Secure OWL Query*

Baowen Xu[1,2], Yanhui Li[1,2], Jianjiang Lu[1,2,3], and Dazhou Kang[1,2]

[1] Department of Computer Science and Engineering,
Southeast University, Nanjing 210096, P.R. China
[2] Jiangsu Institute of Software Quality, Nanjing 210096, P.R. China
[3] Institute of Command Automation,
PLA University of Science and Technology, Nanjing 210007, P.R. China
bwxu@seu.edu.cn

Abstract. With the development of the Semantic Web, the issue on Semantic Web security has received a considerable attention. OWL security is a burgeoning and challengeable sub-area of Semantic Web security. We propose a novel approach about OWL security, especially about inference control in OWL retrieval. We define OWL inference control rules (ICRs) to present the aim of inference control. To achieve ICRs, our approach introduces a novel concept "semantic related view" (SRView) w.r.t an OWL knowledge base, which describes semantic relations in it. We present a construction process of SRViews from OWL knowledge bases and define pruning operations to rewrite them. Based on pruned SRViews, a query framework to achieve inference control is proposed, followed by analysis of correctness and complexity.

1 Introduction

The Semantic Web is an extension of the current Web, which supports machine-understandable semantics to enable intelligent information management. With the increasing efficiency of utilizing data in the Semantic Web, there is a considerable attention for security issues. Thuraisingham pointed out security standards for the Semantic Web, in which security of the whole Semantic Web was divided into security of its components: XML, RDF and OWL security etc. [1].

Various research efforts have been done on XML security, especially about access control in the context of XML. Fan et al. introduced a novel approach that the security restrictions were annotated on the schema structure (DTDs) and queries were corresponding rewritten and optimized [2]. Compared with XML security, RDF security just begins. Reddivari et al. proposed a simple RDF access policy framework (RAP), which contained a policy based access control model to achieve control over various actions possible on an RDF store [3]. Largely dissimilar to XML and RDF cases, OWL security contains a great part of inference control that prevents users from infer-

* This work was supported in part by the NSFC (60373066, 60425206 and 90412003), National Grand Fundamental Research 973 Program of China (2002CB312000), National Research Foundation for the Doctoral Program of Higher Education of China (20020286004), Excellent Ph.D. Thesis Fund of Southeast University, Advanced Armament Research Project (51406020105JB8103) and Advanced Research Fund of Southeast University (XJ0609233).

ring blocked or unallowable information. How to present inference control restrictions and achieve them is still an open problem, and to the best of our knowledge, no published paper is known for it.

In this paper, we will give a novel approach about OWL inference control. Firstly we give a brief introduction to OWL knowledge base (KB) in section 2. Then we define OWL inference control rules (ICRs) to formally present OWL inference control restrictions in section 3. In section 4, a new term "semantic related view" (SRView) is introduced to describe whole semantic relation in an OWL KB. We also give a construction process and pruning operations of SRViews. With pruned SRViews, a query framework to achieve inference control is proposed in section 5, followed by conclusion and future work in section 6.

2 A Brief Introduction to OWL Knowledge Base

The proposed OWL recommendation consists of three languages with increasing expressive power: OWL Lite, OWL DL and OWL Full. For inference in OWL Full is undecidable [4], we focus on inference control in OWL DL. In the following, we use "OWL" instead of "OWL DL". We adopt description logic form to compactly express OWL KB, for OWL DL is description logic SHOIN(D) with RDF syntax.

Definition 1. Let R_A, R_D, C_A, C_D, I_A and I_D be pairwise disjoint sets of atomic abstract roles, atomic concrete roles, atomic concepts, data types, individuals and data values. Let $R \in R_A$, a SHOIN(D) abstract role is either an atomic abstract role R or its inverse role R^-. The set of SHOIN(D) role is defined as $\{ R^- \mid R \in R_A \} \cup R_A \cup R_D$. And SHOIN(D) concept constructors are given in Table 1.

Table 1. Syntax and semantics of concept constructors in SHOIN(D)

Syntax	Semantics	Syntax	Semantics
$A \in C_A$	$A^I \subseteq \Delta^I$	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$
$D \in C_D$	$D^I = D^{\mathrm{D}} \subseteq \Delta^{\mathrm{D}}$	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
$R \in R_A$	$R^I \subseteq \Delta^I \times \Delta^I$	$\neg C$	$(\neg C)^I = \Delta^I \setminus C^I$
$U \in R_D$	$U^I \subseteq \Delta^I \times \Delta^{\mathrm{D}}$	$\{w_1, \cdots, w_n\}$	$\{w_1, \cdots, w_n\}^I = \{w_1^I, \cdots, w_n^I\}$
$i \in I_A$	$i^I \in \Delta^I$	$\forall P.E$	$(\forall P.E)^I = \{d \mid \forall d', (d,d') \in P^I \rightarrow y \in E^I\}$
$v \in I_D$	$v^I = v^{\mathrm{D}} \in \Delta^{\mathrm{D}}$	$\exists P.E$	$(\exists P.E)^I = \{d \mid \exists d', (d,d') \in P^I \wedge y \in E^I\}$
R^-	$(R^-)^I = \{(d,d') \mid (d',d) \in R^I\}$	$\geq n\, P$	$(\geq n P)^I = \{d \mid \#(\{d' \mid (d,d') \in P^I\}) \geq n\}$
$\top$	$\top^I = \Delta^I$	$\leq n\, P$	$(\leq n P)^I = \{d \mid \#(\{d' \mid (d,d') \in P^I\}) \leq n\}$
$\perp$	$\perp^I = \varnothing$		

In table 1, Δ^I and Δ^{D} are two nonempty sets standing for abstract and concrete domains. $\top$ and $\perp$ are considered as the top and bottom concepts. $\{ w_1, \cdots, w_n \}$ is a sub set of I_A or I_D; C_1 and C_2 are abstract concepts; P is a SHOIN(D) role; the expressions $\forall P.E$ and $\exists P.E$ satisfy that if $P \in R_D$, $E \in C_D$; if $P \in \{ R^- \mid R \in R_A \} \cup R_A$, E is an abstract concept.

Definition 2. A SHOIN(D) KB $\sum(T_\Sigma, R_\Sigma, A_\Sigma)$ consists of three finite axiom boxes: TBox T_Σ, RBox R_Σ and ABox A_Σ, whose axioms' syntax and semantics are given in table 2.

Table 2. Axioms in a SHOIN(D) KB

	Syntax	Semantics
TBox axiom	$C_1 \sqsubseteq C_2$	$C_1^I \subseteq C_2^I$
RBox axiom	$R_1 \sqsubseteq R_2$	$R_1^I \subseteq R_2^I$
	$\text{Trans}(R_1)$	$R_1^I = (R_1^I)^+$
	$U_1 \sqsubseteq U_2$	$U_1^I \subseteq U_2^I$
ABox axiom	$i : C$	$i^I \in C^I$
	$i_1 = i_2$	$i_1^I = i_2^I$
	$i_1 \neq i_2$	$i_1^I \neq i_2^I$

An interpretation $I(\Delta^I, \cdot^I)$ satisfies an axiom if it satisfies corresponding semantics restriction given in table 2. *I* satisfies a TBox (RBox and ABox respectively), if *I* satisfies any axiom in it. *I* satisfies a SHOIN(D) KB $\sum$, if *I* satisfies its TBox, ABox and RBox. Such interpretation *I* is called a model of KB $\sum$.

3 OWL Inference Control Rule

Before discussing OWL ICR, we will give a short introduction of OWL query mechanism. Answers of OWL queries are a collection of logical entailed axioms of the OWL KB. Using semantics discussed in section 2, we define "logical entail": for any axiom α, a KB $\sum$ logical entails α, denoted as $\sum \models \alpha$, if for any model $I(\Delta^I, \cdot^I)$ of $\sum$, *I* satisfies α. With "rolling-up" technique [5], OWL query problems can be simplified to be instance retrievals *Retrieval*(*C*) of singleton concepts *C*:

$$Retrieval(C) = \{ i \mid \Sigma \models i : C \}$$

Now we give the definition of OWL ICR.

Definition 3. An OWL ICR is a triple (subject, axioms set, sign), where subject is an identification of user, axioms set is a given set of logical entailed ABox axioms of the OWL KB $\sum$, and sign$\in$ {+(positive), -(negative), ?(unknown)}.

For a positive OWL ICR (*s*, $\{\alpha_1,...,\alpha_n\}$, +), it means when the query presenter *s* give a query: *Retrieval*(*C*), if $\alpha_i = i : C$ and $i \in Retrieval(C)$, *i* must be returned to *s*. Negative or unknown ICRs have similar meanings with replacing "must" with "must not" or "maybe". For the set of logical entailed axioms of $\sum$ may be infinite, it is impossible to sign all axioms with +, - or ?. Therefore we set ? as the default value.

Definition 4. For a given OWL ICR set S_R, let $S(s,+) = \bigcup Sa$ for any $(s, Sa, +) \in S_R$ and $S(s,-) = \bigcup Sa$ for any $(s, Sa, -) \in S_R$. S_R is consistent w.r.t an OWL KB $\sum$, if for any subject *s*, there is a OWL KB $\sum^*$, where $\sum \models \sum^*$ and $\sum^* \models Sa(s,+)$ and for any axioms α in $Sa(s,-)$, $\sum^* \not\models \alpha$. We call such KB $\sum^*$ as a suitable sub-KB of $\sum$.

For any consistent OWL ICR set S_R w.r.t Σ, perfect inference control aims to find the most general suitable sub-KB Σ^* of Σ for a given subject *s*, where no other suitable sub-KB Σ' satisfying $\Sigma' \models \Sigma^*$. In OWL retrieval, Σ^* will replace Σ as a special view for the subject *s*. However, this problem is as hard as finding most general consistent sub-ontology in an inconsistent big ontology [6], and there is no efficient method to deal with such problem. In this paper, we give a primitive approach to find a biggish suitable sub-KB. The creation of the biggish suitable sub-KB is considered as pruning KB Σ. We use SRView as an abstraction of Σ and define pruning operations for SRView. A pruned SRView will play a role as a suitable sub-KB in OWL query.

4 Semantic Related View of OWL Knowledge Base

Before turning our attention towards SRView, we introduce some common notions. We define the set *sub*(*C*) of sub-concepts of a SHOIN(D) concept *C*:

$$sub(C) = \{C\} \cup \begin{cases} \varnothing & \text{if } C = A \mid \geq n\,P \mid \leq n\,P \mid \{w_1, \cdots, w_n\} \\ sub(C_1) & \text{if } C = \neg C_1 \\ sub(C_1) \cup sub(C_2) & \text{if } C = C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \\ sub(E) & \text{if } C = \exists P.E \mid \forall P.E \end{cases}$$

Let S_Σ be the set of concepts appearing in an OWL KB Σ, and $sub(S_\Sigma) = \bigcup sub(C)$ for any *C* in S_Σ. After talking about these notions, we present a formal structure of a SRView.

Definition 5. A SRView is a graph *G*(*V*, *E*), where *V* is a set of labels and *E* is a set of edges with connectives. Every edge can be denoted as a triple (l, con_i, l_1), where $l, l_1 \in V$, $con_i \in \{\exists P, \forall P, \neg, \sqcap, \sqcup, \geq n\,P, \leq n\,P, \sqsubseteq\}$. For any label *l* in *V*, let $E_{out}(l) = \{(l, con_i, l_1) \mid (l, con_i, l_1) \in E\}$, $E_{in}(l) = \{(l_1, con_i, l) \mid (l_1, con_i, l) \in E\}$.

Additionally, we define a label function L: $sub(S_\Sigma) \to V$ and it satisfies that for any two concepts (or data types) C_1 and C_2 in $sub(S_\Sigma)$, $L(C_1) \neq L(C_2)$.

Now we will give the detailed creation of a SRView from an OWL KB Σ. At the beginning, we initialize $V = \{L(\top), L(\bot)\}$, $E = \varnothing$, and following creation consists of three ordinal steps:

Developing step: for any concept *C* in S_Σ, we add a concept tree *Tree*(*C*) in *G*. The addition process *AddTree*(*C*) is described in Figure 1. Obviously, *Tree*(*C*) is a syntax tree to denote the structure of concept *C*.

Connection step: for any axiom $C_1 \sqsubseteq C_2$ in the TBox of Σ, we connect $Tree(C_1)$ and $Tree(C_2)$ by adding $(L(C_1), \sqsubseteq, L(C_2))$ in *E*.

Merging step: if two trees have the same structure, we will merge the two trees into one. For any label *l* in *V*, let $C_{out}(l) = \{(con_i, l_1) \mid (l, con_i, l_1) \in E_{out}(l)$ and $con_i \neq \sqsubseteq\}$ An merging rule is presented as follows: for any two labels *l* and l^*, if $C_{out}(l) = C_{out}(l^*)$, then merge two labels into one label *l* and add a set of new edges

$\{(l, con_i, l_1) | (l^*, con_i, l_1) \in E_{out}(l^*)\} \cup \{(l_1, con_i, l) | (l_1\ con_i, l^*) \in E_{in}(l^*)\}$ in E. When no merging rule can be applied to G, we call G is a SRView of KB $\sum$ and denote it as $G=SRV(\sum)$.

Obviously, a SRView $G(V, E)$ expresses syntax structures of concepts by concept tree and "subsumption" relationships by "$\sqsubseteq$" edge. For any edge in E can be seen as a production rule, a SRView explicitly creates a production system to explain complex concepts by containing all restrictions on these concepts.

```
Procedure AddTree(C) begin
If $L(C)$ is not defined
{add a new label $l$ in $V$ and let $L(C)=l$;
Switch ($C$) {
    Case $\neg C_1$ : AddTree ($C_1$); add a new edge ($L(C), \neg, L(C_1)$) in $E$;
    Case $C_1 \sqcap C_2$ : AddTree($C_1$); AddTree($C_2$); add ($L(C), \sqcap, L(C_1)$) and ($L(C), \sqcap, L(C_2)$) in $E$;
    Case $C_1 \sqcup C_2$ : AddTree($C_1$); AddTree($C_2$); add ($L(C), \sqcup, L(C_1)$) and ($L(C), \sqcup, L(C_2)$) in $E$;
    Case $\exists P.E$ : AddTree($E$); add ($L(C), \exists P, L(E)$) in $E$;
    Case $\forall P.E$ : AddTree($E$); add ($L(C), \forall P, L(E)$) in $E$;
    Case $\geq n\ P$ : add ($L(C), \geq n\ P, L(\top)$) in $E$;
    Case $\leq n\ P$ : add ($L(C), \leq n\ P, L(\top)$) in $E$; }
}
end
```

Fig. 1. *AddTree*( ) procedure

After talking about creating SRView, we will go into pruning it. The eight pruning operations are defined as triples (operation, old edge, substituted edge) in table 3.

Table 3. Pruning operations

Operation	Old Edge	Substituted Edge
Del	(l, con_i, l_1)	
$\neg$	$(l, \neg, l_1)$	$(L(\bot), \sqcap, l)$, $(L(\bot), \sqcap, l_1)$
$\sqcap$	$(l, \sqcap, l_1)$,$(l, \sqcap, l_2)$	$(l, \sqsubseteq, l_1)$, $(l, \sqsubseteq, l_2)$
$\sqcup$	$(l, \sqcup, l_1)$,$(l, \sqcup, l_2)$	$(l_1, \sqsubseteq, l)$,$(l_2, \sqsubseteq, l)$
$\exists P$	$(l, \exists P, l_1)$	$(l^*, \exists P^*, l_1)$, $(l^*, \sqsubseteq, l)$, where $P^* \sqsubseteq P$
$\forall P$	$(l, \forall P, l_1)$	$(l^*, \forall P^*, l_1)$, $(l^*, \sqsubseteq, l)$, where $P \sqsubseteq P^*$
$\geq n\ P$	$(l, \geq n\ P, l_1)$	$(l^*, \geq m\ P, l_1)$, $(l^*, \sqsubseteq, l)$, where $m>n$
$\leq n\ P$	$(l, \leq n\ P, l_1)$	$(l^*, \leq m\ P, l_1)$,$(l^*, \sqsubseteq, l)$, where $n>m$

In table 1, l^* is a new label added by pruning operations and "$P^* \sqsubseteq P$" means P^* is a sub-role of P. A pruning operation will replace old edges with substituted ones in E. Pruning operations rewrite SRView to release some restrictions and such rewriting will affect reasoning so that some blocked axioms will not be inferred by the pruned SRView.

Definition 6. An SRView $G^*(V^*, E^*)$ is a pruned view of $G(V, E)$ w.r.t a operation set $S_O=\{O_1, \dots, O_n\}$, if $G^*(V^*, E^*)$ is the new graph after applying all operations to G.

5 Inference Control with Pruned SRView

Now we give an overview of our query framework (figure 2), which describes data exchange between subject and knowledge base system. The knowledge base system consists of four processors (rectangles) and two stores (rounded rectangles). We will detailedly discuss how knowledge base system deals with a query.

1) When the knowledge base system receives a query $Q=Retrieval(C)$ from subject s, Parser parses C into its concept tree $Tree(C)$ and labels C as $L(C)$.

2) Connector connect $Tree(C)=G_C(V_C, E_C)$ and $SRV(\Sigma)=G_{KB}(V_{KB}, E_{KB})$ into a mixed SRView G, where $V=V_C \cup V_{KB}$ and $E=E_C \cup E_{KB}$. After exhaustively applying merging rules to G, Connector sends G as the final result.

3) Pruner gets G from Connector, and a pruning operation set $So(s)$ from Pruner store. Pruner applies $So(s)$ to prune G and return a pruned SRView G^* w.r.t $So(s)$.

4) Reasoner provides three sub-processors: SRView interpreter, ABox rewriter and RBox preprocessor (Figure 3). SRView interpreter sends an direct production $DP(C_{out}(l))$ and sup-label set $Lsup(l)$ of any label l to Tableau Reasoner.

$$DP(C_{out}(l))=\begin{cases} \neg l^* & \text{if } \#C_{out}(l)=1 \text{ and } (\neg, l^*)\in C_{out}(l) \\ con_i.l^* & \text{if } \#C_{out}(l)=1,\ (con_i, l^*)\in C_{out}(l) \text{ and } con_i \neq \neg \\ l_1 \sqcup l_2 & \text{if } \#C_{out}(l)=2 \text{ and } (\sqcup, l_1), (\sqcup, l_2)\in C_{out}(l) \\ l_1 \sqcap l_2 & \text{if } \#C_{out}(l)=2 \text{ and } (\sqcap, l_1), (\sqcap, l_2)\in C_{out}(l) \end{cases}$$

$$Lsup(l)= \quad \{l^* | (l, \sqsubseteq, l^*)\in E\}$$

ABox rewriter replaces any concept C with its label $L(C)$ in ABox A_Σ and new ABox is denoted as $L(A_\Sigma)$. For a RBox R_Σ, we introduce $\sqsubseteq_R$ as the transitive-reflexive closure of $\sqsubseteq$ on $R_\Sigma \cup \{S^- \sqsubseteq R^- \mid S \sqsubseteq R \in R_\Sigma\}$. RBox preprocessor computes "$\sqsubseteq_R$" relations that are repeatedly used in reasoning process. Finally Tableau Reasoner will return $Retrieval(L(C))$ as an answer set of the query to subject s.

Now we will prove that any answer in $Retrieval(L(C))$ is a validate answer to $Q=Retrieval(C)$, that means $Retrieval(L(C)) \subseteq Retrieval(C)$.

Definition 7. For a SRView $G=(V, E)$ and any l in V, let $TBox(l)=\{DP(C_{out}(l)) \sqsubseteq l, l \sqsubseteq DP(C_{out}(l))\} \cup \{l \sqsubseteq l^* | l^* \in Lsup(l)\}$, $TBox(G)=\bigcup_{l\in V}(TBox(l))$.

Lemma 1. For a query $Retrieval(C)$, a KB $\Sigma(T, R, A)$ and its SRView G_{KB}, let $G=Merge(G_{KB}, Tree(C))$, we can get that $\Sigma(T, R, A) \models i: C \Leftrightarrow \Sigma^*(TBox(G), R, L(A)) \models i: L(C)$.

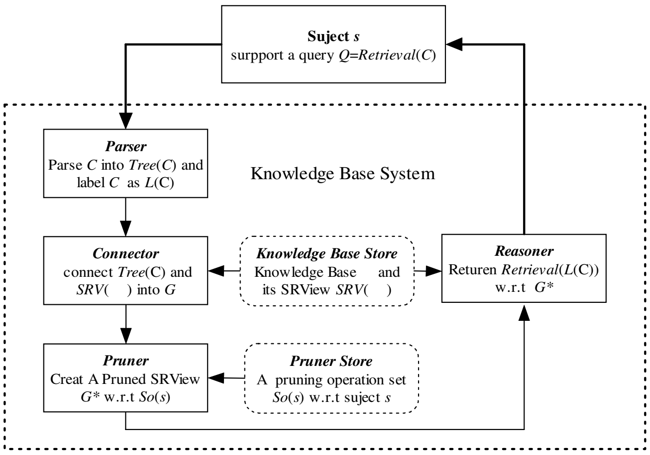
Suject *s*
surpport a query *Q=Retrieval*(*C*)
Parser
Parse *C* into *Tree*(*C*) and label *C* as *L*(C)
Knowledge Base System
Connector
connect *Tree*(C) and *SRV*( ) into *G*
Knowledge Base Store
Knowledge Base and its SRView *SRV*( )
Reasoner
Returen *Retrieval*(*L*(C)) w.r.t *G**
Pruner
Creat A Pruned SRView *G** w.r.t *So*(*s*)
Pruner Store
A pruning operation set *So*(*s*) w.r.t suject *s*

Fig. 2. A query framework with pruned SRViews

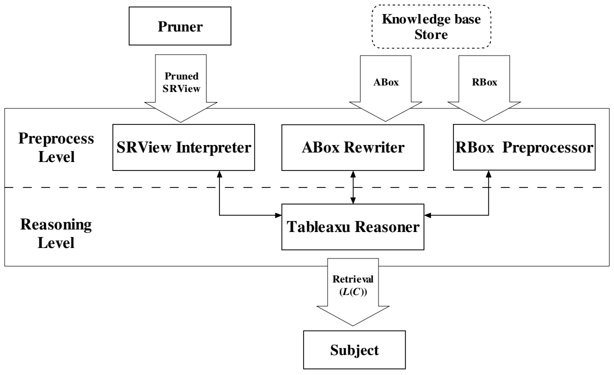

Fig. 3. Level structure of Reasoner

Lemma 2. For a subject *s* and corresponding pruning operation set $S_O(s)$ and a SRView *G*=(*V*, *E*), let *G**=(*V**, *E**) be a pruned SRView of *G* w.r.t $S_O(s)$, we have *TBox*(*G*)|=*TBox*(*G**).

For any pruning operation, it relaxes constraint of complex concepts. The pruned SRView can be inferred by original SRView. For example, $\geq n\ P$ rule adds a new label *l** by replacing *n* with a larger number *m* and states *l** is subsumed by *l*, directly $l = \geq n\ P\,.\top$ $l^* = \geq m\ P\,.\top$ can infer that $l^* \sqsubseteq l$. Therefore, lemma 2 holds.

Lemma 3. For a query *Retrieval*(*C*), a KB $\Sigma(T, R, A)$, a pruned SRView *G**=(*V**, *E**) and its label function *L*(), *Retrieval*(*L*(*C*)) w.r.t *G** is the set $\{i|\ \Sigma^*(TBox(G^*), R, L(A) \models i: L(C)\}$.

This Lemma obviously holds, for Tableau Reasoner considers *G**, *R* and *L*(*A*) as TBox, RBox and ABox of new KB respectively. From above three lemmas, we will have the following theorem.

Theorem 1. Let s be a subject, $S_O(s)$ a pruning operation set, $Retrieval(C)$ a query, $\Sigma(T, R, A)$ a KB and G_{KB} a SRView of Σ. $G=Merge(G_{KB}, Tree(C))$ and G^* is the pruned SRView of G w.r.t $S_O(s)$. We can get that if $\Sigma^*(TBox(G^*), R, L(A)) \models_i: L(C)$, $\Sigma(T, R, A) \models_i: C$. That also means $Retrieval(L(C)) \subseteq Retrieval(C)$.

After proving the correctness of our framework, we will go into complexity issue.

Lemma 4. For any $\Sigma(T, R, A)$, the creation process of its SRView $G_{KB}=(V_{KB}, E_{KB})$ can be performed in polynomial time of $(|sub(S_\Sigma)|+\#T)$.

In developing step: for any concept C in S_Σ, its concept tree $Tree(C)$ can be performed in polynomial of $sub(C)$. In connection step: for the number of edges with connective "$\sqsubseteq$" in E_{KB} is the number $\#T$ of axioms in T, this step can be performed in polynomial time of $\#T$. Finally in merging step, merging rule can at most be applied polynomial times of $(|sub(S_\Sigma)|+\#T)$. Lemma 4 also guarantees that G_{KB} can be stored in polynomial space of $(|sub(S_\Sigma)|+\#T)$.

Lemma 5. For a SRView G_{KB} and $Tree(C)$, $G=Merge(G_{KB}, Tree(C))$, the creation of G can performed in polynomial time of $(|sub(S_\Sigma)|+\#T+sub(C))$.

Lemma 6. For any SRView $G=(V, E)$ and its pruned SRView $G^*=(V^*, E^*)$ w.r.t $So(s)$, let $|G|=|V|+|E|$, $|G^*|$ is a linear function of $|G|$.

Based on the definition of pruning operations, any operation replaces one or two old edges with at most one new label and two edges. This guarantees that $|G^*| \leq 3|G|$.

Theorem 2. All procedures to deal with SRView in our framework can be performed in polynomial time of $(|sub(S_\Sigma)|+\#T+sub(C))$ and all SRViews can be stored in polynomial space of $(|sub(S_\Sigma)|+\#T+sub(C))$.

From Lemma 4-6, theorem 2 holds. For inference problem in SHOIN(D) is NEXPTIME-complete [4], the additional time and space spent on SRView will not be a bottleneck of a knowledge base system.

6 Conclusion and Further Work

In this paper, we define OWL ICR to formally present OWL inference control restrictions. To achieve ICRs, a new concept "SRView" is proposed to describe the whole semantic relation in an OWL KB. We also present a construction process of SRView and define pruning operations of them. Based on pruned SRView, we construct a query framework to achieve inference control and prove the correctness and complexity of it. Creating pruning operation sets w.r.t OWL ICRs is still a manual work in our framework. Future work includes designing some assistant method to semiautomatic or automatic create them, which will make our framework less manually interfered.

References

[1] Thuraisingham, B.: Security standards for the Semantic Web. Computer Standards & Interfaces, vol. 27 (2005) 257-268

[2] Fan, W., Chan, C., Garofalakis, M.: Secure XML Querying with Security Views. In: Proceedings of SIGMOD 2004, (2004) 587-598

[3] Reddivari, P., Finin, T., Joshi, A.: Policy based Access Control for a RDF Store. In: Proceedings of Policy Management for the Web workshop (position paper), 14th Int. World Wide Web Conf, Chiba, (2005) 78-81

[4] Horrocks, I., Patel-Schneider, P.F.: Reducing OWL entailment to description logic satisfiability. In: Proceedings of the 2003 Description Logic Workshop, (2003) 1-8

[5] Horrocks, I., Tessaris, S.: Querying the Semantic Web: a formal approach. In: Proceedings of 2002 Semantic Web Conf. (ISWC 2002), (2002) 177-191

[6] Haase, P., van Harmelen F., Huang Z., Stuckenschmidt, H., Sure Y.: A Framework for Handling Inconsistency in Changing Ontologies. In: Proceedings of the Fourth International Semantic Web Conference (ISWC2005), (2005) 353-367