EQuIKa System: Supporting OWL applications with local closed world assumption

Anees Mehdi* and Jens Wissmann

Institute AIFB, Karlsruhe Institute of Technology, DE
anees.mehdi@kit.edu
Forschungszentrum Informatik Karlsruhe, DE
jens.wissmann@gmail.com

Abstract. One of the major advantages of semantically annotating resources on Web is the facilitation of web services discovery. Languages based on OWL are prune to several problems for web services discovery due to the open-world assumption when handling incomplete information. Thus standard OWL reasoner are usually not suitable for the discovery purposes. The aforementioned problems can easily be fixed by considering some non-monotonic extension of OWL. We present *EQuIKa*, a tool for discovery web resources annotating semantically based on non-monotonic extension of OWL called *autoepistemic description logics (ADL)*. EQuIKa uses a standard reasoner as a black-box and is available as Protégé and NeOn Toolkit plugins.

Introduction

The OWL ontology language and its toolchain have been used in numerous applications to ensure more interoperability, reveal inconsistencies and find new relationships. The underlying open world assumption serves many web applications well as no single agent has complete knowledge about all facts on the web. classical tasks based such as contraints. excluded many applications that are know from classical databases. For example, integrity contraint checks rely on assumptions about the boundaries of facts to consider.

Another example are web services. Web Service Modelling Language (WSML) is specifically designed for annotating Web Services [dBLPF06]. For the purpose of computational feasibility, OWL-DL and WSML-DL are the variants of the OWL and WSML based on description logics [BCM+03]. In [TBP02,GMP04,LH03] the authors suggest the use of standard OWL reasoning for service discovery– *the process of locating a web service as per business request*. Nevertheless in [GMP06] several problems have been identified when OWL inferencing is used for the purpose of semantic services discovery due to the open-world semantics. [GMP06] suggests the use ADL as the query language that allows for

* supported by BMBF project, Sync Tech.

local closed-world reasoning by referring to facts which are explicitly known. Due to space limitation we refer to [MRW13,MR11] for details of such a formalism. Just as an example, to represent the class of all things which are known to be red wine can represented by **K**`RedWine`[1], where **K** is usually called as the epistemic operator and allows to interpret any class or property that it precedes, under the close-world assumption. [MRG11,MR11] presents an approach (along with a prototype) of using standard OWL reasoners for answering ADL queries when put to an ontology. We develop this prototype further in to a full-fledged tool where we improve its overall performance as well as develop plug-ins for the standard ontology editor like Protégé and NeON Toolkit.

EQuIKa: The Epistemic Querying Interface

EQuIKa takes a black box approach in the sense that given an epistemic query it recursively translates the query into a standard reasoning tasks by intermediate calls to the core reasoner. For example, for a given ontology, the member of the class **K**`RedWine` is all those individuals which are explicitly (via assertion or inference) known to be red wines. EQuIKa translate **K**`RedWine` in to the standard **K**-free class $\{w_1, \ldots, w_n\}$ such that the ontology entails `RedWine`(w_i) for each $1 \leq i \leq n$. Such translation indeed requires many intermediate reasoner calls but we have invented several optimization rules (see [MRW13]) for EQuIKa which guarantees boost up in the overall performance.

Implementation and Evaluation

The EQuIKa system is implemented on top of the OWL-API.[2] It can be used as an API as well as within Protégé or NeOn Toolkit. The following considerations and design decisions underly our implementation:

- Since the standard OWL-API does not support epistemic constructs, we extended several classes of the API. The **K**-operator syntactically behaves similar like the *complement construct* (¬) for concepts and like the *inverse role construct* for roles. We therefore followed the same implementation patterns.
- For parsing we created an EpistemicSyntaxParser based on the ManchesterOWLSyntaxOntologyParser. The **K**-operator is expressed by the token `KnownConcept` for concepts and by the token `KnownRole` for the roles.
- In order to support epistemic querying within the Protégé editor, we implemented an additional tab based on the DL Query tab. Figure 1 shows a snapshot of epistemic querying in Protégé.

[1] We use description logic syntax in this paper.
[2] http://owlapi.sourceforge.net/

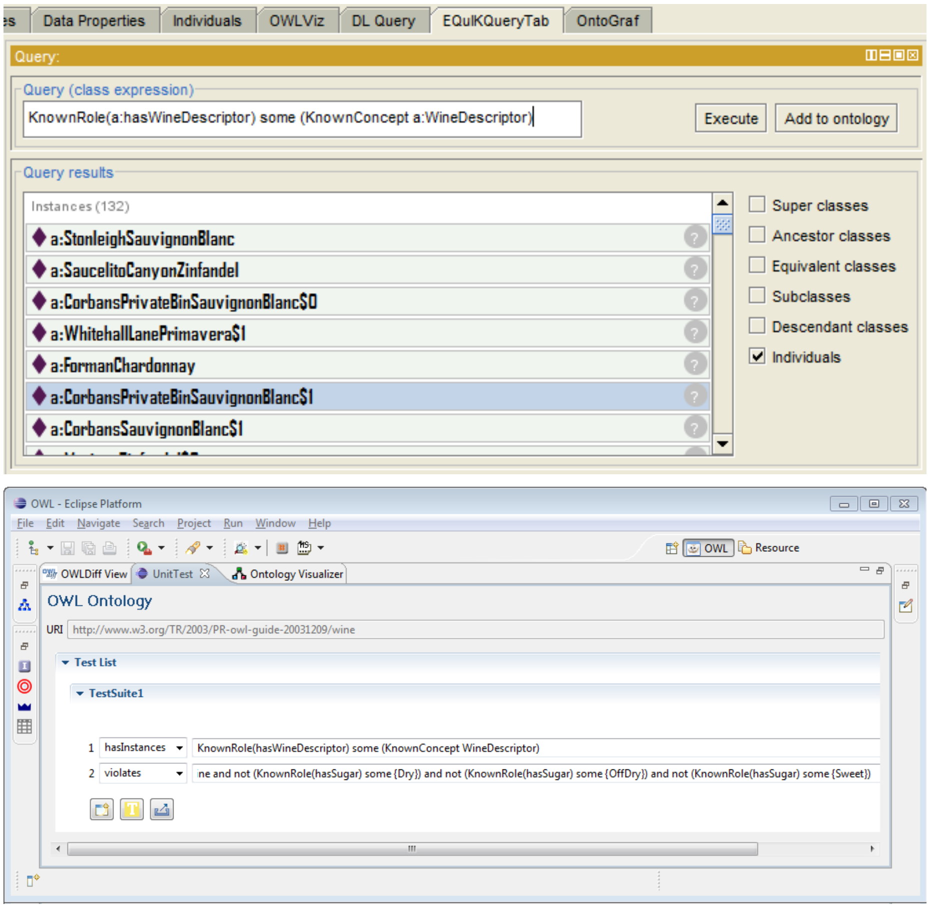

Fig. 1. EQuIKa integration in ontology development tools: Epistemic Querying in Protégé (above) and Integrity Constraint Checking within the NeOn Toolkit (below).

– In order to support epistemic querying within the NeOn Toolkit, we extended the unit testing component of the ontology evolution plugin CHRONOS[3]. Figure 1 shows a snapshot of constraint checking in NeOn.

The class diagram for EQuIKa is displayed in Figure 2. The new types OWLObjectEpistemicConcept and OWLObjectEpistemicRole are derived from the respective standard types OWLBooleanClassExpression and OWLObjectPropertyExpression to fit the design of the OWL-API. As our translation method depends on intermediate calls to a standard reasoner, the class EQuIKaReasoner implements the OWLReasoner interface. As already mentioned, EQuIKa

[3] http://chronos-update.fzi.de

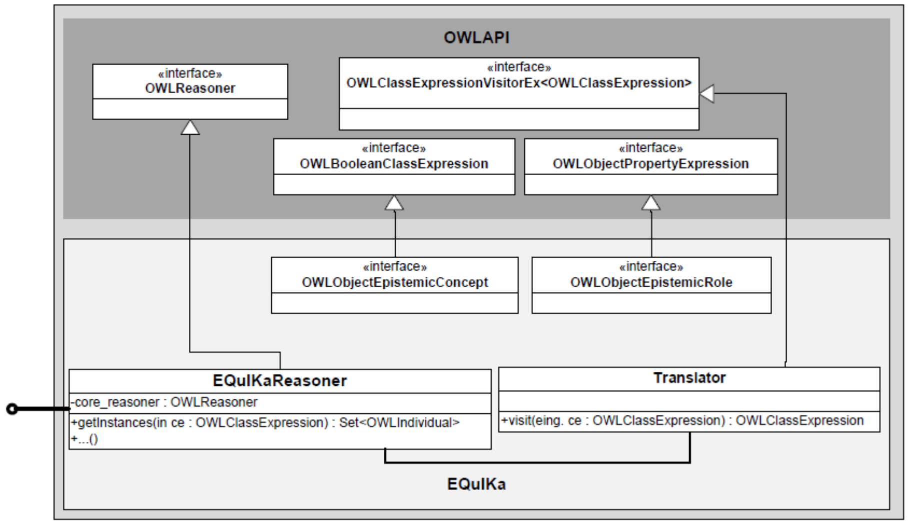

Fig. 2. The *EQuIKa*-system extending the OWL-API

translates an epistemic concept into a **K**-free one in a recursive fashion using the class Translator that implements the OWLClassExpressionVisitor.

Since Protégé and NeOn can utilize any reasoner that implements the OWL-Reasoner interface, EQuIkaReasoner can be easily integrated. Last but not least, EQuIKa has been shared on googlecode for testing purposes.[4] The plugin is provided as jar file[5] that can be installed via the Protégé 4.1 plugin folder.

For the purpose of evaluation, we performed several experiments with the following setup:

- We used a Thinkpad T60: 2 GHz dual core, 2GB RAM, Windows 7.
- For benchmark tests, we used a populated version (483 individuals) of the Wine ontology[6], using most of the OWL 2 DL constructs.
- We constructed several epistemic concepts and translated them into **K**-free ones (Table 1) where $r_1, \ldots, r_{108}$ are individuals representing wine regions in the ontologies.

To the best of our knowledge, EQuIKa is the only reasoner of its nature for epistemic query answering, such that there is no other existing reasoner with

[4] http://code.google.com/p/epistemicdl/
[5] https://epistemicdl.googlecode.com/svn/EpistemicQueryTab/equika.protege.querytab.jar
[6] https://code.google.com/p/epistemicdl/source/browse/trunk/EQuIK/wine_1.owl

Table 1. Concepts used for instance retrieval experiments.

EC_1	$\exists\mathbf{K}\texttt{hasWineDescriptor}.\mathbf{K}\texttt{WineDescriptor}$
EC_2	$\exists\mathbf{K}\texttt{hasWineDescriptor}.\mathbf{K}\texttt{WineDescriptor} \sqcap \exists\mathbf{K}\texttt{madeFromFruit}.\mathbf{K}\texttt{WineGrape}$
EC_3	$\mathbf{K}\texttt{RoseWine}$
EC_4	$\mathbf{K}\texttt{RoseWine} \sqcap \mathbf{K}\texttt{WhiteWine}$
EC_5	$\mathbf{K}\texttt{RoseWine} \sqcap \mathbf{K}\texttt{WhiteWine} \sqcap \{r_1, \ldots, r_{108}\}$
EC_6	$\mathbf{K}\texttt{Wine} \sqcap \neg\exists\mathbf{K}\texttt{hasSugar}.\{Dry\} \sqcap \neg\exists\mathbf{K}\texttt{hasSugar}.\{OffDry\}$ $\sqcap \neg\exists\mathbf{K}\texttt{hasSugar}.\{Sweet\}$

these capabilities against which we could compare EQuIKa's performance. To give an impression about the runtime behavior, we performed two kind of experiments and as a measure, we consider the time required to translate the epistemic concepts (given in Table 1) to $\mathbf{K}$-free equivalent ones and the instance retrieval time of the translated concept. In the first series of experiments, we evaluated

Table 2. Prototpye vs EQuIKa.

Concept	Prototype			EQuIKa		
	T_{trans}	T_{inst}	$\#_{inst}$	T_{trans}	T_{inst}	$\#_{inst}$
EC_1	4	192.7	132	21	97.8	132
EC_2	9	198.9	3	3	37.5	3
EC_3	110	110.1	3	26	26.5	3
EC_4	203	211.7	0	122	122.1	0
EC_5	206	400.6	0	121	121.9	0
EC_6	13	–	–	0.5	487.3	119

EQuIKa against the prototype presented in [MR11]. The corresponding results are shown in Table 2 where T_{trans}, T_{inst} and $\#_{inst}$ represent the translation time (seconds), instance retrieval time (seconds) and the number of instances respectively. One can see that T_{inst} for EQuIKa is far less than for the prototype. In particular for concept $EC6$, the prototype did not responded for almost an hour and we had to abandoned the process, whereas EQuIKa translated $EC6$ and retrieved its instances in few seconds. This shows that the optimization rules we introduced are of high importance toward the feasibility of EQuIKa in practice.

In the second series of experiments, we evaluated the computation time of EQuIKa in general against standard reasoning tasks[7]. For this purpose, we consider non-epistemic concepts $C_1, \ldots, C_6$ obtained by removing $\mathbf{K}$ from $EC_1, \ldots, EC_6$.

[7] T_{inst} and T_{trans} in Table 2 and Table 3 for the same concepts are different reasoning being difficulties in making computation environment (of the machine used) constant.

Table 3. Evaluation epistemic vs. standard instance retrieval

Concept	T_{inst}	$\#_{inst}$	Concept	T_{trans}	T_{inst}	$\#_{inst}$
C_1	2.18	159	EC_1	20	95.7	132
C_2	41.9	159	EC_2	3	36.5	3
C_3	10.7	3	EC_3	10	10.8	3
C_4	2.68	0	EC_4	2	2.9	0
C_5	0.2	0	EC_5	2	2.9	0
C_6	61.1	80	EC_5	0.5	487.3	119

Note that an epistemic concept EC_i and the corresponding C_i are semantically different concepts. Table 3 shows the results of our experiments. It can be seen that even when comparing to the **K**-free counterpart of the epistemic concepts, the computation time of EQuIKa is roughly in the same order of magnitude. This indicates that an explosion of reasoning runtime which often occurs when nonmonotonic features are added to DLs can be avoided in our case.

References

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[dBLPF06] Jos de Bruijn, Holger Lausen, Axel Polleres, and Dieter Fensel. The Web Service Modeling Language WSML: An Overview. In *ESWC*, 2006.

[GMP04] Stephan Grimm, Boris Motik, and Chris Preist. Variance in e-business service discovery. In *In Proceedings of the ISWC Workshop on Semantic Web Services*, 2004.

[GMP06] Stephan Grimm, Boris Motik, and Chris Preist. Matching semantic service descriptions with local closed-world reasoning. In *Proceedings of the 3rd European conference on The Semantic Web: research and applications*, ESWC'06, pages 575–589, Berlin, Heidelberg, 2006. Springer-Verlag.

[LH03] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. 2003.

[MR11] Anees Mehdi and Sebastian Rudolph. Revisiting Semantics for Epistemic Extensions of Description Logics. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI-11)*. AAAI Press, 2011.

[MRG11] Anees Mehdi, Sebastian Rudolph, and Stephan Grimm. Epistemic Querying of OWL Knowledge Bases. In Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z. Pan, editors, *Proceedings of the 8th Extended Semantic Web Conference (ESWC11)*, pages 397–409, 2011.

[MRW13] Aneesl Mehdi, Sebastian Rudolph, and Jens Wissman. EQuIKa: Epistemic Querying in OWL 2 Ontologies. Techreport, AIFB, Karlsruhe, April 2013.

[TBP02] David Trastour, Claudio Bartolini, and Chris Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, 2002.