A Methodology for Conducting Knowledge Discovery on the Semantic Web

Dimitrios A. Koutsomitropoulos, Markos F. Fragakis, Theodoros S. Papatheodorou

University of Patras, School of Engineering, Computer Engineering and Informatics Dpt. High Performance Information Systems Laboratory
Building B, 26500 Patras – Rio, Greece
kotsomit@hpclab.ceid.upatras.gr,
fragakis@ceid.upatras.gr,
tsp@hpclab.ceid.upatras.gr

Abstract. One of the most prominent features that the Semantic Web promises to enable is the discovery of new and implied knowledge from existing information that is scattered over the Internet. However, adding reasoning capabilities to the existing Web infrastructure is by no means a trivial task. Current methods and / or implementations do not seem to be declarative and expressive enough to provide the kind of reasoning support that the Semantic Web users will benefit from. In this paper we propose a methodology based on which, the user can construct and pose intelligent queries to Semantic Web documents in an intuitive manner, without prior knowledge of the document's contents or structure. This methodology is implemented through a knowledge discovery prototype interface that relies on an existing inference engine found suitable for this task.

1 Introduction

One of the most important features promised by the Semantic Web is the capability of reasoning and knowledge discovery. Several solutions have been proposed focused on founding or improving reasoning tasks since the advent of the Semantic Web. Ontology description languages, inference systems and engines, as well as implementations with reasoning capabilities have been developed in order to improve information retrieval and enable knowledge discovery, if possible. Examining existing approaches reveals that no well established and standardized methodology is followed in general. In addition, the expressiveness achieved varies, not always fulfilling the expressiveness needs of the Semantic Web. Finally, current approaches for composing and performing intelligent queries do not seem to be satisfyingly declarative. Most of the time, the user is burdened with the task of collecting the appropriate information needed to construct the query.

In this paper we propose a methodology for conducting knowledge discovery on the Semantic Web. This methodology consists of three phases: First, the selection of the appropriate logical formalism that will enable the use of existing AI tools and reasoners to perform inferences on Semantic Web documents. Second, the identification of some key aspects that need to be taken into account in order for our method to be suitable for a Web environment. Third, the selection of a specific inference engine that meets the criteria posed in the previous phase.

An integral part of our methodology is the Knowledge Discovery Interface (KDI). The KDI is a web application that has been developed as an implementation of the decisions and criteria developed during the three phases. As such, its target is threefold:

- To be *expressive* enough, in order to allow for powerful inferences on ontology documents.
- To be *declarative*, being independent of the specific ontology schema or contents.
- To be *intuitive*, by aiding the user to compose his query in a user-friendly manner and allowing transparent query composition.

The rest of this paper is organized as follows: In Sect. 2 we review and discuss some previous work on information retrieval and knowledge discovery. Then, in Sect. 3, we propose our methodology for knowledge discovery on the Semantic Web. The KDI is presented in Sect. 4, followed by some experimental results that demonstrate its capabilities. Finally, Sect. 5 summarizes our conclusions.

2 Approaches for Reasoning on the Web

Even though the idea of the Semantic Web has only recently begun to standardize, the need for inference extraction and intelligent behaviour on the Internet has long been a research goal. As expected, there have been some efforts in that direction. Such efforts include ontology description languages, inference engines and systems and implementations, based on them.

Knowing the constraints of knowledge discovery in a random environment like the Internet, and taking in to account the advantages of information retrieval, recent research has tried to combine these two approaches. OWLIR [18] for instance, is a system conducting retrieval of documents that are enriched with markup in RDF, DAML+OIL or OWL. A text editing and extraction system is used to enrich the documents, based on an upper level ontology. This extra information is processed by a rule-based inference system. Search is conducted using classical retrieval methods; however, the results are refined using the inference system results.

The TAP framework [9] seeks as well to improve the quality of search results by utilizing the semantic relationships of web documents and entities.

However, no inference takes place here. Instead, the RDF/OWL documents are treated as structured metadata sets. These sets can be represented as directed graphs, whose edges correspond to relations, and vertices correspond to existing internet resources.

The growth and maintenance of a knowledge base is a strenuous procedure, often demanding a great extent of manual intervention. The Artequakt system [1] tries to overcome this obstacle following an automated knowledge extraction approach. Artequakt applies natural language processing on Web documents in order to extract information and uses CIDOC-CRM as its knowledge base conceptual schema. Nevertheless, it should be noted that no inference – and thus knowledge discovery – takes place.

The Wine Agent system [15] was developed as a demonstration of the knowledge discovery capabilities of the Semantic Web. This system uses a certain domain ontology written in DAML+OIL/OWL and performs inferences on it. The Wine Agent employs a first order logic theorem prover (JTP).

The need for formal querying methods with induction capabilities, has led to DQL [5], as well as OWL-QL [6]. DQL and OWL-QL play an important role in terms of interoperability, expansion and enablement of intelligent systems on the Semantic Web. Nevertheless, they do not provide a direct answer to the knowledge discovery issue. Instead, they serve mainly as communication protocols between agents.

3 An Inference Methodology for the Semantic Web

In this section we propose the basic characteristics that a concrete and declarative method for designing intelligent queries should have in order to be appropriate for the Semantic Web environment. Having decided on an appropriate logical formalism, the resulting methodology relies on an inference engine that should meet certain criteria and be suitable for this purpose. By combing low-level functions of such an engine, our methodology aims at helping the user construct his query in a declarative manner and enables expressive intelligent queries to web ontology documents.

3.1 Choosing the Appropriate Formalism

Choosing an underlying logical formalism for performing reasoning is crucial, as it will greatly determine the expressiveness to be achieved. In this subsection we will attempt to examine some available formalisms, as well as a number of existing tools for each of them.

Description Logics (DL) form a well defined subset of First Order Logic (FOL). It has been shown [12] that OWL DL can be reduced in polynomial time into *SHOIN*(D), while there exists an incomplete translation of *SHOIN*(D) to *SHIN*(D). This translation can be used to develop a partial, though powerful reasoning system for OWL DL. A similar procedure is followed for the reduction of OWL Lite to *SHIF*(D), which is completed in polynomial time as well. In that manner, inference engines like FaCT and RACER can be used to provide reasoning services for OWL Lite/DL.

A fairly used alternative are inference systems that obtain reasoning using applications based in **FOL (theorem provers)**. Such systems are Hoolet, using the Vampire theorem prover, Surnia, using the OTTER theorem prover and JTP [7] , used by the Wine Agent. Inference takes place using axioms reflecting the semantics of statements in OWL ontologies. Unfortunately, these axioms often need to be inserted manually. This procedure is particularly difficult not only because the modeling axioms are hard to conceive, but also because of their need for thorough verification. In fact, there are cases where axiom construction depends on the specific contents of the ontology [15].

Another alternative is given by **rule based reasoning systems**. Such systems include DAMLJessKB [16] and OWLLisaKB. The first one uses Jess rule system to conduct inference on DAML ontologies, whereas the second one uses the Lisa rule system to conduct inference on OWL ontologies. As in the case of theorem provers, rule based systems demand manual composition of rules that reflect the semantics of statements in OWL ontologies. This can also be a possible reason why such systems can presently support inference only up to OWL Lite.

DLs seem to constitute the most appropriate available formalism for ontologies expressed in DAML+OIL or OWL. This fact also derives from the design process of these languages. In fact, the largest decidable subset of OWL, OWL DL, was explicitly intended to show well studied computational characteristics and feature inference capabilities similar to those of DLs. Furthermore, existing DL inference engines seem to be powerful enough to carry out the inferences we need.

3.2 Suitability for the Web

Research regarding the use of ontologies and Description Logics for semantic matchmaking of Web Services descriptions [8, 17, 20] as well as other sources [19] has led to the recognition of a common query method, using subsumption relationships between concepts. According to this method, the query is modeled as a new concept, using the Description Logic constructors, and then classified in the hierarchy. Subsumption relationships then determine the absolute and relevant answers to the query. Furthermore, every instance can be modeled as an atomic concept, so queries demanding use of instances can be conducted the same way [14]. The method just described, could be characterized as *taxonomic* since it is entirely based on the functions provided by the TBox of a knowledge base.

Although artificial class creation seems adequate for the matchmaking of Web Service descriptions, querying the Semantic Web demands the added expressiveness provided by the instances. The ABox intelligent functions, such as instance checking, are of crucial importance when domain modelling calls for the fine-grained analysis that instances enable. This is especially true in an environment like the Web, where most of the semantic structures are of unspecified and arbitrary detail. In any case, inserting instances in an ontology enables inferences and expressions that would have been impossible to accomplish using concept classification solely.

The KDI presented in the following section utilizes therefore an *instance-based* method for conducting inferences that employs in its core two basic ABox functions: *instance checking*, that finds the concepts an instance belongs to and *role fillers retrieval*, which, given a specific instance and role, infers all related instances through this role. By utilizing these two features, in combination with the TBox functions, we can achieve not only the retrieval of information that has been explicitly expressed in the ontology, but also the discovery of knowledge that is logically deduced by this information. Therefore TBox as well as ABox support is an important criterion for selecting an appropriate reasoning back-end.

3.3 Choosing a Reasoning Back-End

Having chosen to use the DLs as the underlying formalism for our methodology, and having noted the most important characteristics a suitable implementation should have, we will now examine three inference engines based on DLs. Our evaluation is carried out in terms of expressiveness, support for OWL, reasoning about ABox and other main features provided by each of the three systems.

Cerebra, by Network Inference is a recent commercial system, providing reasoning as well as ontology management features. Cerebra supports nearly all constructors and axioms that would normally classify it to OWL DL expressiveness level. However, our experimentation with the system has shown some deficiencies which, in combination with its inability to reason about the ABox finally ranks Cerebra's expressiveness at *SHIQ* level, at most. On the other hand, the relation model used by Cerebra allows the submission of very powerful queries, based on XQuery syntax. Still, the results are based only on the explicitly expressed information of the ontology, and not on information that may be inferred.

FaCT/FaCT++ [13] is a freely available reasoning software, that is being developed at Manchester. FaCT implements optimized, sound and complete algorithms to compute subsumption in the *SHIQ*(D) DL. FaCT does not support reasoning in ABox, neither concrete domains. It is also syntactically incompatible with OWL, since its knowledge bases are expressed in a Lisp-like or XML syntax.

FaCT++ [21] features greater expressiveness, aiming ultimately at OWL DL, by fully supporting concrete domains, while the underlying logic is *SHIF*(D). OWL syntax is not supported; however a transformation tool to the Lisp intermediate form is available. Individuals (and thus nominals) survive this transformation, but they are not yet fully supported, as they are approximated as primitive concepts.

RACER [10, 11] is an inference engine for very expressive DLs. It is the first system in its category to support reasoning in ABox as well as TBox, and this is its main asset in comparison to the other inference engines. RACER provides reasoning for *SHIQ*(D), including instances (ABox). There is also full support for the OWL syntax. On the other hand, nominals are only approximated by creating a new concept for each of them.

RACER seems to be closer to the expressiveness needed by the Semantic Web mostly because of its enhanced support for OWL and its clear ability to reason about the ABox. Its utilisation in the KDI produced a number of interesting results, some of which are presented in Sect. 4.2.

4 The Knowledge Discovery Interface

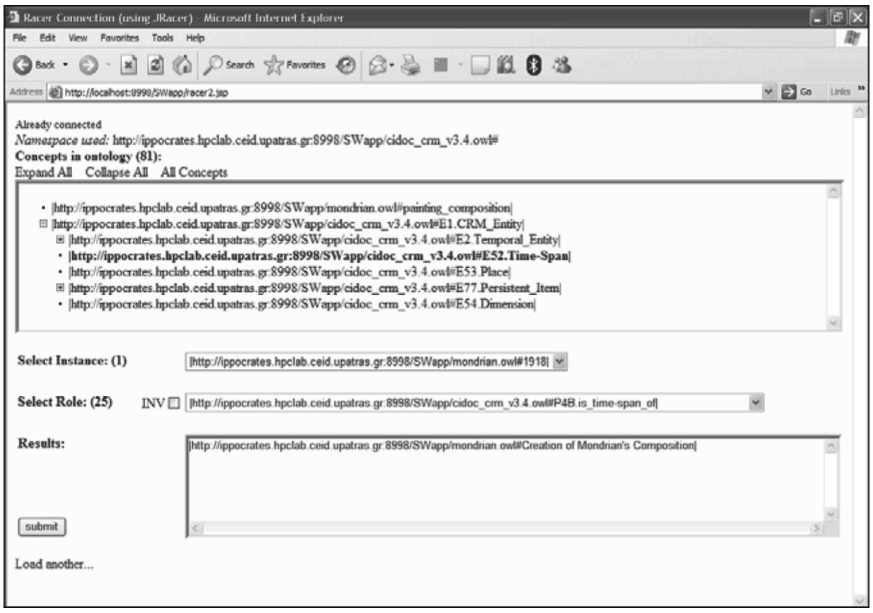
In this section we describe the prototype Knowledge Discovery Interface. The KDI has been developed as an implementation of the decisions and criteria identified in the previous section. First we give a general description of the KDI along with a brief description of its functionality and the relation of its programmatic components to RACER functions. Then, we present two experimental inferences on OWL documents performed using the KDI and their results.

4.1 General Description and Architecture

The KDI is a web application, providing intelligent query submission services on Web ontology documents. We use the word *Interface* in order to emphasize the fact that the user is offered a simple and intuitive way to compose and submit queries. In addition, the KDI interacts with RACER to conduct inferences.

KDI is capable of loading ontologies of any structure and content. Furthermore, the user is capable of browsing the existing classes, their instances and their corresponding roles, thus presenting the user with all the needed information to compose his query. We have identified such a *declarative* behaviour to be of crucial importance for the Semantic Web knowledge discovery process; after all, the user should be able to pose queries even to unknown ontologies, encountered for the first time (Fig. 1).

KDI helps the user compose a query by selecting a concept, an instance and a role in a user friendly manner. After the query is composed, it is decomposed into several lower level functions that are then submitted to RACER. This procedure is transparent to the user, withholding the details of the knowledge base actual querying. This kind of *intuitive* composition of intelligent queries,


Racer Connection (using JRacer) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Go
Links

http://localhost:8998/SWapp/racer2.jsp

Already connected
Namespace used: http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#
Concepts in ontology (81):
Expand All Collapse All All Concepts

- |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/mondrian.owl#painting_composition|
- ⊟ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E1.CRM_Entity|
  - ⊞ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E2.Temporal_Entity|
  - **|http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E52.Time-Span|**
  - |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E53.Place|
  - ⊞ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E77.Persistent_Item|
  - |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#E54.Dimension|

Select Instance: (1) |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/mondrian.owl#1918|

Select Role: (25) INV ☐ |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/cidoc_crm_v3.4.owl#P4B.is_time-span_of|

Results: |http://ippocrates.hpclab.ceid.upatras.gr:8998/SWapp/mondrian.owl#Creation of Mondrian's Composition|

submit

Load another...

Fig. 1. Ontology classification and intelligent query composition using the KDI. The concept "E52.Time-Span" has been selected, thus only its instances and relevant roles are available. System correctly infers that "1918" is the year of the "Creation of Mondrian's composition"

significantly improves the knowledge discovery process by facilitating the user to pose precise and correct queries.

Finally, Fig. 2 displays the programming architecture of the KDI. This figure shows the various parts of the implemented business-logic (right part) as well as their relation to and dependence on RACER's lower level functions (left part): a functions pattern depicts the components it participates in.

4.2 Results

In the following we present the results from two different inference actions performed using the KDI, so as to demonstrate its capabilities as well as its limitations. In order to conduct these inferences we use the CIDOC Conceptual Reference Model [3, 4] as our knowledge base.

Firstly, we ported version 3.4 of the CRM to OWL format. Secondly we semantically enriched and extended CRM with concrete instances and more expressive structures, available only in OWL (like cardinality restrictions, inverse roles, existential and universal quantifications and so on). We then created a document named mondrian.owl that includes CRM concept and role instances which model facts from the life and work of the Dutch painter Piet Mondrian. In this document we also included axiom and fact declarations that

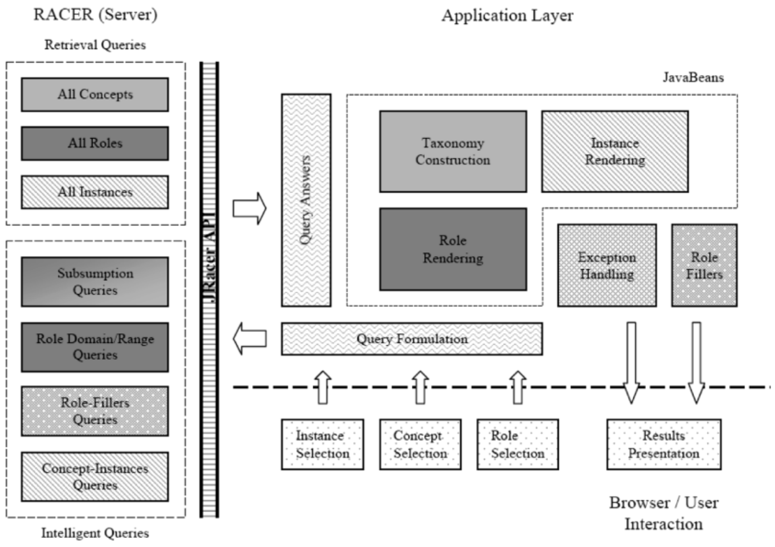


Fig. 2. The Architecture of the Knowledge Discovery Interface

OWL allows to be expressed, as well as new roles and concepts making use of this expressiveness.

The following code is a fragment from mondrian.owl stating that a "Painting_Event" is in fact a "Creation_Event" that "has_created" "Painting" objects only:

```
<owl:Class rdf:ID="Painting_Event">
<rdfs:subClassOf rdf:resource="&crm;E65.Creation_Event"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="&crm;P94F.has_created"/>
<owl:allValuesFrom rdf:resource="#Painting"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<Painting_Event rdf:ID="Creation of Mondrian's composition">
<crm:P94F.has_created rdf:resource="#Mondrian's composition"/>
</Painting_Event>
```

The above fragment is graphically depicted in the left part of Fig. 3. "Creation of Mondrian's Composition" ($\boldsymbol{i}_1$) is an explicitly stated "Painting_Event" that "has_created" ($\boldsymbol{R}$) "Mondrian's composition" ($\boldsymbol{i}_2$). Now, asking the KDI to infer "what is a painting?" it infers that i_2 is indeed a painting (right part of Fig. 3), correctly interpreting the value restriction on role R.

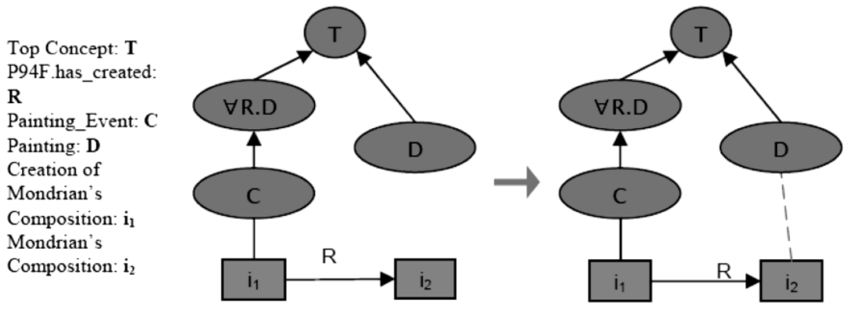


Fig. 3. Inference example using Value Restriction

Let's now examine another example that involves the use of nominals. The following fragment from mondrian.owl states that a "Painting" is a "Visual_Item" that its "Type" is "painting_composition".

```
<owl:Class rdf:ID="Painting">
<owl:subClassOf rdf:resource="&crm;E36.Visual_Item"/>
<owl:equivalentClass>
<owl:Restriction>
<owl:onProperty rdf:resource="&crm;P2F.has_type"/>
<owl:hasValue
rdf:resource="#painting_composition"/>
</owl:Restriction>
</owl:equivalentClass>
</owl:Class>
<crm:E55.Type rdf:ID="painting_composition"/>
<Painting rdf:ID="Mondrian's composition"/>
```

The above fragment is graphically depicted in the left part of Fig. 4.

"Mondrian's Composition" ($\boldsymbol{i}_1$) is explicitly declared as a "Painting" instance which in turn is defined as a hasValue restriction on "has_type" ($\boldsymbol{R}$). "Painting_composition" ($\boldsymbol{i}_2$) is declared as a "Type" object. While the fact

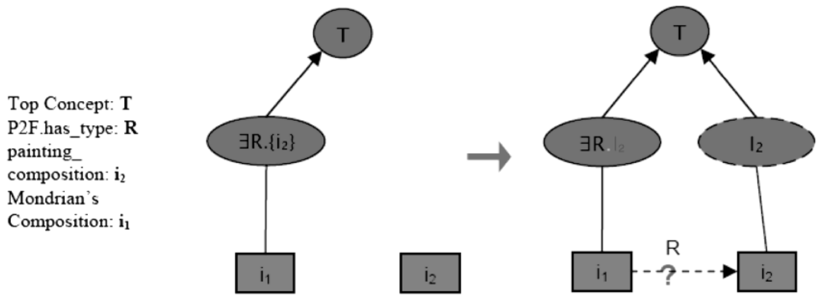


Fig. 4. Inference example using Existential Quantification and nominals

that “Mondrian’s Composition” “has_type” “Painting” is straightforward, the KDI is unable to infer so and returns *null* when asked “what is the type of Mondrian’s composition?”

This example clearly demonstrates the inability of RACER as well as every other current DL based system to reason about nominals. Given the $\{i_2\}$ nominal, RACER creates a new synonym concept I_2 and makes $i_2$an instance of I_2. It then actually replaces the hasValue restriction with an existential quantifier on *concept* I_2 and thus is unable to infer that $R(i_1, i_2)$ really holds.

5 Conclusions

In this paper we have shown how the Semantic Web may fulfill a large part of what it promises, mainly through its knowledge discovery features. These features are grounded on a well-studied background and their adaption to the Web environment is now mature and even standardized to some extent. The KDI and its underlying methodology demonstrate proper evidence of how these features can be practically applied so as to be beneficial for a number of applications.

Our proposed methodology comes to fill the gap between existing approaches by designating a process that can be followed to achieve knowledge discovery on the Semantic Web. Our choices depend on the current state-of-the-art in inference engines and Semantic Web standardization efforts. We trust that at the near future most of the difficulties and incompatibilities identified throughout our work would be overridden by the evolution of systems and the refinement and possibly enrichment of the Ontology Web Language.

The KDI for example is greatly hampered by the limited expressiveness and scalability of current DL inference engines, regarding the use of nominals and the processing of large ontology documents respectively. Despite that fact, the KDI displays a combination of innovations and distinctive features that are not, to our knowledge, simultaneously met by any other system. Among the most important of them is an intuitive and declarative way of constructing and submitting queries and the implementation of an original inference methodology that is especially suitable for the Semantic Web environment.

References

1. H. Alani, S. Kim, D.E. Millard, M.J. Weal, W. Hall, P.H. Lewis and N.R. Shadbolt. Automated Ontology-Based Knowledge Extraction from Web Documents. IEEE Intelligent Systems, 18(1): 14–21, 2003.
2. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider and L.A. Stein. OWL Web Ontology Language Reference. W3C Recommendation, 2004. http://www.w3.org/TR/owl-ref/

3. N. Crofts, M. Doerr and T. Gill. The CIDOC Conceptual Reference Model: A standard for communicating cultural contents. Cultivate Interactive, issue 9, 2003. http://www.cultivate-int.org/issue9/chios/
4. M. Doerr. The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. AI Magazine, 24(3): 75–92, 2003.
5. R. Fikes, P. Hayes and I. Horrocks. DQL – A Query Language for the Semantic Web. KSL Technical Report 02–05, 2002.
6. R. Fikes, P. Hayes and I. Horrocks. OWL-QL: A Language for Deductive Query Answering on the Semantic Web. KSL Technical Report 03–14, 2003.
7. R. Fikes, J. Jenkins, and F. Gleb. JTP: A System Architecture and Component Library for Hybrid Reasoning. In Proc. of the Seventh World Multiconference on Systemics, Cybernetics, and Informatics, 2003.
8. J. González-Castillo, D. Trastour and C. Bartolini. Description Logics for Matchmaking of Services, In Proc. of KI-2001 Workshop on Applications of Description Logics, 2001.
9. R. Guha, R. McCool. TAP: A Semantic Web Platform. Computer Networks, 42(5):557–577, 2003.
10. V. Haarslev and R. Möller. Racer: A Core Inference Engine for the Semantic Web. In Proc. of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), pp. 27–36, 2003.
11. V. Haarslev and R. Möller. RACER User's Guide and Reference Manual Version 1.7.19. http://www.sts.tu-harburg.de/∼r.f.moeller/racer/racer-manual-1-7-19.pdf
12. I. Horrocks and P.F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. In D. Fensel, K. Sycara, and J. Mylopoulos (eds.): Proc. of the 2003 International Semantic Web Conference (ISWC 2003), number 2870 of LNCS, pp. 17–29. Springer, 2003.
13. I. Horrocks and U. Sattler. Optimised reasoning for SHIQ. In Proc. of the 15th Eur. Conf. on Artificial Intelligence (ECAI 2002), pp. 277–281, 2002.
14. I. Horrocks and S. Tessaris. Querying the Semantic Web: a Formal Approach. In I. Horrocks and J. Hendler (eds.): Proc. of the 13th Int. Semantic Web Conf. (ISWC 2002), number 2342 in LNCS, pp. 177–191. Springer, 2002.
15. E. Hsu and D. McGuinness. Wine Agent: Semantic Web Testbed Application. In Proc. Of Workshop on Description Logics, 2003.
16. J. Kopena and W.C. Regli. DAMLJessKB: A tool for reasoning with the Semantic Web. IEEE Intelligent Systems, 18(3): 74–77, 2003.
17. L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In Proc. of the Twelfth International World Wide Web Conference (WWW 2003), pp. 331–339. ACM, 2003.
18. J. Mayfield and T. Finin. Information retrieval on the Semantic Web: Integrating inference and retrieval. In Proc. of SIGIR Workshop on the Semantic Web, 2003.
19. Network Inference Ltd. Description Logics (white paper), http://www.networkinference.com
20. M. Paolucci, T. Kawamura, T.R. Payne and K. Sycara. Semantic Matching of WebServices Capabilities. In Proc. of International Semantic Web Conference (ISWC), 2002.
21. D. Tsarkov and I. Horrocks, Reasoner Prototype: Implementing new reasoner with datatype support. IST-2001-33052 WonderWeb Del. 13, http://wonderweb.semanticweb.org