OWL-BASED KNOWLEDGE DISCOVERY USING DESCRIPTION LOGICS REASONERS

Dimitrios A. Koutsomitropoulos, Dimitrios P. Meidanis, Anastasia N. Kandili
and Theodore S. Papatheodorou
High Performance Information Systems Laboratory,School of Engineering, University of Patras, Buidling B, 26500 Patras-Rio, Greece

Keywords: Inference, OWL, Semantic Web, Reasoning, Description Logics.

Abstract: The recent advent of the Semantic Web has given rise to the need for efficient and sound methods that would provide reasoning support over the knowledge scattered on the Internet. Description Logics and DL-based inference engines in particular play a significant role towards this goal, as they seem to have overlapping expressivity with the Semantic Web de facto language, OWL. In this paper we argue that DLs currently constitute one of the most tempting available formalisms to support reasoning with OWL. Further, we present and survey a number of DL based systems that could be used for this task. Around one of them (Racer) we build our Knowledge Discovery Interface, a web application that can be used to pose intelligent queries to Semantic Web documents in an intuitive manner. As a proof of concept, we then apply the KDI on the CIDOC-CRM reference ontology and discuss our results.

1 INTRODUCTION

Regarding the success of the Semantic Web, it may be encouraging that relevant applications and systems that utilize its standardized "toolkit" of languages and specifications tend to proliferate day by day. Even these very specifications are subject to ongoing research that attempts to push the limits of the current Semantic Web idea some steps further. Nevertheless, measuring the success of the Semantic Web could also be regarded by the point of view of the goals achieved so far: Web knowledge management; semantic resource description; and distributed knowledge discovery, as one of the most prominent. In order for this promise not to be failed, Semantic Web surely could only benefit from efficient and sound methods that would provide reasoning support for its underlying knowledge.

Description Logics and DL-based inference engines in particular play a significant role towards this goal, as they seem to have overlapping expressivity with the Semantic Web de facto language, OWL (Bechhofer et al. 2003). In addition, implemented algorithms and reasoning systems for DLs already exist that could be used to provide knowledge discovery facilities on the Semantic Web. Combined, these two facts make the use of DLs one of the most tempting available formalisms to typically support reasoning with OWL.

In this paper we first compare DL-based systems with alternatives based on other formalisms, like rule based systems and theorem provers, and argue that DLs are currently the most suitable means to build reasoning services for the Semantic Web. Then, we present and survey four popular systems from the DLs world and evaluate them in terms of their availability, expressivity and ability to reason with individuals (ABox support): Cerebra, FaCT, FaCT++ and RACER. In order to demonstrate the ability to perform Semantic Web reasoning using DL based systems, we have chosen one of the inference engines above as the core of our *Knowledge Discovery Interface* (KDI). The KDI is a web application that can be used to pose intelligent queries to Semantic Web documents in an intuitive manner. In order to answer these queries, the KDI relies on the reasoning services provided by the underlying inference engine. Finally we construct and use some instances of the CIDOC-CRM ontology, which we then feed in to the KDI and discuss the results from a series of intelligent queries posed.

A. Koutsomitropoulos D., P. Meidanis D., N. Kandili A. and S. Papatheodorou T. (2006).
OWL-BASED KNOWLEDGE DISCOVERY USING DESCRIPTION LOGICS REASONERS.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - SAIC*, pages 43-50
DOI: 10.5220/0002441400430050

The rest of this paper is organized as follows: In section 2 we review and discuss some previous work on information retrieval and knowledge discovery on the Web. Then, in section 3, we compare available reasoning formalisms, survey the DL-based systems and explain our evaluation criteria. The KDI is presented in section 4; we describe its functionality and architecture followed by some experimental results on the CIDOC-CRM ontology that demonstrate its capabilities. Finally, section 5 summarizes the conclusions from our work.

2 APPROACHES TOWARDS KNOWLEDGE DISCOVERY 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.

SHOE (Heflin et al. 1998), was initially developed as an extension to HTML. It enables web page authors to annotate their web documents with machine-readable knowledge. In that way, these documents can be more efficiently retrieved by knowledge-based search engines and then manipulated by agents. Although SHOE has a number of features, some of which are not present in other languages (e.g. n-ary relations), it lacks the expressiveness needed by the Semantic Web (for example see Gomez-Perez & Corcho 2002).

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 (Mayfield & Finin 2003) 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 (Guha & McCool 2003) 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. This representation is conducted based on the information of a local knowledge base.

The growth and maintenance of a knowledge base is a strenuous procedure, often demanding a great extent of manual intervention. The Artequakt system (Alani et al. 2003) 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 (Hsu & McGuinness 2003) 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 (Fikes et al. 2002), as well as OWL-QL (Fikes et al. 2003a). 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 INFERENCE SYSTEMS FOR THE SEMANTIC WEB

In this section we first briefly compare some inference methods for the Semantic Web, alternative to DLs, with DL-based systems. We present the formal relation of DLs with OWL and then discuss a number of systems based either on full First Order Logic (FOL) or rule-based systems. Next we review and evaluate four inference engines that are based on Description Logics and could be used to provide reasoning services in OWL ontologies. Our most important criteria for this evaluation include the expressivity supported by these systems, as well as their ability to reason with the ontology instance space as well. As reasoning in OWL Full is

undecidable, we at least seek a system that could properly support OWL DL.

3.1 Available Formalisms

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 (DLs) form a well defined subset of First Order Logic (FOL). OWL Lite and OWL DL are in fact very expressive description logics, using RDF syntax (Horrocks et al. 2003). Therefore, the semantics of OWL, as well as the decidability and complexity of basic inference problems in it, can be determined by existing research on DLs. OWL Full is even more tightly connected to RDF, but its typical attributes are less comprehensible, and the basic inference problems are harder to compute (because OWL Full is undecidable). Inevitably, only the examination of the relation between OWL Lite/DL with DLs may lead to useful conclusions. On the other hand, even the limited versions of OWL differ from DLs, in certain points, including the use of namespaces and the ability to import other ontologies.

It has been shown (Horrocks & Patel-Schneider 2003) 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.

The selection of a DL system to conduct knowledge discovery is not the only option. A fairly used alternative are inference systems that achieve 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 (Fikes et al. 2003b) 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 (Hsu & McGuinness 2003).

Another alternative is given by **rule based reasoning systems**. Such systems include DAMLJessKB (Kopena & Regli 2003) 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.

Rule based inference services are also included in the Jena framework (McBridge 2002), developed by Hewlett-Packard. Jena provides a programming environment for web ontologies and supports inference up to OWL Lite level. IBM's corresponding solution is the SNOBASE ontology management system, featuring similar reasoning capabilities. The popular Protégé environment is also capable of processing ontology documents. Protégé's recent support for OWL enables its interconnection with inference systems, but only to provide classification and subsumption services in the class hierarchy.

On the other hand, neither the currently available Description Logic systems nor the algorithms they implement, support the full expressiveness of OWL DL. Even if such algorithms are implemented, their efficiency will be doubtful, since the corresponding problems are solved in non deterministic exponential time.

Nevertheless, DLs seem to constitute the most appropriate available formalism for ontologies expressed in DAML+OIL or OWL. This fact also derives from the designing 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 DL Systems Evaluation

Having discussed the pros and cons of DLs as the underlying reasoning formalism for the Semantic Web we will now examine four inference engines based on DLs: Cerebra, FaCT, FaCT++ and RACER. Our evaluation, summarized in Table 1, is carried out in terms of their availability, expressiveness, support for OWL, reasoning about ABox and interconnection capabilities provided.

3.2.1 Cerebra

Cerebra, by Cerebra Inc. (formerly Network Inference) is a recent commercial system, providing reasoning as well as ontology management features. Cerebra differs from traditional DL based systems, in that it provides some extra features that may be desirable in a production environment. Nevertheless, its expressive power is by no means exceedingly different.

Indeed, one interesting feature of Cerebra is the ability to add persistency to the knowledge bases that is able to process. Cerebra can load OWL documents either from the local file system or directly from the Web, provided the corresponding URL. The ontology information is stored, following an internal data model, in a relational database and can then be reloaded if needed.

Cerebra provides for connecting with client applications written in Java or .NET. Further, any web service may use its functionality through its SOAP interface. Clients written in Java can connect to the system either through RMI or SOAP, by using the classes provided by Cerebra for this purpose. For .NET, Cerebra provides a .dll library which can be used to connect with the SOAP interface. In both cases there is an API that provides for processing, managing and posing queries to ontologies. Query composition, especially when involving instances, follows to some extent the XQuery standard.

To our knowledge, there is no formal documentation for Cerebra's expressive power. It is known however that Cerebra's internal semantic model for conducting inferences is based on DLs. Our experimental evaluation of the system has shown that, for the taxonomic part of the ontology (the TBox), Cerebra supports nearly all constructors and axioms for classes and roles (including set-theoretic operations) that would normally classify it to OWL DL expressiveness level. However, further experimentation with the system has revealed the following:

- Symmetric roles cannot be recognized. This was confirmed as a system's bug.
- Minimum cardinality greater than 1 cannot be expressed (e.g. minCardinality=2), which is especially useful when modelling number restrictions.The most important, inference based on instances (ABox) is not supported. One possible exception is the instanceProperty function. However, given a class and a role, instanceProperty returns all the instance pairs that are inferred to be related through the given role, and its left argument comes from the given class.

The above rank Cerebra's expressiveness at *SHIQ* level, at most. On the other hand, the relational model used by Cerebra allows the submission of very powerful instance-retrieval queries, based on XQuery syntax. These queries may involve data types as well, like strings and numbers, as well as operands between them (equation, comparison). Still, the results are based only on the explicitly expressed information of the ontology, and not on information that could be inferred.

3.2.2 FaCT

FaCT (Horrocks & Sattler 2002) is a freely available reasoning software, that is being developed at Manchester University under Prof. Ian Horrocks. Initially, FaCT supported the *SHF* DL and then evolved to include *SHIF* and finally *SHIQ*. FaCT's latest versions allow its interconnection with other applications following the client – server model through a CORBA interface. Furthermore, they support the DIG/1.0 standard, which prescribes a simple communication protocol through the exchange of XML requests and responses over HTTP.

FaCT implements optimized complete and sound algorithms to solve the subsumption problem in the Description Logics mentioned. Even though a pioneering system in its age, whose performance used to rank it very high among other traditional DL systems, FaCT's lack of support for inference in the ABox renders it inappropriate for OWL. Indeed, during our evaluation we attempted to convert a simple OWL ontology to the intermediate form supported by FaCT. This conversion has been

Table 1: Comparison summary of some DL-based inference engines.

	Availability	Connectivity	Reasoning Strength	Native OWL support (syntax)	Reasoning with instances (ABox)
Cerebra	Commercial	RMI, SOAP	*SHIQ*	Yes	No
FaCT	Free	CORBA, DIG/1.0	*SHIQ*	No	No
FaCT++	Free	DIG/1.1	*SHIN*(D)	No	No
RACER	Free(before 1.8)	TCP, DIG/1.0	*SHIQ*(D)	Yes	Yes

achieved using a tool available through the WonderWeb IST project, under which the next version of the system (FaCT++) is also developed. This conversion had the following results:

- Individuals are transformed into primitive concepts.
- Relations between individuals are not preserved.
  The new concepts that were created to represent individuals are now subsumed by the concepts the individuals initially belonged to.

Besides the lack of support for ABox and data types (concrete domains), the system is also syntactically incompatible with OWL. Apart from the intermediate, lisp-like knowledge base format, FaCT also supports ontologies in XML format, following a proprietary schema. Naturally however,

the transition to and from OWL would result in significant information loss.

3.2.3 FaCT++

Many of FaCT's disadvantages are being coped with in the system's next version, FaCT++ (Tsarkov & Horrocks 2003), which is developed as a part of the Wonderweb project. FaCT++ differs from FaCT in many aspects. It is a re-implementation of FaCT in C++, featuring however greater expressivity, aiming ulimately to support OWL DL.

Specifically, full support for concrete domains is added, while the underlying logic is *SHIF*(D). OWL syntax is not supported; however a transformation tool to the Lisp intermediate form supported by FaCT++ is provided. Individuals (and thus nominals) survive this transformation, but they are not yet fully supported, as they are all approximated as primitive concepts.

FaCT++ document version could only run in Linux and provides no communication interface with other applications. Further it can be used only in "batch mode": The user has first to include in a configuration file information about the inp`t ontology and the requested inferences, along with preferred parameters and then let it to be processed by the system. System's next version however (ver. 0.99) appears to support DIG/1.1 plus unqualified number restrictions.

3.2.4 RACER

RACER (Haarslev & Möller 2003) 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 is being developed by profs. Volker Haarslev and Ralf Moeller in Concordia University and Hamburg Technical University respectively. It is freely available for research purposes, while only recently a corporation has been established for the commercial exploitation of the system (Racer v1.8+ aka RacerPro).

RACER's communication with other applications is achieved through the TCP/IP interface provided or through HTTP, since the system supports the DIG/1.1 standard. For TCP communication there are APIs available in C++ as well as in Java. In addition, RACER can been run in "file mode", where the ontology and queries files are given as parameters in the command line.

Apart from the lisp-like knowledge base format, RACER can load and process natively ontologies written in XML, RDF, RDFS, DAML+OIL and finally OWL (since version 1.7.7). The underlying DL is *SHIQ*(D), including instances (ABox). In fact, RACER expressiveness is superior to OWL DL as regards to qualified number restrictions and concrete domains.

Indeed, RACER implements algorithms for conducting inferences based on min/max relations between integers, linear polynomial equalities and inequalities of reals, non-linear polynomial equations of complex numbers and string comparison. On the other hand, OWL allows only expressing equality between an individual and an instance of the concrete domain.

However, OWL semantics are more expressive than the RACER language as far as *nominals* are concerned, because they are not supported by the system. This seems to be the main problem that prevents full compatibility with OWL DL. RACER deals with nominals by creating a new concept for each of them and making the corresponding individual an instance of this new concept.

Despite these limitations 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 subsection 4.2.

4 DL-BASED KNOWLEDGE DISCOVERY

In this section we demonstrate the use of DLs for knowledge discovery on the Semantic Web. First we give a general description of the KDI and the main technologies that were used, along with a brief description of its functionality. Then, using the KDI,

we present two experimental inferences on CIDOC-CRM instances expressed in OWL DL, and their results.

4.1 The Knowledge Discovery Interface

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. The interface design follows the traditional 3-tier model, with an important variation: Where a database server would be typically used, we now use a knowledge base management system (Figure 1). Note that each of the three levels may be physically located on different computer systems.

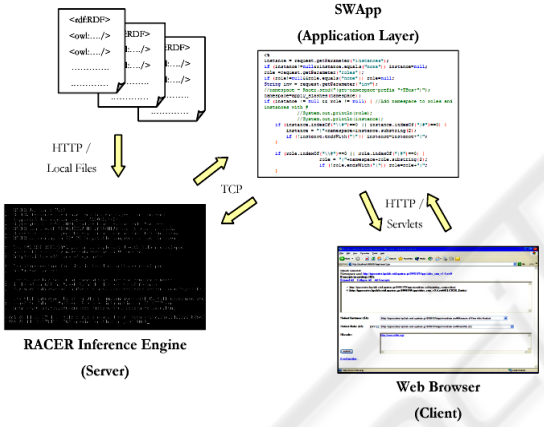


Figure 1: The three levels of the Knowledge Discovery Interface.

The interface can load OWL documents that are available either on the local file system, or on the Internet. A temporary copy of every document is stored locally on the application server and is then loaded by the *knowledge base server* (RACER). RACER creates and stores in memory an internal model for each ontology that it classifies. Classification takes place once for each ontology, during its initial loading. Furthermore, other documents imported by the ontology may be loaded too.

The Interface business logic was implemented using the Java programming language, as well as JSP, JavaBeans and Java Servlets technologies. Tomcat (version 5.0) was used as an *application server*. Business logic is mostly responsible for document loading, proper rendering of the ontological information to the user, composition and submission of queries and formulation and formatting of results. Ontological data and reasoning results are fetched by interacting with RACER over the TCP/IP protocol. This interaction is greatly facilitated through the JRacer API. The latter has been modified in places, mainly in regard to the processing of web documents links and to the processing of synonym concepts.

The user interacts with the *client level*, where the appropriate JSP pages are rendered by his browser. Communication with the application layer is conducted over the HTTP protocol, using forms. At the same time, servlets are used for the administration of multiple user requests and for controlling simultaneous access. Furthermore, when a loaded ontology is not used any more, it is erased from memory, in order to improve the utilisation of system resources.

After connection to RACER has successfully been established, the ontology is loaded and its information is shown on the browser. The user may navigate through the concept hierarchy, which is visualised in a tree form, and select any of the available classes. Upon selection, the page is reloaded, now containing in two drop down menus all of the instances of the selected class, as well as all of the roles whose domain is in this class. The user is able to select an instance and a role and then submit his query by pressing a button. Note that an option is available to invert the selected role, thus resulting in a different query.

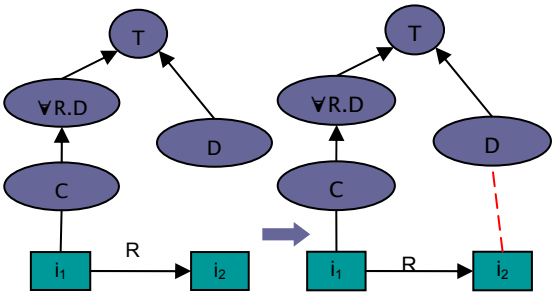
The Interface helps the user compose a query by selecting a concept, an instance and a role in a user friendly manner. After the query is composed, it is decomposed into several lower level functions that are then submitted to RACER. This procedure is transparent to the user, withholding the details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

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 (Crofts et al. 2003) 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 OWL allows to be expressed, as well as new roles and concepts making use of this expressiveness.



Top Concept: **T**
P94F.has_created: **R**
Painting_Event: **C**
Painting: **D**
Creation of Mondrian's Composition: $\mathbf{i_1}$
Mondrian's Composition: $\mathbf{i_2}$

Figure 2: Inference Example using Value Restriction.

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>
```

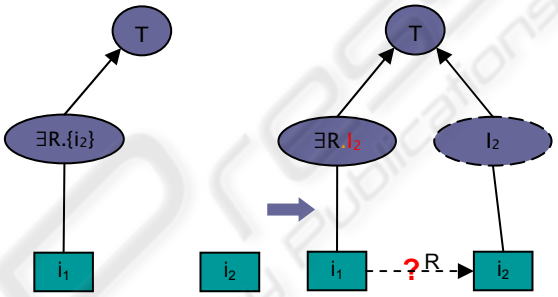
The above fragment is graphically depicted in the left part of Figure 2.

"Creation of Mondrian's Composition" ($\boldsymbol{i_1}$) is an explicitly stated "Painting_Event" that "has_created" (***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 Figure 2), correctly interpreting the value restriction on role R.

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 Figure 3.



Top Concept: **T**
P2F.has_type: **R**
Painting_Composition: $\mathbf{i_2}$
Mondrian's Composition: $\mathbf{i_1}$

Figure 3: Inference Example using Existential Quantification and Nominals.

"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" (***R***). "Painting_composition" ($\boldsymbol{i_2}$) is declared as a "Type" object. While the fact 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 primarily argued about how a well-studied logical formalism, Description Logics, can be utilized in order to enable intelligent querying

of Semantic Web documents. In order to achieve this, a key step was the review of available AI formalisms and system families that could be used to ground reasoning services upon. As the scene is currently set, DL-based systems appear to be the most promising choice to achieve streamlined inference results even in the short term. At the same time, DLs show adequate compatibility and corresponding systems tend to exploit the greatest part out of the Semantic Web ontological formalism expressiveness, as it is now standardized in OWL.

We believe that our hands-on experimentation with a number of state-of-the-art DL inference engines has produced at least two lessons learned: First, the need for instance-based reasoning, which we have shown to be of crucial importance for the Semantic Web environment, is not self-evident in the majority of the systems reviewed; second, we confirmed that even the most advanced DL-systems have problems fully supporting OWL's decidable expressivity.

The potential as well as the limits of the DL-based approach are clearly demonstrated through our "wrapper prototype", the KDI: On the one hand, we have succeeded in demonstrating tangible and meaningful knowledge discovery results on Semantic Web documents. On the other hand, we found that the KDI 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. We trust though 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.

REFERENCES

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.

S. 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/

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/

R. Fikes, P. Hayes and I. Horrocks. DQL - A Query Language for the Semantic Web. KSL TR 02-05, 2002.

R. Fikes, P. Hayes and I. Horrocks. OWL-QL: A Language for Deductive Query Answering on the Semantic Web. KSL TR 03-14, 2003.

R. Fikes, J. Jenkins, and F. Gleb. JTP: A System Architecture and Component Library for Hybrid Reasoning. In Proc. of the 7th World Multiconference on Systemics, Cybernetics, and Informatics, 2003.

A. Gomez-Perez and O. Corcho. Ontology Languages for the Semantic Web. IEEE Intelligent Systems, 17(1):54-60, 2002.

R. Guha, R. McCool. TAP: A Semantic Web Platform. Computer Networks, 42(5):557-577, 2003.

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.

J. Heflin, J. Hendler and S. Luke. Reading between the Lines: Using SHOE to Discover Implicit Knowledge from the Web. In AI and Information Integration: Papers from the 1998 Workshop, pp.51-57. AAAI Press, 1998.

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 2nd Int. Semantic Web Conference (ISWC 2003), pp. 17-29. Springer, 2003.

I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics, 1(1):7-26, 2003.

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.

E. Hsu and D. McGuinness. Wine Agent: Semantic Web Testbed Application. In Proc. Of Workshop on Description Logics, 2003.

J. Kopena and W. C. Regli. DAMLJessKB: A tool for reasoning with the Semantic Web. IEEE Intelligent Systems, 18(3): 74-77, 2003.

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.

B. McBridge. Jena: A Semantic Web Toolkit. IEEE Internet Computing, 6(6):55-59, 2002.

D. Tsarkov and I. Horrocks, Reasoner Prototype: Implementing new reasoner with datatype support. IST-2001-33052 WonderWeb Del. 13, http://wonderweb. semanticweb.org