EQuIKa: Epistemic Querying in OWL 2 Ontologies

Anees Mehdi, Sebastian Rudolph, and Jens Wissmann

Institute AIFB, Karlsruhe Institute of Technology, DE
{anees.mehdi,sebastian.rudolph}@kit.edu
Forschungszentrum Informatik Karlsruhe, DE
wissmann@fzi.de

Abstract. Extending ontology querying facilities with the epistemic operator provides practically useful additional functionalities like ontology introspection, integrity constraints checking, etc. In this paper, we present a practical system called EQuIKa capable of epistemic inferencing on OWL 2 DL Ontologies. It implements our recently developed reduction method of epistemic queries to several subsequent standard reasoning steps. EQuIKa is implemented as a Protégé plugin, featuring a convenient querying interface alike the DL Query tab, enabling epistemic querying directly within the editor. Besides the implementation details, we discuss several optimization issues important for the feasibility of the system in practice. First experiments demonstrate practical feasibility of our system, as the system's runtime lies in the same order of magnitude as standard reasoning tasks.

1 Introduction

OWL 2 DL is the most expressive yet decidable dialect of the Web Ontology Language (OWL) [**?**]. It is based on the description logic (DL) $\mathcal{SROIQ}$ [**?**]. Being a decidable first-order logic fragment, reasoning in $\mathcal{SROIQ}$ is inherently monotonic; addition of information to the ontology never invalidates entailed conclusions. Nevertheless, certain aspects of non-monotonicity are sometimes desired in various semantic applications. In [**?**], the inadequacy of DLs for the task of matching semantic services is discussed and a solution based on a non-monotonic extension of DLs is proposed. A mild form of non-monotonicity which can still be handled rather conveniently but is sufficient for many purposes is when only the querying language is endowed with non-monotonic constructors. In early 1980s, Hector J. Levesque was the first to present the idea of enriching the query language with the epistemic operator $\mathbf{K}$ [**?**]. In [**?**], Raymond Reiter makes a similar arguments and discusses why a language enriched with epistemic operator is desirable for integericty constraint (IC) checking.

Also in the DL community, extensions by epistemic operators have been considered [**?**,**?**,**?**]. In epistemic extension of DLs, the epistemic operator, also called **K**-operator and paraphrased as "known to be, is allowed in front of a concept (role) to represent the individual (pair of individual) known by an ontology to possess the property described by the concept (role). In other words, a query language with **K**-operators allows for ontology introspection.

The **K**-operator allows for epistemic querying. E.g., in order to formulate queries like "known white wine that is not known to be produced in a French regions we could perform an instance retrieval w.r.t. the concept $\mathbf{K}\,\mathit{WhiteWine} \sqcap \neg\exists\mathbf{K}\mathit{locatedIn}.\{\mathit{FrenchRegion}\}$. This concept represents all the wines that are explicitly excluded from being French wines but for which there is also no evidence of being French wines either (neither directly nor indirectly via deduction). For an ontology containing

$$\{\mathit{WhiteWine(MountadamRiesling)}, \mathit{locatedIn(MountadamRiesling,\ AustralianRegion}\}$$

the query would yield *MountadamRiesling* as a result, since it is known to be a white wine (stated explicitly in the ontology) but not known to be produced in France. Note that querying for the instances of the concept $\mathit{WhiteWin} \sqcap \neg\exists \mathit{locatedIn}.\{dlFrenchRegion\}$ yields an empty result. Further, a query language enriched with the **K**-operator allows for IC checking. Such constraint checking capabilities help ontology engineers to determine whether certain properties are satisfied by different elements occurring in the ontology. For example, testing whether the axiom

$$\mathbf{K}\,\mathit{Wine} \sqsubseteq \exists\mathbf{K}\mathit{hasSugar}.\{Dry\} \sqcap \exists\mathbf{K}\mathit{hasSugar}.\{OffDry\} \sqcap \exists\mathbf{K}\mathit{hasSugar}.\{Sweet\}$$

is entailed by an ontology allows to check if for every named individual in that ontology that is known to be a wine, it is also known (i.e., it can be logically inferred from the ontology) what degree of sugar it has. Note that this cannot be taken for granted even if $\mathbf{K}\,\mathit{Wine} \sqsubseteq \exists\mathbf{K}\mathit{hasSugar}.\{Dry\} \sqcap \exists\mathbf{K}\mathit{hasSugar}.\{OffDry\} \sqcap \exists\mathbf{K}\mathit{hasSugar}.\{Sweet\}$ is stated in (or can be derived from) the ontology.

In this paper, we present a system called *EQuIKa*[1] which allows for epistemic querying and IC checking. It basically implements and significantly improves the algorithm devised in our early works [**?**,**?**]. For the ease of use, we have also implemented a Protégé plugin that allows a convenient deployment of epistemic querying during the ontology design

[1] Epistemic Querying Interface Karlsruhe

Table 1. Syntax and semantics of role and concept constructors in $\mathcal{SROIQ}$.Thereby a denotes an individual name, R an arbitrary role name and S a simple role name. C and D denote concept expressions.

Name	Syntax	Semantics
inverse role	R^-	$\{(x,y) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (y,x) \in R^\mathcal{I}\}$
universal role	U	$\Delta^\mathcal{I} \times \Delta^\mathcal{I}$
top	$\top$	$\Delta^\mathcal{I}$
bottom	$\bot$	$\emptyset$
negation	$\neg C$	$\Delta^\mathcal{I} \setminus C^\mathcal{I}$
conjunction	$C \sqcap D$	$C^\mathcal{I} \cap D^\mathcal{I}$
disjunction	$C \sqcup D$	$C^\mathcal{I} \cup D^\mathcal{I}$
nominals	$\{a_1, \ldots, a_n\}$	$\{a_1^\mathcal{I}, \ldots, a_n^\mathcal{I}\}$
univ. restriction	$\forall R.C$	$\{x \mid \forall y.(x,y) \in R^\mathcal{I} \rightarrow y \in C^\mathcal{I}\}$
exist. restriction	$\exists R.C$	$\{x \mid \exists y.(x,y) \in R^\mathcal{I} \wedge y \in C^\mathcal{I}\}$
Self concept	$\exists S.\mathsf{Self}$	$\{x \mid (x,x) \in S^\mathcal{I}\}$
qualified number	$\leqslant n\, S.C$	$\{x \mid \#\{y \in C^\mathcal{I} \mid (x,y) \in S^\mathcal{I}\} \leq n\}$
restriction	$\geqslant n\, S.C$	$\{x \mid \#\{y \in C^\mathcal{I} \mid (x,y) \in S^\mathcal{I}\} \geq n\}$

phase. Our paper is organized as follows: We first provide some background knowledge as preliminaries in Section **??**, where we also discuss a method for epistemic query answering. A major contribution in this work are several optimization rewrite rules in order to ensure the feasibility of EQuIKa in practice. These along with the correctness proof are the matter of Section **??**. In Section **??**, we discuss several implementation issues. We also provide results of experiments we performed and evaluate EQuIKa with and without our new optimization. Finally, we conclude our work in Section **??** and identify some lines of future work.

2 Preliminaries

We now present an overview of the description logic $\mathcal{SROIQ}$ the underlying logic of OWL 2 DL (for details see [**?**]). For the signature of $\mathcal{SROIQ}$ we have finite, disjoint sets N_I, N_C, and N_R called *individual names*, *concept names* and *role names* respectively, with N_R being partitioned into *simple* and *non-simple* roles. These atomic entities can be used to form complex ones as displayed in Table **??**.

A $\mathcal{SROIQ}$ TBox $\mathcal{T}$ is a finite set of TBox axioms as in Table **??**. Similarly, an RBox $\mathcal{R}$ and an ABox $\mathcal{A}$ are finite sets of the corresponding

Table 2. Syntax and semantics of $\mathcal{SROIQ}$ axioms

Axiom α	$\mathcal{I} \models \alpha$, if	
$R_1 \circ \cdots \circ R_n \sqsubseteq R$	$R_1^{\mathcal{I}} \circ \cdots \circ R_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$	RBox axioms
$\mathsf{Dis}(S,T)$	$S^{\mathcal{I}} \cap T^{\mathcal{I}} = \emptyset$	
$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	TBox axioms
$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$	ABox axioms
$R(a,b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$	
$a \doteq b$	$a^{\mathcal{I}} = a^{\mathcal{I}}$	
$a \not\doteq b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$	

axiom types. An ontology Σ is a tuple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $\mathcal{T}$ is a $\mathcal{SROIQ}$ TBox, $\mathcal{R}$ is a regular $\mathcal{SROIQ}$ role hierarchy[2] and $\mathcal{A}$ is a $\mathcal{SROIQ}$ ABox.

The semantics of $\mathcal{SROIQ}$ is defined via interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ composed of a non-empty set $\Delta^{\mathcal{I}}$ called the *domain of* $\mathcal{I}$ and a function $\cdot^{\mathcal{I}}$ mapping individual names to elements of $\Delta^{\mathcal{I}}$, concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. This mapping is extended to complex role and concept expressions as in Table **??** and finally used to define satisfaction of axioms (see Table **??**). We say that $\mathcal{I}$ satisfies an ontology $\Sigma = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ (or $\mathcal{I}$ is a model of Σ, written: $\mathcal{I} \models \Sigma$) if it satisfies all axioms of $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{A}$. We say that an ontology Σ *entails* an axiom α (written $\Sigma \models \alpha$) if all models of Σ are models of α.

Next we present the syntax and semantics of $\mathcal{SROIQK}$, the extension of $\mathcal{SROIQ}$ by $\mathbf{K}$, where we allow $\mathbf{K}$ to appear in front of as well as within concept expressions. Similary we allow $\mathbf{K}$ in front of role expressions. We call a $\mathcal{SROIQK}$-role an *epistemic role* if $\mathbf{K}$ occurs in it. An epistemic role is *simple* if it is of the form $\mathbf{K}S$ where S is a simple $\mathcal{SROIQ}$-role. As the traditional semantics leads to unintuitive effects when employed for expressive languages like $\mathcal{SROIQK}$, different solutions have been proposed, e.g., in [**?**,**?**]. We adopt the semantics presented in [**?**] for $\mathcal{SROIQK}$.

To this end, we present the definition of extended interpretations.

Definition 1. An *extended* $\mathcal{SROIQ}$*-interpretation* $\mathcal{I}$ is a tuple $(\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}}, \varphi_{\mathcal{I}})$ such that

- $(\Delta_{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a standard DL interpretation,
- $\varphi_{\mathcal{I}} : N_I \cup \mathbb{N} \twoheadrightarrow \Delta^{\mathcal{I}}$ is a surjective function from $N_I \cup \mathbb{N}$ onto $\Delta^{\mathcal{I}}$, such that for all $a \in N_I$ we have that $\varphi_{\mathcal{I}}(a) = a^{\mathcal{I}}$.

[2] We assume the usual regularity assumption for $\mathcal{SROIQ}$, but omit it for space reasons.

Note that the function $\varphi_{\mathcal{I}}$ returns the actual interpretation of an individual, given its (abstract) name, under the interpretation $\mathcal{I}$. We lift $\varphi_{\mathcal{I}}$ to sets of names and let $\varphi_{\mathcal{I}}^{-1}$ denote the corresponding inverse. Based on the notion of extended interpretations, we define an *extended epistemic interpretation* for $\mathcal{SROIQK}$ as a pair $(\mathcal{I}, \mathcal{W})$, where $\mathcal{I}$ is an extended $\mathcal{SROIQ}$-interpretation and $\mathcal{W}$ is a set of extended $\mathcal{SROIQ}$-interpretations. The extended interpretation function $\cdot^{\mathcal{I},\mathcal{W}}$ is then defined as follows:

$$
\begin{aligned}
a^{\mathcal{I},\mathcal{W}} &= a^{\mathcal{I}} \quad \text{for } a \in N_I \\
A^{\mathcal{I},\mathcal{W}} &= A^{\mathcal{I}} \quad \text{for } A \in N_C \\
R^{\mathcal{I},\mathcal{W}} &= R^{\mathcal{I}} \quad \text{for } R \in N_R \\
\top^{\mathcal{I},\mathcal{W}} &= \Delta^{\mathcal{I}} \quad \text{(the domain of } \mathcal{I}) \\
\bot^{\mathcal{I},\mathcal{W}} &= \emptyset \\
(C \sqcap D)^{\mathcal{I},\mathcal{W}} &= C^{\mathcal{I},\mathcal{W}} \cap D^{\mathcal{I},\mathcal{W}} \\
(C \sqcup D)^{\mathcal{I},\mathcal{W}} &= C^{\mathcal{I},\mathcal{W}} \cup D^{\mathcal{I},\mathcal{W}} \\
(\neg C)^{\mathcal{I},\mathcal{W}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I},\mathcal{W}} \\
(\forall R.C)^{\mathcal{I},\mathcal{W}} &= \{p_1 \in \Delta^{\mathcal{I}} \mid \forall p_2.(p_1, p_2) \in R^{\mathcal{I},\mathcal{W}} \rightarrow p_2 \in C^{\mathcal{I},\mathcal{W}}\} \\
(\exists R.C)^{\mathcal{I},\mathcal{W}} &= \{p_1 \in \Delta^{\mathcal{I}} \mid \exists p_2.(p_1, p_2) \in R^{\mathcal{I},\mathcal{W}} \wedge p_2 \in C^{\mathcal{I},\mathcal{W}}\} \\
(\leq nR.C)^{\mathcal{I},\mathcal{W}} &= \{d \mid \#\{e \in C^{\mathcal{I},\mathcal{W}} \mid (d, e) \in R^{\mathcal{I},\mathcal{W}}\} \leq n\} \\
(\geq nR.C)^{\mathcal{I},\mathcal{W}} &= \{d \mid \#\{e \in C^{\mathcal{I},\mathcal{W}} \mid (d, e) \in R^{\mathcal{I},\mathcal{W}}\} \geq n\} \\
(\mathbf{K}C)^{\mathcal{I},\mathcal{W}} &= \varphi_{\mathcal{I}}\left(\bigcap_{\mathcal{J} \in \mathcal{W}} \varphi_{\mathcal{J}}^{-1}\left(C^{\mathcal{J},\mathcal{W}}\right)\right) \\
(\mathbf{K}R)^{\mathcal{I},\mathcal{W}} &= \varphi_{\mathcal{I}}\left(\bigcap_{\mathcal{J} \in \mathcal{W}} \varphi_{\mathcal{J}}^{-1}\left(R^{\mathcal{J},\mathcal{W}}\right)\right)
\end{aligned}
$$

Again, for an epistemic role $(\mathbf{K}R)^-$, we set $[(\mathbf{K}R)^-]^{\mathcal{J},\mathcal{W}} := (\mathbf{K}R^-)^{\mathcal{J},\mathcal{W}}$.

The semantics of TBox, RBox and ABox axioms follows exactly that for the classical semantics. We now define the notions of modelhood. An *extended epistemic model* of an ontology $\Sigma = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ is a maximal non-empty set $\mathcal{M}$ of extended interpretations such that $(\mathcal{I}, \mathcal{M})$ satisfies $\mathcal{T}$, $\mathcal{R}$ and $\mathcal{A}$ for each $\mathcal{I} \in \mathcal{M}$. An ontology Σ is *satisfiable* (under the extended semantics) if it has an extended epistemic model. Similarly the ontology Σ *entails* an axiom α, written $\Sigma \mid\!\models \alpha$, if for every extended epistemic model $\mathcal{M}$ of Σ, we have that for every $\mathcal{I} \in \mathcal{M}$, the extended epistemic interpretation $(\mathcal{I}, \mathcal{M})$ satisfies α. Note that a $\mathbf{K}$-free ontology Σ, one without any occurrence of $\mathbf{K}$, admits a unique extended epistemic model, which is the set of all models of Σ extended by all possible surjective mappings that map individuals names and elements of $\mathbb{N}$ to the elements of their domain. We denote this model by $\mathcal{M}(\Sigma)$. A detail discussion on the extended semantics can be found in [**?**], where a method of translating epistemic concept expressions into equivalent $\mathbf{K}$-free ones is also presented. Since EQuIKa implements this method, we recall the translation function in the following. Note that the translation itself re-

quires to check entailment of (**K**-free) axioms, hence it is not strictly syntactical and it depends on the underlying knowledge base.

Definition 2. Given a $\mathcal{SROIQ}$ knowledge base Σ, we define a function $\tilde{\Phi}_\Sigma$ mapping $\mathcal{SROIQK}$ concept expressions to $\mathcal{SROIQ}$ concept expressions (where we let $\{\} = \emptyset = \bot$):

$$
\begin{aligned}
C &\mapsto C \quad \text{if } C \text{ is an atomic or one-of concept, } \top \text{ or } \bot;\\
\mathbf{K}D &\mapsto \begin{cases} \top & \text{if } \Sigma \models \tilde{\Phi}_\Sigma(D) \equiv \top \\ \{a \in N_I \mid \Sigma \models \tilde{\Phi}_\Sigma(D)(a)\} & \text{otherwise} \end{cases}\\
\exists \mathbf{K} S.\mathsf{Self} &\mapsto \begin{cases} \exists S.\mathsf{Self} & \text{if } \Sigma \models \top \sqsubseteq \exists S.\mathsf{Self} \\ \{a \in N_I \mid \Sigma \models S(a,a)\} & \text{otherwise} \end{cases}\\
C_1 \sqcap C_2 &\mapsto \tilde{\Phi}_\Sigma(C_1) \sqcap \tilde{\Phi}_\Sigma(C_2)\\
C_1 \sqcup C_2 &\mapsto \tilde{\Phi}_\Sigma(C_1) \sqcup \tilde{\Phi}_\Sigma(C_2)\\
\neg C &\mapsto \neg \tilde{\Phi}_\Sigma(C)\\
\exists R.D &\mapsto \exists R.\tilde{\Phi}_\Sigma(D) \quad \text{for non-epistemic role } R\\
\exists \mathbf{K} P.D &\mapsto \begin{cases} \bigsqcup_{a \in N_I} \{a\} \sqcap \exists P.(\{b \in N_I \mid \Sigma \models P(a,b)\} \sqcap \tilde{\Phi}_\Sigma(D)) \\ \sqcup \exists P.\big(\{b \in N_I \mid \Sigma \models \top \sqsubseteq \exists P.\{b\}\} \sqcap \tilde{\Phi}_\Sigma(D)\big) \\ \sqcup \{a \in N_I \mid \Sigma \models \top \sqsubseteq \exists P^-.\{a\}\} \sqcap \exists P.\tilde{\Phi}_\Sigma(D) \\ \sqcup \begin{cases} \tilde{\Phi}_\Sigma(D) & \text{if } \Sigma \models \top \sqsubseteq \exists P.\mathsf{Self} \\ \bot & \text{otherwise} \end{cases} \end{cases}\\
\forall R.D &\mapsto \forall R.\tilde{\Phi}_\Sigma(D) \quad \text{for non-epistemic role } R;\\
\forall \mathbf{K} P.D &\mapsto \neg \tilde{\Phi}_\Sigma(\exists \mathbf{K} P.\neg D)\\
\geqslant n S.D &\mapsto \geqslant n S.\tilde{\Phi}_\Sigma(D) \quad \text{for non-epistemic role } S;\\
\geqslant n \mathbf{K} S.D &\mapsto \begin{cases} \bigsqcup_{a \in N_I} \{a\} \sqcap \geqslant n S.(\{b \in N_I \mid \Sigma \models S(a,b)\} \sqcap \tilde{\Phi}_\Sigma(D)) \\ \sqcup \{a \in N_I \mid \Sigma \models \top \sqsubseteq \exists S^-.\{a\}\} \sqcap \geqslant n S.\tilde{\Phi}_\Sigma(D) \\ \sqcup \geqslant n S.(\{b \in N_I \mid \Sigma \models \top \sqsubseteq \exists S.\{b\}\} \sqcap \tilde{\Phi}_\Sigma(D)) \\ \sqcup \begin{cases} \geqslant (n-1) S.\big(\{b \in N_I \mid \Sigma \models \top \sqsubseteq \exists S.\{b\}\} \sqcap \tilde{\Phi}_\Sigma(D)\big) \sqcap \\ \tilde{\Phi}_\Sigma(D) \sqcap \neg\{a \mid a \in N_I\} & \text{if } \Sigma \models \top \sqsubseteq \exists S.\mathsf{Self} \\ \bot & \text{otherwise} \end{cases} \end{cases}\\
\leqslant n S.D &\mapsto \leqslant n S.\tilde{\Phi}_\Sigma(D) \quad \text{for non-epistemic role } S;\\
\leqslant n \mathbf{K} S.D &\mapsto \neg \tilde{\Phi}_\Sigma(\geqslant (n+1) \mathbf{K} S.D)\\
\Xi \mathbf{K} R.D &\mapsto \Xi R.\tilde{\Phi}_\Sigma(D) \quad \text{for } \Xi \in \{\forall, \exists, \geqslant n, \leqslant n\} \text{ and } \Sigma \models R \equiv U
\end{aligned}
$$

EQuIKa implements this translation function. Note that a naive implementation of $\tilde{\Phi}_\Sigma$ might not be feasible in practice. In the next section, we discuss several optimization techniques that we devised in order to ensure the run-time feasibility of EQuIKa.

3 Optimization

A naive implementation of the translation function presented in Definition **??** does not need to be feasible in practice, in particular when dealing with ontologies containing relatively large number of individuals. We came up with several optimization rules. As we will see in the next section, these rules speed up the computation time of EQuIKa in retrieving instances of an epistemic concept. Basically, every rule checks the structure of a given epistemic concept and reduces either the number of $\mathbf{K}$'s occurring in the concept or the number of calls to the core reasoner during the translation of the concept such that the correctness of the answers is preserved. For the proof of the correctness of the answers via translation based upon the optimization rules, we show that for an epistemic concept C, the extensions of the translation of C coincide when the translation is done with and without the optimization rules.

In the following we discuss each of these rules and provide the proof of their correctness, where by $\hat{\Phi}_{\Sigma}$ we mean the translation function based on the optimization rules.

- ***Rule 1 (Nominals):*** For individual names $a_1, \ldots, a_n$ we have

$$\hat{\Phi}_{\Sigma}(\mathbf{K}\{a_1, \ldots, a_n\}) \mapsto \{a_1, \ldots, a_n\}$$

Proof. For the left to right direction, let $x \in \mathbf{K}\{a_1, \ldots, a_n\}^{\mathcal{I},\mathcal{M}(\Sigma)}$. By semantics, therefore,

$$x \in \varphi_{\mathcal{I}}\big(\bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} \varphi_{\mathcal{J}}^{-1}(\{a_1, \ldots, a_n\}^{\mathcal{J}})\big)$$

In other words, $x \in \varphi_{\mathcal{I}}(\varphi_{\mathcal{J}}^{-1}(\{a_1, \ldots, a_n\}^{\mathcal{J}}))$ for each $\mathcal{J} \in \mathcal{M}(\Sigma)$. In particular, $x \in \varphi_{\mathcal{I}}(\varphi_{\mathcal{I}}^{-1}(\{a_1, \ldots, a_n\}^{\mathcal{I}}))$. Since $\varphi_{\mathcal{I}}$ is a surjective mapping, we get that $x \in \{a_1, \ldots, a_n\}^{\mathcal{I}} = \{a_1, \ldots, a_n\}^{\mathcal{I},\mathcal{M}(\Sigma)}$.
For the right to left direction, let $x \in \{a_1, \ldots, a_n\}^{\mathcal{I},\mathcal{M}(\Sigma)}$ and suppose that $x \notin \mathbf{K}\{a_1, \ldots, a_n\}^{\mathcal{I},\mathcal{M}(\Sigma)}$. Note that by definition of $\varphi_{\mathcal{J}}$, we have that $a_i \in {\varphi_{\mathcal{J}}}^{-1}(a_i^{\mathcal{J}})$ for each $\mathcal{J} \in \mathcal{M}(\Sigma)$ and $1 \leq i \leq n$. It means that $\{a_1, \ldots, a_n\} \subseteq {\varphi_{\mathcal{J}}}^{-1}(\{a_1, \ldots, a_n\}^{\mathcal{J}})$ for each $\mathcal{J} \in \mathcal{M}(\Sigma)$ i.e.,

$$\{a_1, \ldots, a_n\} \subseteq \bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} {\varphi_{\mathcal{J}}}^{-1}(\{a_1, \ldots, a_n\}^{\mathcal{J}})$$

Applying $\varphi_{\mathcal{I}}$ we get

$$\varphi_{\mathcal{I}}(\{a_1, \ldots, a_n\}) \subseteq \varphi_{\mathcal{I}}\big(\bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} {\varphi_{\mathcal{J}}}^{-1}(\{a_1, \ldots, a_n\}^{\mathcal{J}})\big)$$

In other words,

$$\varphi_{\mathcal{I}}(\{a_1,\ldots,a_n\}) \subseteq \mathbf{K}\{a_1,\ldots,a_n\}^{\mathcal{I},\mathcal{M}(\Sigma)}$$

By assumption, since $x \notin \mathbf{K}\{a_1,\ldots,a_n\}^{\mathcal{I},\mathcal{M}(\Sigma)}$, consequently we get that $x \notin \varphi_{\mathcal{I}}(\{a_1,\ldots,a_n\}) = \{\varphi_{\mathcal{I}}(a_1),\ldots,\varphi_{\mathcal{I}}(a_n)\}$. By definition of $\varphi_{\mathcal{I}}$ we have that $\varphi_{\mathcal{I}}(a) = a^{\mathcal{I}}$ for each $a \in N_I$. Therefore we get that $x \notin \{a_1{}^{\mathcal{I}},\ldots,a_n{}^{\mathcal{I}}\} = \{a_1,\ldots,a_n\}^{\mathcal{I},\mathcal{M}(\Sigma)}$, which is a contradiction. Hence, $x \in \mathbf{K}\{a_1,\ldots,a_n\}^{\mathcal{I},\mathcal{M}(\Sigma)}$ must hold.

– ***Rule 2 (Conjunction):*** Let $C_1,\ldots,C_n$ be concepts where each C_i is either an epistemic concept of the form $\mathbf{K}D$ for some concept D or a one-of concept for $1 \leq i \leq n$. By C_i' we denote the concept obtained from C_i by dropping $\mathbf{K}$ or $C_i' = C_i$ otherwise. Then,

$$\hat{\Phi}_{\Sigma}(C_1\sqcap,\ldots,\sqcap C_n) \mapsto \mathbf{K}(C_1'\sqcap,\ldots,\sqcap C_n')$$

Proof. For a one-of concept $\{a_1,\ldots,a_k\}$, it follows from the proof of Rule 1 that $\{a_1,\ldots,a_k\}$ is equivalent to $\mathbf{K}\{a_1,\ldots,a_k\}$. Hence, we assume that each concept in $C_1 \sqcap \cdots \sqcap C_n$ is of the form $\mathbf{K}D$ for some concept D. Now

$x \in [\mathbf{K}D_1 \sqcap \ldots \mathbf{K}D_n]^{\mathcal{I},\mathcal{M}(\Sigma)}$
$\Leftrightarrow x \in (\mathbf{K}D_1^{\mathcal{I},\mathcal{M}(\Sigma)} \cap \cdots \cap \mathbf{K}D_n^{\mathcal{I},\mathcal{M}(\Sigma)})$
$\Leftrightarrow x \in \varphi_{\mathcal{I}}(\bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} \varphi_{\mathcal{J}}^{-1}(D_1^{\mathcal{J}})) \cap \cdots \cap \varphi_{\mathcal{I}}(\bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} \varphi_{\mathcal{J}}^{-1}(D_n^{\mathcal{J}}))$
$\Leftrightarrow x \in \varphi_{\mathcal{I}}\Big(\bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} \big(\varphi_{\mathcal{J}}^{-1}(D_1^{\mathcal{J}}) \cap \cdots \cap \varphi_{\mathcal{J}}^{-1}(D_n^{\mathcal{J}})\big)\Big)$
$\Leftrightarrow x \in \varphi_{\mathcal{I}}\Big(\bigcap_{\mathcal{J}\in\mathcal{M}(\Sigma)} \varphi_{\mathcal{J}}^{-1}\big((D_1 \sqcap \cdots \sqcap D_n)^{\mathcal{J}}\big)\Big)$
$\Leftrightarrow x \in [\mathbf{K}(D_1 \sqcap \cdots \sqcap D_n)]^{\mathcal{I},\mathcal{M}(\Sigma)}$ □

– ***Rule 3 (Existential Quantification):***
By Definition **??**, for a concept $\exists\mathbf{K}R.D$ we get that

$$\exists\mathbf{K}P.D \mapsto \begin{cases} \bigsqcup_{a\in N_I}\{a\} \sqcap \exists P.(\{b \in N_I \mid \Sigma \models P(a,b) \sqcap \tilde{\Phi}_{\Sigma}(D)\}) \\ \sqcup\, \exists P.\big(\{b \in N_I \mid \Sigma \models \top \sqsubseteq \exists P.\{b\}\} \sqcap \tilde{\Phi}_{\Sigma}(D)\big) \\ \sqcup\, \{a \in N_I \mid \Sigma \models \top \sqsubseteq \exists P^{-}\{a\}\} \sqcap \exists P.\tilde{\Phi}_{\Sigma}(D) \\ \sqcup \begin{cases} \tilde{\Phi}_{\Sigma}(D) & \text{if } \Sigma \models \top \sqsubseteq \exists P.\mathsf{Self} \\ \bot & \text{otherwise} \end{cases} \end{cases}$$

Suppose $\mathsf{Inst}(D)$ returns the set of instances of a concept D and let $\mathsf{OUT}_P = \mathsf{Inst}(\exists P.\tilde{\Phi}(D))$ and $\mathsf{IN}_P = \mathsf{Inst}(\exists P^{-}.\top)$. Now Rule 3 is as follows

$$\hat{\Phi}_{\Sigma}(\exists\mathbf{K}P.D) \mapsto \begin{cases} \bigsqcup_{a\in \mathsf{OUT}_P}\{a\} \sqcap \exists P.(\{b \in \mathsf{IN}_P \mid \Sigma \models P(a,b) \sqcap \tilde{\Phi}_{\Sigma}(D)\}) \\ \sqcup\, \exists P.\big(\{b \in \mathsf{IN}_P \mid \Sigma \models \top \sqsubseteq \exists P.\{b\}\} \sqcap \tilde{\Phi}_{\Sigma}(D)\big) \\ \sqcup\, \{a \in \mathsf{OUT}_P \mid \Sigma \models \top \sqsubseteq \exists P^{-}\{a\}\} \sqcap \exists P.\tilde{\Phi}_{\Sigma}(D) \\ \sqcup \begin{cases} \tilde{\Phi}_{\Sigma}(D) & \text{if } \Sigma \models \top \sqsubseteq \exists P.\mathsf{Self} \\ \bot & \text{otherwise} \end{cases} \end{cases}$$

Note that without the optimization, translating a concept of the form $\exists \mathbf{K} P.D$ requires at least $|N_I| \times |N_I| + 2|N_I| + 1 +$ (No. of calls needed to translate D) calls to the core reasoner. With the optimization Rule 3, such a translation reduces the number of calls when $|\mathsf{Inst}(\exists P.\tilde{\Phi}(D))| < |N_I|$ or $|\mathsf{Inst}(\exists P^{-}.\top)| < N_I$.

Proof. We introduce the following abbreviations,

- $C_1 = \bigsqcup_{a \in N_I} \{a\} \sqcap \exists P.(\{b \in N_I \mid \Sigma \models P(a,b) \sqcap \tilde{\Phi}_\Sigma(D)\})$
- $C_1' = \bigsqcup_{a \in \mathsf{OUT}_P} \{a\} \sqcap \exists P.(\{b \in \mathsf{IN}_P \mid \Sigma \models P(a,b) \sqcap \tilde{\Phi}_\Sigma(D)\})$
- $C_2 = \exists P.\big(\{b \in N_I \mid \Sigma \models \top \sqsubseteq \exists P.\{b\}\} \sqcap \tilde{\Phi}_\Sigma(D)\big)$
- $C_2' = \exists P.\big(\{b \in \mathsf{IN}_P \mid \Sigma \models \top \sqsubseteq \exists P.\{b\}\} \sqcap \tilde{\Phi}_\Sigma(D)\big)$
- $C_3 = \{a \in N_I \mid \Sigma \models \top \sqsubseteq \exists P^{-}\{a\}\} \sqcap \exists P.\tilde{\Phi}_\Sigma(D)$
- $C_3' = \{a \in \mathsf{OUT}_P \mid \Sigma \models \top \sqsubseteq \exists P^{-}\{a\}\} \sqcap \exists P.\tilde{\Phi}_\Sigma(D)$

We first show that for an extended interpretation $\mathcal{I} \in \mathcal{M}(\Sigma)$ and $x \in \Delta^{\mathcal{I}}$, $x \in C_1^{\mathcal{I},\mathcal{M}(\Sigma)}$ iff $x \in C_1'^{\mathcal{I},\mathcal{M}(\Sigma)}$. For this note that if $b' \in \mathsf{Inst}(\{b \in N_I | \Sigma \models P(a',b)\})$ for individuals a' and b' then it immediately follows that $b' \in \mathsf{Inst}(\{b \in \mathsf{IN}_P | \Sigma \models (a',b)\})$ and vice versa. Now for $x \in \Delta^{\mathcal{I}}$, $x \in C_1^{\mathcal{I},\mathcal{M}(\Sigma)}$ if and only if there is an individual $a' \in N_I$ such that $x = a'^{\mathcal{I},\mathcal{M}(\Sigma)}$ and $x \in \exists P.(\{b \in N_I \mid \Sigma \models P(a',b) \sqcap \tilde{\Phi}_\Sigma(D)\})^{\mathcal{I},\mathcal{M}(\Sigma)}$. This is the case if and only if $x \in \exists P.(\{b \in \mathsf{IN}_P \mid \Sigma \models P(a',b) \sqcap \tilde{\Phi}_\Sigma(D)\})^{\mathcal{I},\mathcal{M}(\Sigma)}$ which is equivalent to $x \in C_1'^{\mathcal{I},\mathcal{M}(\Sigma)}$.

In the same way, we can prove that $x \in C_2^{\mathcal{I},\mathcal{M}(\Sigma)}$ if and only if $x \in C_2'^{\mathcal{I},\mathcal{M}(\Sigma)}$. Similarly, $x \in C_3^{\mathcal{I},\mathcal{M}(\Sigma)}$ if and only if $x \in C_3'^{\mathcal{I},\mathfrak{M}(\Sigma)}$. Consequently, this establishes the proof of the correctness of Rule 3. □

– ***Rule 4 (Number Restriction):***
Similar to Rule 3, $\hat{\Phi}_\Sigma$ maps the concept $\geq n\mathbf{K}S.D$ to

$$\begin{cases} \bigsqcup_{a \in \mathsf{OUT}_{P,n}} \{a\} \sqcap \geqslant nS.(\{b \in \mathsf{IN}_{P,n} \mid \Sigma \models S(a,b) \sqcap \tilde{\Phi}_\Sigma(D)\}) \\ \sqcup \{a \in \mathsf{OUT}_{P,n} \mid \Sigma \models \top \sqsubseteq \exists S^{-}.\{a\}\} \sqcap \geqslant nS.\tilde{\Phi}_\Sigma(D) \\ \sqcup \geqslant nS.(\{b \in \mathsf{IN}_{P,n} \mid \Sigma \models \top \sqsubseteq \exists S.\{b\}\} \sqcap \tilde{\Phi}_\Sigma(D)) \\ \sqcup \begin{cases} \geqslant (n-1)S.\big(\{b \in \mathsf{IN}_{P,n-1} \mid \Sigma \models \top \sqsubseteq \exists S.\{b\}\} \sqcap \tilde{\Phi}_\Sigma(D)\big) \sqcap \\ \tilde{\Phi}_\Sigma(D) \sqcap \neg\{a \mid a \in N_I\} \qquad \text{if } \Sigma \models \top \sqsubseteq \exists S.\mathsf{Self} \\ \bot \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise} \end{cases} \end{cases}$$

where $\mathsf{OUT}_{P,n} = \geq nP.\tilde{\Phi}(D)$ and $\mathsf{IN}_{P,n} = \geq n.P^{-}.\top$.

Proof. Similar to Rule 3. □

These rules suffice in the sense that for most of the remaining constructs, we use their dual which correspond to one of the rules discussed above.

After showing that the rules preserve the semantics, we need to show that they indeed improve the run time of EQuIKa. For this we performed several experiments, which are discussed in Section **??**.

4 Implementation

The *EQuIKa* system is implemented on top of the OWL-API.[3] It can be used as an API as well as within Protégé using an epistemic query tab. 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* ($\neg$) 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.
- We implemented the translation function in a recursive fashion. For this, we implemented a visitor pattern by extending the OWLClassExpressionVisitor class in order to handle the epistemic operator.
- In order to support epistemic querying within the Protégé editor, we implemented an additional tab based on the DL Query tab. Figure **??** shows a snapshot of epistemic querying in Protégé.

The class diagram for EQuIKa is given in Figure **??**. Note that 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 translates an epistemic concept into a **K**-free one in a recursive fashion using the class Translator that implements the OWLClassExpressionVisitor. Further, since Protégé can utilize any reasoner that implements the OWLReasoner interface, this enables Protégé to use EQuIkaReasoner to answer epistemic queries. Last but not least, EQuIKa has been shared on googlecode for testing purposes.[4] The plugin

[3] http://owlapi.sourceforge.net/
[4] http://code.google.com/p/epistemicdl/

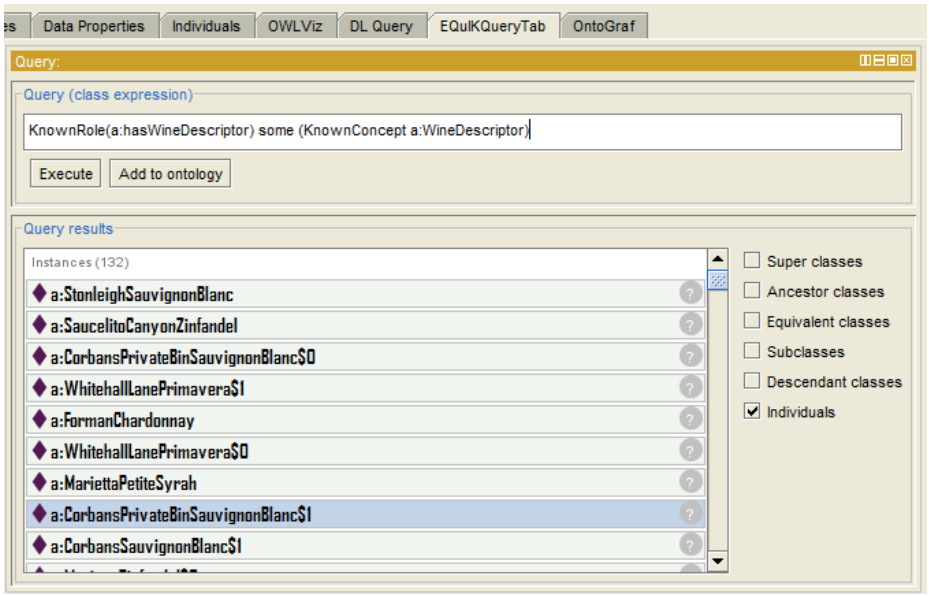

Fig. 1. Epistemic Querying in Protégé

is provided as jar file[5] that can be installed via the Protégé 4.1 plugin folder.

5 Evaluation

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

- We used IBM Thinkpad T60 dual core, 2 GHz each core, with Windows 7 (32-bit) as the operating system. A total of 2 GB memory was available.
- For benchmark tests, we used a populated version of the Wine ontology[6], that contains 483 individuals and uses most of the OWL 2 DL constructs. These ontologies can be downloaded along with EQuIKa.
- To evaluate the performance of EQuIKa, we constructed several epistemic concepts and translated them into **K**-free ones. These concepts are given in Table **??** where $r_1, \ldots, r_{108}$ are individuals representing wine regions in the ontologies.

[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

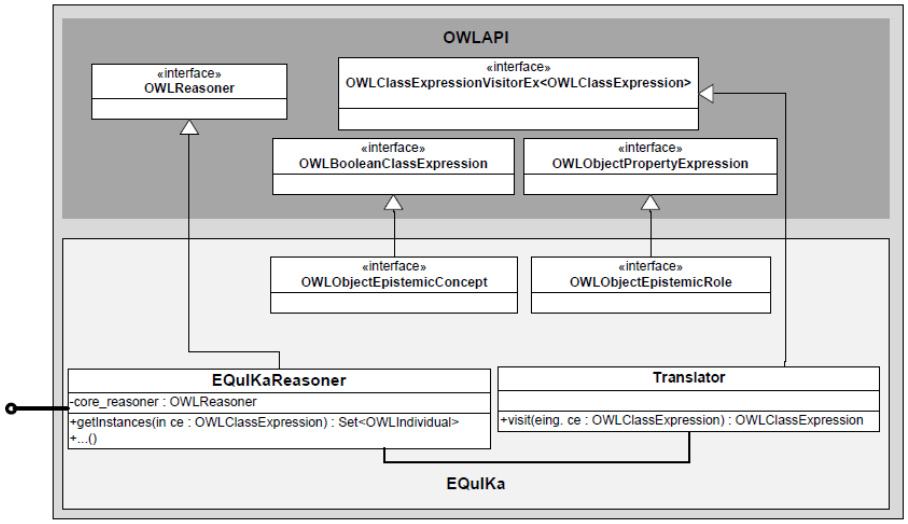

Fig. 2. The *EQuIKa*-system extending the OWL-API

Table 3. Concepts used for instance retrieval experiments.

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

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 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 **??**) to **K**-free equivalent ones and the instance retrieval time of the translated concept. In the first series of experiments, we evaluated the benefit of the optimization rules introduced in Section **??**. We implemented two versions of EQuIKa; a naive one called EQuIKa-N implementing the translation function of Definition **??** as is and an optimized one called EQuIKa-O where the optimization rules were used. The corresponding results are

Table 4. Results of instance retrieval experiments.

Concept	EQuIKa-N			EQuIKa-O		
	$\mathsf{T}_{\mathsf{trans}}$	$\mathsf{T}_{\mathsf{inst}}$	$\#_{\mathsf{inst}}$	$\mathsf{T}_{\mathsf{trans}}$	$\mathsf{T}_{\mathsf{inst}}$	$\#_{\mathsf{inst}}$
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

shown in Table **??** where $\mathsf{T}_{\mathsf{trans}}$, $\mathsf{T}_{\mathsf{inst}}$ and $\#_{\mathsf{inst}}$ represent the translation time, instance retrieval time and the number of instances respectively. A time comparison in seconds between both versions of EQuIKa is presented in Figure **??**. For each concept, the first (brown) bar represents the total time taken by EQuIKa-N and the second (blue) bar represents the total time taken by EQuIKa-O. In both bars, the dark-shaded part represented the time taken for the translation and the unshaded part represents the instance retrieval time by the corresponding EQuIKa version. It can be seen in the chart that the optimized EQuIKa is one or several orders of magnitude faster than the non-optimized one. In particular for concept $EC6$, EQuIKa-N didn't responded for almost an hour and we stopped it, whereas EQuIKa-O translated $EC6$ and retrieved its instances in few seconds. Hence, beside being correct, the optimized rules 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-O in general to provide an impression of how the cost of epistemic querying relates to standard reasoning tasks. For this purpose, we consider non-epistemic concepts $C1, \ldots, C_6$ where each C_i is obtained by dropping K in EC_i for $1 \leq i \leq 6$. Note that an epistemic concept EC_i and the corresponding C_i are semantically different concepts. Table **??** shows the results of our experiments. It can be seen that even when comparing to the K-free counter part of the epistemic concepts, the computation time of EQuIKa-O is roughly in the same order of magnitude.

6 Conclusion

In this paper, we have motivated the importance of epistemic querying of OWL ontologies for ontology introspection and integrity constraint

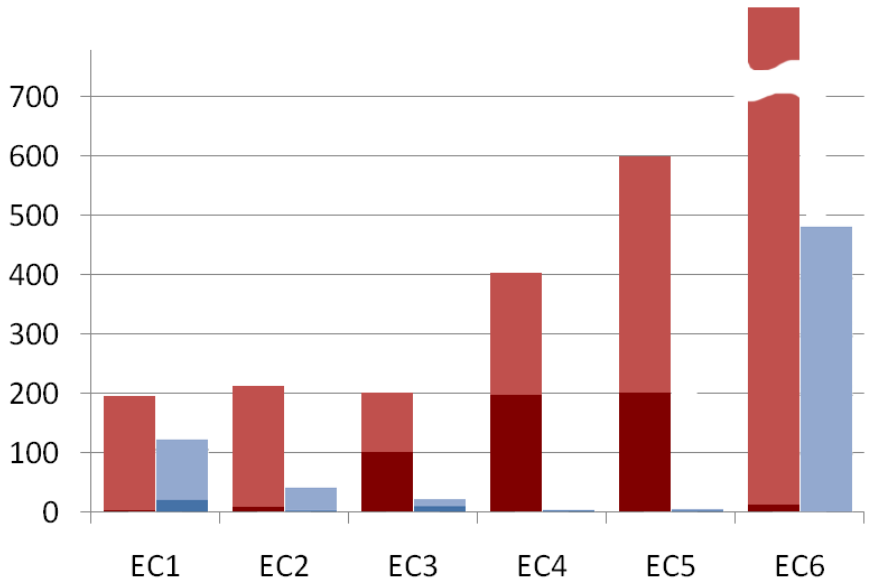

Fig. 3. EQuIKa-N vs. EQuIKa-O

checking. We have presented a system, called EQuIKa, implemented in conformance with the common OWL interfaces such that any off-the-shelf reasoner can be used as its backbone. To support convenient deployment of our tool in the course of the ontology development process, our system also features a user front-end realized as a plugin for the Protégé ontology editor.

EQuIKa is based on a reduction of epistemic queries to standard reasoning. In order to assure its practical feasibility, we have presented, proven and implemented several optimization rules leading to a speed-up of EQuIKa by one to several orders of magnitude. We performed

Table 5. Evaluation epistemic vs. standard instance retrieval

Concept	$\mathsf{t}_{(C_i)}$	$\lvert C_i \rvert$	Concept	$\mathsf{t}_{\mathsf{T}(EC_i)}$	$\mathsf{t}_{(EC_i)}$	$\lvert EC_i \rvert$
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

several experiments checking check computation time of EQuIKa and also evaluated EQuIKa against the standard reasoning task of instance retrieval. We found that EQuIKa performs in the same scale as the standard counter part, witnessing that epistemic querying can be efficiently realized in practice.

Avenues for future research are manifold: we will carry out a more extensive evaluation of our system with data stemming from ontology design scenarios from industry project's use cases. These experiments will provide a clearer view on which – already implemented or still to be defined – optimizations and heuristics in EQuIKa pay off in practice. On the theoretical side, we want to investigate the practical benefits of epistemic constructors as part of the ontology modeling language and try to extent our framework to this case. This is clearly a non-trivial task, since favorable model-theoretic properties get lost.

References

1. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W.: Adding epistemic operators to concept languages. In: Bernhard Nebel, Charles Rich, W.R.S. (ed.) Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). pp. 342–353. Morgan Kaufmann (1992)
2. Donini, F.M., Lenzerini, M., Nardi, D., Schaerf, A., Nutt, W.: An epistemic operator for description logics. Artificial Intelligence 100(1-2), 225–274 (1998)
3. Donini, F.M., Nardi, D., Rosati, R.: Autoepistemic description logics. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97). pp. 136–141. Morgan Kaufmann (1997)
4. Grimm, S., Motik, B., Preist, C.: Matching semantic service descriptions with local closed-world reasoning. In: ESWC. pp. 575–589 (2006)
5. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06). pp. 57–67. AAAI Press (2006)
6. Levesque, H.J.: Foundations of a functional approach to knowledge representation. Artificial Intelligence 23(2), 155–212 (1984)
7. Mehdi, A., Rudolph, S.: Revisiting semantics for epistemic extensions of description logics. In: Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI-11). AAAI Press (2011)
8. Mehdi, A., Rudolph, S.: Revisiting semantics for epistemic extensions of description logics. Technical Report 3015, Institute AIFB, Karlsruhe Institute of Technology (2011), available at http://www.aifb.kit.edu/web/Techreport3015/en
9. Mehdi, A., Rudolph, S., Grimm, S.: Epistemic querying of owl knowledge bases. In: Antoniou, G., Grobelnik, M., Simperl, E.P.B., Parsia, B., Plexousakis, D., Leenheer, P.D., Pan, J.Z. (eds.) Proceedings of the 8th Extended Semantic Web Conference (ESWC11). pp. 397–409 (2011)
10. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)

11. OWL Working Group, W.: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (2009), available at http://www.w3.org/TR/owl2-overview/
12. Reiter, R.: What should a database know? Journal of Logic Programming 14(1-2), 127–153 (1992)