\mathsf{A}^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$, and
- a named property R to a membership function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1]$.

Using the fuzzy set theoretic operations [10], fuzzy interpretations can be extended to interpret f-DL-Lite class and property descriptions. Following Straccia [4], we use the Lukasiewicz negation, c(a)=1-a and the Gödel t-norm for interpreting conjunctions, $t(a,b) = \min(a,b)$. The semantics of f-DL-Lite class and property descriptions, and f-DL-Lite axioms are given in [11]. Given these semantics, it is obvious that crisp assertions $B(a)$, $R(a,b)$ are special forms of fuzzy assertions where $n = 1$.

As indicated by Straccia [4], algorithms for subsumption checking and ontology consistency checking in f-DL-Lite are similar to those in DL-Lite [12]. There are two steps here: 1) *Normalisation of class axioms*: given an ontology $\emptyset$, the set of class axioms $\mathcal{T}$ can be normalised to $\mathcal{T}'$ which consists of class axioms of the forms $B \sqsubseteq B_1$ (Positive Inclusion, or simply PI), $B \sqsubseteq \neg B_2$ (Negative Inclusion, or simply NI), where B, B_1, B_2 are basic classes. Accordingly, subsumption checking can be done as follows: $O \models B \sqsubseteq B_1$ ($O \models B \sqsubseteq \neg B_2$) iff $B \sqsubseteq B_1 \in \mathcal{T}'$ ($B \sqsubseteq \neg B_2 \in \mathcal{T}'$, respectively) [12]. 2) *Normalisation and storage of individual axioms*: Firstly, given an ontology O and the set $\mathcal{A}$ of individual axioms, $\exists R(\mathsf{a}) \geq n$ and $\exists R^-(\mathsf{b}) \geq n$ are added into $\mathcal{A}$ if $R(\mathsf{a},\mathsf{b}) \geq n \in \mathcal{A}$. Secondly, for each basic class B occurring in $\mathcal{A}$, we define a relational table tab_B of arity 2, such that $\langle \mathsf{a}, n \rangle \in tab_B$ iff $B(\mathsf{a}) \geq n$ (where $n > 0$), and, for each property R occurring in $\mathcal{A}$, we define a relational table tab_R of arity 3, such that $\langle \mathsf{a}, \mathsf{b}, n \rangle \in tab_R$ iff $R(\mathsf{a},\mathsf{b}) \geq n$ (where $n > 0$).[2] Accordingly, O is consistent iff the following two conditions are satisfied: (i) there exist no $\mathtt{a}$ such that $B \sqsubseteq \neg B_2 \in \mathcal{T}'$, $B(\mathsf{a}) \geq n_1 \in \mathcal{A}$, $B_2(\mathsf{a}) \geq n_2 \in \mathcal{A}$ and $n_1 + n_2 > 1$ [4]; (ii) there exist no $\mathtt{a, b, c}$ such that $\mathsf{Func}(R) \in O$ (resp. $\mathsf{Func}(R^-) \in O$), $R(\mathsf{a},\mathsf{b}) \geq n \in \mathcal{A}$ and $R(\mathsf{a},\mathsf{c}) \geq m \in \mathcal{A}$ (resp. $R(\mathsf{b},\mathsf{a}) \geq n \in \mathcal{A}$ and $R(\mathsf{c},\mathsf{a}) \geq m \in \mathcal{A}$).

[2] To simplify the presentation, we assume that if there exist multiple fuzzy class assertions about B(a), such as $B(\mathsf{a}) \geq n_1, B(\mathsf{a}) \geq n_2$ with $n_1 \geq n_2$, we only keep $B(\mathsf{a}) \geq n_1$ in O; similarly, this assumption holds for fuzzy property assertions.

3 ONTOSEARCH2

ONTOSEARCH2 [13] has two principal components, namely an ontology repository and query engine. It stores approximations of OWL ontologies in DL-Lite, and allows queries to be executed over all or part of this repository using SPARQL [14]. By using a DL-Lite approximation ONTOSEARCH2 is significantly faster than other comparable tools which perform full OWL DL entailment (up to two orders of magnitude faster on larger datasets [13]).

3.1 Semantic Approximation

Query answering over OWL DL is a hard problem. It has been shown that the complexity of ontology entailment in $\mathcal{SHOIN}(\mathbf{D}^+)$, i.e., OWL DL, is NEXPTIME. This indicates query answering over OWL DL ontologies is at least NEXPTIME. Approximation has been identified as a potential way to reduce the complexity of reasoning over OWL DL ontologies. Previous approaches [15–19] are mainly based on syntactic approximation of ontological axioms and queries. All these approaches can introduce unsound answers. To the best of our knowledge, we have not seen any published framework on sound (and possibly incomplete) approximations for ontology query answering, not to mention efficient ones.

We have reformulated the idea of knowledge compilation [20] as semantic approximation of OWL DL ontologies. The idea of knowledge compilation is simple: users write statements in an expressive representation language and these statements are compiled into a restricted language that allows efficient inference. In this way, users do not have to use a less expressive language which might be too limited for practical applications. In [20], Selman and Kautz showed how propositional logical theories can be compiled into Horn theories that approximate the original information; they also applied this idea on subsumption reasoning for the Description Logic f_L. We have applied knowledge compilation on query answering over OWL DL ontologies [21].

This approach guarantees soundness and completeness for all queries with non-distinguished variables. By using this technique to reduce the complexity of query answering, we can perform conjunctive queries against large knowledge bases. By treating the combined TBoxes of the ontologies in the knowledge base as a single ABox for a meta-ontology representing the underlying structure of DL-Lite, we can allow users to perform searches for structural patterns across multiple ontologies. In tests we have found the performance of the ONTOSEARCH2 query engine to exceed that of other knowledge base systems [13].

3.2 Fuzzy Query Answering

Encouraged by Straccia's work [4], we investigated query languages and query answering (using these languages) over f-DL-Lite. We implemented two different languages for querying a fuzzy DL-Lite knowledge base in ONTOSEARCH2 [11].

We define *conjunctive threshold queries* (CTQ) which extend atoms $\mathsf{A}(v), R(v_1, v_2)$ in conjunctive queries of the form (1) into the following forms $\mathsf{A}(v) \geq t_1, R(v_1, v_2) \geq$

t_2, where $t_1, t_2 \in (0, 1]$ are thresholds. It is obvious that threshold queries are more flexible than queries of the crisp form in that users can specify different thresholds for different atoms in their queries.

Since f-DL-Lite associates assertions with degrees of truth, another useful feature for its query language is to associate degrees of truth with answers in answer sets of queries over f-DL-Lite ontologies. In threshold queries, an evaluation $[X \mapsto S]$ either satisfies the query entailment or not; hence, answers of such queries are crisp. We introduce general fuzzy queries [11] which allow fuzzy answers. Syntactically, general fuzzy conjunctive queries (GFCQ) generalise the fuzzy threshold query language and Straccia's query language, [11].

In performance tests of these query languages, our fuzzy query engine is in most cases close to the performance of the crisp query engine used for DL-Lite queries in ONTOSEARCH2.

3.3 Ontology Search

When a new OWL ontology is submitted to ONTOSEARCH2, the ontology is first examined for metadata. This is stored as a mapping between the objects in the ontology (OWL ontologies, classes, properties, and instances) and keywords that appear in the metadata attached to that object (currently, the label and comment properties, and the URL of the object). How the keyword is related to the object is used to give a weighting for each mapping: for example, a keyword occurring in a label is given the highest weighting, and a keyword in a comment is given the lowest. In addition, keywords can be inherited from parents of an object (super classes or super properties); instances inherit the keywords of their classes, and all objects in an ontology will inherit the metadata of the ontology itself. In all cases, the greater the semantic distance between two objects which have an inheritance relationship, the lower the weighting is for the keywords inherited. Punctuation is removed from keywords before they are added to the metadata repository. If a keyword is less than three characters long or is a common word such as "and", "the", or "some", it is discarded.

From this fuzzy ontology, we find objects within the repository which match the requirements of the original query. The weightings are used to specify minimum weights required for each term in the query, or they are used to rank the results by relevance to the initial terms.

Queries can be made through a simple keyword based search form, or can be submitted as SPARQL queries, optionally containing fuzzy extensions that can specify the degree of confidence required for each term in the query. Keyword based queries are expanded into fuzzy SPARQL queries, so all searches use the same internal process. The most basic search is for a set of keywords, where the results will list ontologies containing all the keywords. The query can be made more specific by adding search directives to the query:

1. *TBox Searching* Restrictions on the search query can specify that a particular keyword should only be matched against a Class or a Property, but not against instance values in the repository. This is done by prefixing the

keyword with *class:*, so the query *class:red wine* would match the keyword "red" in class definitions only, and match the keyword *wine* across the entire ontology. Similarly a keyword can be restricted to only occur within property definitions within the ontology by prefixing the keyword with *property:*. For queries where all keywords should only match class and/or property definitions, the directives *pragma:Class* and/or *pragma:Property* can be added to the query.

2. *ABox Searching* To restrict a search term to only match within ABox (or instance) data, it can be prefixed with *instance:* Similarly the directive *pragma:Instance* will direct the search engine to search instance data only for every keyword. If used with both *pragma:Class* and *pragma:Property*, the search will exhibit its default behaviour, searching the entire contents of the repository (Class and Property definitions as well as instance data).
3. *Other Search Directives* By adding the search directive *pragma:Resource*, the search engine will find the object within each ontology which best matches the search terms. It will also cause the results to be listed as individual resources rather than as ontologies. Therefore if an ontology contains a single class or instance which has a very high match for the keywords, but which as an ontology has a low score, this class will be displayed above other matches. The default behavior is for the whole ontology with the highest sum of its objects' scores to be returned first.

Search results can be ranked as entire ontologies, or as individual objects within each ontology (by using the *pragma:Resource* directive in a query). In the first instance, we sum the total weightings for each object/keyword pairing that matched in an ontology. This total is used to sort the results, with the highest total being returned at the top of the rankings. By ranking results based on the semantic significance of the matches themselves, rather than by tracking the number of links to the ontology ([2], [1]) we are able to find ontologies which are the closest match for a particular query regardless of their popularity.

In the case that results are to be returned as individual objects, the total sum of object/keyword weightings for each object in the repository is used to determine rank. This is used to return a list of all the different objects which matched the search terms.

4 Example

In this section we present a simple example, where we perform the same query in each of the three search mechanisms in ONTOSEARCH2.

4.1 Metadata Keyword Search

The initial search is a keyword based search of the ontology metadata. This matches all objects in the repository, and returns a list of ontologies based on the total value of all matches on objects within each ontology.

The initial search is simply “Chablis Wine”, this returns the ontology which contains the most significant instances of these keywords. Assuming the repository contains only the Wine ontology, this ontology is returned as a match. To improve the search to target the best matching objects within the Wine ontology, the query can be rewritten as “Chablis Wine pragma:Resource”. The *pragma:Resource* directive instructs the search engine to find and rank individual objects in the ontologies. This search then returns a list of all classes, properties, and instances which are associated with these keywords, ranked in order of the semantic significance of those associations. To improve the results further, the directive *pragma:Class* restricts the search to keyword relationships for classes only. The final query “Chablis Wine pragma:Resource pragma:Class” will return a ranked list of all classes in the repository which match the two keywords.

4.2 Query-based Search

This search uses the query answering facilities of ONTOSEARCH2 to find a class which has a particular label, which is a subclass of a class with a different label.

The search is specified in SPARQL. The query is for a class with a label of “Chablis” which is a subclass of a class with a label “Wine”. The query used is shown in listing 1.

```
SELECT ?X WHERE {
  ?X rdfs:label "Chablis" .
  ?X rdfs:subClassOf ?Y .
  ?Y rdfs:label "Wine" .
}
```

Listing 1: SPARQL Query

This is a standard SPARQL query which searches for some class with a *rdfs:label* of exactly “Wine” which has some subclass with a *rdfs:label* of exactly “Chablis”. This query may return classes which are not direct subclasses because of the DL-Lite semantics which underlie the query engine in ONTOSEARCH2, a search engine that used RDF semantics would not match any indirect subclasses.

4.3 Fuzzy Query-based Search

This search uses the fuzzy query engine in ONTOSEARCH2 to find a class which is a subclass of a different class, where both classes have particular metadata associated with them, with a certain level of confidence.

The search is specified using SPARQL with additional fuzzy values included as comments. This current query is for a class with a metadata keyword of “Chablis” with a degree of confidence ≥ 0.7 which is a subclass of a class with a keyword of “Wine” with a confidence of ≥ 0.5. The query is shown in listing 2.

```
SELECT ?X WHERE {
  ?X os2:hasKeyword "Chablis" . #FT# 0.7
  ?X rdfs:subClassOf ?Y .
  ?Y os2:hasKeyword "Wine" .    #FT# 0.5
}
```

Listing 2: fuzzy SPARQL Query

The property *os2:hasKeyword* is a fuzzy property in the ONTOSEARCH2 fuzzy metadata ontology which associates objects with keywords.

5 Related Work

5.1 Ontology Search

Currently the two main ontology search engines are Swoogle [1] and OntoKhoj [2]. These both use indexes of metadata and allow users to search using keywords. Because they both only store metadata about ontologies, but not representations of the ontologies themselves, they do not allow queries which specify how two search terms may be related. Both also use the number of links to an ontology or to an object within the ontology to rank the results. This is in contrast to ONTOSEARCH2 which attempts to determine the semantic strength of the association between an object and each keyword that is applicable to that object.

The approach taken by Swoogle and OntoKhoj offers higher performance for purely keyword based searches, but cannot offer the flexibility of ONTOSEARCH2. It is not possible to check requirements more complex than the presence of keywords using a keyword based search tool, further tools must be used to check whether any matching ontology satisfies these requirements.

5.2 Ontology Querying

Ontology querying can be provided by a tableax DL reasoner such as Pellet or Racer, or by OWL repositories such as KAON2 [22] or OWLIM [22]. When posing queries against an ontology with a large ABox, tableax based reasoners suffer because of the NEXPTIME requirement for querying OWL. In [23], Motik and Sattler compare the performance of various knowledge management tools using several benchmark ontologies. When using the Lehigh University benchmark [24] with large datasets, both the reasoners tested were unable to perform the queries against larger ontologies within the time constraints.

The KAON2 system uses novel algorithms to reduce a $\mathcal{SHIQ}(\mathbf{D})$ ontology to a disjuntive datalog program. This provides sound and complete reasoning for all queries not containing non-distinguished variables, similar to the semantic approximation technique used in ONTOSEARCH2, but preliminary results indicate the performance of KAON2 is less than that of ONTOSEARCH2 for most ontologies tested [21]. OWLIM uses a subset of OWL-DL, a restricted subset of OWL-DLP. This gives OWLIM excellent performance for performing queries against large OWL ontologies, but it does not guarantee sound or complete results.

6 Conclusions and Future Work

In this paper we have presented three methods for semantically searching ontologies. By offering different methods of querying for data, we can allow users to choose the best tool available for their level of expertise and their requirements. We make use of the semantic significance of different forms of metadata to create a fuzzy DL-Lite ontology mapping objects to the keywords used to describe them.

Future work is centered around improving the metrics used to give different weightings. Additionally, we plan to investigate the use of machine learning techniques to evaluate the success of the weightings currently being used by comparing the ontologies users eventually select for their application, the position in the results which that ontology held, and the weightings used to generate the results. To help users to find the best ontology or resource we are working on an ontology browser integrated with the search engine which will allow ontologies to be explored graphically, with matching resources highlighted, on the results page.

Another aim is to expand the metadata captured to include other recognised sources of metadata such as Dublin Core [25] and to eventually index all datatype properties present in an ontology as potential metadata.

Acknowledgement This research was partially supported by (1) the Advanced Knowledge Technologies (AKT) project which is sponsored by the UK Engineering and Physical Sciences Council under grant number GR/N15764/01 and (2) the FP6 Network of Excellence EU project Knowledge Web (IST-2004-507842). We would also like to thank Giorgos Stamou and Giorgos Stoilos for their discussions and contributions on the fuzzy query answering component.

References

1. Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., Kolari, P.: Finding and Ranking Knowledge on the Semantic Web. In: Proceedings of the 4th International Semantic Web Conference. LNCS 3729, Springer (2005) 156–170
2. Patel, C., Supekar, K., Lee, Y., Park, E.K.: Ontokhoj: a semantic web portal for ontology searching, ranking and classification. In: WIDM '03: Proceedings of the 5th ACM international workshop on Web information and data management, New York, NY, USA, ACM Press (2003) 58–61
3. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R., Vetere, G.: DL-Lite: Practical Reasoning for Rich DLs. In: Proc. of the DL2004 Workshop. (2004)
4. Straccia, U.: Answering vague queries in fuzzy DL-Lite. In: Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, (IPMU-06). (2006) 2238–2245
5. Uschold, M., Gruninger, M.: Ontologies: Principles, Methods and Applications. The Knowledge Engineering Review (1996)
6. Brickley, D., Guha, R.: http://www.w3.org/TR/rdf-schema/. http://www.w3.org/TR/rdf-schema/ (2004)

7. Smith, M.K., Welty, C., McGuiness, D.L.: http://www.w3.org/TR/owl-guide/. http://www.w3.org/TR/owl-guide/ (2004)
8. Donini, F.M.: Complexity of Reasoning. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003) 96–136
9. Straccia, U.: Reasoning within fuzzy description logics. Journal of Artificial Intelligence Research **14** (2001) 137–166
10. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice-Hall (1995)
11. Pan, J.Z., Stamou, G., Stoilos, G., Thomas, E.: Efficient query answering over fuzzy dl-lite ontologies. Technical report, University of Aberdeen (2007) http://www.ontosearch.org/TR/f-DL-Lite.pdf.
12. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., , Rosati, R.: DL-Lite: Tractable description logics for ontologies. In: Proc. of AAAI 2005. (2005)
13. Pan, J.Z., Thomas, E., Sleeman, D.: ONTOSEARCH2: Searching and Querying Web Ontologies. In: Proc. of WWW/Internet 2006. (2006) 211–218
14. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF (2006) W3C Working Draft, http://www.w3.org/TR/rdf-sparql-query/.
15. Stuckenschmidt, H., van Harmelen, F.: Approximating Terminological Queries. In: Proc. of FQAS2002). (2002) 329–343
16. Wache, H., Groot, P., Stuckenschmidt, H.: Scalable Instance Retrieval for the Semantic Web by Approximation. In: Proc. of WISE-2005 Workshop on Scalable Semantic Web Knowledge Base Systems. (2005)
17. Hitzler, P., Vrandecic, D.: Resolution-Based Approximate Reasoning for OWL DL. In: Proc. of the 4th International Semantic Web Conference (ISWC2005). (2005)
18. Groot, P., Stuckenschmidt, H., Wache, H.: Approximating Description Logic Classification for Semantic Web Reasoning. In: Proc. of ESWC2005. (2005)
19. Hurtado, C., Poulovassilis, A., Wood, P.: A Relaxed Approach to RDF Querying. In: Proc. of the 5th International Semantic Web Conference (ISWC-2006). (2006)
20. Selman, B., Kautz, H.: Knowledge Compilation and Theory Approximation. Journal of the ACM (JACM) **43**(2) (1996) 193–224
21. Pan, J.Z., Thomas, E.: Approximating owl-dl ontologies. Technical report, University of Aberdeen (2006) http://www.ontosearch.org/TR/approximating.pdf.
22. Kiryakov, A., Ognyanov, D., Manov, D.: Owlim a pragmatic semantic repository for owl. In: Proc. of Int. Workshop on Scalable Semantic Web Knowledge Base Systems. (2005)
23. Motik, B., Sattler, U.: A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In: Proc. of LPAR 2006. (2006) 227–241
24. Guo, Y., Pan, Z., Heflin, J.: LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics **3**(2) (2005) 158–182
25. DCMI: Dublin Core Metadata Element Set, Version 1.1: Reference Description. DCMI Recommendation, URL http://dublincore.org/documents/dces/ (2003)